Burroughs Corporation

COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

2219 0144

B1800/B1700 MCP CONTROL SYNTAX

PRODUCT SPECIFICATION

REV LTR	REVISION ISSUE DATE	APPROVED BY	REVISIONS
A	2/23/78	Wale	Original issue - Software Release Level Mark 7.0.

IPS 2228 3501 (C)
MCP Operators Manual

31-7

TABLE OF CONTENTS

INTRODUCTION

MCP CONTROL SYNTAX describes many of the areas of MCPII that change considerably from release to release and is a companion document to the MCP product specification 2212 5462, Master Control Program II. It contains six sections and an appendix, covering MCP communicates, declarations, halts, enhancements for the VII.0 release, program control attributes and SPO messages. Each section contains its own introduction; and entries, where possible, are indexed alphabetically.

RELATED DOCUMENTATION

Name	Number
B1800/B1700 Software Operational Guide	1068731
Master Control Program II	P.S. 2212 5462

COMMUNICATES

This section contains all the MCP communicates that are implemented at the Mark VII.0 release of MCPII. The Table of Contents sectionalizes them by number and the Alphabetic Index contains a named page reference for each individual communicate.

F O R M A T

VERB	0 - 11
OBJECT	12 - 35
ADVERB	36 - 47
CT.1	48 - 71
CT.2	72 - 95
CT.3	96 - 119
CT.4	120 - 143
CT.5	144 - 167
CT.6	168 - 191
CT.7	192 - 215
CT.8	216 - 239
CT.9	240 - 263
CT.10	264 - 297
CT.11	298 - 321
CT.12	322 - 345
CT.13	346 - 369
CT.14	370 - 393
CT.15	394 - 417

NOTE: ALL COMMUNICATES RETURN A VALUE OF
@000000000000@ OR @000018000000@
IN THE RS.REINSTATE.MSG.PTR UNLESS
OTHERWISE SPECIFIED.

CT.VERB 00

```
CT.VERB      00
             ILLEGAL COMMUNICATE
```

READ - CT.VERB 01

```
READ (MICRO MCP)
----
CT.VERB      01
CT.OBJECT    FILE.NUMBER
CT.ADVERB    BIT
             0      REPORT & RETURN TO USER ON EOF
             1      REPORT & RETURN TO USER ON PARITY
             2      REPORT & RETURN TO USER ON INCOMPLETE I/O
             3      LENGTH ADDRESS PAIR IS PRESENT FOR RESULT MASK FIELD
             4-6    -
             7      STACKERS--STACKER # IS IN CT.3
             8-11   -
CT.1         LOGICAL RECORD BIT LENGTH
CT.2         LOGICAL RECORD BASE RELATIVE BIT ADDRESS
CT.3         RANDOM FILE ACTUAL BINARY DISK KEY
             (RECORD NUMBER INSERTED BY MCP FOR SERIAL FILES)
             OR
             LENGTH OF KEY FOR REMOTE FILES
CT.4         ADDRESS OF KEY FOR REMOTE FILES ONLY
CT.5         LENGTH IN BITS OF RESULT MASK
CT.6         BASE RELATIVE ADDRESS OF RESULT MASK FIELD
REINSTATE.MSG.PTR VALUES
             0     GOOD READ
             1     END OF FILE
             2     I/O ERROR
             3     INCOMPLETE I/O
             4     IMPOSSIBLE SEARCH (RPG SEARCH OP)
```

WRITE = CT.VERB 02

```
WRITE (MICRO MCP)
-----
CT.VERB      02
CT.OBJECT    FILE.NUMBER
CT.ADVERB    BIT
             0      REPORT & RETURN TO USER ON EOF
             1      REPORT & RETURN TO USER ON PARITY
             2      REPORT & RETURN TO USER ON INCOMPLETE I/O
             3      LENGTH ADDRESS PAIR IS PRESENT FOR RESULT MASK FIELD
             4-5    -
             6      QUEUE FILES: WRITE TO FRONT OF
                    QUEUE ("STACK").
             7      STACKERS--STACKER # IS IN CT.3
             8-11   PRINTER SPACING  (4 BIT VALUE)
                    0  NO PAPER ADVANCE
                    1  SKIP TO CHANNEL 1  AFTER PRINTING
                    2  SKIP TO CHANNEL 2  AFTER PRINTING
                    3  SKIP TO CHANNEL 3  AFTER PRINTING
                    4  SKIP TO CHANNEL 4  AFTER PRINTING
                    5  SKIP TO CHANNEL 5  AFTER PRINTING
                    6  SKIP TO CHANNEL 6  AFTER PRINTING
                    7  SKIP TO CHANNEL 7  AFTER PRINTING
                    8  SKIP TO CHANNEL 8  AFTER PRINTING
                    9  SKIP TO CHANNEL 9  AFTER PRINTING
                    A  SKIP TO CHANNEL 10 AFTER PRINTING
                    B  SKIP TO CHANNEL 11 AFTER PRINTING
                    C  SKIP TO CHANNEL 12 AFTER PRINTING
                    D  SKIP TO NEXT CHANNEL AFTER PRINTING
                    E  SINGLE SPACE AFTER PRINTING
                    F  DOUBLE SPACE AFTER PRINTING
CT.1         LOGICAL RECORD BIT LENGTH
CT.2         LOGICAL RECORD BASE RELATIVE BIT ADDRESS
CT.3         RANDOM FILE ACTUAL BINARY DISK KEY
             (RECORD NUMBER INSERTED BY MCP FOR SERIAL FILES)
             OR
             LENGTH OF KEY FOR REMOTE FILES
CT.4         ADDRESS OF KEY FOR REMOTE FILES ONLY
CT.5         LENGTH IN BITS OF RESULT MASK
CT.6         BASE RELATIVE ADDRESS OF RESULT MASK FIELD
```

REINSTATE.MSG.PTR VALUES

0	GOOD WRITE
1	END OF FILE
2	I/O ERROR
3	INCOMPLETE I/O

SEEK = CT.VERB 03

SEEK (MICRO MCP)

CT.VERB	03
CT.OBJECT	FILE.NUMBER
CT.ADVERB	-
CT.1	-
CT.2	-
CT.3	RANDOM FILE ACTUAL BINARY DISK KEY

SORTER CONTROL - CT.VERB 04

```
SORTER CONTROL
--------------
CT.VERB       04
CT.OBJECT     FILE.NUMBER
CT.ADVERB     BIT
              0-5    -
              6      POCKET SELECT
              7      STOP-FLOW
              8      BATCH-COUNT
              9      POCKET LIGHT
              10     -
              11     ENDORSE
CT.1          POCKET NUMBER
```

SORTER READ - CT.VERB 05

```
SORTER READ (MICRO MCP)
-----------
CT.VERB       05
CT.OBJECT     FILE.NUMBER
CT.ADVERB     -
CT.1          READ AREA BIT LENGTH
CT.2          READ AREA BASE RELATIVE BIT ADDRESS
```

OPEN (DM) = CT.VERB 06

```
OPEN (DM)
---------
CT.VERB      06
CT.OBJECT    INVOKE NUMBER & PATH NUMBER
CT.ADVERB    BIT
             0      INCLUDES PACKID OF DICTIONARY
             1      -
             2      DM.STATUS FORMAT
                    0=BINARY
                    1=4-BIT DECIMAL
             3      ON EXCEPTION
             4      UPDATE
             5      REORGANIZATION ( REORG ONLY)
             6-11   -
CT.1         DM.STATUS REGISTER BIT LENGTH
CT.2         DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
CT.3         DATA BASE NAME BASE RELATIVE BIT ADDRESS
CT.4         DATA BASE NAME BIT LENGTH
CT.5         PACKID BASE RELATIVE BIT ADDRESS (BIT 0 OF
               CT.ADVERB = 1)
CT.6         PACKID BIT LENGTH (BIT 0 OF CT.ADVERB = 1)
```

CLOSE (DM) = CT.VERB 07

```
CLOSE (DM)
----------
CT.VERB      07
CT.OBJECT    -
CT.ADVERB    BIT
             0-1    -
             2      DM.STATUS FORMAT
                    0=BINARY
                    1=4-BIT DECIMAL
             3      ON EXCEPTION
             4-11   -
CT.1         DM.STATUS REGISTER BIT LENGTH
CT.2         DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
```

OPEN = CT.VERB 08

```
OPEN
----
CT.VERB      08
CT.OBJECT    FILE.NUMBER
CT.ADVERB    BIT
             0      INPUT
             1      OUTPUT
             2      NEW FILE
             3      PUNCH
             4      PRINT
             5      NO REWIND/INTERPRET (DATA RECORDERS)
             6      REVERSE/POCKET (CARD PUNCH)
             7      LOCK
             8      LOCKOUT
             9      REPORT FILE MISSING
            10      REPORT FILE LOCKED
            11      OVERRIDE NAMING CONVENTION AND SECURITY
REINSTATE.MSG.PTR VALUES
             0     GOOD OPEN
             1      FILE NOT PRESENT (INPUT DISK)
                    PACK NOT PRESENT (OUTPUT DISK)
                    NO MORE FILES ON MULTI-FILE REEL (TAPE)
             2     FILE LOCKED  (DISK FILES ONLY)
```

CLOSE = CT.VERB 09

```
CLOSE
-----
CT.VERB     09
CT.OBJECT   FILE.NUMBER
CT.ADVERB   BIT
             0      REEL
             1      RELEASE
             2      PURGE
             3      REMOVE
             4      CRUNCH
             5      NO REWIND
             6      OVERRIDE NAME CONVENTION AND SECURITY
             7      LOCK
             8      IF NOT CLOSED
             9      ROLLOUT
            10      AUDIT SWITCH
            11      TERMINATE
```

POSITION (MICRO MCP (BACKUP FILES ONLY)) = CT.VERB 10

```
POSITION (MICRO MCP (BACKUP FILES ONLY))
--------
CT.VERB        10
CT.OBJECT      FILE.NUMBER
CT.ADVERB      BIT
               0      REPORT & RETURN TO USER ON EOF
               1      REPORT & RETURN TO USER ON PARITY
               2      REPORT & RETURN TO USER ON INCOMPLETE I/O
               3-7
               8      POSITION TO END OF FILE
               9      CT.1 CONTAINS PRINTER CHANNEL NUMBER
              10      CT.1 CONTAINS RECORD COUNT AS A FIXED NUMBER
              11      CT.1 CONTAINS RECORD NUMBER DESIRED
CT.1           DEFINED BY BITS IN CT.ADVERB
REINSTATE.MSG.PTR VALUES
               0     GOOD POSITION
               1      END OF FILE (OR END OF PAGE ON PRINTER)
               2     I/O ERROR
               3     INCOMPLETE I/O
```

ACCESS FILE PARAMETER BLOCK (FPB) = CT.VERB 11

```
ACCESS FILE PARAMETER BLOCK (FPB)
---------------------------------
CT.VERB        11
CT.OBJECT      FILE.NUMBER
CT.ADVERB      BIT
               0-10  -
               11     0=READ
                      1=WRITE
CT.1           RECEIVING FIELD BIT LENGTH
CT.2           RECEIVING FIELD BASE RELATIVE BIT ADDRESS
```

ACCESS FILE INFORMATION BLOCK (FIB) - CT.VERB 12

```
ACCESS FILE INFORMATION BLOCK (FIB)
-----------------------------------
CT.VERB      12
CT.OBJECT    FILE.NUMBER
CT.ADVERB    BIT
             0-10  -
             11    FORMAT
                   0=CHARACTER
                   1=BINARY
CT.1         RECEIVING FIELD BIT LENGTH
CT.2         RECEIVING FIELD BASE RELATIVE BIT ADDRESS
```

DATA OVERLAY - CT.VERB 13

```
DATA OVERLAY
------------
CT.VERB      13
CT.OBJECT    BASE RELATIVE BIT ADDRESS OF 76 BIT FIELD IN FORMAT OF :
             4 BITS   -
             24 BITS  BEGINNING ADDRESS
             24 BITS  ENDING ADDRESS
             24 BITS  RELATIVE DISK ADDRESS
```

ACCESS DISK FILE HEADER (DFH) - CT.VERB 14

```
ACCESS DISK FILE HEADER (DFH)
-----------------------------
CT.VERB       14
CT.OBJECT     BASE RELATIVE ADDRESS OF 30 CHARACTER FILE IDENTIFIER :
              PACK.ID CAT MFID CAT FID
CT.ADVERB     BIT
              0-5    -
              6      IN.SECURE
              7-8    -
              9-11   0=WRITE
                     1=READ
                     2=READ & FORMAT IN BINARY
                     3=READ & FORMAT IN CHARACTERS
CT.1          RECEIVING FIELD BIT LENGTH
CT.2          RECEIVING FIELD BASE RELATIVE BIT ADDRESS
REINSTATE.MSG.PTR VALUES
              0     COMMUNICATE COMPLETE
              1     FILE NOT PRESENT
```

FIND/MODIFY (DM) - CT.VERB 15

```
FIND/MODIFY (DM)
----------------
CT.VERB       15
CT.OBJECT     INVOKE NUMBER & PATH NUMBER OF THE PATH-NAME
CT.ADVERB     BIT
              0      RETURN LIST HEADS (REORG ONLY)
              1      RETURN LOGICAL ADDRESS (REORG ONLY)
              2      DM.STATUS FORMAT
                     0=BINARY
                     1=4-BIT DECIMAL
              3      ON EXCEPTION
              4      -
              5      MODIFY
              6-10   SELECTION EXPRESSION
                     0  NEXT
                     1  PRIOR
                     2  FIRST
                     3  LAST
                     4  NEXT AT
                     5  CURRENT
                     6  AT
              11     DATA SET SELECTION EXPRESSION
CT.1          DM.STATUS REGISTER BIT LENGTH
CT.2          DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
CT.3          DATASET RECORD WORK AREA BIT LENGTH
CT.4          DATASET RECORD WORK AREA BASE RELATIVE BIT ADDRESS
CT.5          SEARCH KEY (CAT OF COMPONENT NAMES) BASE RELATIVE BIT ADR.
CT.6          INVOKE NUMBER & PATH NUMBER OF DATASET-NAME
```

STORE (DM - CT.VERB 16

```
STORE (DM)
----------
CT.VERB      16
CT.OBJECT    INVOKE NUMBER & PATH NUMBER
                   (SUBSET IF INSERT)
CT.ADVERB    BIT
             0     INSERT
             1     -
             2     DM.STATUS FORMAT
                   0=BINARY
                   1=4-BIT DECIMAL
             3     ON EXCEPTION
             4     BEGIN TRANSACTION (NOT INSERT)
             5     INCLUDES.LIST.HEADS (REORG ONLY)
             6     END TRANSACTION (NOT INSERT)
             7     NO AUDIT (BEGIN OR END TRANSACTION ONLY)
             8     SYNC (END TRANSACTION ONLY)
             9     -
             10    STORE INDEXES ONLY (REORG ONLY)
             11    PSEUDO CREATE (REORG ONLY)
CT.1         DM.STATUS REGISTER BIT LENGTH
CT.2         DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
CT.3         DATASET RECORD WORK AREA BIT LENGTH (NOT INSERT)
             INVOKE NUMBER & PATH NUMBER OF DATASET (INSERT)
```

DELETE (DM) = CT.VERB 17

```
DELETE (DM)
-----------
CT.VERB       17
CT.OBJECT     INVOKE NUMBER & PATH NUMBER
              (SUBSET IF REMOVE)
CT.ADVERB     BIT
              0      REMOVE
              1      -
              2      DM.STATUS FORMAT
                     0=BINARY
                     1=4-BIT DECIMAL
              3      ON EXCEPTION
              4-11   -
CT.1          DM.STATUS REGISTER BIT LENGTH
CT.2          DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
CT.3          DATASET RECORD WORK AREA BIT LENGTH (NOT REMOVE)
              INVOKE NUMBER & PATH NUMBER OF DATASET (REMOVE)
CT.4          DATASET RECORD WORK AREA BASE RELATIVE BIT ADDRESS
                                                    (NOT INSERT)
```

CREATE/RECREATE (DM) = CT.VERB 18

```
CREATE/RECREATE (DM)
--------------------
CT.VERB       18
CT.OBJECT     INVOKE NUMBER & PATH NUMBER
CT.ADVERB     BIT
              0      -
              1      RECREATE
              2      DM.STATUS FORMAT
                     0=BINARY
                     1=4-BIT DECIMAL
              3      ON EXCEPTION
              4-11   -
CT.1          DM.STATUS REGISTER BIT LENGTH
CT.2          DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
CT.3          DATASET RECORD WORK AREA BIT LENGTH
CT.4          DATASET RECORD WORK AREA BASE RELATIVE BIT ADDRESS
```

SWITCH.TAPE.DIRECTION = CT.VERB 19

```
SWITCH.TAPE.DIRECTION
---------------------
CT.VERB       19
CT.OBJECT     FILE.NUMBER
CT.ADVERB     BIT
              0-7  NOT USED
              8-11 0 = READ FORWARD
                   1 = READ REVERSE
                   4 = WRITE
REINSTATE.MSG.PTR VALUES
              0    GOOD SWITCH
              1    FILE NOT OPEN
              2    WRONG DIRECTION OR NOT A TAPE FILE
              3    END OF FILE
```

TERMINATE (STOP RUN) = CT.VERB 20

```
TERMINATE (STOP RUN)
--------------------
CT.VERB      20
```

FREE (DM) = CT.VERB 21

```
FREE (DM)
---------
CT.VERB      21
CT.OBJECT    INVOKE NUMBER & PATH NUMBER
CT.ADVERB    BIT
             0-1    -
             2      DM.STATUS FORMAT
                    0=BINARY
                    1=4-BIT DECIMAL
             3      ON EXCEPTION
             4-11   -

CT.1         DM.STATUS REGISTER BIT LENGTH
CT.2         DM.STATUS REGISTER BASE RELATIVE BIT ADDRESS
```

TIME/DATE/DAY - CT.VERB 22

```
TIME/DATE/DAY
-------------
CT.VERB      22
CT.OBJECT    BASE RELATIVE BIT ADDRESS OF WHERE TO PUT THE RESULT
CT.ADVERB    BIT
             0      1=DATE REQUESTED
             1-2    FORMAT
                    0 YY/DDD   (JULIAN)
                    1 MM/DD/YY
                    2 YY/MM/DD
                    3 DD/MM/YY
             3-4    REPRESENTATION
                    0 BINARY
                    1 4-BIT DECIMAL
                    2 8-BIT DECIMAL
             5      1=TIME REQUESTED
             6-7    FORMAT
                    0 COUNTER
                    1 HH:MM:SS.S (24-HOUR CLOCK)
                    2 HH:MM:SS.S TT (12-HOUR CLOCK, TT=AM/PM)
             8-9    REPRESENTATION
                    0 BINARY
                    1 4-BIT DECIMAL
                    2 8-BIT DECIMAL
             10     1=TODAYS.NAME REQUESTED
             11     -
NOTE : TODAYS.NAME RETURNS 9 CHARACTERS LEFT JUSTIFIED
```

FORMAT	BINARY	4-BIT DECIMAL	8-BIT DECIMAL
YY/DDD (JULIAN)	7+9=16	8+12=20	16+24=40
MM/DD/YY	4+5+7=16	8+8+8=24	16+16+16=48
YY/MM/DD	7+4+5=16	8+8+8=24	16+16+16=48
DD/MM/YY	5+4+7=16	8+8+=24	16+16+16=48
COUNTER	20	24	48
HH:MM:SS.S	5+6+6+4=21	8+8+8+4=28	16+16+16+8=56
HH:MM:SS.S TT	4+6+6+4+16=36	8+8+8+4+16=44	16+16+16+8+16=72
TODAYS.NAME			72 (9 CHAR, LEFT JUST.)

INITIALIZER I/O - CT.VERB 23

```
INITIALIZER I/O
---------------
CT.VERB      23
CT.OBJECT    BASE RELATIVE ADDRESS OF
                    6 BYTE UNIT MNEMONIC
                    OR
                    I/O DESCRIPTOR
CT.ADVERB    VALUE
             0      ASSIGN UNIT TO THIS PROGRAM
             1      RELEASE UNIT
             2      LINK IN THE I/O DESCRIPTOR BUT DO NOT INITIATE
             3      LINK IN THE I/O DESCRIPTOR AND INITIATE
             4      LINK IN, INITIATE, WAIT, AND IGNORE
                    EXCEPTIONS (NON-RELEASE MCPS ONLY)
             5-11   -
REINSTATE.MSG.PTR VALUES
             IF CT.ADVERB=0 THEN
             PORT, CHANNEL AND UNIT OF DEVICE REQUESTED
                  PORT BIT (3)
                  CHANNEL BIT (4)
                  FILLER BIT (1)
                  UNIT BIT (4)
             ALL OTHER CASES RETURN 0
```

WAIT (SNOOZE) = CT.VERB 24

```
WAIT (SNOOZE)
-------------
CT.VERB      24
CT.OBJECT    LENGTH OF TIME IN 10THS OF A SECOND
  FUNCTION   PROGRAM IS PUT TO SLEEP FOR SPECIFIED LENGTH OF TIME
```

ZIP = CT.VERB 25

```
ZIP
---
CT.VERB      25
CT.OBJECT    -
CT.ADVERB    -
CT.1         MESSAGE AREA BIT LENGTH
CT.2         MESSAGE AREA BASE RELATIVE BIT ADDRESS
REINSTATE.MSG.PTR VALUES
             0    NO ERRORS IN ZIP TEXT
             1    ZIPPED INVALID CONTROL CARD
```

ACCEPT = CT.VERB 26

```
ACCEPT
------
CT.VERB      26
CT.OBJECT    -
CT.ADVERB    BIT
             0     RETURN IF NO MESSAGE
             1-11  -
CT.1         MESSAGE AREA BIT LENGTH
CT.2         MESSAGE AREA BASE RELATIVE BIT ADDRESS
REINSTATE.MSG.PTR VALUES
             0    MESSAGE OF LENGTH ZERO
     @FFFFFF@     NO MESSAGE PRESENT
     ANY OTHER VALUE LENGTH OF MESSAGE IN BITS
```

DISPLAY - CT.VERB 27

```
DISPLAY
-------
CT.VERB       27
CT.OBJECT     -
CT.ADVERB     BIT
              0-10  -
              11    0=CRUNCH BLANKS OUT OF MESSAGE
                    1=PRINT MESSAGE AS IS
CT.1          MESSAGE AREA BIT LENGTH
CT.2          MESSAGE AREA BASE RELATIVE BIT ADDRESS
```

USE/RETURN - CT.VERB 28

```
USE/RETURN
----------
CT.VERB       28
```

SORT HANDLER - CT.VERB 29

```
SORT HANDLER
------------
CT.VERB       29
CT.OBJECT     BASE RELATIVE ADDRESS OF SORT INFORMATION TABLE
CT.ADVERB     BIT (12)
              0 -  SORT.RESTART
              1 -  SORT.DUPCHECK
              2 -  SORT.W1.PID
              3 -  SORT.W2.PID
              4-11 FILLER
CT.1          BASE RELATIVE BIT ADDRESS OF SORT KEY TABLE
CT.2          INPUT FILE.NUMBER OR ADDR OF MERGE.INPUT.TABLE IF MERGE
CT.3          OUTPUT FILE.NUMBER
CT.4          TRANSLATE FILE.NUMBER OR NOT 0
CT.5          -
CT.6          DATA.ADDRESS (DELETE.KEY.TABLE)
CT.7          IF (SORT.W1.PID := W1.PID.FLAG) THEN
                 DATA.ADDRESS (W1.PID) ELSE 0
CT.8          IF (SORT.W2.PID := W2.PID.FLAG) THEN
                 DATA.ADDRESS (W2.PID) ELSE 0
```

SDL TRACE - CT.VERB

```
SDL TRACE
---------
CT.VERB      30
CT.OBJECT    TRACE FLAGS
```

EMULATOR TAPE (MICRO MCP) - CT.VERB 31

```
EMULATOR TAPE (MICRO MCP)
-------------------
CT.VERB      31

CT.OBJECT    FILE.NUMBER

CT.ADVERB    BIT
             0-2     OP.CODE
                        0 = READ
                        1 = WRITE
                        2 = SPACE
                        3 = REWIND
                        4 = TEST
             3-8     OP.CODE.VARIANT
                 3 = REVERSE (READ, SPACE), ERASE (WRITE),
                     TEST.WAIT.READY.NOT.REWIND (TEST)
                 4 = ONE.RECORD (SPACE), TAPE.MARK (WRITE),
                     TEST.WAIT.NOT.READY (TEST)
                 5 = ODD.PARITY (READ, SPACE, WRITE)
                 6 = NOISE (READ, SPACE)
               7-8 = NOT USED
             9-11    SCHEDULING.VARIANTS
                 9 = FETCH.RESULT
                10 = DONT.WAIT
                11 = REPORT AND RETURN ON IO ERROR

CT.1         USER TAPE BUFFER BIT LENGTH

CT.2         USER TAPE BUFFER BASE RELATIVE ADDRESS

CT.3         USER ERROR MASK (BIT SET IMPLIES USER WILL HANDLE THE
                              CORRESPONDING ERROR)
             BIT
             0        (MAY NOT USE)
```

1	(MAY NOT USE)
2	NOT READY
3	PARITY (NOT ON TEST)
4	ACCESS (NOT ON TEST)
5	TRANSMISSION (ON TEXT ONLY)
6	END.OF.TAPE
7	BEGINNING.OF.TAPE
8	WRITE.LOCK.OUT
9	END.OF.FILE (NOT ON TEST), UNIT.PRESENT (ON TEST)
10	REWINDING
11	TIME.OUT (NOT ON TEST)
12-16	(MAY NOT USE)
17	SHORT.RECORD
18	LONG.RECORD
19	DROPOUT
20	INITIATE.LATE
21	(MAY NOT USE)
22	TRANSMISSION.ERROR.MEC
23	TRANSMISSION.ERROR.MTC

CT.4 BASE RELATIVE ADDRESS OF USER'S 48 BIT RESULT
BIT 0-23 OF RESULT CONTAIN THE RESULT DESCRIPTOR
BIT 24-47 OF RESULT CONTAIN THE ACTUAL LENGTH

REINSTATE.MSG.PTR VALUES
0 = RESULT RETURNED
1 = IO.ERROR
2 = RESULT NOT AVAILABLE

COBOL PROGRAM ABNORMAL END - CT.VERB 32

COBOL PROGRAM ABNORMAL END

CT.VERB	32

SORT EOJ - CT.VERB

SORT EOJ

CT.VERB	33
CT.OBJECT	FILE.NUMBER
CT.ADVERB	CLOSE TYPE
CT.1	END-OF-FILE POINTER
CT.2	RECORD SIZE

CT.VERB 34 (UNUSED)

FREEZE/THAW RUN STRUCTURE - CT.VERB 35

```
FREEZE/THAW RUN STRUCTURE
-------------------------
CT.VERB       35
CT.OBJECT     BIT 0 (HIGH ORDER BIT)
              0=THAW
              1=FREEZE
```

COMPILE CARD INFORMATION - CT.VERB 36

```
COMPILE CARD INFORMATION
----------------------------

CT.VERB       36
48 BITS       SDL DESCRIPTOR (WHERE TO PUT INFO) IN FORMAT :
              16 BITS=LENGTH
              24 BITS=ADDRESS

              RETURNS COMPILE CARD INFO IN FOLLOWING FORMAT :
              #CHARS  INFO
              .....   ...................................
              30      OBJECT NAME
              02      EXECUTE TYPE
              10      PACK.NAME OF THE RUNNING PROGRAM
              30      INTERPRETER NAME OF THE RUNNING PROGRAM
              10      INTRINSIC NAME (FAMILY)
              02      PRIORITY
              06      SESSION NUMBER
              06      JOB NUMBER
              20      1ST & 2ND NAMES OF RUNNING PROGRAM
              07      CHARGE NUMBER
              01      FILLER
              36 BITS DATE AND TIME COMPILED
              04 BITS FILLER
              10      USERCODE
              10      PASSWORD
              04      PARENT JOB NUMBER
              20      PARENT QUEUE IDENTIFIER
              01      LOG SPO
```

DYNAMIC MEMORY BASE - CT.VERB 37

```
DYNAMIC MEMORY BASE
-------------------
CT.VERB       37
              VALUE IS RETURNED IN COMMUNICATE MESSAGE POINTER AS
              SELF RELATIVE DESCRIPTOR
```

MEMORY DUMP TO DISK - CT.VERB 38

```
MEMORY DUMP TO DISK
-------------------
CT.VERB      38
```

GET SESSION NUMBER - CT.VERB 39

```
GET SESSION NUMBER
------------------
CT.VERB      39
             SESSION IS PUT INTO RS.REINSTATE.MSG.POINTER
```

DC.INITIATE.IO - CT.VERB 40

```
DC.INITIATE.IO
--------------
CT.VERB      40
24 BITS      PORT
24 BITS      CHANNEL
24 BITS      BASE RELATIVE ADDRESS OF I/O DESCRIPTOR
```

NDL/MACRO COMMUNICATES - CT.VERB 41

```
NDL/MACRO COMMUNICATES
----------------------
CT.VERB       41
CT.OBJECT     INDICATES FUNCTION
DESC1         BIT  1-48    MESSAGE AREA 1
DESC2         BIT 49-96    MESSAGE AREA 2
QUEUE.PTR     BIT 97-106   REMOTE FILE NUMBER OR STATION NUMBER
```

DCWRITE

```
DCWRITE
-------
CT.OBJECT     11
DESC1         RESULT AREA
DESC2         DC.WRITE MESSAGE
  NOTE:       NUMBER AT SUBSTR(DESC2,6,2) IS MESSAGE TYPE
              40=FINISH OPEN
              41=NDL/MACRO PRESENT
              42=ATTACH STATIONS TO REMOTE FILE
              43=DETACH STATIONS FROM REMOTE FILE
```

QUICK QUEUE WRITE (REMOTE FILES)

```
QUICK QUEUE WRITE (REMOTE FILES)
--------------------------------
CT.OBJECT     12
DESC1         MESSAGE HEADER
DESC2         MESSAGE
RMT.FL        REMOTE FILE TO WHICH THE MESSAGE IS DESTINED
```

QUICK QUEUE WRITE (STATION NUMBER)

QUICK QUEUE WRITE (STATION NUMBER)	
CT.OBJECT	13
DESC1	MESSAGE HEADER
DESC2	MESSAGE
ST.NR	STATION NUMBER

ACCESS USERCODE FILE

```
ACCESS USERCODE FILE
------ -------- ----
CT.VERB      42
DESC         BIT 0-47 DESCRIPTOR TO PARAMETER LIST.

  PARAMETER LIST LAYOUT
    MODE     BIT (4)
      0      SET ALL PARAMETERS IN LIST EXCEPT USERCODE AND
               PASSWORD.  THESE MUST BE SUPPLIED TO FIND
               CORRECT ENTRY.
      1      SET ALL PARAMETERS IN LIST EXCEPT INDEX. INDEX
               MUST BE SUPPLIED TO FIND ENTRY.
      2      SET OVERRIDE.  USERCODE MUST BE PRESENT TO FIND
               ENTRY. THE OVERRIDE FIELD FOR ALL OCCURRENCES OF
               THIS USERCODE WILL BE SET.
      3      SET OVERRIDE.  INDEX MUST BE SUPPLIED TO FIND ENTRY.
               THE OVERRIDE FIELD FOR ALL OCCURRENCES OF THIS
               USERCODE WILL BE SET.
      4      ADD ENTRY.  ALL FIELDS HAVE TO BE SUPPLIED.
      5      DELETE ENTRY.  USERCODE AND PASSWORD MUST BE
               SUPPLIED TO FIND ENTRY.
      6      INITIALIZE ALL OVERRIDE BITS.
      7      CHANGE BY USERCODE.  ALL ENTRIES FOR A GIVEN USER-
               CODE CAN BE CHANGED WITH ONE COMMUNICATE.  USER-
               CODE MUST BE PRESENT.  PACK FIELD MUST NOT BE
               EQUAL TO ZERO TO CHANGE IT.  CHARGE NUMBER MUST
               NOT BE EQUAL TO ZERO TO CHANGE IT.  PRIORITY MUST
               NOT BE EQUAL TO ZERO TO CHANGE IT.
      8      DELETE ALL RECORDS FOR A GIVEN USERCODE. USER-
               CODE MUST BE PRESENT.
      9      SET ALL PARAMETERS IN LIST EXCEPT USERCODE AND
               PASSWORD.  ONLY USERCODE HAS TO BE SUPPLIED
               BECAUSE SEARCH STOPS ON FIRST ENCOUNTER OF
               GIVEN USERCODE.
     10      CHANGE BY INDEX.  INDEX MUST BE PRESENT.
               PRIORITY CAN BE CHANGED BY SETTING FIELD TO NON-
               ZERO.  CHARGE CAN BE CHANGED BY SETTING CHARGE
               FIELD TO NON-ZERO.  PASSWORD CAN BE CHANGED BY
               SETTING PASSWORD TO NON-ZERO.
     11      CLEAR PACK OVERRIDE FIELD FOR ALL OCCURRENCES OF
               THIS USERCODE.  USERCODE MUST BE SUPPLIED.
     12      CLEAR PACK OVERRIDE BID FOR ALL OCCURRENCES OF THIS
               USERCODE. INDEX MUST BE SUPPLIED.
```

```
INDEX     BIT (10)
USERCODE CHARACTER (10)
          WHEN SET BY PROGRAM (MODE = 0, 2, 4, 5, 7, 8, 9, 11),
            THE USERCODE MAY OR MAY NOT CONTAIN PARENTHESES.
            IF PARENS ARE NOT FOUND, ONLY THE FIRST EIGHT
            USED.
          WHEN SET BY MCP (MODE = 1)
            USERCODE WILL ALWAYS CONTAIN PARENTHESES.
PASSWORD  CHARACTER (10)
PACK NAME CHARACTER (10)
CHARGE #  BIT (24)
PRIORITY  BIT (4)
PRIVILGD  BIT (1)
OVERRIDE BIT  (1)
REINSTATE.MSG.PTR VALUES
          0  NO ERRORS.
          1  ERROR ON INPUT: EITHER INDEX IS WRONG OR
             USERCODE/PASSWORD IS NOT PRESENT.
          2  "(SYSTEM)/USERCODE" FILE NOT IN "US" SLOT.
```

<u>CT.VERB 43 (UNUSED)</u>

<u>PROGRAM CALLER = CT.VERB 44</u>

```
PROGRAM CALLER
--------------
CT.VERB      44
48 BITS      SDL DESCRIPTOR
             24 BIT LENGTH OF TEXT
             24 BIT BASE RELATIVE ADDRESS OF TEXT
```

<u>STACK SIZE CHANGE = CT.VERB 45</u>

```
STACK SIZE CHANGE
-----------------
CT.VERB      45
CT.OBJECT    CODEFILE-RELATIVE DISK ADDRESS OF NEW SPAD
```

LOAD.DUMP MESSAGE = CT.VERB 46

```
LOAD.DUMP MESSAGE
-----------------
CT.VERB      46
CT.OBJECT    BASE RELATIVE ADDRESS OF MESSAGE
CT.ADVERB    BIT
             0      1=LOADED  0=DUMPED
             1-11   -
```

COMPLEX WAIT - CT.VERB 47

```
COMPLEX WAIT (MICRO MCP)
------- ----
CT.VERB      47
CT.OBJECT    NUMBER OF EVENTS
CT.ADVERB    FIRST EVENT TO CHECK (CHECKED IN CIRCULAR
                  FASHION FROM THIS POINT).
CT.1-ETC.    BIT ENCODED EVENTS (NUMBER SPECIFIED BY CT.OBJECT
                  MAX=15).
              0- 3 EVENT TYPE
              4- 7 EVENT PARAM1
              8-15 EVENT PARAM2
             16-24 EVENT PARAM3

             EVENT TYPES:
                   0 - NULL - PARAM1,2,3 : NOT USED
                   1 - SPO INPUT PRESENT - PARAM1,2,3 : NOT USED
                   2 - TIME - PARAM1,2,3 : CONCATENATED BIT 20
                       FIELD CONTAINING THE LENGTH OF TIME TO
                       WAIT IN 10THS OF A SECOND
                   3 - READ OK -PARAM1: NOT USED, PARAM2:
                       FILE NUMBER, PARAM3: MEMBER NUMBER IF FILE IS
                       Q-FILE-FAMILY
                   4 - WRITE OK - PARAM1,2,3: SAME AS READ OK
                   5 - QUEUE WRITE OCCURRED - PARAM1: NOT USED,
                       PARAM2: FILE NUMBER OF Q-FILE-FAMILY,
                       PARAM3: NOT USED
                   6 - DATA COMM IO COMPLETE - PARAM1,2,3: NOT USED
REINSTATE.MSG.PTR VALUES
  ZERO RELATIVE INDEX TO THE COMMUNICATE EVENT LIST ELEMENT
  WHICH IS COMPLETE
```

MESSAGE COUNT - CT.VERB 48

```
MESSAGE COUNT
------- -----
CT.VERB      48
CT.OBJECT    FILE.NUMBER
CT.ADVERB    0      DECIMAL FORMAT RESULTS IF TRUE
                        COBOL ("PIC 999")
                    ELSE BINARY (BIT (24))
             1-11 -
CT.1         RESULT FIELD LENGTH
CT.2         BASE RELATIVE RESULT FIELD ADDRESS
  FUNCTION   RETURN THE COUNT OF THE MESSAGES CONTAINED
             IN THE QUEUE-FILE SPECIFIED. IF THE OBJECT
             IS A QUEUE-FILE-FAMILY, THE COUNT WILL BE
             RETURNED AS A LEFT-JUSTIFIED ARRAY OF
             24-BIT COUNTS, ONE FOR EACH MEMBER OF
             THE FAMILY.
```

DISK READER = CT.VERB 49

```
DISK READER
---- ------
CT.VERB       49
CT.OBJECT     BASE RELATIVE ADDRESS OF RECEIVER FIELD
CT.ADVERB     PORT, CHANNEL AND UNIT OF DESIRED DISK
CT.1          DISK ADDRESS
CT.2          LENGTH (IN BITS) OF RECEIVER FIELD
REINSTATE.MSG.PTR VALUES
              0    READ DONE AS REQUESTED
              1    READ NOT DONE
```

RECOVERY COMPLETE = CT.VERB 50

```
RECOVERY COMPLETE
-----------------
CT.VERB       50
CT.OBJECT
              DATA BASE GLOBALS BASE RELATIVE BIT ADDRESS
```

GET.ATTRIBUTES = CT.VERB 51

```
GET.ATTRIBUTES
--------------
CT.VERB       51
CT.OBJECT     FILE NUMBER
CT.ADVERB     COMMUNICATE LEVEL (MK 7.0 LEVEL=1)
CT.1          TOTAL ATTRIBUTES (MUST BE 1 IN 7.0)
CT.2          BASE RELATIVE ADDRESS OF ATTRIBUTE LIST
```

CHANGE.ATTRIBUTES = CT.VERB 52

```
CHANGE.ATTRIBUTES
-----------------
CT.VERB      52
CT.OBJEXT    FILE NUMBER
CT.ADVERB    COMMUNICATE LEVEL (MK 7.0 LEVEL=1)
CT.1         TOTAL ATTRIBUTES (MUST BE 1 IN 7.0)
CT.2         BASE RELATIVE ADDRESS OF ATTRIBUTE LIST
```

DECLARATIONS

DISK AVAILABLE

```
    DEFINE
    DISK.AVAILABLE.DECLARATION AS#
DECLARE
    01 DUMMY REMAPS DISK.AVAILABLE BIT(SEG.SIZE),
       02 AVL.FOR.LINK        DSK.ADR,
       02 AVL.BACK.LINK       DSK.ADR,
       02 AVL.SELF            DSK.ADR,
       02 FILLER              BIT(4),
       02 AVL.BLOCK(22),
          03 AVL.ADDRESS      DSK.ADR,
          03 AVL.LENGTH       WORD;#;
```

PACK LABEL

```
DEFINE  PACK.LABEL.DECLARATION  AS #%
DECLARE 01 DUMMY REMAPS PACK.LABEL%
,   02 PL.VOL1                 CHAR (4)  %         "VOL1"
,   02 PL.SERIAL.NO            CHAR (6)  %         SERIAL (CAN) NUMBER
,   02 PL.ACCESS.CODE          CHAR (1)  %         ACCESS CODE
,   02 PL.ID                   CHAR (17) %         PACK ID
,        03 PL.NAME            CHAR (10) %
,        03 FILLER             CHAR (7)  %
,   02 PL.SYSTEM.INTERCHANGE CHAR (2)  %         SYSTEM INTERCHANGE/CODE
                                         %  00 = INTERCHANGE
                                         %  17 = B1700 INTERNAL
                                         %  35 = B3500 INTERNAL
                                         %  ETC, ETC, ETC
,   02 PL.CODE                 CHAR (1)  %         PACK CODE 00 = SCRATCH
,   02 FILLER                  CHAR (6)  %
,   02 PL.OWNER.ID             CHAR (14) %
,   02 PL.TYPE                 CHAR (1)  %         "R" = RESTRICTED PACK
                                         %         "U" = USER PACK
                                         %         "S" = SYSTEM.PACK
```

```
,    02 PL.CONTINUE             CHAR (1)  %        CONTINUATION FLAG "C"
,    02 FILLER                  CHAR (26) %
,    02 PL.INT                  CHAR (1)  %
,    02 PL.VOL2                 CHAR (4)  %        "VOL2"
,    02 PL.DATE.INITIALIZED     CHAR (5)  %
,    02 PL.INIT.SYSTEM          CHAR (6)  %        INITIALIZING SYSTEM
,    02 PL.DISK.DIRECTORY       CHAR (8)  %        DIRECTORY ADDRESS
,    02 PL.MASTER.AVAIL         CHAR (8)  %        MASTER AOAILABLE TABLE
,    02 PL.DISK.AVAILABLE       CHAR (8)  %        WORKING AVAILABLE TABLE
,    02 PL.INTEGRITY            CHAR (1)  %        0 = NORMAL
                                          %        1 = RECOVERY REQUIRED
,    02 PL.ERROR.COUNT          CHAR (6)  %
,    02 PL.SECTORS.XD           CHAR (6)  %        REMOVED SECTORS
,    02 PL.TEMP.TABLE           CHAR (8)  %        TEMP TABLE LINK
,    02 PL.PCD                  CHAR (3)  % LAST PORT, CHAN, DRIVE
,    02 PL.ASSIGNED.TO.BPS      CHAR (6)  %    BASE PACK SERIAL NUMBER
,    02 PL.SP.SEC.FLAGS          CHAR (8)  % SPARE.SECTOR.TABLE FOR 225 DSK
,    02 FILLER                   CHAR (23)
;#;%
```

FILE HEADER

```
DEFINE FILE.HEADER.DECLARATION AS #%
    FH.MAP(FILE.HEADER)#,
FH.MAP(FILE.HEADER) AS #%
DECLARE 01 DUMMY REMAPS FILE.HEADER,%
    02  FH.USERS.RANDOM       BIT(8),% FORMERLY FH.CORE.ADDR
    02  FH.NEWFILE            BIT(1),% CLEARED WHEN NEW FILE IS FILED.
    02  FILLER                BIT(1),%
    02  FILLER                BIT(6),%
    02  FH.FILE.KIND          BIT(8),%
    02  FH.SELF               DSK.ADR,
    02  FH.NO.USERS           BIT (8),
    02  FH.USERS.OPEN.OUT     BIT (4),
    02  FH.OPEN.TYPE          BIT (4),
    02  FH.FILE.TYPE          BIT (4),
```

```
02  FH.PERMANENT                 BIT (4),
02  FH.JOB.WAITING.ON.CLOSE  BOOLEAN,
02  FILLER                   BIT(9), % DON'T USE UNTIL 1977
02  FH.HDR.SIZE              BIT(14),% LENGTH OF MYSELF IN BITS.
02  FILLER                   BIT(4),% DON'T REUSE TILL 1977.
02  FH.RECORD.SIZE           BIT(20),% LENGTH IN BITS.
02  FILLER                   BIT(4),% DON'T USE TILL 1977
02  FH.RCDS.BLOCK            BIT(20),%
02  FH.BLOCKS.AREA           WORD,
02  FH.SEGS.AREA             WORD,
02  FH.AREAS.RQST            BIT (12),
02  FH.AREA.CTR              BIT (12),
02  FH.EOF.POINTER           WORD,
02  FILLER                   BIT(4),%DON'T USE TILL 1977
02  FH.BPS.NO                BIT(20),%
02  FH.BLOCK.COUNT           BIT(24),% DON'T USE TILL 1977. IGNORED 5.1.
02  FH.FORMAT                BIT(3),% HITHERTO =0.  FOR RELEASE, =1.
02  FH.MPF                   BIT(1),% HITHERTO 4 BITS.
02  FH.DBM.LINK                  ADDRESS,
02  FH.CREATE.TIME           BIT(16),% HITHERTO 0. HENIGE'S GENEROSITY.
02  FH.DBM.DFH.NO            BIT(8),% HITHERTO 24 BITS.
02  FH.USER.INFO                 WORD,
02  FH.SAVE.FACTOR               BIT (12),
02  FH.CREATION.DATE             BIT (16),
02  FH.ACCESS.DATE           BIT(16),%
02  FH.SER.NO                BIT(24),% DON'T REUSE TILL 1977. 5.1 IGNORE
02  FH.MPF.ADDR             DSK.ADR,  % DONT REUSE TILL 1977
02  FILLER                       BIT(1),
02  FH.UPDATE.VERSION            BOOLEAN,
02  FH.DMS.WRITE.CONTROL,
     03   FH.DMS.TO.BE.WRITTEN      BOOLEAN,
     03   FH.DMS.CONTROLPOINT       BOOLEAN,
02   FH.VERSION                   BIT(36),       % YEAR,JDAY,TIME
02 FH.PROTECTION                       BIT (2),%  HOST RJE
```

```
    02 FH.PROTECTION.IO                    BIT (2),%  HOST RJE
    02 FILLER                              BIT (16),% HOST RJE
    02   FH.AREA.ADDRESS (105)     DSK.ADR,
      03 FH.UNIT                   BIT (12), %
         04 FH.PORT                BIT (3), %
         04 FH.CHAN                BIT (4), %
         04 FH.SER.NO.FLAG         BOOLEAN, %
         04 FH.EU                  BIT (4), %
         03   FH.ADDR              BIT (24);
#;%
```

LABEL SIZE

```
DEFINE LBL.SIZE AS #640#;
DEFINE
    SCRTCH.TYPE AS #"0"#
,   USER.TYPE AS #"1"#%
,   BACKUP.TYPE AS #"2"# %
,   LIBRARY.TYPE AS#"3"# %
,   NOT.ANSI AS #"0"# %
,   BOV  AS #1# %
,   BOF  AS #2# %
,   EOV  AS #3# %
,   EOF AS #4# %
,   PFB  AS #5# %
,   LOST AS #7# %
;       %
DEFINE STANDARD.LABEL.DECLARATION AS # %
DECLARE 01 DUMMY REMAPS L.LABEL.RECORD  %
,   02 L.LABEL      CHAR (9)  %        " LABEL  0"
,   02 L.MFID       CHAR (7)  %                 "
,   02 L.Z1         CHAR (1)  %        "0"      "
,   02 L.ID         CHAR (7)  %
,   02 L.REEL       CHAR (3)  %
,   02 L.DW         CHAR (5)  %        DATE WRITTEN
,   02 L.CYCLE      CHAR (2)  %        "00"
,   02 L.PID        CHAR (5)  %        PURGE DATE
,   02 L.S          CHAR (1)  %        SENTINNEL (1 = END-OF-REEL)
,   02 L.BC         CHAR (5)  %        BLOCK COUNT
,   02 L.RC         CHAR (7)  %        RECORD COUNT
,   02 L.PB         CHAR (1)  %        PRINT BACKUP FLAG
```

```
,   02 L.SERIAL     CHAR (5)   %        SERIAL NUMBER
,   02 L.SYSTEM     CHAR (5)   %        CREATING SYSTEM
,   02 L.BUFSIZE    CHAR(8)    %        NEW FORMAT DECIMAL BLOCK SIZE
,     03 L.BSIZE    BIT(24)    %        OLD FORMAT BINARY
,     03 L.RSIZE    BIT(24)    %        OLD FORMAT BINARY
,   02 L.RECSIZE    CHAR(8)    %        NEW FORMAT DECIMAL RECORD SIZE
,   02 L.MODE       CHAR(1)    %        NEW FORMAT RECORDING MODE FOR
                               %        TAPE FILE
; # ;
DEFINE ANSI.TAPE.DECLARATION AS #
DECLARE
     01 DUMMY REMAPS ANSI.TAPE.LABEL CHAR(80)%
,     02 ID.AND.NUMBER  CHAR(4)   %
,       03 ID           CHAR(3)   %
,       03 NUMBER       CHAR(1)   %
,     02 FILLER         CHAR (76)  %
; #; %
DEFINE VOL.HEADER.DECLARATION AS #
DECLARE%
    01 DUMMY REMAPS ANSI.TAPE.LABEL %
,     02 FILLER         CHAR(4)  %
,     02 VOL.ID         CHAR(6)  %
,     02 ACCESSIBILITY  CHAR(1)  %
,     02 RFS            CHAR(26) % SUPPOSED TO BE RESERVED BUT WE WILL
                                 %USE IT ANYWAY
,       03 MFID         CHAR(17) %"0" IF NO MULTIPLE FILE ID
                                 %"XO" FOR 17  IF SCRATCH
                                 %"BACKUP" FOR BACKUP
,       03 SYS.SYMBOL   CHAR(2)  % "17"
,       03 TAPE.TYPE         CHAR(1)    % 0= SCRATCH
                                        % 1= USER
                                        % 2= BACKUP
                                        % 3=LIBRARY
,       03 FILLER       CHAR(6)  %
,     02 OWNER.ID       CHAR(14) %
,     02 FILLER    CHAR(28)%
,     02 VERSION        CHAR(1)  % 1 FOR THIS STANDARD
; #;
```

```
DEFINE HEADER1.DECLARATION AS# % USE FOR HDR1,EOV1,EOF1
DECLARE%
    01 DUMMY REMAPS ANSI.TAPE.LABEL
,      02  FILLER         CHAR(4)
,      02 FILE.ID         CHAR(17)
,      02 FILE.SET.ID     CHAR(6)
,      02 FILE.SECTION.NO CHAR(4)
,      02 FILE.SEQ.NO      CHAR(4)
,      02 GENERATION.NO    CHAR(4)
,      02 GENERATION.VERSION.NO CHAR(2)
,      02 CREATION.DATE    CHAR(6)
,      02 EXPIRATION.DATE CHAR(6)
,      02 ACCESSIBILITY    CHAR(1)
,      02 BLOCK.COUNT      CHAR(6) %HDR1="000000",EOV & EOF = REAL THING
,      02 SYSTEM.CODE      CHAR(13) %
,      02 FILLER           CHAR(7)  % RFS
; #;
DEFINE HEADER2.DECLARATION AS #        %HDR2,EOV2,EOF2
DECLARE                                %
     01 DUMMY REMAPS ANSI.TAPE.LABEL   %
,      02 FILLER              CHAR(4)  %
,      02 RECORD.FORMAT       CHAR(1)  % F= FIXED,D=VARIABLE,S=SPANNED
,      02 BLOCK.LENGTH        CHAR(5)  %
,      02 RECORD.LENGTH       CHAR(5)  %
,      02 RESV.SYSTEM.USE     CHAR(35) %
,        03 DENSITY           CHAR(1)  % 0=800, 1=556,2=200,3=1600
,        03 SENTINAL          CHAR(1)  %
,        03 PARITY            CHAR(1)  % 0= ALPHA(EVEN),1=BINARY(ODD)
,        03 EXT.FORM          CHAR(1)  % 0= UNSPECIFIED
                                       % 1= BINARY
                                       % 2= ASCII
                                       % 3= BCL
                                       % 4= EBCDIC
,        03 FILLER            CHAR(31) %
,      02 FILLER              CHAR(28) % RFS
; # ;
```

<u>COLD START VARIABLES</u>

```
DEFINE   CSV.SIZE        AS  #934#;
DECLARE COLD.START.VARIABLES TEMPLATE BIT (CSV.SIZE);
DEFINE CSV.ONE AS #,%
    02 CLEAR.START.FLAGS                 BIT (40),
       03 CS.WHICH                       BIT(4),
       03 CS.TRACE                       BIT (4),
       03 CS.INTERP                      BIT (4),
       03 CS.MCP                         BIT (4),
```

```
      03 CS.GISMO                        BIT (4),
      03 CS.INIT                         BIT (4),
      03 CS.EMULATE                      BIT(4),
      03 CS.MICRO.MCP                    BIT (4),
   02 NAME.TABLE                         DSK.ADR,
   02 INTERP.DIC.ENTRIES                 WORD,
   02 CS.SIZE                            WORD,
   02 DUMP.FILE                          DSK.ADR,
   02 CSV.COLD.START.LEVEL                    BIT(24),
      03 L61.NAME.TABLE                       BIT(1),
      03 FILLER        BIT(23),
   02 MPF.TABLE                          DSK.ADR,
   02 LOG.MIX.INFO                       DSK.ADR,
   02 DISK.AVAIL                         DSK.ADR,
   02 DISK.DIRECTORY                     DSK.ADR,
   02 TEMP.TABLE                         DSK.ADR,
   02 SYSTEM.DRIVES                      BIT (16),
      03 SYSTEM.DRIVE                    BOOLEAN,
   02 AVL.TAB.DISP                       BIT (64),
   02 SY.DAY                             BIT (5),
   02 SY.MONTH                           BIT (4),
   02 SY.YEAR                            BIT (7),
   02 SY.JDAY                            BIT (9),
   02 SY.TIME                            BIT (21),
      03 SY.HOUR                         BIT (5),
      03 SY.MIN                          BIT (6),
      03 SY.SEC                          BIT (6),
      03 SY.10THSEC                      BIT (4),
   02 SY.12HOUR                          BIT (5),
   02 SY.DAYNAME                         CHAR (9),
   02 SY.MERIDIAN                        CHAR (2),
   02 SYSTEM.OPTIONS                     BIT (80),
```

```
      03 LOG.OPTION                        BOOLEAN,
      03 CHARGE.OPTION                     BOOLEAN,
      03 LIB.OPTION                        BOOLEAN,
      03 OPEN.OPTION                       BOOLEAN,
      03 TERM.OPTION                       BOOLEAN,
      03 TIME.OPTION                       BOOLEAN,
      03 DATE.OPTION                       BOOLEAN,
      03 CLOSE.OPTION                      BOOLEAN,
      03 PBT.OPTION                        BOOLEAN,
      03 PBD.OPTION                        BOOLEAN,
      03 BOJ.OPTION                        BOOLEAN,
      03 EOJ.OPTION                        BOOLEAN,
      03 SCHM.OPTION                       BOOLEAN,
      03 LAB.OPTION                        BOOLEAN,
      03 RMOV.OPTION                       BOOLEAN,
      03 DUMP.OPTION                       BOOLEAN,
      03 ZIPP.OPTION                       BOOLEAN,
      03 MEM.OPTION                        BOOLEAN,%
      03 SW01.OPTION                       BOOLEAN,
      03 SW02.OPTION                       BOOLEAN,
      03 SW03.OPTION                       BOOLEAN,
      03 PWS.OPTION                        BOOLEAN,
      03 DMS.OPTION                        BOOLEAN,
      03 RFAC.OPTION                       BOOLEAN,
      03 TRMD.OPTION                       BOOLEAN,
      03 DEBUG.OPTION                      BOOLEAN,
      03 DISP.OPTION                       BOOLEAN,
      03 SPOL.OPTION                       BOOLEAN,%
      03 RMSG.OPTION                       BOOLEAN,
      03 SQRM.OPTION                       BOOLEAN,
      03 TABS.OPTION                       BOOLEAN,
      03 BREL.OPTION                       BOOLEAN,
      03 MPRI.OPTION                       BOOLEAN,
      03 THRASHING.OPTION                  BOOLEAN,
      03 FLMP.OPTION                       BOOLEAN,
      03 VLCP.OPTION                       BOOLEAN,
      03 VLIO.OPTION                       BOOLEAN,
      03 BRGR.OPTION                       BOOLEAN%
#;%
DEFINE CSV.TWO AS #,%
    02 FIRST.SCHED.ENTRY                   DSK.ADR,
    02 FIRST.WAITING.SCHED                 DSK.ADR,
    02 GISMO.TRACE.FLAGS                   WORD,
    02 SYS.PC                              BIT (28),%ARRAY OF SYS PORT,CHAN
       03 SPC                              BIT (7),
    02 MIX.LIMIT                           BIT(8)%
   ,02 SPO.Q.CLEAR.START                   BIT(60)% SPO.Q.POINTER, DOWN TO B
   ,02 SPO.FLAGS                           BIT(3)%
```

```
   ,  03  SPO.DISPLAY.TIME                 BOOLEAN%
   ,  03  CRT.SPO.TRANSL                   BOOLEAN%
   ,  03  CRT.SPO.DIRN                     BOOLEAN%
   ,02 SPOLOG.LAST.AREA                   BIT(1)% SPO.LOG.FULL
   ,02  SPO.SUPPRESS                      BIT(1)% IF ON THE DISPLAY ONLY ID
   ,02  FILLER                            BIT(1)%
   ,02  SPOLOG.LAST.SECTORS               BIT(2)% NOT USED TILL 6.1
   ,02  SPOLOG.SIZE                       BIT(16)% 14 BITS ARE ENOUGH.,
   ,02 NEXT.SPOLOG.REC                    BIT(36) %NEXT SPOLOG SEGMENT,
,   02 CSV.CORRECTABLE.ERROR.TABLE.LENGTH   BIT(16)%  40 + 32*#ENTRIES
   ,02 DUMP.FILE.SIZE                       BIT(16)
#;%,
DEFINE CSV.DECLARATION AS #%
DECLARE 01 DUMMY REMAPS COLD.START.VARIABLES
CSV.ONE
CSV.TWO
;#;%
```

DISK COLDSTART VARIABLES

```
DEFINE DCSV.SIZE AS #935#;
DEFINE DCSV.DECLARATION AS #%
DECLARE 01 DUMMY REMAPS DISK.CS.VARIABLES,
    02 MASTER.IOAT                      DSK.ADR,
    02 MASTER.DISK.AVAIL                DSK.ADR,
    02 NEXT.LOG.REC                     DSK.ADR,
    02 LG.SIZE                          WORD,
    02 NEXT.ELOG                        DSK.ADR,
    02 ELOG.SIZE                        WORD,
    02 JOB.NO                           WORD,
    02 PBD.NO                           WORD,
    02 SPO.Q.SIZE                       WORD,
    02 CTLDCK.NO                        WORD,
    02 LOG.NO                           WORD,
    02 Q.DISK                           DSK.ADR,
    02 TRACE.FPB                        DSK.ADR,
    02 AUTO.MASK                        BIT(28),% 4 OF 7 EACH=PRT PC
    02 AB.NUMBER                        BIT(3),% NUM OF SYSTEM/BACKUPS
    02 FILLER                           BIT(5),%
    02 PBD.BLCKS.AREA                   WORD,
    02 LG.LAST.AREA                     BOOLEAN,
    02 ELOG.LAST.AREA                   BOOLEAN,
    02 FILLER                           BIT (2),
    02 PBD.DESIGNATION                  CHAR(10),
    02 SPO.Q                            DSK.ADR,
    02 PROTECTED.UNITS                  BIT (256),
         %ARRAY OF 16 16 BIT ENTRIES OF UNITS TO BE PROTECTED THRU C/S
         %12 BITS FOR PCU
         % 2 BITS FOR LABEL TYPE
         %     0 = ANSII
         %     1 = UNLABELED
         %     2 = BURROUGHS
         % 1 BIT FOR TRANSLATE  0=EBCDIC  1=ASCII
         % 1 BIT TO INDICATE THAT A C/S HAS HAPPENED ON THIS UNIT
    02 SYS.LOG NUMBER                   BIT (24),
    02 JOB.ACCTING.NUMBER               BIT (24),
    02 SESSION.NR                       BIT (10),
    02 XM.TABLE.DISK.ADDRESS            DSK.ADR,
    02 DCSV.NSEC.DISABL.THRASH.FAULT    BOOLEAN,
    02 DCSV.SAMPLING.INTERVAL           BIT (6),
    02 DCSV.OVERLAY.TARGET              BIT (8)
    ;
#;%
```

```
DEFINE  MEMORY.LINK.SIZE AS #179#;
DECLARE MEMORY.LINK TEMPLATE BIT(MEMORY.LINK.SIZE);
DEFINE MEMORY.LINK.DECLARATION AS #
DECLARE 01 DUMMY REMAPS MEMORY.LINK,
           2 ML.DISK                      DSK.ADR,
           2 ML.GROUP,
             3 ML.POINTER                 ADDRESS,
             3 ML.JOB.NUMBER              BIT(16),
             3 ML.TYPE                    BIT(6),
             3 ML.SAVE                    BIT(1),
           2 ML.SIZE                      BIT(24),
           2 ML.PRIORITY.FIELD            BIT(22),
             3 ML.OK.INTERVAL             BIT(6),
             3 ML.CURRENT.OK.INT          BIT(6),
             3 ML.INCOMING.PRIORITY       BIT(5),
             3 ML.RESIDENCE.PRIORITY      BIT(5),
               4 ML.RP.WHOLE              BIT(4),
               4 ML.RP.FRACTION           BIT(1),
           2 ML.FRONT                     BIT(24),
           2 ML.BACK                      BIT(24),
           2 ML.USAGE.BITS                BIT(2),
             3 ML.PREVIOUS.SCAN.TOUCH     BIT(1),
             3 ML.CURRENT.SCAN.TOUCH      BIT(1);#;

USE( ML.DISK
    ,ML.POINTER
    ,ML.JOB.NUMBER
    ,ML.TYPE
    ,ML.SAVE
    ,ML.SIZE
    ,ML.FRONT
    ,ML.BACK
     ) OF MEMORY.LINK.DECLARATION;

DEFINE Q.ML.DECLARATION AS#DECLARE
   01 Q.MEMORY.LINK TEMPLATE
,     02 FILLER                 BIT(MEMORY.LINK.SIZE)
,     02 Q.ML.F.AVL             ADDRESS
,     02 Q.ML.B.AVL             ADDRESS
;#;

DEFINE
   TAKE.LO                      AS#0#
,  TAKE.RIGHTMOST               AS#1#
;
DEFINE         % TYPES FOR "ML.TYPE"
    CODE AS          #0#%

,   AVAILABLE AS     #2#%
,   RN.S AS          #3#%
```

```
,    MCP.TEMP AS      #4#%
,    USER.FILE AS     #5#%
,    SEG.DICTV AS     #6#%
,    MICROCODE        #7#%
,    DICT.MASTER AS #8#
,    QUEUE.DIRECTORY.TYPE
                 AS #9#
,    MSG.BUFFERV
                AS  #10#
,    MESSAGE.LIST.TYPE
                 AS #11#
,    TO.BE.FORGOTTEN AS #12#
,    DATA.SEG      AS #13#
,    DBM.BUFFER AS  #14#
     TERMINATING LINK AS #15#%
,    MCP.PERM      AS #16#%
,    PSR.MEM       AS #17#%
,    MCP.IOAT      AS #18#%
,    DISK.HEADER AS #19#%
,    PACK.MEM      AS #20#%
,    SD.CNTNR      AS #21#%
,    SCHED.MEM     AS #22#%
,    SORT.MEM      AS #23#%
,    DCH.MEM       AS #24#%
,    MICROCODE.NON.OVERLAYABLE AS #25#%
,    QUEUE.AVL.BUF.V AS #26#
,    DMS.DISK.HDR AS#27#%
,    DMS.STRUCTURE AS#28#%
,    DMS.TEMP      AS #29#%
,    DMS.GLOBALS AS #30#%
,    DMS.TEMP.LOCK.DESCR AS #31#
,    XM.MEMORY AS #32#
;%
```

SYSTEM DESCRIPTORS

```
DECLARE
    01 SYSTEM.DESCRIPTOR TEMPLATE BIT(SY.SIZE);
DEFINE SY.DECLARATION AS #SY.DECL(SYSTEM.DESCRIPTOR)#;%
DEFINE SY.DECL(X) AS #DECLARE%
    01  DUMMY REMAPS X,%
        02 SY.IN.USE          BIT(1),    % TO HELP MEMORY MANAGEMENT
        02 SY.MEDIA           BIT(1),    % 0=DISK, 1=S-MEMORY
        02 SY.LOCK            BIT(1),    %
        02 SY.IN.PROCESS      BIT(1),    % TRUE IF THERE IS AN I/O IN
                                         % PROCESS FOR THE INFORMATION
                                         % REPRESENTED BY THIS DESCRIPTOR.
                                         % IF TRUE, "SY.CORE" CONTAINS A
                                         % POINTER TO THE I/O DESCRIPTOR.
```

```
      02 SY.INITIAL           BIT(1),    % "ADDRESS" IS READ-ONLY MOTHER
                                         % COPY, HENCE IF "WRITE" THEN GET
                                         % NEW DISK AND REPLACE ADDRESS.
      02 SY.FILE              BIT(1),    % THE OBJECT OF THIS DESCRIPTOR
                                         % IS A FILE WHOSE USERCOUNT MUST
                                         % BE DECREMENTED WHEN THIS
                                         % DESCRIPTOR IS RETIRED.
      02 SY.DK.FACTOR         BIT(3)     % MEMORY DECAY FACTOR
      02 SY.SEG.PG            BIT(7),    % MEMORY.ACTIVITY AUDITING
      02 SY.TYPE              BIT(4),    % UNITS FOR SY.LENGTH.
                                         % 0 = BITS
                                         % 1 = DIGITS (4 BIT)
                                         % 2 = CHARACTERS (8 BIT)
                                         % 3 = NORMAL DESCRIPTORS
                                         % 4 = DISK SEGMENTS
                                         % 5 = SYSTEM DESCRIPTORS
                                         % 6 = SYSTEM INTRINSIC
                                         % 7 = INDIRECT REFERENCE
                                         %     ADDRESS GIVES RELATIVE
                                         %     DISPLACEMENT IN BITS
                                         %     (SIGNED NUMBER).
                                         % 8=  MICROS
      02 SY.ADDRESS           BIT(36),   %
         03 FILLER            BIT(12),   % PORT,CHANNEL AND UNIT.
         03 SY.CORE           BIT(24),   % CORE, OR ADDRESS WITHIN UNIT.
      02 SY.LENGTH            BIT(24);   % NUMBER OF UNITS, AS DETERMINED
                                         % BY SY.TYPE.
#;                                       %
    DEFINE ND.DECLARATION AS#
    DECLARE
    01 DUMMY REMAPS NORMAL.DESCRIPTOR BIT(ND.SIZE),
      02 ND.DK.FACTOR         BIT(3),
      02 FILLER               BIT(6),
      02 ND.CORE              BIT(24),
      02 ND.TYPE              BIT(3),
      02 ND.LENGTH            BIT(24);#;
```

RUN STRUCTURE STATUS TYPES

```
DEFINE MAX.REASON AS #52#;
DEFINE   %    "RS.STATUS" TYPES
     EXECUTING                    AS #0#
,    NO.FILE                      AS #1#
,    NO.USER.DISK                 AS #2#
,    DUPLICATE.LIBRARY            AS #3#
,    DUPLICATE.INPUT.FILE         AS #4#
,    POSSIBLE.DUP                 AS #5#
,    WAITING.FOR.HARDWARE         AS #6#
,    PROGRAM.STOPPED              AS #7#
,    WAITING.IO.COMPLETE          AS #8#
,    WTG.DATACOMM.MSG             AS #9#%
,    WAITING.OVERLAY              AS #10#
,    WAITING.KBD.IN               AS #11#
,    HDWR.NOT.READY               AS #12#
,    WAITING.OPERATOR.ACTION      AS #13#
,    WAITING.CLOSE                AS #14#
,    WAITING.DS.OR.DP             AS #15#
,    NO.MPF.PACK                  AS #16#
,    NO.FILE.ON.DISK              AS #17#%
,    WAITING.FOR.LOCKED.FILE      AS #18#
,    WAITING.Q.IS.FULL            AS #19#
,    WAIT.STATUS                  AS #20#
,    NOMEM.WAITING.COMM.Q         AS #21#
```

```
,    NOMEM.WAITING.READY.Q      AS #22#
,    STOPPED.CQ                 AS #23#
,    STOPPED.RQ                 AS #24#
,    WAITING.TIME.COMM.Q        AS #25#
,    WAITING.TIME.READY.Q       AS #26#
,    WAITING.RECEIVE            AS #27#
,    WTG.DATACOMM.OPN      AS #28#%
,    TERMINATING                AS #29#
,    IN.READY.Q                 AS #30#
,    IN.COMM.Q                  AS #31#
,    STOPPED.FOR.SORT           AS #32#
,    WTG.DC.DSK.CMPLT           AS #33#
,    WTG.DATACOMM.DSK      AS #34#%
,    NO.CONTROLLER              AS #35#%
,    NO.OUTPUT.PACK             AS #36#
,    VSORT.QSORT.NOT.PRESENT    AS #37#
,    NO.SORT.INPUT.FILE         AS #38#
,    WAITING.CONTENTION         AS #39#
,    WAITING.SYNCPOINT          AS #40#
,    WAITING.RECOVERY           AS #41#
,    WAITING.NEW.AUDIT          AS #42#
,    WAITING.SORTER.IO          AS #43#
,    TERMINATING.WAITING.IO     AS #44#
,    CLOSING.WAITING.IO         AS #45#
,    WAITING.FORMS              AS #46#
,    NO.TRANSLATE.FILE          AS #47#
,    MF.SEARCHING               AS #48# %
     ,NO.DMS.FILE               AS #49#
     ,NO.DMS.DICTIONARY         AS #50#
,    WTG.DMS.REORGANIZATION     AS #51#
,    WTG.INACTIVE.DATA.BASE     AS #52#
;
```

RUN STRUCTURE NUCLEUS

```
DEFINE RS.N.SIZE AS #2868#;
    DECLARE RS.NUCLEUS TEMPLATE BIT (RS.N.SIZE);%
    DEFINE RS.ONE AS #,%
           02 RS.COMMUNICATE.MSG.PTR    BIT (48)    ,%
              03 RS.ITYPE               BIT(2),
              03 RS.INMBR               BIT(6),
              03 RS.ILENGTH             BIT(16),
              03 RS.IADDRESS            BIT(24),
           02 RS.COMMUNICATE.LR         ADDRESS,
           02 RS.REINSTATE.MSG.PTR      BIT(48)     ,%
           02 RS.MY.BASE                ADDRESS     ,%
           02 RS.MY.LIMIT               ADDRESS     ,%
           02 RS.MCP.BIT                BOOLEAN,
           02 RS.NIP                    BIT (32)    ,%
           02 RS.SEG.DIC.PTR            BIT(24)     ,%
           02 RS.DATA.DIC               ADDRESS     ,%
           02 RS.INTERP.ID              BIT(5)      ,%
           02 RS.INTRINSICS.LOC         BIT(10),
           02 RS.M.MACHINE              BIT(S.PAD.SIZE),%
           02 RS.PRIORITY,
             03 RS.PRIORITY.INTEGER     BIT (4),
             03 RS.PRIORITY.FRACTION    BIT (4),
           02 RS.STATE.LIGHT            BIT (16),
             03 RS.VARIABLE.LAMP.CPU    BIT(2),
             03 RS.VARIABLE.LAMP.CODE.OVLY  BIT(2),
             03 RS.VARIABLE.LAMP.DATA.OVLY  BIT(2),
           02 RS.NUMBER.FILES           BIT (8),
           02 RS.BOOLEANS               WORD,
              03 RS.IL                  BOOLEAN,
              03 RS.UL                  BOOLEAN,
              03 RS.OF                  BOOLEAN,
              03 RS.FR                  BOOLEAN,
              03 RS.FM                  BOOLEAN,
              03 RS.OU                  BOOLEAN,
              03 RS.OK                  BOOLEAN,
              03 RS.RM                  BOOLEAN,
              03 RS.MR                  BOOLEAN,
              03 FILLER                 BIT (15),
           02 RS.TYPE                   BIT (6),
           02 RS.PAGED.DICT             BIT(1),
           02 RS.IN.TRANSACTION         BOOLEAN,  %DMS TRANSACTION STATE
           02 RS.SORTER.FLOWING         BOOLEAN,
           02 RS.DM.OPERATION           BOOLEAN,% 1=DM OP IN PROCESS
           02 RS.DISABLE.INTERRUPTS     BIT(6),
```

```
        02 RS.FUTURE                  BIT (9),
      02 RS.EXTERNAL.INTERRUPT.MASK BIT (9),%
           03 RS.EXTERNAL.INTERRUPT.BIT BIT (1),
           03 RS.DC.IO.COMPLETE       BIT (1)%
#;%
    DEFINE RS.TWO AS #,%
        02 RS.Q.IDENT                 ADDRESS,
        02 RS.LAST.OVLY               ADDRESS,
        02 RS.DATA.OVERLAYS           ADDRESS,
        02 RS.LAST.LINK               ADDRESS,
        02 RS.JOB.NUMBER.IN.DECIMAL BIT(16),
        02 FILLER                     BIT(8),
        02 RS.SER.NO                  WORD,
        02 RS.EMULATOR.BITS           BIT(4), % USED BY INTERP
        02 RS.OVLY.DISK.PTR           BIT(24),
        02 RS.JOB.ACCTING.NO          BIT(24),
        02 RS.FIB.DIC                 BIT(24),
        02 RS.TRACE                   ADDRESS, % TRACE INFO ADDRESS
        02 RS.TRACE.FIB               BIT(24),
        02 RS.TRACE.BITS              BIT(8),
        02 RS.MEDIA                   BIT(1),
        02 RS.LENGTH                  BIT(24),
        02 RS.Q.LINK                  ADDRESS,
        02 RS.STATUS                  BIT(24),
        02 RS.JOB.NUMBER              BIT(16),
        02 FILLER                     BIT(8),
        02 RS.CODE.OVLY.COUNT         BIT(20),
        02 RS.TIME                    BIT(20),
        02 RS.PROG.PTR                DSK.ADR,
        02 RS.PAUSE                   WORD,
        02 RS.DONT.REENTER            BIT(1),
        02 RS.SD.PTR.FLAG             BIT(1),
        02 RS.LINKS                   BIT(1),
        02 RS.EMULATOR.TAPE           BIT (1),%
        02 RS.DISK                    DSK.ADR,
        02 RS.ROLLOUT.COMPLETE        BIT(1),
        02 RS.ROLLOUT.IN.PROCESS      BIT(1),
        02 FILLER                     BIT(8),
        02 RS.DICT.MEDIA              BIT(1),
        02 RS.APPARITION              BIT(1),
```

```
          02 RS.SIZECHANGE              BIT(1),
          02 RS.TO.BE.STOPPED           BIT (1),
          02 RS.NOT.A.ROLLOUT.CANDIDATE BIT(1),
          02 RS.TO.BE.ROLLED.OUT        BIT(1),
          02 RS.STOPPED                 BIT(1),
          02 RS.USE.FLAG                BIT(1),
          02 RS.PSEUDO.READER           ADDRESS,
          02 FILLER                     BIT(24), % OVLY ACTUAL END
          02 RS.OVLY.DESC               BIT(DESCRIPTOR.SIZE),
          02 RS.MCP.USE                 BOOLEAN,
          02 RS.FILE                    BIT (8),
          02 RS.ROLLIN.IN.PROCESS       BIT(1),
#;
    DEFINE RS.THREE AS #,
          02 RS.TEMPORARY.FREEZE        BIT(1),
          02 RS.BOJ.TO.EOJ.FREEZE       BIT(1),
           02 RS.DATA.COMM              BIT(1),%
          02 RS.UNIT.INDEX              ADDRESS,
          02 RS.OVLY.DISK.BASE          DSK.ADR,
          02 RS.LOG.PTR                 DSK.ADR
,       02 RS.PATH.DIC                  ADDRESS   % DATA MANAGEMENT
,       02 RS.NMBR.PATHS                BIT(8)
,         02 RS.PROTECTED               BIT (1)
          ,02 RS.PRIVILIGED             BOOLEAN,%
           02 RS.ABORT                  BIT (2)%
,         02 RS.SWITCHES                BIT(40)
,       2 RS.WAIT.LOC                   ADDRESS,
        2 RS.WAIT.LEN                   BIT(12),
        2 RS.NEXT.Q                     BIT(24),
        2 RS.M.PROBLEM                  BIT(48),
            5 RS.M.PROBLEM.TYPE         FIXED,
            5 RS.M.PROBLEM.PARAMETER    BIT(24),
        2 RS.REPORT.EV.INX              BOOLEAN,
        2 RS.INTERVENTION               BOOLEAN,
        2 RS.DUMMY.EV                   BIT(1),
        2 FILLER                        BIT (1),
       2 RS.SPO.Q.KEY                   ADDRESS,
        2 RS.EVENT.SPACE                BIT (24*15),
        2 RS.MAX.TIME                   BIT (24)
,         02 RS.PARENT.QUEUE            BIT (24),% HOST RJE
          02 RS.LOG.SPO                 BIT (1),%  HOST RJE
          02 RS.USERCODE                BIT (10),% HOST RJE
          02 RS.SESSION                 BIT (16),% HOST RJE
          02 RS.PARENT.JOB.NR           BIT (16) % HOST RJE
          02 FILLER                     BIT (8),
         ,02 RS.DATA.OVLY.COUNT         BIT (20)
,       02 RS.NMBR.INVOKES              BIT(6)
         ,02 RS.MEMORY.PRIORITY         BIT(5)
```

```
            03 FILLER                   BIT(4),
            03 RS.HALF.PRIORITY         BIT(1),
          02 RS.SWEEPS.BEFORE.DECAY     BIT(10),
          02 RS.FORCED.SUSPENSION       BIT(1),
          02 FILLER                     BIT(8),
       ,02 RS.DISPLACED                 FIXED,
        02 RS.PREVENT.MOVE              BIT(1)%
,         02 RS.OVLY.DISK.SIZE          BIT(24)%
        02 RS.PRIOR.JOB.NO              BIT(16)
        02 RS.DMS.GLOBAL                BIT(24)
#;%
```

```
DEFINE RS.N.DECLARATION AS #%
DECLARE 01 DUMMY REMAPS RS.NUCLEUS
RS.ONE
RS.TWO
RS.THREE
;#;%
DEFINE IO.DESCRIPTOR.MAP(IO.DESCRIPTOR) AS #
        DECLARE 01 DUMMY                            REMAPS IO.DESCRIPTOR
              ,    02  IO.RESULT                    WORD
              ,      03  IO.RESULT.BIT.1.2          BIT (2)
              ,    02  IO.LINK                      ADDRESS
              ,    02  IO.OP                        WORD
              ,    02  IO.BEGIN                     ADDRESS
              ,    02  IO.END                       ADDRESS
              ,    02  IO.DISK.ADDRESS              ADDRESS
              ,     2 IO.M.EVENTS         BIT(8)
              ,     2 IO.MCP.IO           BIT(16)
              ,    02  IO.FIB                       ADDRESS
              ,    02  IO.FIB.LINK                  ADDRESS
              ,    02  IO.BACK.LINK                 ADDRESS
              ,    02  IO.PORT.CHAN                 BIT (7)
              ,      03  IO.PORT                    BIT (3)
              ,      03  IO.CHANNEL                 BIT (4)
             #;
DEFINE IO.DESC.DECLARATION AS #IO.DESCRIPTOR.MAP(IO.DESC)#;
```

PROGRAM PARAMETER BLOCK

```
DEFINE PPB.LEVEL AS #-4#;
DEFINE PPB.SIZE AS #3870#;
DEFINE WORKING.PPB.SIZE AS #2880#;
DEFINE PPB.DECLARATION AS#
DECLARE 01 DUMMY REMAPS PPB
PPB.MASTER
PPB.SPAD
PPB.SCHED
;#;
DEFINE MASTER.PPB.DECLARATION AS#
DECLARE 01 DUMMY REMAPS MASTER.PPB
PPB.MASTER
;#;
DEFINE SCHED.DECLARATION AS#
DECLARE 01 DUMMY REMAPS SCHED
PPB.SCHED
;#;
DEFINE WORKING.PPB.DECLARATION AS #
DECLARE 01 DUMMY REMAPS WORKING.PPB
PPB.MASTER
PPB.SPAD
;#;%
DEFINE MASTER.PPB.SIZE AS #1440#;
DEFINE PPB.MASTER AS#,
  02 PROG.NAME                          CHAR(30), %
     03 PROG.CURRENT.DIRECTORY          NAME,     %
                   % DIRECTORY IN WHICH PROGRAM IS LISTED.
     03 PROG.NAME.FIRST                 NAME,     %
                   % PROGRAMS FIRST NAME
```

```
    03 PROG.NAME.SECOND               NAME,     %
                 % PROGRAMS SECOND NAME
                 %
                 % FOR COMPILATIONS - IN THE LOG COPY OF THE PPB
                 % PROG.NAME.FIRST = COMPILERS FIRST NAME
                 % PROG.NAME.SECOND = OBJECT PROGRAMS FIRST NAME
                 %
 02 PROG.INTRINSIC                   CHAR (20),
    03 PROG.INTRINSIC.DIRECTORY       NAME,
                 % PACK ID FOR INTRINSICS
    03 PROG.INTRINSIC.NAME            NAME,
                 % FAMILY NAME FOR INTRINSICS
 02 PROG.INTERP.NAME                  CHAR(30), %
    03 PROG.INTERP.DIRECTORY          NAME,     %
                 % PACK ID FOR INTERPRETER
    03 PROG.INTERP.NAME.FIRST         NAME,     %
                 % FAMILY NAME FOR INTERPRETER
    03 PROG.INTERP.NAME.SECOND        NAME,     %
                 % OFFSPRING NAME FOR INTERPRETER
 02 PROG.PRIORITY                     BIT (4),  %
                 % PRIORITY IN THE MIX - COMPILER DEFAULT = 4
 02 PROG.BEGINNING                    BIT (32), %
                 % FIRST INSTRUCTION POINTER
 02 PROG.STATIC.CORE                  WORD,     %
                 % LENGTH IN BITS OF MEMORY TO BE ALLOCATED
                 % IMMEDIATELY AFTER THE BASE REGISTER
                 % IF THERE EXISTS A DATA DICTIONARY, THEN THIS FIELD
                 % MUST BE DESCRIBED BY ITS FIRST ENTRY. IF A DISK
                 % ADDRESS IS PRESENT THERE THEN THE SPACE WILL NOT
                 % ONLY BE ALLOCATED BUT ALSO FILLED FROM THAT
                 % ADDRESS.
 02 PROG.DYNAMIC.CORE                 WORD,     %
                 % DATA OVERLAY AREA IN BITS.
 02 PROG.TOTAL.CORE                   WORD,     %
                 % SMALLEST AMOUNT OF MEMORY REQUIRED TO RUN - IN
                 % BITS.
                 % SHOULD BE EQUAL TO -
                 % PROG.STATIC.CORE + PROG.DYNAMIC.CORE +
                 % DATA DICTIONARY SIZE + FIB DICTIONARY SIZE
 02 PROG.WORKING.SET                  WORD,     %
                 % AMOUNT OF MEMORY (IN BITS) NEEDED TO
                 % RUN THE PROGRAM EFFICIENTLY
 02 PROG.DATA.DIC                     BIT (ND.SIZE),%
                 % DATA DICTIONARY
 02 PROG.SEG.DIC                      BIT (ND.SIZE),%
                 % SEGMENT DICTIONARY
```

```
  02 PROG.FPB.ADDRESS                         ADDRESS,  %
                    % RELATIVE DISK ADDRESS OF THE FIRST FPB IN THE
                    % PROGRAM FILE (RELATIVE TO THE PPB). ALL FPBS IN
                    % THE PROGRAM MUST BE CONTIGUOUS.
  02 PROG.FILES                               BIT (8),  %
                    % TOTAL NUMBER OF FILES - 255 MAX
  02 PROG.VERSION.NO                          WORD,     %
               % REVISION LEVEL OF PPB AND FPB-S
  02 PROG.OVLY.SEG                            BIT (10), %
                    % RESERVED FOR THE SDL OVERLAY HANDLER
  02 PROG.FREEZER                             BIT(1),   %
                    %REQUESTS THAT PROGRAM NOT BE RELOCATED.
  02 PROG.LINKS                               BIT(1),   %
                    % TELLS MCP WHETHER OR NOT MEMORY LINKS ARE
                    % DESIRED IN THE DYNAMIC MEMORY AREA
  02 PROG.TRACE                               BIT (8),  %
                    % TRACE FLAGS TO ENABLE TRACING FROM THE FIRST
                    % EXECUTABLE INSTRUCTION.
  02 PROG.SCHED.PRIORITY                      BIT (4),  %
                    % PRIORITY FOR SCHEDULING
  02 PROG.VIRTUAL.DISK                        WORD,     %
                    % NUMBER OF DISK SEGMENTS DESIRED FOR DATA OVERLAY
                    % IF = 0 AND DATA OVERLAYS ARE REQUIRED MCP WILL
                    % USE 1000.
                    % REQUESTS THAT PROGRAM NOT BE RELOCATED
  02 PROG.IPB                                 ADDRESS,  %
                    % IF THIS IS AN INTERPRETER, THEN THIS IS THE FILE
                    % RELATIVE LOCATION OF THE INTERPRETER PARAMETERS.
  02 PROG.DYNAMIC.SPACES                      BIT (8),  %
               % MAX NUMBER OF SPACES TO BE ALLOCATED IN
               % PROG.DYNAMIC.CORE.
               % USED ONLY IF PROG.LINKS = 1.
  02 PROG.M.MACHINES.LOC                      WORD,     %
               % FOR CHANGE STACK SIZE COMMUNICATE.
  02 PROG.NUM.M.MACHINES                      BIT (8),  %
               % ONE FOR EACH CSS COMMUNICATE.
  02 PROG.SWITCHES                            BIT(40), %
               % FOR RUN-TIME SWITCHES
  02 PROG.PERMANENT.FLAGS,
     03 PROG.DMS                              BOOLEAN,  %
                    % THIS PROGRAM USES DMS.
     03 PROG.INTERPRETER.CHECK.OVERRIDE BOOLEAN, %
                    % USE THE ONE I SAID.
     03 PROG.INTR.AGGR                          BIT(1),   %
                    % THIS PROG CALLS FOR INTRINSIC.AGGREGATE.
     03 PROG.PROTECTED                        BOOLEAN,%
                    % STOPS DS GT SW ST
```

```
  02 PROG.COMPILER.LEVEL                     BIT(8),    %
                    % NEW LEVEL MEANS RECOMPILE RQD.
  02 PROG.PROG.PTR                           DSK.ADR,   %
                    % ABSOLUTE ADDRESS OF THE PPB ON DISK
  02 PROG.EXECUTE.TYPE                       BIT (4),   %
                    % 1 = EXECUTE
                    % 2 = COMPILE AND GO
                    % 3 = COMPILE FOR SYNTAX
                    % 4 = COMPILE TO LIBRARY
                    % 5 = COMPILE AND SAVE
                    % 6 = GO PART OF COMPILE AND GO
                    % 7 = GO PART OF COMPILE AND SAVE
  02 PROG.EOJ.TYPE                           BIT (4),   %
                    % 0 = NORMAL EOJ
                    % 1 = DS OR DP
                    % 2 = ERROR CONDITION IN PROGRAM
                    % 3 = ABORTED
                    % 4 = RS-ED
  02 PROG.GENERATOR.NAME                     CHAR(30),  %
     03 PROG.GENERATOR.DIRECTORY             NAME,      %
                    % PACK ID FOR COMPILER
     03 PROG.GENERATOR.NAME.FIRST            NAME,      %
                    % FAMILY NAME OF COMPILER
     03 PROG.GENERATOR.NAME.SECOND           NAME,      %
                    % OFFSPRING NAME OF COMPILER
  02 PROG.DATE.COMPILED                      BIT(36)    %
                    % COMPILATION DATE -
                    %  YEAR MONTH DAY HOUR MINUTE SECOND
#;%
DEFINE PPB.SPAD AS #,%
  02 PROG.SPAD                               BIT (S.PAD.SIZE),%
            % M-MACHINE
  02 PROG.CHARGE.NUMBER                      WORD,      %
            % CHARGE NUMBER
  02 PROG.PATH.DIC                           ADDRESS,   %
            % DATA MANAGEMENT PATH DICTIONARY ADDRESS
  02 PROG.PATH.SIZE                          BIT (20),  %
            % LENGTH OF PATH DICT IN BITS
  02 PROG.NMBR.PATHS                         BIT (8), %
            % TOTAL NUMBER OF PATHS - MAX=255
  02 PROG.NMBR.INVCKES                       BIT (4),   %
            % TOTAL NUMBER OF INVOKES - MAX=15
  02 PROG.COMPILER.ATTRIBUTES                BIT(80),   %
                    % INTERP MUST HAVE THEM ALL.
  02 PROG.BOOLEANS                           BIT(4),    %
                    % JUST LIKE IT SAYS
     03 PROG.SORT                            BOOLEAN,   %
                    % 1 IF SORT PROGRAM
```

```
  02 PROG.SORT.DATA                          DSK.ADR,   %
                 % SORT PARAMETER ADDRESS
   2 PROG.RS.EV.LIST.SIZE                      BIT(4),
  02 PROG.MAX.TIME                           WORD,
                 % MAXIMUM PROCESSOR TIME ALLOWED IN MINUTES
  02 PROG.6.0.NMBR.INVOKES                   BIT(6),
            % EXPANDED MAXIMUM NUMBER OF DMS INVOKES - MAX = 63
  02 PROG.NAME.TABLE                           BIT(ND.SIZE),%
                 % USED FOR INTRINSIC AGGREGATES FOR NAMES CHECKING.
  02 PROG.LAYOUT.TABLE.ADDRESS               BIT(16),%
            % CODEFILE-RELATIVE SEGMENT ADDRESS OF SDL LAYOUT TABLE
  02 PROG.LAYOUT.TABLE.SIZE                  BIT(12),%
            % SDL LAYOUT TABLE SIZE IN SEGMENTS
    02  PROG.INTRIN.AGGR.USED                  BIT(2),%
  02 PROG.MEMORY.PRIORITY                    BIT(4),
            % PRIORITY FOR CODE SEGMENTS - SYSTEM DEFAULT = 4
  02 PROG.SECONDS.BEFORE.DECAY               BIT(10),
            % NUMBER OF FRACTIONAL SECOND INTERVALS AFTER A SEGMENT
            % IS LAST ACCESSED BEFORE THAT SEGMENTS MEMORY PRIORITY
            % IS LOWERED.
  02 FILLER                                  BIT(330)
            % RESERVED FOR MORE COMPILER GENERATED DATA
#;%
DEFINE SCHED.SIZE AS #990#;%
DEFINE PPB.SCHED AS #,%
  02 PROG.SCHED.LINK                         DSK.ADR,   %
                 % SCHEDULE IS A LINKED LIST
  02 PROG.SCHED.PR.COPY                      BIT (4),   %
                 % SCHEDULE PRIORITY
  02 PROG.SCHED.SIZE                         WORD,      %
                 % TOTAL MEMORY REQUIRED FOR THIS JOB
                 % PROG.TOTAL.CORE
  02 PROG.JOB.NUMBER                         BIT(16), %
                 % JOB NUMBER - LIVES FROM SCHEDULE TO EOJ
  02 FILLER                                  BIT(8), %
  02 PROG.FLAGS,                                        %
     03 PROG.UNCONDITIONAL                   BIT(1),    %
                 % IF TRUE THEN RUN JOB EVEN IF PREDECESSOR ABORTS.
     03 PROG.DONT.REENTER                    BIT(1),    %
                 % MY SEGMENT DICTIONARY IS NON.STANDARD.
     03 PROG.NODIF                           BIT (1),%
                 % IF SPAWNED, CONTINUE IF SPAWNER GOES AWAY
     03 PROG.WAIT.OPERATOR                   BOOLEAN,   %
                 % WAITING FOR OPERATOR TO FS OR RS THIS JOB
  02 PROG.EX.AFTER.NAME                      CHAR(30), %
                 % THE FOLLOWING NAMES ARE USED WHEN AN EX AFTER
                 % IS ENTERED AFTER THE FIRST PROGRAM IS ALREADY
                 % IN THE MIX
     03 PROG.EX.AFTER.DIRECTORY              NAME,      %
                 % PACK ID FOR THE EX AFTER
```

```
     03 PROG.EX.AFTER.NAME.FIRST     NAME,     %
              % FAMILY NAME OF THE EX AFTER
     03 PROG.EX.AFTER.NAME.SECOND    NAME,     %
              % OFFSPRING NAME OF THE EX AFTER
  02 PROG.REMOTE.JOB.STN             BIT(8),   %
              % STATION NO. OF RJE JOB.
  02 PROG.REMOTE.JOB                 BOOLEAN,  %
              % FLAGS RJE (REMOTE) JOB
   02 PROG.SORT.JOB.NO               BIT(24), %
              % JOB NO OF SORT INTRINSIC
  02 PROG.SPAWNER                    BIT (1),%
              % THIS PROGRAM SPAWNED ANOTHER ONE.
  02 PROG.SEG.DECAY.SET              BIT(1),
              % THIS PROGRAM HAS HAD ITS NO.DK.FACTORS SET.
  02 FILLER                          BIT(1),
              % 1 BIT AVAILABLE
  02 PROG.JOB.ACCTING.NO             BIT(24), %
              % THIS PROGRAMS UNIQUE ACCOUTING NUMBER
  02 FILLER                          BIT(8),
  02 PROG.PRIOR.JOB.NO               BIT(16), %
              % PROGRAM CALLERS JOB NUMBER
  02 FILLER                          BIT(8), %
  02 PROG.SCHED.DATE                 BIT(36),  %
              % YEAR MONTH DAY HOUR MINUTE SECOND.
  02 PROG.BOJ.DATE                   BIT(36),  %
              % YEAR MONTH DAY HOUR MINUTE SECOND.
  02 PROG.EOJ.DATE                   BIT(36),  %
              % YEAR MONTH DAY HOUR MINUTE SECOND.
  02 PROG.PROCESS.TIME               BIT(24),  %
              % TOTAL PROCESSOR TIME
  02 PROG.CODE.OVLY.COUNT            BIT (24), %
         % NUMBER OF CODE (USER, INTERP + MICRO.MCP) OVERLAYS FOR
         % THIS PROGRAM. USED BY THE LOG.
  02 PROG.OBJ.NAME                   CHAR(30), %
     03 PROG.OBJ.DIRECTORY           NAME,     %
              % PACK ID FOR OBJECT CODE           COMPILES ONLY
     03 PROG.OBJ.NAME.FIRST          NAME,     %
              % FAMILY NAME FOR OBJECT CODE       COMPILES ONLY
     03 PROG.OBJ.NAME.SECOND         NAME,     %
              % OFFSPRING NAME FOR OBJECT CODE    COMPILES ONLY
  02 PROG.PSEUDO.READER              ADDRESS,
              % PSEUDO.READER TABLE ADDRESS
  02 PROG.PORT.CHAN                  BIT (7)
         % PORT AND CHANNEL OF THE DISK ON WHICH
         % THE MOTHER COPY RESIDES
, 02 PROG.ME.FACTOR          FIXED %
, 02 PROG.PARENT.JOB.NR              BIT(16) %
              % JOB NR OF WHO ZIPPED THIS ONE
, 02 FILLER                          BIT(8) %
,   02 PROG.PARENT.QUEUE BIT (24)%
```

```
          % PRESENCE INDICATES JOB SPAWNED. SPECIAL BOJ, EOJ AND BACKUP
          % FILE PRESENT MESSAGES WILL BE INSERTED INTO THIS QUEUE.
          % GENERAL SPO MESSAGES WILL BE PUT INTO THIS QUEUE IF
          % PROG.LOG.SPO IS ON.
,   02 PROG.LOG.SPO BIT (1)%
,   02 PROG.USER.CODE BIT (10)%
          % THIS WILL BE A RECORD POINTER, RELATIVE TO ZERO, INTO A DISK
          % FILE LABELED SYSTEM/USERCODES.
,   02 PROG.SESSION BIT (16)%
          % UNIQUE NUMBER ASSIGNED BY RJE. IT IS TO BE INSERTED INTO ANY
          % MESSAGES PASSED BACK THRU THE PARENT QUEUE.
    , 02 PROG.PRIVILIGED                 BOOLEAN%
               % PROGRAM RUNNING WITH A PRIVILIGED USERCODE
    , 02 PROG.MFID.CHGD                    BIT (2)%
    , 02 PROG.PKID.CHGD                    BIT (1)%
    , 02 PROG.OBJ.MFID.CHGD                BIT (2)%
    , 02 PROG.OBJ.PKID.CHGD                BIT (1)%
            % SET IF NAME CHANGED RY RJE. USED IN IDENTIFY MIX AND ECT
    , 02 FILLER                            BIT (9)%
, 02 PROG.DATA.OVLY.COUNT                BIT (24)  %
            % NUMBER OF DATA OVERLAYS DONE BY THIS PROGRAM
            % USED BY THE LOG
#;
```

FPB DECLARATIONS

```
DEFINE FPB.SIZE AS                     #1263#;%
DEFINE FPB.DECLARATION AS #%
DECLARE 01 DUMMY REMAPS FPB,
  02 FPB.FILE.NAME                     NAME,
            % INTERNAL FILE NAME
  02 FPB.NAMES                         CHAR (30),
     03 FPB.PACK.ID                    NAME,
            % PACK NAME
     03 FPB.MULTI.FILE.ID              NAME,
            % FAMILY NAME
     03 FPB.FILE.ID                    NAME,
            % OFFSPRING NAME
         10 FILLER                     CHARACTER(5),      % DMS
         10 FPB.AUDITFILE.NUMBER       CHARACTER(5),      % DMS
  02 FPB.HDWR                          BIT(6),
            % HARDWARE TYPE
  02 FPB.MODE                          BIT (4),
            % RECORDING MODE
     03 FPB.EVEN.PARITY                BOOLEAN,
            % 1 = EVEN
            % 0 = ODD
     03 FPB.CODE.TYPE                  BIT (3),
            % 000 = EBCDIC
```

```
            % 001 = ASCII
            % 010 = BCL
            % 011 = BINARY
 02 FPB.BUFFERS                        BIT (24),
            %NUMBER OF BUFFERS REQUESTED-MAX=16,777,215
 02 FPB.BACKUP                         BIT (2),
            % 00 = INVALID CASE
            % 01 = TAPE ONLY
            % 10 = DISK ONLY
            % 11 = EITHER TAPE OR DISK
 02 FPB.BACKUP.OK                      BOOLEAN,
            %SEND TO BACKUP IF NECESSRY
 02 FPB.HDWR.OK                        BOOLEAN,
            % SEND TO HARDWARE IF POSSIBLE
 02 FPB.BOOLEANS                       BIT (24),
   03 FPB.FORMS                        BOOLEAN,
            % OUTPUT FILE REQUIRES SPECIAL FORMS
   03 FPB.OPTIONAL                     BOOLEAN,
            % THIS IS AN OPTIONAL FILE
   03 FPB.VARIABLE                     BOOLEAN,
            % THIS FILE CONTAINS VARIABLE LENGTH RECORDS
   03 FPB.LOCK                         BOOLEAN,
            % LOCK THIS FILE AT TERMINATE TIME
   03 FPB.COBOL                        BOOLEAN,
            % COBOL FILE - IMPLIED OPEN NOT ALLOWED
   03 FPB.EOP                          BOOLEAN,
            % INDICATES PRESENCE OF END-OF-PAGE ACTION LABELS
   03 FPB.DEFAULT                      BOOLEAN,
            % BLOCK.SIZE RECORD.SIZE ETC. TO BE FILLED IN BY THE MCP
            % FOR DISK FILES ONLY
   03 FPB.PSEUDO                       BOOLEAN,
            %FLAGS CTLDCK AND PSEUDO READER FILES
   03 FPB.RMT.KEY                      BOOLEAN,
            % KEY FIELD HAS BEEN ASSIGNED FOR NDL
   03 FPB.NO.LABEL                     BOOLEAN,
             % IF SET USE FPB.UNIT.NAME AND IGNORE LABEL
    03 FPB.WORK.FILE                   BOOLEAN, %
            % FORCE JOB NO. INTO THE MIDDLE OF THE MFID
   03 FPB.QUEUE.FILE                   BOOLEAN,
            % IF 0 THEN SINGLE QUEUE IF 1 THEN MULTIPLE QUEUES
   03 FPB.DMS.FLAG                     BOOLEAN,
            % DMS AUDIT FILE
  03 FPB.LDDMP                         BOOLEAN,%
            %THIS WILL BE A LIBRARY TAPE
   03 FPB.EMULATOR.TAPE                BOOLEAN,
            % NO MCP TAPE SERVICES PROVIDED
   03 FPB.HEADER                       BOOLEAN,%
            % USERS REMOTE READS AND WRITES WILL INCLUDE THE HEADER
   03 FPB.TRANSLATE                    BOOLEAN,
            % TRANSLATE ALL RECORDS USING FPB.TRANSLATE.NAME FILE
```

```
      03 FPB.USER.BACKUP.NAME               BOOLEAN,
               % USE USER PRINT FILE NAME FOR NAME OF PRINTER BACKUP FILE
               % RATHER THAN MCP GENERATED NAME.
      03 FPB.ATTRIBUTE.ERROR                BIT(1),
               % INDICATES WHETHER ERROR OCCURRED ON LAST
               % FILE ATTRIBUTE REQUEST
   02 FPB.RECORD.SIZE                       WORD,
               % MAX LOGICAL RECORD SIZE IN BITS
   02 FPB.RECORDS.PER.BLOCK                 WORD,
               % NUMBER OF LOGICAL RECORDS PER PHYSICAL RECORD
   02 FPB.MAX.BLOCK.SIZE                    WORD,
               % MAX PHYSICAL RECORD SIZE IN BITS FOR VARIABLE
               % LENGTH RECORDS ONLY. NOT CONSULTED IF
               % FPB.VARIABLE = 0
   02 FPB.ADVERB                            BIT (12),
      03 FPB.INPUT                          BOOLEAN,
               %
      03 FPB.OUTPUT                         BOOLEAN,
               %
      03 FPB.NEW                            BOOLEAN,
               %
      03 FPB.WITH.PUNCH                     BOOLEAN,
               %
      03 FPB.WITH.PRINT                     BOOLEAN,
               %
      03 FPB.NO.REWIND                      BOOLEAN,
               % DOUBLES WITH "WITH.INTERPRET".
      03 FPB.REVERSE                        BOOLEAN,
               % DOUBLES WITH "WITH STACKERS".
      03 FPB.OPEN.LOCK                      BOOLEAN,
               %
      03 FPB.OPEN.LOCKOUT                   BOOLEAN,
               %
      03 FPB.REPORT.NO.FILE                 BOOLEAN,
               %
      03 FPB.REPORT.LOCKED.FILE             BOOLEAN,
               %
      03 FPB.OPEN.CODE                      BOOLEAN,

               % ADVERB USED FOR IMPLIED OPENS
   02 FPB.LABEL                             BIT (48),
               % DESCRIPTOR POINTING TO LABEL IN MEMORY
      03 FPB.LABEL.LENGTH                   WORD,
               % SIZE OF LABEL IN BITS
      03 FPB.LABEL.ADDRESS                  ADDRESS,
               % ADDRESS OF LABEL
   02 FPB.LABEL.USE                         BIT (32),
               % SEGMENT AND DISPLACEMENT OF LABEL USE ROUTINE
   02 FPB.LABEL.TYPE                        BIT (4),
               % 0 = DEFAULT ANSI LABEL
```

```
              % 1 = UNLABELED
              % 2 = 7 CHAR BURROUGHS STANDARD
  02 FPB.SAVE                              WORD,
              % SAVE FACTOR FOR MAG TAPE AND DISK
  02 FPB.REEL                              WORD,
              % REEL NUMBER
  02 FPB.EXCEPTION.MASK                   WORD,%
  02 FPB.UNIT.NAME                         CHAR (6),
              % MNEMONIC OF DESIRED UNIT
  02 FPB.NUMBER.STATIONS                   BIT (8),
              % NUMBER OF REMOTE STATIONS ASSIGNED TO THIS FILE
  02 FPB.USE.ROUTINE                       BIT (32),
              % SEGMENT AND DISPLACEMENT OF THE FIRST INSTRUCTION
              % IN THE USE ROUTINE
    10 FPB.DMS.AUDIT.SERIAL.NUMBER         BIT(32),               % DMS
              % SERIAL NUMBER OF FIRST AUDIT IN CURRENT AUDIT FILE
  02 FPB.USE.AREA                          BIT (48),
              % A 24 BIT LENGTH AND 24 BIT ADDRESS
              % OF THE USE ROUTINE WORK AREA
  02 FPB.SR.STATION                        BIT (4),
              % READ STATION FOR READER/SORTER
  02 FPB.ACCESS                            BIT (4),
              % 0 = SERIAL
              % 1 = RANDOM
              % 6 = DELAYED.RANDOM
  02 FPB.AREAS                             WORD,
              % MAX AREAS DESIRED
              % LIMIT = 40 NOW LATER IT WILL BE EXPANDED TO 740
  02 FPB.BLCKS.AREA                        WORD,
              % NUMBER OF PHYSICAL RECORDS PER AREA
  02 FPB.EU.DRIVE                          BIT (4),
              % SPECIAL EU OR DRIVE.NO USED WITH
              % FPB.INC.EU AND FPB.SPECIAL.EU
  02 FPB.ALL.AT.OPEN                       BOOLEAN,
              % ALLOCATE ALL AREAS AT OPEN TIME
  02 FPB.CYL.BOUNDARY                      BOOLEAN,
              % ALLOCATE AREAS ON A CYLINDER BOUNDARY
  02 FPB.MULTI.PACK.FILE                   BOOLEAN,
              % FILE CAN GO ON MULTI-PACK
  02 FPB.SPECIAL.EU                        BOOLEAN,
              % FILE MUST GO ON EU OR DRIVE SPECIFIED BY FPB.EU.DRIVE
  02 FPB.INC.EU                            BOOLEAN,
              % INCREMENT EU OR DRIVE FOR EACH AREA
  02 FPB.RESTORE.IMAGE                     BIT (1),%
              % RESTORE FPB IMAGE ON CLOSE RELEASE LOCK REMOVE ECT
  02 FILLER                                BIT (2)%
  FPB.CONTINUE.1#,%
FPB.CONTINUE.1 AS #,%
  02 FPB.AUTOPRINT                         BIT (1),
              % 0=AUTOPRINTABLE
```

```
  02 FPB.SYSTEM.BACKUP.BIT                  BIT (1),
            % TO IDENTIFY AUTO BACKUP PROGRAM, SYSTEM/BACKUP
  02 FPB.REPETITIONS                        BIT(6),
            % NUMBER OF COPIES OF BACKUP FILE
  02 FPB.OPEN                               WORD,
            %LAST TIME THIS FILE WAS OPENED
  02 FPB.1ST.OPEN                           WORD,
            %1ST TIME FILE WAS OPENED
  02 FPB.RECORD.COUNT                       WORD,
            %NUMBER OF RECORDS ACCESSED SINCE LAST OPEN
  02 FPB.BLOCK.COUNT                        WORD,
            %NUMBER OF BLOCKS ACCESSED SINCE LAST OPEN
  02 FPB.NO.OPEN.AND.CLOSE                  BIT (16),
            %NUMBER OF OPENS AND CLOSES
  02 FPB.CUMULATIVE                         WORD,
            %TOTAL OPEN TIME
  02 FPB.ERRORS                             WORD,
            %NUMBER OF IRRECOVERABLE ERRORS
  02 FPB.MCPDATA                            DSK.ADR,
  02 FPB.MCPINTERNAL                        BOOLEAN,
            %
  02 FPB.BACKUP.ALREADY                     BOOLEAN,
            %THIS IS A BACKUP FILE - DO NOT PUT TO BACKUP AGAIN
  02 FPB.NEW.FORMAT                         WORD,
            %THIS WILL = @FFFFFF@ FOR THE NEW FORMAT
            %
  02 FPB.FILE.TYPE                          BIT (4),
            % FILE TYPE TO BE USED AT CLOSE ON DISK FILES.
            %     0 = DATA
            %     3 = PSEUDO CARD
            %     7 = INTERPRETER
            %     8 = CODE
            %     9 = DATA
            %    12 = INTRINSIC
            % FILE TYPES 7,8 OR 12 CAUSE CLOSE TO MODIFY THE CLOSE
            % ADVERB TO CLOSE LOCK,CRUNCH
  02 FPB.PSEUDO.RDR                         ADDRESS,
            % PSEUDO READER FOR THIS FILE
  02 FPB.SAVE.HDWR                          BIT(6),%SAVE HDWR TYPE  IF FILE
                                              % EVER CHANGED BY OPEN
  02 FPB.INV.CHARS                          BIT(2),%
            % 0 = REPORT ALL INVALID CHARACTER LINES
            % 1 = REPORT ALL AND STOP AT THAT LINE
            % 2 = REPORT FIRST ONE ONLY
            % 3 = DONT REPORT ANY INVALID CHARACTERS
  02 FPB.SERIAL                             CHAR (6),%
            %ALPHA-NUMERIC TAPE SERIAL  ID
  02 FPB.Q.FAMILY.SIZE                      BIT(8),
            % IF FPB.QUEUE.FILE IS SET, THIS FIELD IS NUMBER OF Q-S.
  02 FPB.Q.MAX.MESSAGES                     BIT(8)
```

```
              % MAXIMUM NUMBER OF MESSAGES IN ANY ONE QUEUE
,  02 FPB.TRANSLATE.NAME                 CHARACTER(10)
              %TRANSLATE FILE NAME ("TRANSLATE"/FPB.TRANSLATE.NAME)
, 02 FPB.PROTECTION                      BIT (2) %  HOST RJE
              % 0 = PUBLIC FILE
              % 1 = PRIVATE FILE
              % 2 = GUARD FILE
, 02 FPB.PROTECTION.IO                    BIT (2)%      HOST RJE
              % 0 = I/O
              % 1 = INPUT ONLY
              % 2 = OUTPUT ONLY
,   2 FPB.MERGED.ATTRIBUTES                    BIT(12)
,        5 FILLER                              BIT(1)          % 0
,        5 FPB.MERGED.INPUTF                   BOOLEAN         % 1
,        5 FPB.MERGED.OUTPUTF                  BOOLEAN         % 2
,        5 FPB.MERGED.OPEN.OPTIONS             BIT(4)          %  3 TO 6
,             10 FPB.MERGED.WITH.PUNCH         BOOLEAN         % 3
,             10 FPB.MERGED.WITH.PRINT         BOOLEAN         %  4
,             10 FPB.MERGED.WITH.INTERPRET     BOOLEAN         %  5
,             10 FPB.MERGED.WITH.STACKERS      BOOLEAN         %  6 DANGER
,               11 FPB.MERGED.REVERSEF         BOOLEAN         %  6 DANGER
,        5 FPB.MERGED.LOCKF                    BOOLEAN         %  7
,        5 FPB.MERGED.LOCKOUTF                 BOOLEAN         %  8
,        5 FPB.MERGED.RRRNOFILE                BOOLEAN         %  9
,        5 FPB.MERGED.RRRLOCK.FILE             BOOLEAN         % 10
,        5 FPB.MERGED.OPENCODE                 BOOLEAN         % 11
,   02 FPB.MLTI.CHGD                     BIT (2)%
,   02 FPB.PKID.CHGD                     BIT (1)%
,   02 FPB.PROTOCOL                      BIT(8)
               % USED BY APPLICATION PROGRAM TO TELL MCS
                 MESSAGE FORMAT
  ;
#;%
```

__IPB DECLARATIONS__

```
DEFINE IPB.SIZE AS #1440#;
    DEFINE IPB.DECLARATION AS#
    DECLARE 01 DUMMY REMAPS IPB BIT(1440),
               02 FILLER BIT(1192),
               02 IPB.HARDWARE CHAR(1),
               02 IPB.ARCHITECTURE.NAME CHAR(10),
               02 IPB.COMPILER.LEVEL BIT(8),
               02 IPB.MCP.LEVEL BIT(8),
               02 IPB.GISMO.LEVEL BIT(8),
               02 IPB.ARCHITECTURE.ATTRIBUTES BIT(80),
               02 FILLER BIT(56);
#;
```

HINTS

```
 % THE FOLLOWING DECLARATIONS MUST ALWAYS REMAIN IN THE BEGINNING AND
 % THEIR ORDER MUST NOT BE CHANGED. THE FIRST THREE ARE RESERVED FOR THE
 % SDL INTERPRETER, THE REST ARE FOR THE DUMP ANALYZER.
 %
 % THE FOLLOWING DECLARATIONS MUST ALWAYS REMAIN IN THE BEGINNING AND
 % THEIR ORDER MUST NOT BE CHANGED. THE FIRST THREE ARE RESERVED FOR THE
 % SDL INTERPRETER, THE REST ARE FOR THE DUMP ANALYZER.
 %
DECLARE  HINTS     BIT(2047);
DEFINE%
     DISK.TRACE.BLOCK.ADDR     AS #SUBBIT(HINTS,0048,024)#%  HEX  030
 ,   LOCN.MAKE.MCP.BE.HERE     AS #SUBBIT(HINTS,0096,033)#%  HEX  060
 ,   LOCN.INTERP.DICT          AS #SUBBIT(HINTS,0129,024)#%  HEX  081
 ,   KI.KO                     AS #SUBBIT(HINTS,0153,001)#%  HEX  099
 ,   NO.REINSTATES             AS #SUBBIT(HINTS,0154,001)#%  HEX  09A
 ,   FIRE.UP.CONTROLLER        AS #SUBBIT(HINTS,0155,001)#%  HEX  09B
 ,   N.SECOND.COUNTER          AS #SUBBIT(HINTS,0156,002)#%  HEX  09C
 ,   MCP.LIMIT                 AS #SUBBIT(HINTS,0158,024)#%  HEX  09E
 ,   HINTS.LAST.OVLY           AS #SUBBIT(HINTS,0182,024)#%  HEX  0B6
 ,   MICR.DEBUG.BIT            AS #SUBBIT(HINTS,0206,1)   #%  HEX  0CE
 ,   Q.NOT.LOCKED              AS #SUBBIT(HINTS,0207,001)#%  HEX  0CF
 ,   DFH.DIR.AD                AS #SUBBIT(HINTS,0208,024)#%  HEX  0D0
 ,   AUTO.GUARD                AS #SUBBIT(HINTS,0232,003)#%  HEX  0E8
 ,   FIRE.SYSTEM.BACKUP        AS #SUBBIT(HINTS,0235,003)#%  HEX  0EB
 ,   FOUND.BACKUP.DESIGNATION  AS #SUBBIT(HINTS,0238,001)#%  HEX  0EE
 ,   TRACE.WORD                AS #SUBBIT(HINTS,0243,024)#%  HEX  0F3
 ,   TRACE.FLAG                AS #SUBBIT(HINTS,0261,003)#%  HEX  105
 ,   TRACE.OPT                 AS #SUBBIT(HINTS,0264,003)#%  HEX  108
 ,   HINTS.FIRST.QUEUE         AS #SUBBIT(HINTS,0267,024)#%  HEX  10B
 ,   ADDR.OF.COLD.START.VAR    AS #SUBBIT(HINTS,0291,024)#%  HEX  123
 ,   MAXM                      AS #SUBBIT(HINTS,0339,004)#%  HEX  153
 ,   ELOG.FULL                 AS #SUBBIT(HINTS,0343,001)#%  HEX  157
 ,   NON.RELEASE.MCP           AS #SUBBIT(HINTS,0344,001)#%  HEX  158
 ,   GISMO.LEVEL               AS #SUBBIT(HINTS,0345,008)#%  HEX  159
 ,   PSR.CHANGE.BIT            AS #SUBBIT(HINTS,0354,001)#%  HEX  162
 ,   RELEASE.LEVEL             AS #SUBBIT(HINTS,0355,008)#%  HEX  163
 ,   FIRST.LINK                AS #SUBBIT(HINTS,0363,024)#%  HEX  16B
 ,   SYCOUNTER                 AS #SUBBIT(HINTS,0391,020)#%  HEX  187
 ,   SY.CNTR.MSK               AS #SUBBIT(HINTS,0411,020)#%  HEX  19B
 ,   SY.PRIOR.TIME             AS #SUBBIT(HINTS,0431,020)#%  HEX  1AF
 ,   SYSTEM.PACK.INFO          AS #SUBBIT(HINTS,0471,024)#%  HEX  1D7
 ,   SYSTEM.UNIT               AS #SUBBIT(HINTS,0502,012)#%  HEX  1F6
 ,   SYSTEM.PORT.CHAN          AS #SUBBIT(HINTS,0502,007)#%  HEX  1F6
 ,   SYSTEM.PORT               AS #SUBBIT(HINTS,0502,003)#%  HEX  1F6
 ,   SYSTEM.CHANNEL            AS #SUBBIT(HINTS,0505,004)#%  HEX  1F9
 ,   SYSTEM.UNIT.EU            AS #SUBBIT(HINTS,0510,004)#%  HEX  1FE
 ,   CONSOUL.SWITCHES          AS #SUBBIT(HINTS,0514,028)#%  HEX  202
 ,   MICRO.TRACE.FLAG          AS #SUBBIT(HINTS,0542,001)#%  HEX  21E
```

```
,    PORT.CHANNEL.TABLE         AS #SUBBIT(HINTS,0567,192)#%   HEX  237
,    BYPASS.CLEANUP             AS #SUBBIT(HINTS,0759,001)#%   HEX  2F7
,    CONTRL.CRD.FLG             AS #SUBBIT(HINTS,0760,001)#%   HEX  2F8
,    EXT.RESULT.DESC.CHAIN      AS #SUBBIT(HINTS,0761,024)#%   HEX  2F9
,    T.FILES                    AS #SUBBIT(HINTS,0785,008)#%   HEX  311
,    MICR.COUNT                 AS #SUBBIT(HINTS,0793,006)#%   HEX  319
,    CHANGE.BIT                 AS #SUBBIT(HINTS,0799,001)#%   HEX  31F
,    RELEASE.VERSION            AS #SUBBIT(HINTS,0800,008)#%   HEX  320
,    SCHEDULER.LOCKED           AS #SUBBIT(HINTS,0808,001)#%   HEX  328
,    IOAT.POINTER               AS #SUBBIT(HINTS,0809,024)#%   HEX  329
,    IOAT.END                   AS #SUBBIT(HINTS,0833,024)#%   HEX  341
,    SYSTEM.PAUSE.DESC          AS #SUBBIT(HINTS,0857,024)#%   HEX  359
,    PSEUDO.TABLE.ADDRESS       AS #SUBBIT(HINTS,0881,024)#%   HEX  371
,    DATE.SET                   AS #SUBBIT(HINTS,0905,001)#%   HEX  389
,    TIME.SET                   AS #SUBBIT(HINTS,0906,001)#%   HEX  38A
,    GISMO.OPTIONS              AS #SUBBIT(HINTS,0907,024)#%   HEX  38B
,    CHECK.RA                   AS #SUBBIT(HINTS,0907,001)#%   HEX  38B
,    COMM.TRACE                 AS #SUBBIT(HINTS,0908,001)#%   HEX  38C
,    GISMO.TRACE                AS #SUBBIT(HINTS,0909,001)#%   HEX  38D
,    XFER.24                    AS #SUBBIT(HINTS,0910,001)#%   HEX  38E
,    DMS                        AS #SUBBIT(HINTS,0911,001)#%   HEX  38F
,    READR.SORTR                AS #SUBBIT(HINTS,0912,001)#%   HEX  390
,    ANY.MAG.TAPE               AS #SUBBIT(HINTS,0913,001)#%   HEX  391
,    NRZ.MAG.TAPE               AS #SUBBIT(HINTS,0914,001)#%   HEX  392
,    CASSETT                    AS #SUBBIT(HINTS,0915,001)#%   HEX  393
,    PAPER.TAPE                 AS #SUBBIT(HINTS,0916,001)#%   HEX  394
,    DATA.COMM                  AS #SUBBIT(HINTS,0917,001)#%   HEX  395
,    PORT.DEVICES               AS #SUBBIT(HINTS,0918,001)#%   HEX  396
,    EXCHANGES                  AS #SUBBIT(HINTS,0919,001)#%   HEX  397
,    PORT.TRACE                 AS #SUBBIT(HINTS,0920,001)#%   HEX  398
,    B1720.CODE                 AS #SUBBIT(HINTS,921,001)#%    HEX  399
,    B1860.CODE                 AS #SUBBIT(HINTS,0922,001)#%   HEX  39A
,    MPROC.CODE                 AS #SUBBIT(HINTS,0923,001)#%   HEX  39B
,    B1830.CODE                 AS #SUBBIT(HINTS,0924,001)#%   HEX  39C
,    READ.AFTER.WRITE.CHECK     AS #SUBBIT(HINTS,0925,001)#%   HEX  39D
,    PRIORITY.MEMORY.MGMT       AS #SUBBIT(HINTS,0926,001)#%   HEX  39E
,    FIFO.MEMORY.MGMT           AS #SUBBIT(HINTS,0927,001)#%   HEX  39F
,    THRASHING.COUNTING         AS #SUBBIT(HINTS,0928,001)#%   HEX  3A0
,    DCH.SCRATCH.MEM.ADDR       AS #SUBBIT(HINTS,0931,024)#%   HEX  3A3
,    S.MCP.TRACE                AS #SUBBIT(HINTS,0955,001)#%   HEX  3BB
,    INTERRUPT.SWITCH.SET       AS #SUBBIT(HINTS,0956,001)#%   HEX  3BC
,    DISABLE.INTERRUPT.SW       AS #SUBBIT(HINTS,0957,001)#%   HEX  3BD
,    MEMORY.USAGE.BIT           AS #SUBBIT(HINTS,0958,001)#%   HEX  3BE
,    MIN.MEMORY.SIZE            AS #SUBBIT(HINTS,0959,012)#%   HEX  3BF
,    BEEN.THRU.MCP.BE.HERE      AS #SUBBIT(HINTS,0971,001)#%   HEX  3CB
,    REMOTE.REROUTE             AS #SUBBIT(HINTS,0972,001)#%   HEX  3CC
,    QUEUE.REROUTE              AS #SUBBIT(HINTS,0973,001)#%   HEX  3CD
,    SPOLOG.NEEDS.TRANSFERING   AS #SUBBIT(HINTS,0978,001)#%   HEX  3D2
,    INTERPRETER.TABLE.ADDR     AS #SUBBIT(HINTS,0979,036)#%   HEX  3D3
,    SPO.PORT.CHAN              AS #SUBBIT(HINTS,1015,007)#%   HEX  3F7
```

```
,   SPO.PORT                    AS #SUBBIT(HINTS,1015,003)#%  HEX  3F7
,   SPO.CHANNEL                 AS #SUBBIT(HINTS,1018,004)#%  HEX  3FA
,   KEYBOARD.SPO.DESC           AS #SUBBIT(HINTS,1022,024)#%  HEX  3FE
,   SPO.SQ.AD                   AS #SUBBIT(HINTS,1046,24)#%   HEX  416
,   LAMP.FLAGS                  AS #SUBBIT(HINTS,1086,024)#%  HEX  43E
,   LAMP.DISK.TABLE.ADDR        AS #SUBBIT(HINTS,1110,024)#%  HEX  456
,    VARIABLE.LAMP.ALL.USER.CPU        AS#SUBBIT(HINTS,1113,1)#% 459
,    VARIABLE.LAMP.ALL.USER.CODE.OVLY  AS#SUBBIT(HINTS,1114,1)#% 45A
,    VARIABLE.LAMP.ALL.USER.DATA.OVLY  AS#SUBBIT(HINTS,1115,1)#% 45B
,    VARIABLE.LAMP.SMCP.CPU.1ST.BIT    AS#SUBBIT(HINTS,1122,1)#% 462
,    VARIABLE.LAMP.SMCP.OVLY           AS#SUBBIT(HINTS,1128,2)#% 468
,   SEGMENT.HALT                AS #SUBBIT(HINTS,1134,004)#%  HEX  46E
,   HALT.MASK                   AS #SUBBIT(HINTS,1138,024)#%  HEX  472
,   MMCP.SEGMENT.HALT           AS #SUBBIT(HINTS,1162,004)#%  HEX  48A
,   MMCP.HALT.MASK              AS #SUBBIT(HINTS,1166,024)#%  HEX  48E
,   COMPILE.TIME.OPTIONS        AS #SUBBIT(HINTS,1190,008)#%  HEX  4A6
,   RELEASE.VERSION.MCP         AS #SUBBIT(HINTS,1190,001)#%  HEX  4A6
,   DEBUG.OPTION                AS #SUBBIT(HINTS,1191,001)#%  HEX  4A7
,   CONTROL.MEMORY.SIZE         AS #SUBBIT(HINTS,1198,004)#%  HEX  4AE
,   MCP.VERSION.DATE            AS #SUBBIT(HINTS,1202,016)#%  HEX  4B2
,   MAIN.MEMORY.SIZE            AS #SUBBIT(HINTS,1218,012)#%  HEX  4C2
,   DM.GLOBALS                  AS #SUBBIT(HINTS,1230,024)#%  HEX  4CE
,   QUEUE.ROOT.ADDRESS          AS #SUBBIT(HINTS,1254,024)#%  HEX  4E6
,   DC.CHAIN                    AS #SUBBIT(HINTS,1302,024)#%  HEX  516
,   MESSAGE.LIST.TAIL           AS #SUBBIT(HINTS,1326,024)#%  HEX  52E
,   ABSOLUTE.S.MEM              AS #SUBBIT(HINTS,1350,012)#%  HEX  546
,   S.C.Q.EV                    AS #SUBBIT(HINTS,1362,001)#%  HEX  552
,   M.C.Q.EV                    AS #SUBBIT(HINTS,1363,001)#%  HEX  553
,   S.M.Q.EV                    AS #SUBBIT(HINTS,1364,001)#%  HEX  554
,   S.I.Q.EV                    AS #SUBBIT(HINTS,1365,001)#%  HEX  555
,   M.M.Q.EV                    AS #SUBBIT(HINTS,1366,001)#%  HEX  556
,   M.I.Q.EV                    AS #SUBBIT(HINTS,1367,001)#%  HEX  557
,   M.CAUSE.LOCK                AS #SUBBIT(HINTS,1368,001)#%  HEX  558
,   M.MCP.LR                    AS #SUBBIT(HINTS,1374,024)#%  HEX  55E
,   M.NUMBER.PAGES              AS #SUBBIT(HINTS,1542,024)#%  HEX  606
,   TRACE.ADDR                  AS #SUBBIT(HINTS,1566,024)#%  HEX  61E
,   M.MCP.Q.IDENT               AS #SUBBIT(HINTS,1590,008)#%  HEX  636
,   COMM.SPLITTER               AS #SUBBIT(HINTS,1598,072)#%  HEX  63E
,   MIKES.HALT.SPACE            AS #SUBBIT(HINTS,1670,096)#%  HEX  686
,   LAMP.CPU.BASE               AS #SUBBIT(HINTS,1766,001)#%  HEX  6E6
,   FIXED.LAMP.DISPLAY          AS #SUBBIT(HINTS,1767,001)#%  HEX  6E7
,   VAR.LAMP.BASE               AS #SUBBIT(HINTS,1768,001)#%  HEX  6E8
,   VAR.LAMP.CPU.OVLY           AS #SUBBIT(HINTS,1769,001)#%  HEX  6E9
,   VAR.LAMP.IO                 AS #SUBBIT(HINTS,1770,001)#%  HEX  6EA
,   VAR.LAMP.BARGRAPH           AS #SUBBIT(HINTS,1771,001)#%  HEX  6EB
,   LAST.MEM.LINK               AS #SUBBIT(HINTS,1790,024)#%  HEX  6FE
,   SYSTEM.ID                   AS #SUBBIT(HINTS,1862,012)#%  HEX  746
,   CPU.ID                      AS #SUBBIT(HINTS,1862,004)#%  HEX  746
,   MEMORY.ID                   AS #SUBBIT(HINTS,1866,004)#%  HEX  74A
,   IO.ID                       AS #SUBBIT(HINTS,1870,004)#%  HEX  74E
```

```
,   ELOG.HERE                  AS #SUBBIT(HINTS,1874,024)#%  HEX  752
,   QLOCK.COUNT                AS #SUBBIT(HINTS,1898,004)#%  HEX  76A
,   CHIP.TABLE.ADDRESS         AS #SUBBIT(HINTS,1902,024)#%  HEX  76E
,   MIX.MEMORY.PRIORITIES      AS #SUBBIT(HINTS,1926,016)#%  HEX  786
,   STOP.SCHED.INPUT           AS #SUBBIT(HINTS,1942,001)#%  HEX  796
,   NSEC.DISABL.THRASH.FAULT AS #SUBBIT(HINTS,1943,001)#%  HEX  797
,   DISABLE.THRASHING.FAULT    AS #SUBBIT(HINTS,1944,001)#%  HEX  798
,   MCP.VARIABLE.MEM.PRIORITY AS #SUBBIT(HINTS,1945,005)#% HEX  799
,   MEM.SWEEP.PENDING          AS #SUBBIT(HINTS,1952,001)#%  HEX  7A0
,   SAMPLING.CLOCK             AS #SUBBIT(HINTS,1953,006)#%  HEX  7A1
,   SAMPLING.INTERVAL          AS #SUBBIT(HINTS,1959,006)#%  HEX  7A7
,   MEM.SWEEP.INTERVAL         AS #SUBBIT(HINTS,1965,010)#%  HEX  7AD
,   MAX.SWEEP.INTERVAL         AS #SUBBIT(HINTS,1975,010)#%  HEX  7B7
,   MEM.EXTEND.COUNT           AS #SUBBIT(HINTS,1985,002)#%  HEX  7C1
,   OVERLAY.COUNTER            AS #SUBBIT(HINTS,1987,008)#%  HEX  7C3
,   OVERLAY.TARGET             AS #SUBBIT(HINTS,1995,008)#%  HEX  7CB
,   MCP.SWEEPS.BEFORE.DECAY    AS #SUBBIT(HINTS,2003,010)#%  HEX  7D3
,   MEM.DUMP.COMPLETE          AS #SUBBIT(HINTS,2013,001)#%  HEX  7DD
,   FOUNTAIN                   AS #SUBBIT(HINTS,2022,024)#%  HEX  7E6
,   CLEAR.START.REQD           AS #SUBBIT(HINTS,2046,001)#%  HEX  8FE
;
& IF STACK.CHECK
DECLARE 01 THRESHOLD BIT(24),  01 PREVIOUS.LOW BIT(24);
& END
```

HALTS

When a software-controlled halt occurs on a B1800/B1700 system, various registers are used to display information about the halt. The most important of these is the L REGISTER, which is considered the primary halt definition. Some halts display further descriptive information in other registers (usually X, Y and T).

The REGISTER is functionally divided into two portions.

(1) The left-most 16 bits (bits 0-15) describe the specific program or routine that halted, as follows:

@0000@	SDL Interpreter STATE light ON -- MCP STATE light OFF -- Normal-state SDL program
@0200@	MICRO.MCP
@0D00@	GISMO
@00F0@	CLEAR/START
@000F@	SYSTEM/INIT
@B000@	BASIC Interpreter
@C000@	COBOL Interpreter
@E000@	RPG Interpreter
@F000@	FORTRAN Interpreter
@5000@	B500 Interpreter
@1100@	1130 Interpreter
@1400@	1401 Interpreter

(2) The rightmost 8 bits (bits 16-23) give the halt identification. This portion of the halt code is dependent upon the specific routine that halted (given in bits 0-15).

SDL INTERPRETER HALTS (L=@0000xx@)

@01@	Evaluation/Program Pointer Stack Overflow.
@02@	Control Stack Overflow.
@03@	Name/Value Stack Overflow.
@04@	REMAPS size error.
@05@	Invalid Parameter passed to a PROCEDURE.
@06@	Invalid Substring (SUBBIT or SUBSTR).
@07@	Invalid Subscript.
@08@	Invalid Value returned from a TYPED PROCEDURE.
@09@	Invalid CASE.

@0A@	Divide by Zero.
@0B@	Invalid Index.
@0C@	Invalid Operator.
@0E@	Invalid Parameter in VALUE.DESCRIPTOR.
@10@	Console Halt (INTERRUPT switch).
@11@	HALT Operator (T REGISTER contains further definition of the halt). Complete information on MCP halts is given below.
@12@	Write Out-of-Bounds.
@13@	No memory for MCP TRACE.
@1E@	Invalid Parameter in DYNAMIC Declaration.
@1F@	Invalid TRANSLATE.
@35@	1860 Console Cassette Data Error
@37@	1860 RWOAM detected

BASIC INTERPRETER HALTS (L=@B000xx@)

@10@ Console Halt (INTERRUPT switch)

COBOL INTERPRETER HALTS (L=@C000xx@)

@10@ Console Halt (INTERRUPT switch)

RPG INTERPRETER HALTS (L=@E000xx@)

@10@ Console Halt (INTERRUPT switch)

FORTRAN INTERPRETER HALTS (L=@F000xx@)

@10@ Console Halt (INTERRUPT switch)

B500 INTERPRETER HALTS (L=@5000xx@)

@10@ Console Halt (INTERRUPT switch)

1130 INTERPRETER HALTS (L=@1100xx@)

@10@ Console Halt (INTERRUPT switch)

1401 INTERPRETER HALTS (L=@1400xx@)

@10@	Console Halt (INTERRUPT switch)

FIRMWARE HALTS (L=@00F0xx@ OR @000Fxx@)

@02@	Device on selected Channel is not disk (T=device id).
@03@	Control is not idle (T=control status).
@04@	Timeout while waiting for SERVICE REQUEST. (software timed).
@05@	Bad Reference Address (X=good, Y=bad, T=Port/Channel)
@06@	Bad status count after SERVICE REQUEST (T=control status).
@07@	Bad Result Descriptor (T=Result Descriptor).
@08@	Seek timeout (software timed).
@0A@	Memory parity error on I/O.
@0B@	Timeout while waiting for OP.COMPLETE.
@0C@	Exception condition after 15 retries (T=Result Descriptor).
@0F@	No disk found on system.
@12@	Insufficient memory for this routine.
@13@	Memory parity error after overlay.
@14@	Memory parity error (no diagnosis of location).
@15@	NAME TABLE entry is empty.
@17@	SYSTEM/DUMPFILE disk address is zero. (DUMP option is probably reset).
@1C@	No SPO on system.
@1E@	Checksum error. T = SOFTWARE.ID
@22@	Cassette data error.
@24@	No empty channel for SC.SPO.
@26@	NAME TABLE is not proper level.
@27@	Other than test op to nonexistent port device.
@28@	Extended toggle entry point (load registers and hit start).
@29@	Dumpfile port-channel does not match system disk port-channel.
@83@	Optional GISMO segment required which does not exist in this version of GISMO.
@86@	Software compatibility problem. TE & TF contain values which identify the incompatible programs: 1=SYSTEM/INIT 2=GISMO 3=MICRO.MCP 4=MCP 5=SDL Interpreter If TD=1 then X=Level, Y=Level If TD=7 then X="NEEDS", Y="HAS"

@87@	Pseudo MAXM (LR bits 15-23) greater than real MAXM.
@8A@	Not enough M.Memory.
@8B@	Pseudo MAXS too large (X=pseudo MAXS, Y=maximum value we can use). Push START to use value in Y.
@90@	S.Memory space needed, but area was XM-ed.
@91@	Bad op to soft console.
@92@	Soft console out of sync with driver.
@93@	Time out on soft console.
@94@	No service request on soft console.
@95@	Soft console message without ETX.
@96@	Test op to soft console returned unknown ID, see T

__GISMO HALTS (L=@0D00xx@)__

@10@	Console Halt (INTERRUPT switch).
@20@	Reference Address error (X=good, Y=bad, TA=channel, TE-TF=control status). Push START to assume good address and continue.
@21@	Irrecoverable Reference Address error (same register settings as halt @0D0020@).
@22@	Reference Address is zero (TB=channel, X=Reference Address).
@24@	Unknown device id (TB=channel, X=Reference Address).
@25@	Invalid device id for second OP.COMPLETE bit off (TB=Channel, X=Reference Address).
@26@	Illegal request (STAND.ALONE GISMO).
@27@	Pause on uninitialized Channel (TB=Channel, X=Reference Address).
@28@	SOFT.IO seek complete problem (TB=Channel, X=Reference Address).
@29@	Bad exchange entry in CHANNEL.TABLE (TE=Port, TF=Channel).
@30@	INTERRUPT.QUEUE Overflow (X=number attempted, Y=maximum allowable).
@31@	MLC hung (TE=Port, TF=Channel, X=Reference Address).
@32@	Memory parity error detected by MLC. Push START to find parity error.
@33@	M.WAIT: MCP waiting only on time or does not have S.I.Q.EV set in WAIT.LIST.
@34@	Memory parity error (T=address of byte in error). Pushing START will force good parity at the address in T and start another scan of memory. If T=@FFFFFF@, then there was a read outside the physical bounds of memory or an intermittant parity error that could not be isolated.
@36@	COMMUNICATE.WITH.GISMO: Bad verb (t=fa value when reading communicate).
@37@	COMMUNICATE.WITH.GISMO: Bad adverb (X=Communicate

function, T=FA value when reading Communicate).

@38@ COMMUNICATE.WITH.GISMO/GISMO.COMMUNICATE: Parameter list length error (T=FA address).

@39@ GISMO.COMMUNICATE: Bad verb (X=verb).

@40@ COMMUNICATE.WITH.GISMO/GISMO.COMMUNICATE: Call by non-MCP (T=LIMIT.REGISTER).

@41@ USE.COMMUNICATE: Code not present.

@42@ Deleted function (T=LIMIT.REGISTER, X=SWAPPER value).

@43@ HI.PRI: MCP not in READY.Q.

@44@ MCP clobbered RS field (TB=Channel, X=Reference Address), or a Reference Address error halt because RFAC option was reset.

@46@ Disk exchange EU.BUSY from non-busy EU.

@47@ Hi-priority Interrupt request from MLC on DISPATCH.READ (T=Port/Channel, X=Reference Address+24).

@48@ SOFT.IO: Hi-priority request for a non-sorter device (T=Channel, X=Reference Address, Y=RS field).

@50@ Attempt to reinstate non present RS

@51@ Part, Data Transfer on Disk Pack T=RD, X=Reference Address

@53@ 1860 Cass Data error at entry to GISMO. Y=STATE.FLAGS, X=LR

@54@ 1860 CPU multiple (parity). T=ELOG, Y=STATE.FLAGS, X=LR

@55@ 1860 RWOAM detected. T=0, Y=STATE.FLAGS, X=LR

@56@ 1860 RWOAM during memory scan to check location of parity error. Probably indicative of CPU failure. T, X, and Y are as they would have been had we halted for parity.

@57@ 1860 NON-CPU parity. T=ELOG, Y=Nothing, X=LR. If error was detected via port int (rather than ELOG) we wait for awhile to get ELOG. It may never be reported, and T will not have ELOG. NON.CPU.MULTIPLE on.

@58@ In STAND.ALONE GISMO, 1860 correctable error in S memory. T=ELOG. Push START to continue processing; to suppress further reports, modify T Register prior to pushing START.

@59@ Illegal op with verify variant. TB=CHANNEL, X=REF.ADR

@60@ Service req from missing or ignored CHANNEL. TB = CHANNEL

@80@ IQ: Queue empty but an MCP count not equal to zero.

@81@ IQ: Queue empty but an IQ event is set.

@82@ IQ: Queue not empty but total count is zero.

@83@ IQ: Queue not empty but sub-queue count is zero.

@84@ IQ: Queue not empty but event is reset.

@85@ COMM: ITYPE=00, COMM=59.
@86@ Q.OUT: On READY.Q, not called by SCHEDULER.
@87@ Q.OUT: An MCP COMMUNICATE.Q not empty but event bit is reset.
@88@ HANG.RS: On time only, but clock is zero.
@89@ LRU: Old, presence bit reset.
@90@ ADJUST.INTERP: RS not MCP.
@91@ USE: See code.
@92@ USE: Not MCP who tried to transfer to USE.
@93@ USE: See code.
@94@ Bad event descriptor (T=RS).

MICRO.MCP HALTS (L=@0200xx@)

@01@ INT: OP.COMPLETE off.
@02@ INT: IO.MCP.IO is invalid.
@03@ CD: Block and S.IOC.
@04@ CD: IO.MCP.IO not USER.IO.
@05@ CD: USE.IT, not IOC.E.
@06@ CD: Not IOC.E and IOC set.
(the above all halt with T=Reference Address, Y=Result Descriptor, X=M.EVENTS CAT IO.MCP.IO).
@07@ HANDLE.INT: Bad result (T=result from GISMO).
@08@ DISPATCH: Bad result (T=result from GISMO).
@09@ MMCP.COMM: Bad RS.COMMUNICATE.MSG.PTR.
@10@ MMCP.COMM: COMM.VERB not 1, 2, 3, or 10.
@11@ Irrecoverable error T=HEX.SEQ.NO
@12@ Conditional halt
@13@ On a delayed random I/O, RD indicates that the I/O has been initiated but MCP.IO is in an improper state
@20@ FETCH.MSG returned null empty
@21@ Incomplete prior dispatch
@22@ LF NEQ 1 after dispatch
@23@ Not POCKET.COMPLETE or DOUBLE.DOCUMENT
@24@ Not POCKET.COMPLETE or TOO.LATE.2.POCKT
@75@ M.I.Q.EV set.
@76@ IQE.VALID off.
@77@ IQE.MMCP off.
@79@ COMMUNICATE.WITH.GISMO not zero-relative.
@80@ Ex. SIOC but not BEEN.THRU.ERROR

CLEARSTART: TAPE-IO-RUN MODE (L=@AAAAAA@)

Toggle entry point load registers, select run mode, start.

X = Disk port and channel
Y = Printer port and channel
T = Toggles

0 = Memory dump
1 = Standalone program
2 = Unused
3 = TRACE.SWITCH
4 = Experimental MCP
5 = Experimental Init
6 = Experimental Interpreter
7 = Experimental GISMO
8 = Experimental MMCP
9 - 11 = Standalone program number
12 = Halt at selected points in Init with L=HEX.SEQ.NO
13 = Override the XM Table
14 = Unused
15 = Halt a second time for more entries

BR = GISMO trace flags, overrides those in Coldstart Variables.
LR = Pseudo MAXS, Pseudo MAXM
FA = Ignore soft IO channels
FB = Keep specific GISMO segments, overrides discarding rules.

@28@ Second Halt for more entries (LOAD and START):
X = Location to be temporarily XM-ed.
Y = Length to be temporarily XM-ed.
FB = More GISMO segments to be kept.

MCP HALTS (L=@000011@)

When the MCP executes an explicit HALT instruction, the L REGISTER is set to @000011@ and a parameter is loaded into the T REGISTER to further define the nature of the halt. Usually, this parameter is the first six digits of a sequence number in the MCP itself; however, some halt conditions occur at more than one place in the MCP. These are given a common identifier, as follows:

T REGISTER	REASON
@111111@	No memory

@Cxxxxx@ @Dxxxxx@ @Exxxxx@	Disk I/O Error
@000Cxx@	Systems Software Compatibility Error The low-order eight bits of the T REGISTER contains two 4-bit numbers identifying the two programs that are incompatible, as follows:

1 SYSTEM/INITIALIZER
2 GISMO
3 MICRO.MCP
4 SDL MCP
5 SDL Interpreter

All other halts point to a specific sequence number in the MCP. Unless otherwise noted, all halts are irrecoverable; a memory dump should be taken and submitted with supporting documentation and a B1700 Field Trouble Report.

T REGISTER	REASON
@0432750@	M.IN.M.OUT failed
@0433150@	Integrity of system disk is bad
@0435500@	Attempt to initiate in-process I/O
@0438750@	DISPATCH to invalid Port or Channel
@0439080@	QLOCK.COUNT overflow
@0439090@	QLOCK.COUNT underflow
@0483660@	Disk I/O did not complete in 10 seconds. Push START to retry.
@0484750@	MSG.TYPE=0 in S.MSG.Q
@0484790@	DISK.TRACE msg with release MCP
@0613500@	Invalid USERCODE
@0613510@	Tried to verify a usercode but no one is running
@0613530@	Tried to verify a usercode but no one is running
@0613580@	Tried to verify a usercode but no one is running
@0639600@	Could not find Pseudo Reader that was in use
@0717940@	Problem with skipping an empty area of a disk file
@0717940@	Call on GET.NEW.AREA with POSITION communicate is invalid
@0717950@	Could not update Base Pack header
@0717960@	Unrecognizable communicate called GET.NEW.AREA
@0985860@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@0989310@	Irrecoverable error not handled by IO.ERROR
@0989460@	Checking for INV.CHAR in Pseudo Reader but there are no Readers

@0996672@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@0999482@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@1006682@	Incorrect interface between Q.DRIVER & Q.FILES
@1007252@	Incorrect interface between Q.DRIVER & Q.FILES
@1009382@	File not open not screened off by R.W.CALLER
@1009392@	File positioning not screened off by R.W.CALLER
@1009462@	Enhanced I/O not allowed
@1015522@	DISPATCH to invalid port or channel
@1015552@	MCP is lost during Punch Check Recovery
@1015582@	DISPATCH to invalid Port or Channel
@1019612@	Got an INTERRUPT from the MMCP
@1020152@	Memory parity on DISPATCH (Interrupt from Channel 15)
@1026032@	Invalid IO.TYPE - 24
@1026912@	In IOCOMPLETE procedure and I/O in question is not complete
@1026922@	Invalid IO.TYPE - 32
@1027592@	Begin address of I/O higher than MCP.LIMIT
@1027742@	DISPATCH to invalid Port or Channel
@1029282@	Invalid USERCODE
@1097252@	DISK.ADDR not part of Q.DISK
@1097952@	RETURN.Q.DISK - space already returned
@1122282@	Self-checking in Q.DISK.DRIVER
@1122992@	I/O Descriptor not complete
@1123942@	No message found
@1127402@	No message found
@1136752@	Queue memory link data structure broken
@1138802@	Queue must be empty here
@1140572@	No disk passed to RETURN.DISK
@1140852@	Incorrect use of CLOSE.QUEUE parameters
@1154702@	Queue FIB not in memory
@1202682@	Invalid return from CLOSE.QUEUE
@1233052@	Invalid parameters passed to OPEN.QUEUE
@1239052@	Invalid parameters passed to OPEN.QUEUE
@1254652@	Invalid parameters passed to CLOSE.QUEUE
@1306352@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@1306532@	Invalid value returned from PUT.QUEUE.MESSAGE
@1357922@	Failed to Open QUEUE for AUTOBACKUP
@1357932@	Failed to Open QUEUE for AUTOBACKUP
@1357942@	Failed to Open QUEUE for AUTOBACKUP
@1357952@	Failed to Open QUEUE for AUTOBACKUP
@1357982@	Failed to QUEUE a message for AUTOBACKUP
@1358082@	Failed Receiving Msg from QUEUE for AUTOBACKUP
@1358122@	Failed Receiving Msg from QUEUE for AUTOBACKUP
@1358132@	Failed Receiving Msg from QUEUE for AUTOBACKUP
@1358142@	Failed Receiving Msg from QUEUE for AUTOBACKUP
@1358172@	Failed closing QUEUE for AUTOBACKUP
@1403302@	No such job scheduled
@1405352@	Scheduled a job that does not exist on disk
@1415722@	Invalid USERCODE
@1460502@	Cannot handle type INDIRECT in a Code Dictionary

@1541553@	CLUB.SOMEONE failed
@1713003@	CLUB.SOMEONE failed
@1715403@	Terminating a job that does not exist on disk
@1721253@	Terminating a job that does not exist on disk
@1734333@	No schedule entry for successor on unconditional execute
@1741663@	Q.KEY stored in QPTR is now bad
@1756063@	CLUB.SOMEONE failed
@1795953@	No disk space available for ROLLOUT
@1805153@	CLUB.SOMEONE failed
@1814633@	Cannot find the file that we are currently using
@1821753@	Sy.type = system.desc and sy.file on
@1897653@	Bad call on OPEN.QUEUE
@1897673@	Bad call on OPEN.QUEUE
@1903393@	Bad call on OPEN.QUEUE
@1903403@	Bad call on OPEN.QUEUE
@1903633@	Bad call on GET.QUEUE.MESSAGE
@1903643@	Bad call on GET.QUEUE.MESSAGE
@1903653@	Bad call on GET.QUEUE.MESSAGE
@1903663@	Bad call on GET.QUEUE.MESSAGE
@1964603@	Invalid MICR Communicate
@2124663@	System disk with no available disk space
@2124673@	Cannot find an IOAT for this system disk
@2148303@	DISPATCH to invalid Port or Channel
@2158773@	A control did not respond to TEST OP after a 10-second wait (at CLEAR/START). Push START once -- T will contain Port and Channel. Push START again -- T will contain I/O Descriptor address.
@2180683@	SPO did not respond to a TEST OP
@2201773@	CLEAR/START again (increased COLD.START.VARIABLES size
@2224583@	Disk is not a system disk
@2247303@	Failed to absolutize GISMO or SDL Interpreter
@2338823@	I/O won't complete while trying to ready a disk
@2375743@	SPO.QUEUE has been destroyed
@2378673@	Invalid recursion in CONTROL.CARD.DRIVER
@2381733@	CRT SPO had read data beyond the buffer limits
@2381903@	Invalid recursion in CONTROL.CARD.DRIVER
@2431203@	Bad value passed to HEADER.UPDATE
@2446253@	Cannot find MPF.INFO.TABLE already in use
@2462653@	Bad disk file header
@2473823@	Cannot find entry in MPF.TABLE
@2474323@	Cannot find MPF.INFO.TABLE already in use
@2632653@	No disk available for temporary available table
@2730253@	Bad entry passed to RETURN.DISK. Push start to resolve conflict and continue.
@2730263@	Bad address passed to RETURN.DISK
@2730273@	Zero Port and Channel passed to RET.DISK
@2730623@	Bad disk address passed to RETURN.DISK
@2736393@	Bad disk address passed to RETDSKENMASSE

@2736643@	RETURN.DISK problem. Push START to continue
@2787718@	Tape I/O did not complete after a 10-second wait. Push START once -- T will contain I/O Descriptor address. Push START again -- T will show whether OP was complete.
@2921159@	Unexpected return from GET.QUEUE.MESSAGE
@2921161@	Unexpected return from GET.QUEUE.MESSAGE
@2921162@	Unexpected return from GET.QUEUE.MESSAGE
@2970066@	NCR = " " in Scanner after reduce
@3032833@	Unexpected return from PUT.QUEUE.MESSAGE
@3033305@	Unexpected return from PUT.QUEUE.MESSAGE
@3036665@	Failed receiving msg from QUEUE for AB
@3036667@	Failed closing QUEUE for AB
@3040422@	Incorrect call on OPEN.QUEUE
@3040432@	Incorrect call on OPEN.QUEUE
@3040593@	Incorrect call on PUT.QUEUE.MESSAGE
@3040602@	Incorrect call on PUT.QUEUE.MESSAGE
@3197771@	Incorrect call on PUT.QUEUE.MESSAGE
@3197772@	Incorrect call on PUT.QUEUE.MESSAGE
@3611155@	Bad call on CLOSE.QUEUE
@3932342@	Invalid USERCODE
@4217725@	Bad Pseudo Reader chain
@4262733@	Verify USERCODE failed in "RV"
@4298402@	No disk available for available table
@4335142@	Bad entry in SPC.TABLE during SQUASH
@4335172@	No entry in AVL.TABLE segment
@4335212@	Bad virtual block number
@4335412@	RETURN.DISK failed
@4335612@	Problems in MERGE phase
@4335862@	Missing entry in SQ.FILE
@4335882@	Cannot RETURN.DISK while shrinking SQ.FILE
@4335932@	Cannot find an SQ.FILE area in SQ.DIRECTORY
@4336032@	Got the wrong buffer
@4336042@	File header disk address missing
@4336062@	File AREA.ADDRESS missing
@4336082@	CHILD.DIRECTORY address missing
@4336102@	Bad data block being returned to disk
@4336112@	Bad data block being returned to disk
@4336302@	Cannot RETURN.DISK for SQ.FILE.SPACE
@5074492@	Cannot find translate file that was already in use
@5119952@	Cannot find pack that is already in use
@5126132@	No space allocated for ANSI.BUFFER.SPACE
@5128322@	Illegal hardware type
@5128562@	Illegal hardware type
@5239392@	Missing file header for a multi-pack file
@5240042@	Cannot find pack that is already in use
@5247192@	Verify USERCODE failed when OPEN.SET.OVERRIDE
@5249352@	Pseudo Reader links are blown
@5469902@	QUEUE not a queue or not a disk queue

@5470088@	Cannot find disk
@5473356@	Invalid parameters passed to OPEN.QUEUE
@5474431@	Invalid parameters passed to CLOSE.QUEUE
@5474452@	Invalid parameters passed to CLOSE.QUEUE
@5476653@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@5483373@	Bad call on OPEN.QUEUE
@5484430@	Cannot backtrack (CLOSE.QUEUE problem)
@5488876@	Device is not a DATA.RECORDER
@5491104@	Attempted to open DATA.RECORDER other than INPUT or OUTPUT
@5491146@	Attempted to open DATA.RECORDER other than INPUT or OUTPUT
@5558820@	Bad disk directory
@5565520@	Irrecoverable disk I/O error on file header read
@5590005@	No disk available for directory expansion
@5594403@	Irrecoverable disk I/O error on file header read
@5672260@	Partial block on Disk with no area allocated
@5675555@	Could not update Base Pack header
@5713395@	Could not power down Continuation Pack
@5716695@	Cannot find Base Pack
@5731187@	Lost a Disk File Header for a Pseudo Reader
@5734450@	MPF.TABLE has been destroyed
@5788895@	Blown Q.KEY in QUEUE file structure
@5818873@	Invalid parameters passed to PUT.QUEUE.MESSAGE
@5820042@	Invalid parameters passed to CLOSE.QUEUE
@5820081@	Invalid parameters passed to CLOSE.QUEUE
@5835526@	Failed to Queue a msg for AUTOBACKUP
@5855525@	Q.KEY for SPO.QUEUE is blown
@5878862@	FIB indicates partial block but block count = 0
@5905517@	No label on multi-file tape
@5947766@	Tape I/O did not complete after a 10-second wait. Push START once -- T will contain I/O Descriptor address. Push START again -- T will show whether OP was complete.
@6026613@	Non-card device in CARD.STATUS
@6053622@	SCHED.OUT failed
@6082210@	Some I/O did not complete after a 10-second wait in the IO.ERROR routine. Push START once -- T will contain I/O descriptor address. Push START again -- T will show whether OP was complete.
@6375530@	Bad Port and Channel in Descriptor
@6375550@	Bad Descriptor
@6386630@	Structure has invalid DFH pointer
@6420050@	Invalid AUDIT.EXCEPTION.STATUS value
@6444400@	Invalid AUDIT.EXCEPTION.STATUS value
@6453360@	Bad AUDIT.TYPE Length
@6454410@	Bad STR.NUMBER Length
@6454430@	Bad data descriptor
@6454450@	Bad data descriptor
@6454470@	Bad data descriptor

@645510@	Bad DISK.L.ADDR Length
@6455560@	Bad DISK.L.ADDR Length
@6456600@	Bad DISK.L.ADDR Length
@650106@	DMS Audit User Count is incorrect
@6531113@	DMS Audit User Count is incorrect
@6534333@	Files/Structures linked into globals at last DB close
@6534733@	DB dictionary not found on disk at DB close
@6586373@	Cannot find DFH area for data record of disk search
@6587473@	Interrupt with op incomplete in IO.CLEAN.UP
@6587483@	Interrupt with op in-process in IO.CLEAN.UP
@6699403@	Exception while backing out DMS operation
@6737033@	GET.RECORD requested address exceeds file bounds
@6979723@	The number of entries in a Fine Table exceeded
@6986023@	The number of entries in a Coarse Table exceeded
@7042503@	Could not find key
@7103603@	Invalid mix number in currents
@7233703@	No key match
@7234063@	Cannot find Index Sequential key to delete
@7234203@	Problem with WA.CT.ENTRY
@7286403@	Bad parameter (CASE 0)
@7431013@	WA.STR.ADDR equal zero with non-zero WA.REASON
@7442003@	DMC.MAIN invalid reason
@7442803@	DMC.MAIN invalid reason
@7447603@	DMC.MAIN invalid reason
@7817103@	Ran out of disk during DMS
@7835253@	Missing LOG file
@7835403@	Missing LOG file
@7835423@	Missing LOG file
@7835443@	No disk available for new LOG area. LOG option will be reset after CLEAR/START
@7897843@	Changed system disk without CLEAR/START
@7932993@	Invalid Queue parameter for Autobackup Queue
@9904513@	Invalid communicate (CODE overlay)
@9906613@	Invalid communicate (INTERPRETER overlay)
@9906623@	Invalid communicate (GISMO overlay)
@9952533@	Logical I/O problems
@9952543@	Cannot INTERVENE yet

COLDSTART HALTS

HALT CODE	DESCRIPTION
1	Disk I/O error, hit Start once to get result descriptor.
2	Tape I/O error, hit Start once to get result descriptor.
3	Unexpected data or result descriptor from tape.
4	No tape control on system.
5	No disk control on system.
6	Disk not initialized in MARK 3.3 format.
7	Trying to coldstart user pack or cartridge.
8	Missing system tape. Make available and hit Start.
9	Missing files from tape. Hit Start until @0C0009@ is displayed again to obtain list of missing file numbers: 1 = MCPII 1 = SDL/INTERP1S 3 = SDL/INTERP1M 4 = GISMO 5 = SYSTEM/INIT 6 = MCPII/MICRO.MCP 7 = SYSTEM/LOAD.DUMP 8 = FILE LOADER
A.	Missing device on dispatch.
B.	Insufficient disk for COLDSTART.
C.	Went past tape mark.
D.	Missing tape mark.
E.	No SPO control on system (Non-released).
F.	SPO not ready (Non-released).

ENHANCEMENTS FOR VII.0

PRIORITIZED MEMORY MANAGEMENT

Prioritized Memory Management, as it is implemented in the Mark VII.0 level of MCPII, can be defined as an automatic mechanism for detecting and eliminating thrashing on B1800/B1700 systems. Thrashing, as it exists in the B1800/B1700 environment, is an overcommittment of memory that results in a degradation, sometimes very serious, of the system's peformance. Prioritized memory management is designed to eliminate thrashing by automatically notifying the operator when thrashing occurs, if the THRashing option is set, and by allowing the operator a range of recourses to address the problem that has been identified.

No single memory management system is ideal for all situations. Consequently, the VII.0 memory management system is implemented on four separate levels of sophistication to minimize the possible impact of memory management on those who do not need or want it. Those who are satisfied with lower levels of this system need not be concerned with the details of higher levels. This approach also allows users to ease into the more sophisticated aspects of the new system smoothly without being forced into an all-or-nothing situation.

This section discusses the implementation of Prioritized Memory Management and also provides background information on general approaches to memory management and the GETSPACE routine as it has been implemented in MCPII prior to the VII.0 release.

DEFAULT LEVEL

The first (default) level of memory management implements the old "GETSPACE" algorithm (see "GETSPACE" ALGORITHM PRIOR TO 7.0 below) with the exception that the "victim selector" routine, which decides which piece of memory to de-allocate next, is microcoded. This change was made for ease of implementation with the higher levels of the new memory management system, and not for increased speed of operation. No increase in speed should be expected.

Advantages

The advantages of this implementation are that:

1. Most users who are happy with the old "GETSPACE" algorithm should not notice any change when using this level of the new algorithm.

2. It requires the least amount of memory of any of the new memory management levels.

Disadvantages

The disadvantages are that:

1. All the disadvantages of the old "GETSPACE" algorithm listed below are still there; i.e., no mechanism to detech thrashing, relative activity not know, job priority not considered, etc.

2. It requires 50 more bytes of non-overlayable memory than the old implementation of this algorithm, though the performance appears approximately equal.

3. Memory links are one byte larger. A rough estimate of the number of memory links in a system is one per thousand bytes of physical memory.

SPO Messages

There are no SPO messages associated with the Level One implementation of VII.0 memory management.

DEFAULT-WITH-THRASHING-MESSAGE LEVEL

The second level of the new memory management algorithm implements thrashing detection. The "victim selector" of the Default Level is still used. When GISMO, which is monitoring overlay activity, decides that thrashing is occurring which is not a temporary phenomenon, it sends a communicate to the S-MCP. The S-MCP does the following two things:

a. Stops any more jobs from being automatically started. This can be overridden by using the "PS" SPO message to prod the schedule. Otherwise the schedule will not be automatically restarted until some job goes to to EOJ.

b. Sends the following message to the SPO: "***SYSTEM IS THRASHING, SCHEDULE STOPPED***". This message is repeated every N.SECOND if thrashing continues, however, there is a SPO message (see below) to cause the thrashing warning to only be repeated once per job entered into or removed from the mix.

When the system is shifting from one working set to another, memory is often overcommitted for a short period of time. This condition is acceptable provided it does not persist for long. However, one installation may be willing to tolerate an overcommittment of memory for longer time intervals than another. Therefore, a mechanism is provided to adjust the sensitivity of the memory management system's thrashing detection mechanism. (See "MM O.T" SPO message below)

Advantage

The advantage of this level is that users will now know when their memory is overcommitted and therefore, be able to do a much better job of maintaining a mix of jobs which utilizes most of memory but does not cause thrashing to occur.

Disadvantage

The only disadvantage of this level is that 140 more bytes of non-overlayable memory are required, beyond the requirement of Level One.

SPO Messages

SPO messages associated with Level Two are the following:

SO THR — Invokes Level Two of the new memory management system after the next Clear/Start. If the THR option is set, the initializer binds into GISMO the code to detect thrashing.

MM THRASH ON
MM THRASH OFF — When THRASH is on, the thrashing message will

be redisplayed on the SPO once per N.SECOND as long as thrashing continues. This is the default value. When THRASH is off the thrashing warning message is displayed only each time a job is added to or removed from the mix as long as thrashing continues.

<Job#>ST EOJ — This message will stop the program indicated by <Job#> and automatically restart it when another program goes to EOJ. ST EOJ is useful when the system gets into a thrashing state. The operator can then use the "ST EOJ" message to stop jobs until the mix is not thrashing. These jobs will then be automatically resumed after other jobs go to EOJ. Of course a "<Job#>GO" message will restart a stopped job right away.

MM O.T <decimal #> — Sets the overlay target to <decimal #> which may vary between 4 and 31. The default value is 8. These values denote the number of overlays per item interval. One time interval, in this instance, is equivalent to 800 milliseconds. Higher values will make the thrashing detector less sensitive and lower values will make it more sensitive.

PRIORITIZED-MEMORY LEVEL

The third level of the new memory management system implements the thrashing detection of Level Two with a new "victim selector" based on job priority and segment usage. The Prioritized-Memory Level is invoked by simply setting the "MPRI" option and Clear/Starting which causes the initializer to incorporate a different "victim selector" into GISMO.

In a "flat mix", one with all priorities equal, those segments which are actively in use will tend to stay in memory while those segments which are no longer being used will tend to be overlayed. This cannot be made an absolute policy in a memory management scheme based on segmentation due to problems of "geography". A very small inactive segment which has been allocated between two active segments may remain in memory longer than it otherwise would because of its location.

In a mix with varying priorities, those segments of high priority jobs which are actively in use will be protected from segments of lower priority jobs. When a high priority segment is not accessed for a period of time (equal to 1.2 seconds on the average) it is lowered to the priority of the next lower priority job in the system and is again protected at that priority for the same period of time, etc. If a segment is accessed at any time before being overlayed it is restored to its original priority. In this way segments of high priority jobs are protected from low priority jobs' segments and yet unused segments of any job tend to be overlayed.

Advantages

There are two advantages to Level Two implementation:

1. Changes in job priority will protect active segments of higher priority jobs from being overlayed by segments of lower priority jobs.

2. All the advantages of Level Two listed above.

Disadvantages

The disadvantages are that:

1. 150 more bytes of non-overlayable memory than that required by Level Two are required.

2 If jobs are run at varying priorities fragmentation of memory is increased.

SPO Messages

SO MPRI	Invokes Prioritized-Memory Level of the new memory management system after the next Clear/Start. If the MPRI option is set the initializer binds into GISMO the code to detect thrashing as well as the prioritized "victim selector".
<Job#>PR <decimal #>	Sets both the processor priority and the memory priority of the indicated job to <decimal #> which must be > 0 and < 15. Default value is 4.
<Job#>MP <decimal #>	Sets only memory priority of the indicated job to <decimal #> which must be > 0 and <

15. Default value is 4. (The multipack SPO command which used to be "MP" is now "MU".)

<Job #>PP <decimal #> Sets only the processor priority of the indicated job to <decimal #> which must be > 0 and < 15. Default value is 4.

PROLONGED-SEGMENT-DECAY LEVEL

The fourth level of the new memory management algorithm implements the protection of segments from overlay by segments of lower priority jobs for an extended period of time after they were last accessed. Level Four was devised primarily for datacomm users who had no way of insuring that key segments of their handlers and other jobs remained in memory. As a result response time suffered when low priority background jobs caused datacomm segments to be overlayed. It was not advisable to permit users to mark their segments as "save", however, the system discussed below is only a little short of that.

There are two aspects to protecting key segments. First, those segments which are to be protected for a prolonged period must be marked. The mechanism for accomplishing this task is a normal state program called "SYSTEM/MARK.SEGS". Below is a sample deck for marking three segments of the program "CANDE/HANDLER" as important and one segment which had previously been marked important as unimportant.

```
?EX SYSTEM/MARK.SEGS;
?FILE CODE NAME CANDE/HANDLER;
?FILE NEWCODE NAME CANDE/HND.MARKED;
?DATA CARDS
(0,1)  1    % MARK PAGE 0 SEGMENT 1 AS IMPORTANT
   (0,3) 1      (1,7)1
  (0,5) 0    % MARK PAGE 0 SEGMENT 5 AS UNIMPORTANT
?END
```

Note 1: For one level segment dictionaries only the segment number would be specified.
Note 2: A free format is used in specifying segments as important or unimportant.

"SYSTEM/MARK.SEGS" copies CANDE/HANDLER to CANDE/HND.MARKED, updates the disk copy of the segment dictionaries of CANDE/HND.MARKED, and produces a listing of all of the segments of that program as they are currently marked. If no data cards are contained in the input file "CARDS", then a listing of the current status of segments of the object program is produced.

The second requirement for protecting key segments is to specify how long to retain them. This is done by setting the program attribute SECONDS.BEFORE.DECAY (abbreviated SB) to a value from 0 to 600. The priority of segments which have been marked as important will not be degraded until and unless those segments are not accessed for SB seconds. It should be noted that if SB = 0 for a particular job then both segments marked as important and unimportant will be treated the same. Furthermore, SECONDS.BEFORE.DECAY is completely subserviant to memory priority. A segment with a higher memory priority can overlay a segment with a lower memory priority no matter what the lower priority segment's SECONDS.BEFORE.DECAY is set to. SB simply determines how long after a segment was last accessed it will be able to retain a given priority.

In order to facilitate the use of SB where all segments of a particular program are to be considered important (i.e., a datacomm handler), SB should be set to a non-zero value and none of its segments should be marked as important. Under these conditions, all of the program's segments are protected from decay of its memory priority for SB seconds after it was last accessed.

Advantages

The advantages of Level Four are that:

1. Datacomm users will be able to guarantee that key segments of their handlers and other programs will not be overlayed by lower priority jobs for any fixed period of time between 0 and 600 seconds after they are last accessed.

2. Uses the same amount of memory and runs at the same speed as Level Three.

Disadvantages

The disadvantage of Level Four is that users of extended segment decay can lock up more memory than they really need and thereby degrade the performance of background jobs more than necessary.

SPO Messages

SECONDS.BEFORE.DECAY A new program attribute, abbreviated as SB, which can be used with "MO" or "DY" or in an "EXECUTE" statement to set the seconds before decay for a program which has some segments marked as important. The value must be GEQ 0 and LEQ 600.

Examples:

```
MO <job#> SB 10;
DY <job#> SB 15;
EX CANDE/HANDLER SB 5;
```

BACKGROUND FOR MEMORY MANAGEMENT CONCEPTS

Memory Fragmentation

This is the failure to allocate all of memory for useful purposes. Two varieties of fragmentation, internal and external, occur depending on whether a system uses a paging or segmentation mechanism.

Paging vs. Segmentation:

In a paging system all of memory is divided into equal sized pages. Therefore 100% of memory is assigned to a usable page and there is 0% external fragmentation. However, since memory requests are typically of varying sizes, the last page required to fulfill a memory request is usually not full. This is internal fragmentation.

In a system based on segmentation, segment sizes are variable so that only enough memory to satisfy a request is allocated. Therefore, no internal fragmentation exists in a segmented memory management system. However, some memory is required for a memory link to describe each segment. This is something less than 25 bytes per segment. A more serious problem is that areas of memory too small for anyone to use become available between two pieces of memory which are being used. This is external fragmentation.

Neither paging or segmentation is clearly superior to the other. Each has advantages and disadvantages. The primary advantage of paging is that it is straightforward from a memory management point of view. Segmentation, on the other hand, causes "geography" problems because external fragmentation checkerboards memory. However, segmentation provides a much more reasonable structuring of memory since only the space logically required for a given function is provided. Therefore, programmers need not be concerned with trying to structure their memory requirements into mod <page size> requests. And the less unnecessary details programmers have to think about the more quickly and accurately they can complete their actual tasks.

Burroughs has traditionally opted to use segmentation in its approach to memory management and the B1800/B1700 are no exception to this rule. Therefore, prioritized memory management is concerned with memory management schemes based on segmentation.

WORKING SET

This term refers to the set of all memory pages or segments which are accessed during a specific time interval. Using Denning's terminology the "working set" at time T, W(T, t), is the set of all segments accessed in the time interval from T - t to T. The working set for a specific job then is the set of segments which that job can use during the last time interval. The working set for the entire system is simply the sum of the working sets of all currently active jobs. The working set for a program and especially for the system as a whole can and and often does change over successive time intervals as jobs go from one phase to another.

THRASHING

This is the condition which exists when the "working set" of the system does not fit in real memory. In order to bring in the next segment the system has to overlay a currently active segment. Then that segment has to be brought back in and another active segment must be overlayed, etc. System performance suffers tremendously when "thrashing" occurs. Throughput degradation of 100% is not unusual. In fact, in the worst case absolutely nothing except overlays gets done.

"GETSPACE" ALGORITHM PRIOR TO 7.0

Description

In order to understand the new memory management system and under what circumstances it might be useful, some familarity with the old "GETSPACE" algorithm is required. The old algorithm is basically a round robin or first-in first-out memory management scheme. Overlayable memory is allocated starting from a "left-off pointer" which is then updated to point to the next lower segment in memory. Thus, the left-off pointer sweeps from high to low memory addresses until it hits the first link, at which time it starts from the last link again.

Save memory, which cannot be reassigned until it is explicitly forgotten, is allocated at the high end of memory so that it will tend to be pushed together, thereby cutting down the external fragmentation which save memory inherently creates.

Advantages

1. External fragmentation of memory is minimized since small available chunks of memory tend to be swept up and used as the left-off pointer sweeps through memory.

2. Although a simplistic decision about who to deallocate is made, this decision is made quickly. This is a very important feature. If enough memory is available to contain the working set of the currently active jobs then the first priority of the memory management system is to get that working set in as quickly as possible.

3. Current users are accustomed to the old algorithm and many are satisfied with it either because they have learned to avoid its pitfalls or because it suits their needs well.

Disadvantages

1. The most serious flaw of the old "GETSPACE" algorith is that there is no mechanism to detech thrashing. Users have no clean way of determining when they have overcommitted their memory. They instead have to rely on elapsed time to tell them after the fact that their

system was performing poorly and even then thrashing is just one of several possible causes.

2. The relative activity of a segment or whether or not it is currently in use is not considered or even known when deciding to overlay that segment. Therefore, segments which are no longer in use may be retained while active segments are overlayed.

3. The priority of a job using a segment is not considered when deciding to overlay that segment. Therefore, segments of high priority jobs are not protected from being overlayed by segments of lower priority jobs. High priority datacomm jobs are a prime example of jobs which sometimes suffer because their segments are not protected from background jobs (notably compiles and LOAD.DUMPs).

SPO Messages

There are no special SPO messages associated with the old "GETSPACE" algorithm.

MEASUREMENT TOOLS (VII.0)

The VII.0 release of MCPII provides measurement tools through a facility called Console Lamps and through additional information in the System Log and in displays on the SPO. The debug version of GISMO turns the console lights into "lamps" by using the lights to represent various system activities. Through these console lamps, it is possible to determine when a selected disk unit is busy or when the MCP is performing a code overlay for a selected program specified by job number.

Through the lamps, users may monitor:

- I/O activity by channel or unit on a given channel
- Subsets of CPU activity that include:
  - Idle time
  - MMCP and scheduler time
  - SMCP time

- User time (by job number or total user time)

- Overlay activity of:

  - SMCP interpreter segments, SMCP segments, and MMCP segments

  - User code segments and user interpreter segments

  - User data

In terms of lamps, the B1700 and B1800 have the same console. The console contains 24 main exchange lights, 24 switches, and the state light. Users make monitor requests through the console switches on either console. However, the two processors do not have the same capabilities for displaying information through the lights. The B1700 allows only the state light to be controlled by the software. For this implementation, there is only one lamp. If any of the requested activities is in process, then the light is on. Otherwise, it is off.

The B1800 has two additional capabilities that may be used to enhance the display. The processor allows software control of the 24 exchange lights and it contains a hardware timer that can be used to accumulate a reasonably accurate account of the time involved in the activity. In this implementation, the 24 exchange lights are divided into two groups of lamps. The left 16 lamps are used as a "bar graph", and the right six are used as a "fixed" display of various activities. The remaining two are not used.

INVOKING THE LAMPS

The lamps are made active under the following conditions:

- A 7.0 debug GISMO dated 8/1/77 or later must be used.

- Appropriate SPO options must be set.

- Appropriate console switches must be set.

SPO OPTIONS

The implementation for the lamps adds six discardable segments to GISMO. Four system options control which of these segments are kept at Clear/Start time, and thus which of the lamp features may be used. The following two options are available to either processor. They control the segments which use only the state light for a lamp. These options are:

VLIO Allows specification and display of activity on the IO subsystem.

VLCP Displays CPU usage and overlay activity.

Setting of either of these options does not, in itself, cause any activities to be displayed; they only control which segments are to kept at Clear/Start time.

The following options are used only for the B1800 to control the main exchange lights:

BRGR The activities displayed on this graph are the same, at any given moment, as the ones being shown on the state lamp. Setting this option does not, by itself, cause anything to be shown; the console switches are used in the same manner described for the state lamp. This code is a "subsidiary" of the state lamp options; if both VLIO and VLCP are reset, this option is also considered reset.

FLMP When this option is set, the discardable segments of GISMO used in console-lamp displays are kept at Clear/Start time. Since there are no further options in this lamp group, keeping the code at Clear/Start time is all that is necessary to cause the display described later.

The following caveats should be noted about these options. If a system has been Clear/Started with the debug GISMO and any of the options have been set, the main exchange lights behave quite differently from normal. This behavior may fool the operator into believing that the system has broken down, but that, ordinarily is not the case. Another problem related to the options occurs if any of the options is set and the system is Clear/Started with a non-debug GISMO. The initializer will look for the segments requested by the options, will not find them, and will halt the system with @00F083@ in the L register. Though the halt is an encumbrance, it may be pushed through in order to Clear/Start.

Specifying Activities to be Shown

Specifying an activity to be shown implies that whenever that activity is "in process", the STATE LAMP will be ON, and whenever that activity is not "in process", the STATE LAMP will be OFF.

Given that the appropriate discardable segments have been kept at Clear/Start time, the specifications for what is to be shown may be changed at any time. If the discardable segments supporting the activity specified have not been kept, specifications from the switches will be treated as "null", and no activities will be shown.

Activities are selected on the console switches from one of three groups: IO, CPU and Overlay. In order to make these selections, the switches must first be "enabled" for GISMO; this is done with the SPO message "SE4" (Switch Enable 4 (GISMO)). The switches then have these meanings (numbered left to right, 0 thru 23):

0 and 1	Are always the B1820 scale factor (see B1820, below).
2	When set, flags specifications for IO activity, and 3 through 23 are interpreted as discussed below, IO ACTIVITY.
3	When 2 is reset and 3 is set, the specificattons are for CPU usage and 4 through 23 are interpreted as discussed below, CPU USAGE.
4	When 2 and 3 are reset and 4 is set, the specifications are for overlay activity, and 5 through 23 are interpreted as discussed below, OVERLAY ACTIVITY.
	When 2, 3, and 4 are reset, the specifications are treated as "null" and no activities are shown (the STATE LAMP is always OFF).

Summary of Switch Specifications

```
       |---------|---------|---------|---------|---------|---------|
       | 0 1 2 3 | 4 5 6 7 |8 9 0 1 | 2 3 4 5 | 6 7 8 9 | 0 1 2 3|
       |---------|---------|-----------------------------------------|
IO     | S-S 1 C | CHANNEL |  UNITS (left to right, 0 thru 15)       |
       |---------|---------|-----------------------------------------|
CPU    | S-S 0 1 | U M I S |  USER NUMBER (right justified)          |
       |---------|---------|-----------------------------------------|
OLAY   | S-S 0 0 | 1 S U U |  USER NUMBER (left justified)           |
       |         |   O P D |                                         |
       |---------|---------|-----------------------------------------|
```

S-S	B1820 scale factor
C	IO control activity
CHANNEL	Channel number, 0 through 14 (hex E)
U	User CPU usage
I	Idle CPU
M	Micro MCP and Micro Scheduler CPU usage
S	SMCP CPU usage
SO	Overlay of SMCP S-code and interp segs, MMCP segs
UP	Overlay of User S-code and interp segs
UD	Overlay of User data

IO ACTIVITY

```
|-------------------------------------------------------------|
| 0 1 2 3 | 4 5 6 7 | 8 9 0 1 | 2 3 4 5 | 6 7 8 9 | 0 1 2 3 |
|---------|---------|---------|---------|---------|---------|
| S-S 1 C | CHANNEL |  UNITS (left to right, 0 through 15    |
|-------------------------------------------------------------|
```

Here it is specified that the STATE LAMP is to be ON when there is IO in process on the selected channel. The C bit (Control activity) and the Unit bits (Unit activity) have meaning dependent on the device bit as discussed below.

For channels other than DISK and MAG TAPE, the unit bits are ignored. Selecting the channel and setting the C bit will cause all activity through the channel to be shown.

For MAG TAPE, selecting the channel and setting the C bit will show all activity through the channel, regardless of which unit is involved. If the C bit is reset, then activity only for the unit(s) selected in the unit mask will be shown.

For DISK which does not have overlapped seeks (HPT, FLOPPY), the specification is as for MAG TAPE.

For DISK that does have overlapped seeks (CARTRIDGE, PACK), it is possible that the channel will be "idle" when there is IO "in process"; that is, when a unit(s) is seeking. The C bit, when set, specifies that the STATE LAMP is to be on only when the channel is busy (will typically exclude seek time). When a unit(s) is selected in the unit mask, the STATE LAMP will be on whenever there is an IO "in process" for the unit(s), including the time the unit is seeking.

For exchanges, if the primary (lowest) channel is selected, all activity to the selected units will be shown, regardless of what channel in the exchange actually does the IO. The C bit will cause activity only on the selected channel to be shown. If a secondary channel is selected, then the unit mask is ignored (and no unit activity shown); the C bit will show activity through the selected channel.

Note: TEST operations (including TEST&WAITs) and PAUSE operations are never shown.

CPU USAGE

```
|-----------------------------------------------------------|
| 0 1 2 3 | 4 5 6 7 | 8 9 0 1 | 2 3 4 5 | 6 7 8 9 | 0 1 2 3 |
|---------|---------|---------|---------|---------|---------|
| S-S 0 1 | U M I S |  USER NUMBER (right justified)        |
|-----------------------------------------------------------|
```

In this group are specifications that the STATE LAMP is to be ON during the execution on the CPU of:

- USER code
- Micro MCP and Micro Scheduler code
- IDLE CPU
- SMCP code

Any combination of the four may be specified; if all four are specified together, the STATE LAMP is always ON. If the USER bit is set, then if the user number field is zero, then the STATE LAMP is ON when any USER is running; if the user number field is non-zero, then the STATE LAMP is ON, during user time, only when the selected user is running. For example, if IDLE CPU and USER are selected, and the user number field is zero, then the STATE LAMP will be on when any user is running or the CPU is IDLE; if the user number field is non-zero, then the STATE LAMP will be on when the selected user is running or the CPU is IDLE.

Note: SOFTIO time (in GISMO) is not isolated and will appear as CPU time used by whoever was running when SOFTIO was invoked.

OVERLAY ACTIVITY

```
|-----------------------------------------------------------|
| 0 1 2 3 | 4 5 6 7 | 8 9 0 1 | 2 3 4 5 | 6 7 8 9 | 0 1 2 3 |
|---------|---------|---------|---------|---------|---------|
| S-S 0 0 | 1 S U U |  USER NUMBER (right justified)          |
|         |   O P C |                                         |
|-----------------------------------------------------------|
```

This group specifies that the STATE LAMP is to be ON when overlay is "in process"; that is, from the time that the overlay request is recognized by the SMCP until the overlay is complete (including any disk operations) and the requestor returned to the READYQ.

There are three types of overlay which may be shown:

- MCP overlays, including SMCP S-code overlay, SDL Interpreter segments requested for the SMCP, and Micro MCP overlays.

- User code overlays, including user S-code and Interpreter segments requested for a user.

- User data overlays.

When user code or user data overlays are requested, and the user number field is zero, then all user code and/or user data overlays are shown; if the user number field is non-zero, then user code and/or user data overlays are shown only for the selected job. For example, if MCP overlays and user code overlays are selected and the user number field is zero, then the STATE LAMP will be on when MCP overlay or any user code overlay is in process; if the user number field is non-zero then the STATE LAMP will be on when MCP overlay or user code overlay for the selected user is in process.

<u>BAR GRAPH</u>

Again, the activities that are currently being shown on the STATE LAMP are the ones that are displayed on the Bar Graph. The hardware timer is used to accumulate the time involved in the selected activities. At each internal N (its duration is discussed below), this time is converted to a percentage of N, and shown on the Bar Graph. Each lamp in the Bar Graph group represents 1/16th of the interval N (left to right, 0 to 100%), with these two restrictions:

- If the activity(s) occurred at all in the interval N, regardless of how little time it accumulated, at least the leftmost lamp will be on for the succeeding interval N.

- If the activity(s) did not occupy 100% of the interval N, the rightmost lamp will not be on for the succeeding interval N.

BAR GRAPH SCALE FACTOR (INTERVAL N)

The interval N acquires its duration from the SCALE FACTOR field in the console switches (leftmost two switches) according to the following table:

Switches	Interval N	Each lamp represents
00	400 milliseconds	25 milliseconds
01	800 milliseconds	50 milliseconds
10	1600 milliseconds	100 milliseconds
11	3200 milliseconds	200 milliseconds

For an example of what might be seen on the Bar Graph, assume that some particular user's CPU Usage is being shown on the STATE LAMP. Further assume that the SCALE FACTOR Switches are set at 01 (800 milliseconds).

Every 800 milliseconds the 16 lamps in the Bar Graph group of the console lamps will be changed to show what percentage of the just-ended 800 millisecond interval that user was executing on the CPU.

If the 24 lamps are (where 1=ON, 0=OFF, X=does not apply):

1111 0000 0000 0000 XXXX XXXX

then the user was executing on the CPU 25% of the previous 800 milliseconds (that is, 200 milliseconds).

If the lamps are:

    1100 0000 0000 0000 XXXX XXXX

then the user was executing on the CPU 12.5% of the previous 800 milliseconds (that is, 100 milliseconds).

As another example, assume that activity on DPA and DPB are being shown on the STATE LAMP, and that the SCALE FACTOR is at 00 (400 milliseconds).

If the lamps are:

    1111 1111 1111 0000 XXXX XXXX

then there was IO in process on either DPA or DPB 75% of the previous 400 milliseconds (that is, 300 milliseconds).

FIXED LAMP DISPLAY

The rightmost six console lamps form the "fixed" display group. On these lamps six activities are shown exactly as they would have appeared, if selected, on the STATE LAMP. This group is completely independent of whatever is being shown (if anything) on the STATE LAMP and Bar Graph. There are no optional displays; setting the appropriate MCP option and Clear/Starting (thus causing the necessary segments to be kept) is sufficient to "turn on" this display. None of the other three options need be set.

The six activities displayed are (lamps numbered left to right, 0 through 23);

23 All user CPU Usage. The same as is shown on the STATE LAMP when user CPU usage is selected and the user number field is zero.

22 Micro MCP and Micro Scheduler.

21 IDLE CPU.

20 SMCP CPU Usage.

19 Any overlay in process. The same as is shown on the STATE

LAMP when all three of the overlay options are selected and the user number field is zero.

18 Any disk IO in process. Has no equivalence on the STATE LAMP. Here activity is shown for all disk channels and units on the system, whereas on the STATE LAMP only one channel may be shown at a time.

Lamps 20 through 23 are mutually exclusive. No two will be ON at the same time, and together they make up 100% of the CPU usage.

JOB NUMBERS (VII.0)

Mix numbers have been replaced by job numbers. From schedule time to end-of-job, any reference to a job must be through one number--the job number. Job numbers begin with 0001 and continue through 9999, at which time the MCP will recycle back to 0001. A new number, Job Accounting Number, has been added to aid TABS and other log-handling programs. The Job Accounting Number can only be reset by COLDSTARTing. Previously, if a job called a sort, the sort came into the mix as the same mix number. If that job was DS-ed, all jobs with that mix number were DS-ed. By definition, job numbers must be unique, so sorts can no longer be hidden as they once were. Now, if a job calls a sort, the sort comes into the mix with its own job number and it is identified by the following message:

```
SORT/VSORT=38 BOJ. (#37) PR=4 TIME=18:38:17.6
```

where (#37) identifies the job number of the calling program.

Its corresponding end message is:

```
SORT/VSORT=38 EOJ. (#37) TIME=18:55:13.0
```

If, while the sort was running, either #37 or #38 were DS-ed, both would be terminated.

All occurrences of job numbers in the MCP have been cut in size to 16 bits or 4 characters, depending on its use.

DISK ALLOCATION (VII.0)

In order to allow users to selectively allocate their disk storage, a new file attribute is being implemented in the VII.0 MCP. This attribute is used only by the DISK/ALLOCATOR utility program, which is discussed in the UTILITIES section below. The attribute allows the utility routine to allocate areas of user disk to specified files by disk address.

SAVING S-MEMORY (VII.0)

Users can now mark selected memory locations as unusable through the "XM" SPO message. This feature is implemented as a function of the SYSTEM/INIT program and takes effect during Clear/Start. XM, without parameters, lists all entries in the XM Table and reports whether or not each entry is currently active, will be active upon the next Clear/Start, or could not be activated at the last Clear/Start. Any temporary XM entered through the toggles at Clear/Start is also listed, along with its temporary status.

At the transition from Tape to Run mode, toggles 13 and 15 in the T register, respectively, may be used to override the XM Table or to cause a second halt so that more entries may be added. When toggle 15 is set in T, more entries may be added through:

X = TEMPORARY XM LOCATION (in addition to table)
Y = TEMPORARY XM LENGTH

The XM Table provides a complete list of all areas of memory which the initializer will attempt to remove from use at the next Clear/Start. Note, however, that the XM Table may be incomplete in describing current memory since deletions from the table are merely removed from the table. In the current design of this function, no attempt is made to prevent overlapping or duplicate entries. The advantage of this structure is that the users can reverse any entries they have made by entering the same entry preceded by a "-", e.g., XM- <previous entry>. Otherwise, users wishing to reverse entries would have to consider carefully how to return the table to its prior state.

The XM table, maintained in sorted order from low-to-high address, may contain a maximum of 28 entries. One entry is required as a terminator to the serial list of entries, and the last entry in the table is reserved for a temporary XM entry if it was entered at Clear/Start time.

The initializer has the responsibility of resolving XM Table entries which describe overlapping space and for maintaining the current status of all the entries in the table. The disk address of the table is contained in the disk cold start variables.

PROGRAM CONTROL ATTRIBUTES

Program Control Attributes are used to inquire about the attributes of a compiled program (object code file) or to control the execution or compilation of such programs. The syntax of such requests, including that for FILE statements which are part of compilations and executions, is given below. Some attributes, such as COMPILE, are described here only briefly since they are defined more fully in another section of this document.

Unless otherwise noted, each command is presumed to apply to both the library copy and the working copy of the program, as the syntax in figure 6.1 shows. If individual messages are restricted, e.g., LEVEL is only permitted with inquiry (QF or QP), the restriction(s) are noted. Except for SPO input, semicolons are required as terminators.

Each command, unless defined otherwise, is allowed with the syntax:

```
>------------------ EX ---------- <program-name> ------->>
>-- ? -->| | |----- CO ---->|                          |
           | |----- MO ---->|                          |
           | |----- QF ---->|                          |
           |                                           |
           |------- QP ---------- <job-number> ----->|
             |----- DY ---->|


     |<--------------------- ; --------------|
   >>-- <attribute> ------------------------------------------------>#
                    |------------- <value> -->|  |-- ; --->|
                        |-- = -->|
```

Figure 6.1 Program Control Attribute Syntax

Attributes in figure 6.1 fall into two classes, depending on which copy of the program the user wishes to modify: the library copy or the working copy. After a code file has been compiled to library, its attributes can be queried and/or permanently modified. Executing programs, i.e., working copies, can be queried or modified also. Any changes made to this copy of the program, however, affect only the working copy of the program. Modifications can be made to non-library code files in a compile-and-go situation through the OBJ (object) attribute. These modifications also affect a single execution of a program only.

The execution of a program causes the MCP to create a working copy of the library file which can also be queried or modified, but only through the job number under which it is executing. Any changes to this copy of the program do not affect the library copy.

Valid program-control commands are developed from following the path of the arrow from its beginning (>) to its termination (#). In general, those commands which involve inquiry (QF, QP) do not require a control attribute value because the operating system will inform the user as to what the value is and those which involve modification, execution, or compilation ordinarily require a control-attribute value to be supplied, although there are some exception to this, as noted below. The attributes listed below are given in alphabetical order and any restrictions, in terms of the syntax in Figure 6.1, are noted. Those which are explained more fully in the SPO MESSAGES section of this document are only briefly defined in this section.

ABBR	PROGRAM ATTRIBUTE	FUNCTION
AT	ATTRIBUTES	Queries the interpreter requirements of a compiled program (object code file). See ATTRIBUTES in EXPANDED DEFINITIONS.
CG	CHARGE	Assigns a charge number into the log record for a program. CHARGE must be followed by an integer of seven digits or less (leading zeros assumed). If the CHRG option is set in the MCP, the CHARGE statement must be used before the program will be scheduled. The CHARGE number is entered in the MCP log file and is available for subsequent analysis.
DS	DYNAMIC.SPACES	Assigns the maximum number of overlays that will ever be present in a program's dynamic memory. The command reserves space in dynamic memory for memory links that will be associated with the overlayable data within a program. At run time, the MCP assigns a value of 10 to DYNAMIC.SPACES if the value is zero.
FI	FILE	Interrogates the file attributes of a compiled program or overrides these compiled attributes at run time. See FILE in EXPANDED DEFINITIONS.
FR	FREEZE	Prohibits rolling a program out to disk at any time during its execution, thereby remaining in the same memory location, regardless, until EOJ.

ABBR	PROGRAM ATTRIBUTE	FUNCTION
ID	INTRINSIC.DIRECTORY	Used to specify the name of the pack on which the program's intrinsics are located. That information is used by the MCP. INTRINSIC.DIRECTORY must then be followed by the <disk-pack-id> of the disk on which the referenced intrinsic files reside.
IN	INTERPRETER	Assigns a specified interpreter for use by a program. The attribute is followed by the <file-identifier> of the interpreter which the operator wishes to use. Also abbreviated INTERP.
IT	INTRINSIC.NAME	Assigns another multifile-id to the intrinsic file requested by a program. Note that the file-name portion of the intrinsic-file-id may not be changed--it must remain as "AGGREGATE".
LE	LEVEL	Reports on the level of the compiler used to compile the program. This option can be used for queries only. See LEVEL in EXPANDED DEFINITIONS.
ME	MEMORY	Assigns a non-default dynamic memory size to a given program. MEMORY is followed by an <integer> which describes, in bits, the amount of memory requested. When MEMORY is used with COMPILE statements, the dynamic memory requested is reserved for the compiler, not the program compiled. This attribute cannot be modified dynamically (DY) in the working copy of the program.

ABBR	PROGRAM ATTRIBUTE	FUNCTION
MP	MEMORY.PRIORITY	Assigns the priority with which segments of an executing program are overlaid in memory. Lower-priority jobs are overlaid before ones with higher priorities. See also PR below and MP in the SPO MESSAGES section of this document.

When a MEMORY.PRIORITY of nine or greater is specified, the following action occurs in multiprogramming environment:

a. If necessary, jobs which are running and which have a lower priority are "rolled out" of memory to disk in order to create space for the higher-priority jobs. This action is called "crashout".

b. A high-priority job entered in the schedule will not automatically suspend any other high-priority job running in memory. However, the system operator may stop (ST) them.

c. Upon termination of the high-priority job, the suspended programs will automatically be reinstated to memory.

d. Maximum priorities for remote jobs that run under the security mechanism are determined by the maximum priority established for the particular usercode in the (SYSTEM)/USERCODE file. Attempts to ignore that priority cause the operating system to display an error message at the remote terminal.

ABBR	PROGRAM ATTRIBUTE	FUNCTION
MS	MEMORY.STATIC	Assigns a non-default static memory size to a program.
OV	OVERRIDE	Bypasses the compatibility check normally made between a program and its interpreter. Compatibility is checked by the MCP for the following: a. HARDWARE.TYPE (U, S, or M), depending on the processor. b. MCP.LEVEL c. GISMO.LEVEL d. COMPILER.LEVEL d. ARCHITECTURE (i.e., correct interpreter name and attributes.)
PA	PAD	Assigns new scratchpad settings in the program. See PAD in EXTENDED DEFINITIONS.
PP	PROCESSOR.PRIORITY	Specifies the priority with which programs in a multiprogramming environment are given access to the processor. The default value is four. See also PR below and PP in the SPO MESSAGES section of this document.
PR	PRIORITY	Specifies the operational priority (MP and PP) assigned to a given program in order to maximize output. When the VII.0 memory-management system is not invoked (MPRI=0), the operational priority specified is assigned to PROCESSOR.PRIORITY only. When the memory-management option is set

<u>ABBR</u>	<u>PROGRAM ATTRIBUTE</u>	<u>FUNCTION</u>
		(MPRI=1), PR sets both MP and PP to the same value. (Memory management is discussed in the ENHANCEMENTS section of this document). Priorities range in value from zero to fifteen (0-15), where zero is the lowest and fifteen is the highest. Jobs which are run at priorities of less than eight are governed by a mix-limit (see ML in SPO MESSAGES) which specifies the total number of active jobs (priority 8 or less) in a multiprogramming mix. When the mix is full, any new jobs must wait for an executing job to go to EOJ before it may begin processing. Jobs which users wish to run at the higher priorities (9-15) in a multiprogramming mix of jobs, e.g., a text-editing program which allows remote editing, are not governed by the restrictions of the mix limit. They may enter the schedule at any time and begin processing.
PT	PROTECT	Prevents accidental tampering with the executing program by causing the MCP to reject SPO commands that would impair its execution. The designated commands are: CL, DP, DS, QC, ST, and SW. If a protected program reaches an abnormal termination, the MCP unlocks it automatically so that it may be DS-ed or DP-ed. The PROTECT attribute may not be specified in a MODIFY or DYNAMIC control instruction. It is allowed only with EXECUTE and COMPILE statements. See also PROTECT in EXPANDED DEFINITIONS and LP in SPO

ABBR	PROGRAM ATTRIBUTE	FUNCTION
		MESSAGES.
SB	SECONDS.BEFORE.DECAY	Specifies the minimum time, in seconds, before the memory priority (MEMORY.PRIORITY) of a program's code segment will be downgraded to the next lower priority. See also SB in SPO COMMANDS and PRIORITIZED MEMORY MANAGEMENT . in the ENHANCEMENTS section of this document.
SC	SCHEDULE.PRIORITY	Assigns priorities to programs in the schedule (i.e., awaiting execution). The maximum number for this attribute is 14. Jobs in the active schedule having the same assigned priority are further discriminated by the actual time the jobs have been in the schedule. Note: Once the program has been placed in the schedule, the SP console message must be used to change the scheduled priority.
SW	SWITCH =	Allows the operator to set or reset any or all program switch(es) to the <value> specified. Ordinarily, switches are set to an integer value, but character values are permitted if the language in which the program is written permits a character value. Users must consult specific language manuals for the format of these respective declarations. If a particular switch is not referenced, then all (40 bits) are implied. This command requires an "=". After a program has gone to BOJ, any queries or changes to the program's switches in the working copy of the program must be done

ABBR	PROGRAM ATTRIBUTE	FUNCTION
		through the SPO COMMANDs "TS" and "SW", respectively.
TC	TRACE <integer>	Activates the tracing features of the interpreter, provided the conditions for a trace have been met. See also GT and NT in the SPO MESSAGES section of this document.
TI	TIME	Specifies the maximum allowable processor time a job may accumulate. When a program exceeds the maximum run time specified by this attribute, it is DS-ed after a message is displayed on the console CRT. The integer specifies minutes in processor time, not elapsed time.
UF	UNFREEZE	Removes the FREEZE option, thus permitting it to be rolled out to disk when it is in an interrupted state.
UV	UNOVERRIDE	Resets the compatibility-check bypass caused by OVERRIDE.
VI	VIRTUAL.DISK	Changes the number of disk segments assigned by the compiler for saving data overlays during execution. The default size is 1000 segments.

RUN-TIME ATTRIBUTES

The following attributes are also used in program control but only during a single execution or compilation. They are therefore specified as a separate class. Illegal uses of these attributes are reported on the SPO by the operating system.

```
>---------------- EX ---------- <program-name> --------->>
>-- ? -->| | |--- CO--->| |

     |<---------------------------------------------------|
  >>-- <condit'l-attr> ---------------------------------------->#
                         |----------- <attribute-value> --->|
                          |-- = -->|
```

Figure 6.2 Conditional Control Attribute Syntax

ABBR	ATTRIBUTE	FUNCTION
AF	AFTER <program-name>	Schedules a program (e.g., BETA) by program-name, not job-number, for execution or compilation after the termination of another program (e.g., ALPHA). When ALPHA reaches EOJ, BETA is placed in the active schedule for execution as soon as memory resources are available. If ALPHA was neither executing nor scheduled when BETA is scheduled, BETA remains in the waiting schedule until ALPHA is executed and reaches EOJ or until BETA is FS-ed by the system operator.
AN	AFTER.NUMBER <number>	Schedules a program (e.g., BETA) for execution or compilation after the termination of a particular job (e.g., 238). The value required, <number>, refers to the assigned job-number of an executing or scheduled program. When the job specified by job-number reaches EOJ, BETA is placed in the active schedule for execution as soon as

memory resources are available. If <number> was only scheduled when BETA is scheduled, BETA remains in the waiting schedule until the specified job (e.g., 238) is executed and reaches EOJ or until BETA is FS-ed by the system operator.

CA CONDITIONAL

Inhibits a program from being executed unless its predecessor successfully reaches EOJ. CONDITIONAL is the system default and is implied in every AFTER, AFTER.NUMBER, and THEN scheduling of jobs. This attribute does not require a <value>.

CTLDCK

A unique <attribute-value>, not a <control-attribute>, which marks the beginning of a pseudodeck. The CTLDCK-value must follow the DATA-attribute (?DATA) and becomes a <file-id> under DATA (see DATA below and also LD in SPO MESSAGES).

DA DATA <file-id>

Identifies to the program that an input file is coming from a card reader with an external file name of <file-id>. The <file-id> value must conform to B1800/B1700 file-naming conventions.

END

Marks the end of a datafile, ordinarily, a card file.

ENDCTL

A unique <attribute-value>, not a <control-attribute>, which marks the end of a pseudodeck. The ENDCTL-value must follow the END-attribute (?END) and becomes a special <attribute-value> for END that is used only with pseudodecks. See also LD in SPO MESSAGES.

HO	HOLD	Allows the system operator to place a program into the waiting schedule until it is forced (FS-ed) into the active schedule. This attribute may not be used with MODIFY or DYNAMIC control statements and does not require a <value>.
	OBJ	Modifies an object code file after compilation and before execution (e.g., in a COMPILE and GO situation). This attribute does not require a <value>.
	STREAM <file-id>	Identifies, by name, the beginning of a punched card file. See STREAM in EXPANDED DEFINITIONS.
	TERMINATE <file-id>	Identifies, by name, the end of a STREAM input file. See TERMINATE in EXPANDED DEFINITIONS.
TH	THEN <file-id>	Schedules one program in relation to another, in same fashion as AFTER. THEN and AFTER (or AFTER.NUMBER) are mutually exclusive attributes, i.e., both cannot be used in the same execute string. The <value> of <file-id> must not violate B1800/B1700 file-naming conventions.
UC	UNCONDITIONAL	Forces a program to be executed whether its predecessor successfully reached EOJ or not. UNCONDITIONAL must be specified; it is not a default condition. This attribute does not require a <value>.

EXPANDED DEFINITIONS

ATTRIBUTES

```
*****************************************************************
*                                                               *
*  ------------ QF ---- <program-name> --- ATTRIBUTES ---->#    *
*           | |       |                    --                   *
*  -- ? -->| |- QP ->|                                          *
*                                                               *
*****************************************************************
```

ATTRIBUTES makes it possible to query the compiler attributes of a program. The ATTRIBUTES statement may only appear after a QF or QP statement.

Compiler attributes is an 80-bit field in the Program Parameter Block (PPB). This field is used by the compiler to indicate that the compiled program requires special features in its interpreter. This may be a special feature which is not in the normal interpreter, i.e., the COBOL MICR interpreter. This feature might also be a new S-op which has been added since the last S-machine level change (see LEVEL) as in the SDL EXTENDED ARITHMETIC ops. Generally, each bit indicates a special feature.

At BOJ time, the MCP will insure that the interpreter has all the attributes required by the object program. If the object program and interpreter are not compatible, the MCP will display a message containing a bit string of the missing attributes.

The compatibility check may be skipped by using the OVERRIDE control attribute.

FILE

Format:

```
*****************************************************************
*                                                               *
*                     |<-------------  ,  ---------------|     *
*                     |                                  |     *
* >-- FILE -- <name> -- <attribute>-------------------------->| *
* ?-| --                            |------------ <value>--|    *
*                                   |-- = -->|                  *
*                                                               *
*****************************************************************
```

The FILE statement may be used to query or override various file attributes. Each element within the statement must be separated by at least one space, and the statement must be terminated by a semicolon or and end-of-text (ETX). If more than one card is required for a FILE statement, each of the continuation cards may have a question mark (?) is column one.

The FILE statement can be used with COMPILE, EXECUTE, MODIFY, DYNAMIC, QF, or QP statement. If the FILE statement follows a MODIFY statement, the MCP will modify the information in the program's File Parameter Block (FPB). If the FILE statement follows a COMPILE, EXECUTE or DYNAMIC statement, the MCP will modify the information in a working copy of the program's FPB so that the change will be in effect for only that run. If the FILE statement follows a QF or QP statement, the MCP will display the current setting of the FILE attributes listed.

The internal-file-id used in the FILE statement must refer to the name used in the program that opens the file. For example, if the external file-id is to be changed for this run only, the FILE statement would be as follows:

```
? EXECUTE <program-name>
? FILE <internal-file-id> NAME <file-id>;
```

ABBR	FILE ATTRIBUTE	FUNCTION
ADV	ADVERB [=] <number>	The implied open adverb. Number is a 12-bit value with the following meanings: 0 = INPUT 1 = OUTPUT 2 = NEW 3 = WITH.PUNCH 4 = WITH.PRINT 5 = REWIND or WITH.INTERPRET 6 = REVERSE or WITH.STACKERS 7 = OPEN.LOCK 8 = OPEN.LOCKOUT 9 = Reserved 10 = Reserved 11 = Reserved
ALL	ALLOCATE.AT.OPEN	All of the areas requested by this file will be allocated at the time the file is opened.
ANS	ANSI	Defines the tape label type as ANSI (for non-released MCPs only).
ARE	AREAS [=] <integer>	The number of areas assigned to the file.
ASC	ASCII	The recording mode of the file is ASCII.
ATP	AUTOPRINT	Allow the file to go to auto backup. See also PB in SPO MESSAGES.
BAC	BACKUP	The output of the file will be allowed to go to backup. This sets BACKUP.DISK and BACKUP.TAPE.
BDK	BACKUP.DISK	Allows the file to go to disk backup.
BTP	BACKUP.TAPE	Allows the file to go to tape backup.

ABBR	FILE ATTRIBUTE	FUNCTION
BCL	BCL	The recording mode of the file is BCL.
BIN	BINARY	The recording mode of the file is BINARY.
B.A	BLOCKS.AREA [=] <integer>	The number of blocks (physical records) to an area.
BUF	BUFFERS [=] <integer>	The number of buffers assigned to the file. The integer must be a positive number from 1 to 255.
CAS	CASSETTE	Identifies that the file to be processed is located on a cassette.
CPC	CARD.PUNCH	Identifies the device to which an output file is to be sent.
CPY	COPY	The entire File Parameter Block, except the internal file-id, will be copied to the receiving file's FPB and the internal file-id is not changed. The syntax for COPY is given below:

```
*******************************************************************
*                                                                 *
*  ------------ COPY --- <file-id> ---------------------->>       *
*          ! !         !          !          !                    *
*  -- ? ->! !-- CPY -->!          !--- FROM --->!                 *
*                                                                 *
*                                                                 *
*  >>------ JOB.NUMBER -- <job-number> -------------------->#     *
*     ! !-- JN ------->!              !                           *
*     ! !-- # -------->!              !                           *
*     !                               !                           *
*     !                               !                           *
*     !-- PROGRAM.NAME -- <program-name> !                        *
*       !-- PN -------->!                                         *
*******************************************************************
```

ABBR	FILE ATTRIBUTE	FUNCTION
CRD	CARD.READER	Identifies the device on which an input file is located.
CYL	CYLINDER.BOUNDARY	Each area of the disk file will start at the beginning of a CYLINDER when the file is directed to a disk pack or a disk cartridge. Not implemented yet.
DCG	DISK.CARTRIDGE	Identifies that the file to be processed is located on a disk cartridge.
DFL	DISK.FILE	Identifies that the file to be processed is located on disk.
DEF	DEFAULT	Overrides the record length and blocking factor declared for an input file. The input file's external label directs the MCP to use the record length and blocking factor that it finds in the file header for that <filename>. Note: if the input file is a disk file and the original input-file declaration was CARDS, DISK or DSK must be declared as the input hardware device; otherwise, the MCP is unable to locate the file. This is the only case in which a hardware declaration is also required.
DPC	DISK.PACK	Indicates that the file to be processed is located on a disk pack.
D.R	DELAYED.RANDOM	The file is to be accessed as DELAYED.RANDOM.
DRC	DATA.RECORDER.80	Indicates the device on which the input file is to be found or where to output the file.

ABBR	FILE ATTRIBUTE	FUNCTION
DRI	DRIVE [=] <integer>	The file will be directed to the DRIVE or EU specified by the integer. The drive must be a system disk. The integer must be a positive number from 0 to 15.
DSK	DISK	The input or output file is located on disk, regardless of the compiled program declaration.
EBC	EBCDIC	The recording mode of the file is EBCDIC.
EMT	EMULATOR.TAPE	States that the program will do all of its own I/Os on tape and will use the emulator-tape communicate.
EOP	EOP	Allows the ON END-OF-FILE branch to be taken if the END-OF-PAGE is sensed.
EU	EU [=] <integer>	Same as DRIVE.
EVN	EVEN	The file will use EVEN parity.
EXT	EXTEND	Allows I/O sequential (disk) files to be extended beyond end-of-file. When a sequential file is opened I/O EXTEND, the current record pointer is incremented on successive writes and writes are allowed past end-of-file. Without the EXTEND option, successive writes do not increment the current record pointer and writing past end-of-file is not allowed. If an attempt is made to write past end-of-file on an I/O sequential file which has been opened under the default method (NO EXTEND), a "DS" or "DP" condition is displayed in the SPO message: <file-name> DECLARED FILE SIZE EXCEEDED. If an attempt is made to open an

ABBR	FILE ATTRIBUTE	FUNCTION
		I/O sequential file NEW without also specifying EXTEND, a "DS" or "DP condition is displayed in the SPO message: <program-identifier> ATTEMPTED TO OPEN FILE IO SEQUENTIAL NEW WITHOUT EXTEND.
FTP	FILE.TYPE	An output disk file will be assigned the specified type when it is closed and has been entered in the disk directory.

```
******************************************************************
*                                                                *
*  FILE.TYPE -------------------- DATA ------------------->#     *
*  FTP ------>|  |--  =  ->| |--- CODE -------->|                *
*                          |-- INTERPRETER -->|                  *
*                          | |-- INTERP --->| |                  *
*                          | |-- INT ------>| |                  *
*                          |--- INTRINSIC --->|                  *
*                          |--- PSR.DECK ---->|                  *
*                                                                *
******************************************************************
```

ABBR	FILE ATTRIBUTE	FUNCTION
FMS	FORMS	The program will be suspended and the MCP will display a message for the operator to load special forms in the device (printer or punch) before the file is opened.
HAR	HARDWARE	A printer or punch file will be allowed to go to the hardware device assigned.
HDR	HEADER	First 50 characters of message read will be header information. If a file is opened with HEADERS, the program becomes an MCS.
INC	INCREMENT.DRIVE INCREMENT.EU	Each area of a disk file will start on the next disk drive (pack or cartridge) or EU (head-per-track). When the last system drive has been used, it will start over from drive

ABBR	FILE ATTRIBUTE	FUNCTION
		zero. Not yet implemented.
IMP	IMPLIED.OPEN	IMPLIED.OPEN denial set. (Also known as FPB.COBOL).
INP	INPUT	Implied open is INPUT.
INV	INVALID.CHARACTERS [=] <integer>	The <integer> may contain the value of 0, 1, 2 or 3, and determines the course of action for invalid character output to a train printer. 0 = Report all lines that contain invalid characters. The following message will be displayed for each occurrence: FILE <file-name> IS PRINTING INVALID CHARACTERS ON LP<x>. 1 = Report all lines that contain invalid characters and stop the program at that point. Not yet implemented. 2 = Report once that the file is printing invalid characters. The following message will be displayed: FILE <file-name> IS PRINTING INVALID CHARACTERS ON LP<x>. (ONE TIME WARNING) 3 = Do not notify operator of invalid character output.
ISL	INPUT.SELECTIVITY	Reads from remote files will return only messages from the station specified in the key portion of the read. Not yet implemented.

ABBR	FILE ATTRIBUTE	FUNCTION
LAB	LABEL.TYPE [=] <integer>	The integer value and associated label types are as follows: 0 = ANSI 1 = Unlabeled 2 = Burroughs
LOC	LOCK	The file will be LOCKED, if still open at program termination (DS or normal EOJ). This bit may be modified dynamically, if the user desires, regardless of the open status of the file.
MAX	MAXIMUM.BLOCK.SIZE [=] <integer>	Fixed block size to be used for variable length records.
	MCPDATA <number>	Used to pass the address of a list of files to be loaded or dumped to SYSTEM/LOAD.DUMP.
MUL	MULTI.PACK	The file will be considered a multipack file. MPF is also an acceptable abbreviation for MULTI.PACK.
NAM	NAME [=] <file-id>	The external pack-id and file-id. If only the disk pack-id is to be changed, the PACK.ID attribute may be used.
	NEW NEW.FILE	Implied open is NEW.
	NO <file-attribute> NOT	When this option is used, it will negate the file-attribute following the word NO or NOT. For example, a file assigned to go to a backup could be changed to go to the printer by entering a NO BACKUP file statement. The following is a list of file attributes that the NO or NOT statement will negate. AUTOPRINT OPEN.LOCK

ABBR	FILE ATTRIBUTE	FUNCTION
		ALLOCATE.AT.OPEN, BACKUP, BACKUP.DISK, BACKUP.TAPE, CYLINDER.BOUNDARY, DEFAULT, EOP, FORMS, HARDWARE, HEADER, INCREMENT.DRIVE, INCREMENT.EU, IMPLIED.OPEN, INPUT, INPUT.SELECTIVITY, LOCK, MULTI.PACK, NEW, OPEN.LOCKOUT, OPTIONAL, OUTPUT, QUEUE.OLD, REVERSE, REWIND, TRANSLATE, USER.BACKUP.NAME, VARIABLE, WITH.INTERPRET, WITH.PRINT, WITH.PUNCH, WITH.STACKERS, WORK.FILE
	NODIF	An option that allows jobs executed (spawned) from a remote terminal to continue processing even when the MCS goes down. The output is routed to some other location, e.g., SPO or printer. When set, this option cannot be modified.
NST	NUMBER.STATIONS	Maximum number of stations with which this file is to communicate.
	ODD	The file will use ODD parity.
OLK	OPEN.LOCK	Implied open is OPEN.LOCK.
OLO	OPEN.LOCKOUT	Implied open is OPEN.LOCKOUT.
OPT	OPTIONAL	Select OPTIONAL file.
OUT	OUTPUT	Implied open is OUTPUT.
PID	PACK.ID [=] disk-pack-id	The name of the disk pack on which the file(s) is located. The identifier can be reset with a null string (""), e.g., PID "".

ABBR	FILE ATTRIBUTE	FUNCTION
PTL	PROTOCOL	The means by which a remote file tells a MCS whether or not it can accept a specified range of message formats.
PTP	PAPER.TAPE.PUNCH	Specifies output to a paper-tape-punch device.
PTR	PAPER.TAPE.READER	Specifies input from a paper-tape-reader device.
PRT	PRINTER	The output file is to be sent to an available printer.
PSE	PSEUDO	Makes a file pseudo type.
PTN	PROTECTION DEFAULT [=] PUBLIC PRIVATE	If a NEW file is created, it will have the protection specified. DEFAULT for running with a usercode is set by SYSTEM/MAKEUSER. Without a usercode, the default is PUBLIC. See also the MH input command.
PIO	PROTECT.IO I.O [=] INPUT OUTPUT	If a NEW file is created, it will have the input/output protection specified. Default is I.O. See also the MH SPO command.
QFO	QUEUE.OLD	Identifies an old queue as opposed to a new queue.
QFS	Q.FAMILY.SIZE [=] <integer>	The number of families (sub-queues) in this queue (maximum = 255).
QMX	Q.MAX.MESSAGES [=] <integer>	The maximum depth to which the queue is allowed to grow before suspending the writer (maximum = 255).
QUE	QUEUE	Identifies the file as a queue file.

ABBR	FILE ATTRIBUTE	FUNCTION
R96	READER.96	Identifies input from a 96-column card reader.
RAN	RANDOM	Defines the access mode as random.
R.B	RECORDS.BLOCK [=] <integer>	The number of logical records per block for fixed-record-length files.
REE	REEL [=] <integer>	The value of the integer will determine the number of the first reel.
REM	REMOTE	Declares a file as a remote file.
REP	REPETITIONS [=] <integer>	Requests that <integer> copies of an individual printer backup be printed. The information is permanently stored as part of the file header but can be overridden (see PB in SPO MESSAGES). If the <integer> is greater than one, the output print file is automatically sent to backup disk or tape.
REV	REVERSE	Implied open is REVERSE.
REW	REWIND	Implied open is REWIND. Tape will not be rewound before being opened.
RPP	READER.PUNCH.PRINTER	Assigns an input or an output file to this hardware device.
RS2	READER.SORTER.2	Assigns an input file to this hardware device.
RSR	READER.SORTER	Assigns an input file to this hardware device.
RST	READER.SORTER.STATIONS	Declares the number of read heads on the reader-sorter. The allowable range is one to three (1-3).

ABBR	FILE ATTRIBUTE	FUNCTION
RSZ	RECORD.SIZE <integer>	Declares that the record length of an input or an output file is of size <integer> bytes.
SAV	SAVE [=] <integer>	Only used by programs which remove files according to last access date. The <integer> specified by save factor is added to the creation date to determine the last "relative" access date.
SER	SERIAL	File is to be accessed sequentially.
SNO	SERIAL.NUMBER	The volume serial number of the tape to be used. Not yet implemented.
TAP	TAPE	Declares an output file is to be written on this hardware device (most general type of reference).
TPN	TAPE.NRZ	Declares that an output file is to be written in Non-Return-to-Zero (NRZ) tape mode only.
TP7	TAPE.7	Declares an output file for 7-track tape only.
TP9	TAPE.9	Declares that an output file is to be written in 9-track tape mode only.
TPE	TAPE.PE	Declares that an output file is to be written in Phase-Encoded (PE) tape mode only.
TRN	TRANSLATE	Soft translation is to be performed on each record as it is being transferred to or from the user's buffer.
TNM	TRANSLATE.NAME [=] <file-id>	Associates a translate table file with a file to be translated.

ABBR	FILE ATTRIBUTE	FUNCTION
UNI	UNIT.NAME [=] <unit-mnemonic>	The file will be directed to the device specified by unit-mnemonic if opened output.
UNL	UNLABELED	There are to be no labels on this file.
U.N	USER.BACKUP.NAME	Overrides the default naming convention for backup print or punch files (e.g., BACKUP.PRT/<number>) and allows the user to name (U.N NAM <file-id>) the file. Also abbreviated as USER.NAME.
VAR	VARIABLE	The file to be processed contains variable-length records.
WIN	WITH.INTERPRET	Implied open is WITH.INTERPRET. Punched cards will be interpreted.
WPR	WITH.PRINT	Implied open is WITH.PRINT. Punched cards will be printed from buffer #2.
WPC	WITH.PUNCH	Implied open is WITH.PUNCH. Cards will be punched from buffer #1.
WST	WITH.STACKERS	Implied open is WITH.STACKERS. Cards will be stacker selected.
WFL	WORK.FILE	Assign a unique external file name based upon the combination of the name and the job-number. Allows multiple copies of the same program to run at the same time, with no "DUPLICATE-FILE" error condition existing.

LEVEL

Format:

```
*******************************************************************
*                                                                 *
*  ---------------- QF <program-name> ------- LEVEL ---->#        *
*            |    |                    |     --                   *
*  --- ? -->|    |-- QP <job-number> --->|                        *
*                                                                 *
*******************************************************************
```

The LEVEL attribute makes it possible to query the compiler level of a program. The LEVEL statement may appear only after a QF or QP statement.

The compiler LEVEL is an indication of what S-OPs generated by the compiler mean. If the meaning of an S-OP is changed or an S-OP is removed from the interpreter, the LEVEL is increased by 1 and the user is required to recompile his programs. The compiler LEVEL is not affected by the addition of a new S-OP as it is usually added as a new feature in the compiler ATTRIBUTES.

At BOJ the MCP will insure that the program and its interpreter are the same LEVEL. This check may be bypassed with the OVERRIDE control attribute.

OVERRIDE

Format:

```
*****************************************************************
*                                                               *
* --------------------- <program-name>----- OVERRIDE ---->#    *
*          |  |  |        |                 |   --              *
* -- ? ->|  |  |- MO ->|                 |                      *
*           |  |- OF ->|                 |                      *
*           |                            |                      *
*           |---- OP ---- <job-number> -->|                     *
*                                                               *
*****************************************************************
```

The OVERRIDE attribute makes it possible to bypass the compatibility check normally made between a program and its interpreter.

At BOJ time the MCP will perform a compatibility check of a program and its interpreter unless OVERRIDE is specified. The compatibility check consists of the following:

Interpreter HARDWARE TYPE is "U" (Universal) or matches the type of the processor it is running on.

Interpreter's MCP LEVEL matches the MCP's level.

Interpreter's GISMO LEVEL matches GISMO's level.

Interpreter's COMPILER LEVEL matches the program's LEVEL.

Interpreter's ARCHITECTURE (language) matches the program's INTERPRETER FIRST NAME.

Interpreter has at least every ATTRIBUTE required by the program.

OVERRIDE will not bypass the MCP's name generation process.

OVERRIDE may be reset with UNOVERRIDE (see UNOVERRIDE below), thus causing the compatibility check to be performed.

PAD

Format:

```
*********************************************************************
*                                                                   *
*                                                                   *
* ----------------------- <program-name>---------------->>          *
*         |  |  |         |                   |                     *
* -- ? ->|  |  |- MO ->|                     |                     *
*            |  |- QF ->|                     |                     *
*            |                                 |                     *
*            |---- QP ---- <job-number> -->|                        *
*                                                                   *
*                                                                   *
* >>--- PAD ----- <integer> --- A ---- <number> ----------->#       *
*        --                      |- B ->|                           *
*                                                                   *
*********************************************************************
```

The PAD attribute makes it possible to query or alter the program's initial scratch pad settings. The <integer> must be a positive number from 0 to 15 and specifies which scratch pad is to be accessed. The following letter (A or B) specifies which hemisphere is to be accessed. The <number> must be not more than 24 bits in length.

PROTECT

Format:

```
*****************************************************************
*                                                               *
* ----------------- QP ---------- <program-name> -- PT -->#     *
*              |    |     .       |                             *
* --- ? --->|       |- EXECUTE ->|                              *
*                      --                                       *
*                                                               *
*****************************************************************
```

The PROTECT attribute protects the job from certain input messages. The PROTECT control statement may appear only after an EXECUTE or QP statement.

The following input messages are disallowed if the PROTECT flag is set:

CL, DS, DP, GT, ST or SW.

These messages may be entered from the remote console which spawned the job if the PROTECT flag is set.

If the job goes to DS or DP, the PROTECT flag is automatically reset to allow the DS or DP. See also the LP input message.

STREAM

Format:

```
*******************************************************************
*                                                                 *
* ---- ?STREAM ------------- <file-identifier> ---------->#       *
*                                                                 *
*******************************************************************
```

The STREAM attribute informs the MCP to treat the sequence of cards that occurs between the STREAM and the TERMINATE as a data deck, regardless of the control instructions that actually occur on the cards. In this way, for example, compile and execute decks can be passed from a remote system to a host system for compilation or for execution. STREAM cannot be abbreviated.

A stream of data consists of all data cards contained between the stream beginning and the stream end, as delimited by the STREAM and TERMINATE control instructions. The <file-identifier> on the TERMINATE card must be the same as the <file-identifier> on the corresponding STREAM card in order to properly terminate the data stream; otherwise, the card reader will be left open.

STREAM and TERMINATE are somewhat analogous to DATA CTLDCK and ENDCTL used in pseudoreader loading, but are more generalized.

When reading a data stream, the exception branch is taken any time a card with a question mark (invalid character) occurs in column one of a card read. The MCP replaces column one of that card image with binary zeros (@00@), prior to passing the card image to the reading program. The program receives the EOF branch only when the proper terminate card has been read.

Example

```
?STREAM CARDS
?COMPILE TEST WITH COBOL LIBRARY
?CHARGE 123456
?FI CARDS NAME SOURCE/TEST DISK;
?EX TEST CHARGE 123456
?DA CARDIN
--- data cards ---
?END CARDIN
?TERMINATE CARDS
```

TERMINATE

Format:

```
*****************************************************************
*                                                               *
* --- ?TERMINATE ----------- <file-identifier> --------->|      *
*                                                               *
*****************************************************************
```

The TERMINATE attribute informs the MCP that the stream input file has reached an end-of-file (EOF). TERMINATE cannot be abbreviated.

The <file-identifier> specified on the terminate card must be the same as the one specified on the corresponding stream card in order to clear the card reader that has been assigned to the data stream. See STREAM above.

SPO MESSAGES

This section contains an alphabetized list of all the console printer or CRT (SPO) input messages implemented at the VII.0 release level of MCPII. Each message is syntaxed and explained, usually with examples included. Messages are indexed by the two-letter abbreviation that the user would normally use in a request to the MCP.

Messages are formatted for SPO application (CRT or TTY). If a message is to be entered through a card reader, it must be preceded by an invalid character (?) and terminated by a semicolon (;). Messages may be concatenated, as long as individual messages are delimited by semicolons. Maximum length for any message, or series of commands, is 160 characters, for SPO application.

Messages are evaluated, also, from the standpoint of the HOST/RJE application, since remote users are not allowed the full complement of SPO commands. Following the expansion of the two-letter command, and in the index, each message is designated as illegitimate, restricted, or legitimate where HOST/RJE usage is concerned. Restrictions are placed within parentheses. If the parentheses contain a dollar sign ($), the command is invalid from a remote terminal; if the character is a pound sign (#), the command must be preceded by a usercode and password to be legal; and if the command is followed by neither of these, it is legal as it is defined.

The syntax of each SPO message is shown in railroad syntax, which implies the following assumptions:

A valid MCP command is developed from following the main line of syntax from its beginning (>) to its termination point (#). Where optional entries are concerned, the direction of flow proceeds down on the left-hand line and up on the right. Unless otherwise noted, optional entries are expected by the program in the order (top to bottom) in which they are presented. When no return line (<----) is given, the user must choose one of the options presented.

Required entries occur on the main line of development; optional ones below the line. Required keywords appear in upper-case letters; variables in lower-case letters. Underlined portions of required keywords are acceptable abbreviations for the key words. Other abbreviations used are:

Variable	Abbreviation
identifier	id
multifile-identifier	mfid

INITIAL LETTERS A-E

AB (AUTOBACKUP) ($)

```
**********
*   AB   *
**********
```

Format:

```
*****************************************************************
*                                                               *
*  >-- AB ------------------------------------------------>#    *
*      --     |                                    |            *
*             |-- <integer> ------------------->|               *
*             |-- <device specifier list> ----->|               *
*                                                               *
*****************************************************************
```

```
<integer>                  ::=  0-4 number of SYSTEM/BACKUPs
                                (Maximum number = 4)

<device specifier list>    ::=  <unit> | -<unit> |
                                <unit> <device specifier list> |
                                -<unit> <device specifier list>

<unit>                     ::=  LPA | LPB | LPC (IOAT UNIT_NAME)
```

AUTOPRINT

The AUTO BACKUP mechanism retrieves backup disk files and prints them automatically, directing the output to designated printer(s). The function will not relieve the operator of the responsibility for matching jobs with their printer listings but the printing procedure will be automated. When AUTO BACKUP is invoked, a designated number of SYSTEM/BACKUPs will be executed. The names of "candidate" backup disk files are entered in a community queue file named "AUTOPRINT". Each SYSTEM/BACKUP in turn reads an entry in the queue file, maps the name on to an output printer file's FPB, and prints the file as in the case of a simple PB (FPB.REPETITIONS are honored). The backup files are removed after printing.

SYSTEM/BACKUPs remain in the mix as long as there are backup files to be printed and terminates when the AUTOPRINT queue is empty. A bit in the FPB, FPB.SYSTEM.BACKUP.BIT, gives SYSTEM/BACKUP special privileges in acquiring designated printers.

All backup disk files with an FPB.MULTI.FILE.ID equal to BACKUP.PRT (usercode for HOST/RJE) will be candidates for AUTO BACKUP. To bypass AUTO BACKUP, when the option is set, the user must specify a USER.BACKUP.NAME with an FPB.MULTI.FILE.ID not equal to BACKUP.PRT.

AUTO BACKUP is initiated by the MCP control string "AB". AB has three forms which are:

AB <integer> — allows <integer> number of copies of SYSTEM/BACKUP to be executed, if or when appropriate.

Note: "AB 0" purges the AUTOPRINT queue and causes the program to terminate.

AB — without parameters causes a display of the AB <integer> value and identification of the printers reserved for AUTO BACKUP.

AB <UNIT.NAME> — reserves printers for AUTO BACKUP. "AB -<UNIT.NAME>" resets the option.

NOTE: It is the operator's responsibility to match the number of printers reserved for AUTO BACKUP with the number of SYSTEM/BACKUPs specified by AB <integer>. It would not be desirable, for example, to specify a multiple number of SYSTEM/BACKUPs while reserving a single printer for AUTO BACKUP. Likewise, nothing is gained from reserving more printers than SYSTEM/BACKUPs specified.

FPB.AUTOPRINT

An FPB bit called FPB.AUTOPRINT has been added to allow avoidance of the AUTO BACKUP mechanism. If this bit is set (default is reset) the backup file created will not be retrieved by AUTO BACKUP.

The FPB.AUTOPRINT bit may be set/reset using file equation. NO ATP or NO AUTOPRINT sets the bit. Omitting NO resets the bit.

```
>------ FILE -- <filename> ----------------- AUTOPRINT -------->|
                           |          |  |                |
                           |-- NO -->|  |----- ATP -----|
```

Note: The AUTOPRINT bit may be OY-ed while the file is open.

AD (ADD) (#)

```
**********
*   AD   *
**********
```

Format:

```
*******************************************************************
*                                                                 *
* >--- ADD ----------------------------------------------------># *
*      --    |-- TO <disk-pack-id> -->|     |              |      *
*                --                         |              |      *
*                                           |              |      *
* |<----------------------------------------|              |      *
* |-- FROM <library-tape-id> ----- =/= ------------------|      *
*     ----                   |<-----------  ,  ---------->|      *
*                            |-- <mfid> ----------------->|      *
*                                         |-- /= ---------->|      *
*                                         |-- /<filename> ->|      *
*                                                                 *
*******************************************************************
```

The AD (ADD) statement will cause a file or files on a library tape to be placed on disk only if the file is not already on disk. The =/= option causes every file on the tape to be added. The <mfid>/= option causes every file with the specified <mfid> to be added.

Example:

CARD	SPO
?AD TO USERPACK FROM LIBTAPE PAYROLL/=, ACCPAY/=, MASTER/FILE;	AD TO USER FROM LIB =/= AD TO USER FROM LIB MASTER/FILE, PAYROLL/=
?AD FROM LIBRARY =/=;	

AX (Response to ACCEPT)

```
**********
*   AX   *
**********
```

Format:

```
*****************************************************
*                                                   *
*  >--- <job-number> --- AX <input message> ----->| *
*                        --                         *
*****************************************************
```

The AX message is a response to an ACCEPT message requested by an object program through the MCP.

All responses are assumed to be alphanumeric format. The input message starts in the first position after the AX on the input line, and continues until the END OF MESSAGE button is pressed.

If the End-of-Message is depressed immediately after the AX, the MCP fills the area in the requesting program with blanks.

Example:

```
632AX  CHECK  VOID  IF OVER 500 DOLLARS
```

Input messages shorter than the receiving field in the program will be padded with trailing blanks. Longer messages will be truncated on the right.

The AX message has an unsolicited console feature in that the operator may enter as many AX message responses as needed for a given program prior to the actual ACCEPT. The AX messages must be entered in the order they will be used, since the queue is structured on a first-in, first-out basis.

The queue is automatically cleared at program EOJ or an abort.

BB (Backup Blocks per Area) ($)

```
**********
*   BB   *
**********
```

Format:

```
**************************************
*                                    *
* >--  BB ----------------------># *
*      --   |                   |    *
*           |-- <integer> -->|       *
**************************************
```

The BB input message allows the operator to specify the number of blocks to assign each area of a printer or punch backup disk file.

The value of the backup blocks per area is set to 200 by COLDSTART, and if the <integer> entered in the BB message is less than 5, a value of 200 is assigned by default.

If an <integer> is not entered with the BB message, the MCP displays the current setting of the backup blocks per area. At OPEN time, the MCP will use this number or the blocks per area specified in the file declaration, whichever is larger.

Examples:

```
BB

BB 350
```

BD (Backup Designate) ($)

```
**********
*   BD   *
**********
```

Format:

```
*****************************************
*                                       *
* >-- BD ----------------------------># *
*     --    |                      |    *
*           |-- <disk-pack-id> -->|    *
*****************************************
```

The BD input message allows the operator to designate a specific disk pack or disk cartridge for backup files.

When the BD message is entered without the <disk-pack-id> the MCP will cause the current setting of the default backup disk to be displayed.

If a null string ("") is entered as a <disk-pack-id>, the default backup disk is changed to the SYSTEM disk.

Examples:

BD (Responses: PBD DESIGNATION = SYSTEM DIRECTORY)

BD USERPACK

BD ""

BF (Display Backup Files) (#)

```
**********
*   BF   *
**********
```

Format:

```
************************************************************
*                                                          *
* >-- BF ---------------------------- =/= ----------->#    *
*     --  |                         |  |- <integer> -->|   *
*         |-- <disk-pack-id> --|  |- PRT/= ------>|        *
*                                      |- PRN/= ------>|   *
*                                      |- PCH/= ------>|   *
*                                      |- DMP/= ------>|   *
************************************************************
```

The BF input message lists disk backup files and dumpfiles on the console printer. PRT/=, PCH/=, and DMP/= list all printer backup, punch backup, and dumpfiles on disk, respectively.

The =/= option lists all printer backup, punch backup and dumpfiles that are stored on system disk.

PRN and PRT are both to be assumed to mean printer backup files. That is, PRN and PRT are equivalent. The pack-identifier option designates backup files on a removable disk cartridge or pack which is specified by name (not by unit-id). Default specification is system disk, as mentioned above.

CC (Control Card)

```
***********
*   CC    *
***********
```

Format:

```
**************************************
*                                    *
* >-- CC ----------------------->#   *
*     --                             *
**************************************
```

The CC input message is solely for the purpose of those who are former B3500 users. On the B3500, all control strings had to be preceded with CC. The B1800/B1700 systems ignore this message.

CD (List Card Decks in Pseudo Readers)

```
***********
*   CD    *
***********
```

Format:

```
************************************
*                                  *
* >-- CD ----------------------->| *
*     --                           *
************************************
```

The CD input message allows the system operator to obtain a list of the pseudo card files and their file numbers that have been previously placed on disk by SYSTEM/LDCONTRL.

The MCP displays the number of each pseudo deck and the first fifty (50) characters of the first card in the deck.

If a deck is in use, its name and the program using it are displayed.

CH (CHANGE) (#)

```
**********
*   CH   *
**********
```

Format:

```
*****************************************************************
*                                                               *
*               |<------------------  ,  ------------|          *
*               |                                     |         *
*  >-- CHANGE ---- <file-id1> ---- TO --- <file-id2> -------->#  *
*      --                          --                           *
*                                                               *
*****************************************************************
```

The CH input message changes the file identifier of a disk file, causing the file to be referenced by the new file identifier.

Any CHANGE statement affecting more than one file must have the file identifiers separated by commas.

The CHANGE statement will cause the MCP to change the file identifier of specified disk files from one name to another. If the file referenced in the CHANGE statement resides on a removable disk, the diskpack-id must precede the file identifier in order for the MCP to locate the proper file to change. The CHANGE statement is not allowed on a multipack file.

Example:

CH may consist of additional cards (or lines on a SPO) for multiple changes. Termination will occur when a semicolon is detected. For example:

```
?CH A/B/C B/C/C, A/B/ C/D/, X Y, Z Q, ABC DEF;
```

If the CHANGE statement is entered and the MCP cannot locate the file or if the file is in use, the following message is displayed on the console printer (SPO):

--<file identifier> NOT CHANGED . . . (reason) . . .

Error conditions (e.g., <file-identifier> NOT CHANGED . . .) that the MCP encounters are reported whether the LIB(RARY) option is set or not; however, the LIB(RARY) option must be set before the results of CH are displayed.

CL (Clear Unit) (S)

```
***********
*   CL    *
***********
```

Format:

```
*****************************************
*                                       *
*  >-- CL ----- <unit-mnemonic> ----->| *
*      --                               *
*****************************************
```

The CL input message allows the operator to clear a unit on the system because of an apparent system software loop or hardware malfunction. Any program using the unit that has been cleared using the CL message will be discontinued (DS-ed).

The CL message cannot be used with disk devices (DCx, DKx, DPx).

Examples:

    CL LPA

CM (Change System Software) ($)

```
**********
*  CM    *
**********
```

Format:

```
****************************************************************
*                                                              *
* >-- CM -- <software-mnemonic> ---- <program-id> -------># *
*     --                          |                     |    *
*                                 |-- PURGE ----------->|    *
*                                     -----                    *
****************************************************************
```

The CM input message allows the operator to identify programs to the system for subsequent usage. The purpose of the CM message is to identify a file that contains the program to be used for a designated function as part of the system's software.

The resultant action of the CM message does not take affect until the next Clear/Start, except for Network Controllers and usercode files. For them, the effect is immediate.

The PURGE option removes the file from the designated NAME TABLE entry.

Refer to the Clear/Start procedure in the Software Operational Guide for a list of the system software mnemonics that are used in the NAME TABLE.

Example:

```
    Delayed                              Immediate
    -------                              ---------

  CM MX MCP/XYZ                      CM C CANDE/HANDLER
```

The CM message at the next Clear/Start makes the program MCP/XYZ the experimental MCP. CANDE/HANDLER becomes the current Network Controller, providing no system software controller is active.

CN (Change to Non-Trace System Software) ($)

(DEBUG Only)

```
**********
*   CN   *
**********
```

Format:

```
*************************************
*                                   *
* >--- CN ----------------------->#  *
*      --                           *
*************************************
```

The CN input message allows the system operator to change the operating environment to non-trace system software after the next Clear/Start (if the DBUG option is set).

CAUTION

The CN input message is "strictly" for system software development and debugging. It should not be used in the standard operating environment.

CO (COMPILE) (#)

```
**********
*   CO   *
**********
```

Format:

```
******************************************************************
*                                                                *
* >-- COMPILE -- <program-name> -------- <compiler-name> ---->#  *
*     --                        |-WITH-|                 |  |    *
*                                ----                    |  |    *
*   |<---------------------------------------------------|  |    *
*   |------------------------------------------------------>|    *
*     |------------ LIBRARY --->| |                         |    *
*     | |         |  --         | |<---------  ;  ----------|    *
*     | |-- TO ->|              | |                         |    *
*     |                         | |-- <program-control      |    *
*     |-- FOR SYNTAX ---------->| |      attribute> ------->|    *
*     |    --  --               | |                         |    *
*     |-- SAVE ---------------->| |---- <file-statement> --->|   *
*         --                                                     *
******************************************************************
```

The COMPILE statement must be the first statement in a set of control statements. CO has four options:

1. COMPILE
2. COMPILE TO LIBRARY
3. COMPILE SAVE
4. COMPILE FOR SYNTAX

CO is a "compile and go" operation. Providing the compilation is error-free, the MCP schedules the object program for execution. The program will not be entered into the disk directory, and must be recompiled to be used again. The "compile and go" is the default option of the CO statement.

CO TO LI will leave the program object file on disk and will enter the program-name into the disk directory after error-free compilation. The program is not scheduled for execution.

CO and SA combines the execute and library options. The MCP will enter the program-name into the disk directory, and will leave the object program file on disk, as well as schedule the program for execution after error-free compilation. The program remains in the disk directory.

CO FO SY provides a diagnostic listing as the only output. This option does not enter the program-name into the disk directory or leave the program object file on disk. Some uses are as a debugging tool, first time compilation, or a new source listing.

Program-control attributes and file statements are discussed in Section Seven of this document.

CP (Compute)

```
**********
*   CP   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- CP --- <integer1> -------------- <integer2> ----------->| *
*     --                |-operator-|             |- ; -|        *
*                                                               *
*****************************************************************
```

The CP input message allows the operator to perform simple arithmetic functions on the console printer, as well as decimal/ hexadecimal conversion.

The valid operators recognized by the CP messages are as follows:

+	addition
-	subtraction
*	multiplication
/	division
M	MOD (remainder divide)

The semicolon (;) terminates the expression and must be the last entry when entered from a card reader.

The CP message will evaluate an arithmetic expression strictly on a left-to-right basis. Therefore, quantities contained in parentheses or brackets are invalid. Spaces are not used as delimiters and are ignored. Operands and intermediate results are considered positive integers, and overflow beyond 16777215 will be truncated.

The response is displayed in both decimal and hexadecimal formats.

Examples:

```
Request:     CP @3A@ * 4 + @F@

Response:    CP:@0000F7@=247

Request:     CP @F@

Response:    CP:@00000F@=15
```

CQ (Clear Queue) ($)

```
**********
*   CQ   *
**********
```

Format:

```
************************************
*                                  *
* >--- CQ ---------------------->| *
*      --                          *
************************************
```

The CQ input message causes all messages stored in the console printer queue to be cleared.

<u>CS (Change to Standard System Software) (S)</u>

(DEBUG Only)

```
**********
*   CS   *
**********
```

Format:

```
*************************************
*                                   *
* >--- CS ----------------------->| *
*      --                           *
*************************************
```

The CS input message allows the system operator to insure that during the next Clear/Start MCPII system software and firmware will be loaded on the system (if the DBUG option is set).

CT (Change to Trace System Software) ($)

(DEBUG Only)

```
**********
*   CT   *
**********
```

Format:

```
*************************************
*                                   *
* >--- CT ----------------------->| *
*      --                           *
*************************************
```

The CT Input message allows the system operator to change the operating environment to trace system software after the next Clear/Start (providing the DBUG options has been set).

CAUTION

The CT input message is strictly for system software development and debugging. It should not be used in the standard operating environment.

CU (Core Usage)

```
**********
*   CU   *
**********
```

Format:

```
*******************************************
*                                         *
* >--------------------- CU ------->#     *
*     |-- <job-number> --|   --           *
*                                         *
*******************************************
```

CU (Core Usage), with a job-number, displays the amounts of save and overlayable memory (in bytes) in use by a program, and CU without a job-number displays system totals and the core usage for all programs in the mix. If the DBUG option is set, an analysis of linked memory, certain run structure fields, total user code space, and size and location of user files will also be displayed. The examples below show printouts with both the DBUG option reset (0) and set (1).

Examples:

CU (DBUG=0)

```
CORE USAGE : 19:40:33.4
SAVE = 29367 BYTES
OVERLAYABLE = 232275 BYTES
AVAIL = 89860 BYTES
DUMP/ANALYZER = 34  SAVE = 6020 BYTES,  OVERLAYABLE = 24625 BYTES
SDL = 32,  SAVE = 708 BYTES, OVERLAYABLE = 0 BITS, (ROLLED OUT).
DMPALL = 31,  SAVE = 3347 BYTES, OVERLAYABLE = 724 BYTES
```

31CU

```
DMPALL = 31  SAVE = 3347 BYTES, OVERLAYABLE = 724 BYTES
```

32CU

```
SDL = 32  SAVE = 708 BYTES,  OVERLAYABLE = 0 BITS (ROLLED OUT).
```

```
34CU

  DUMP/ANALYZER = 34  SAVE = 1003 BYTES,  OVERLAYABLE = 16150 BYTES

CU   (DBUG = 1)
---------------

  CORE USAGE : 19:37:11.2
  SAVE = 33500 BYTES
  OVERLAYABLE = 228768 BYTES
  AVAIL = 118390 BYTES
  WITHIN LINKED MEMORY:
  IN USE = 129257 BYTES
  LARGEST OVERLAYABLE = 182572 BYTES
  LARGEST AVAIL = 76921 BYTES
  SAVE SPACE = 18880 BYTES
  LINKED MEMORY = 247648 BYTES
  FIRST.LINK = @0181EE@
  LAST.LINK = @1FECF1@
  FENCE = @02040A@
  NUMBER OF LINKS = 196

613CU

  COBOL = 613  PR = 4
  COBOL = 613  LR = @1EE820@, BR = @1DE984@, LENGTH = 8680 BYTES.
  COBOL = 613  MEDIA, STOPPED, INTERVENTION, DUMMY.EV = 1,0,0,1
  COBOL = 613  ROLLOUT.COMPLETE, IN.PROCESS, TO.BE,
               ROLLIN.IN.PROCESS = 1,0,0,0
  COBOL = 613  RS.Q.IDENT, RS.NEXT.Q = WATE.Q, READY.Q
  COBOL = 613  RS.STATUS =10
  COBOL = 613  RS.DISPLACED = @000000@, 00000000
  COBOL = 613  RS.CODE.OVLY.COUNT = 16
  COBOL = 613  RS.DATA.OVLY.COUNT = 0
  COBOL = 613  CODE SPACE = 10701 BYTES.
  COBOL = 613  FILES = 17
  COBOL = 613  -6 AT @1D050E@  SIZE = 330 BYTES EXCLUDING DFH ON DISK
               AT @F200004AC@.
  COBOL = 613  -5 AT @1DDF61@  SIZE = 330 BYTES EXCLUDING DFH ON DISK
               AT @F200004494@
  COBOL = 613  -4 AT @1D81F0@  SIZE = 312 BYTES EXCLUDING DFH ON DISK
               AT @F20000487@
  COBOL = 613  -2 AT @1DC7D8@  SIZE = 422 BYTES INCLUDING DFH AT
               @1D0288@
  COBOL = 613  --TOTAL.FILE.SPACE = 1395 BYTES
  COBOL = 613  SAVE = 10864 BYTES, OVERLAYABLE = 10664 BYTES.
```

DB (Data Base) ($)

```
**********
*   DB   *
**********
```

Format:

```
*******************************************
*                                         *
* >--- DB ------------------------------># *
*       --   |-- <data-base-name> -->|     *
*                                         *
*******************************************
```

DB gives the user the ability to interrogate the status of an active data base (if <data-base-name> is specified), or to get a list of the currently active data bases. If there are no active data bases, the MCP responds with:

NO ACTIVE DATA BASES

If a <data-base-name> is specified and that data base is not active, the MCP responds with:

DATA BASE <data-base-name> IS NOT ACTIVE

If the <data-base-name> is active, the MCP identifies the current status of the data base with:

```
DATA BASE <data-base-name> ACTIVE ---
      <integer> OPERATIONS,
      <integer> READS,
      <integer> WRITES,
      <integer> EXCEPTIONS
      IN <time>
```

If no <data-base-name> is specified and any data bases are active, the MCP responds with a list of the active data bases.

```
ACTIVE DATA BASES ARE <data-base-name-1>
      [,<data-base-name-2>[...]]
```

DD (S)

(DEBUG Only)

```
**********
*   DD   *
**********
```

Format:

```
**************************************
*                                    *
* >--- DD ------------------------># *
*      --                            *
*                                    *
**************************************
```

Note: Requires a special MCP and special firmware.

Information to be supplied.

DF (Date of File) (#)

```
**********
*   DF   *
**********
```

Format:

```
**************************************************************
*                                                            *
* >-- DF -------------- <file-id> ---------------># *
*     --    |                                       |        *
*           |--------------------- <mfid>/= --->|           *
*             |-- <pack-id>/ -->|                            *
*                                                            *
**************************************************************
```

The DF input message allows the operator to display on the console printer the compilation data and time for code and interpreter files, and the creation date for all other types of files.

DM (Dump Memory and Continue)

```
**********
*   DM   *
**********
```

Format:

```
*******************************************
*                                         *
* >---------------------- DM ------># *
*    |-- <job-number> -->|    --          *
*                                         *
*******************************************
```

The DM input message allows the system operator to dump the contents of a program's memory space to disk for subsequent analysis by DUMP/ANALYZER. When the DEBUG option is not set (DBUG = 0), DM without a job-number produces an error message.

Processing automatically continues when the dump is finished.

The DM message will create a file called DUMPFILE/<integer>. The integer will be incremented by one each time a DM is performed in order to make each DUMPFILE unique.

The DUMPFILE may be printed by the DUMP/ANALYZER program. Refer to the "PM" message.

Example:

```
2 DM          % DBUG = 0
DM            % DBUG = 1
```

When the DBUG option is set, DM without a job-number produces a SYSTEM dump and places it in the SYSTEM/DUMPFILE. The DUMPFILE may be printed with MCPII/ANALYZER via the PM message.

DP (Dump Memory and Discontinue)

```
**********
*   DP   *
**********
```

Format:

```
**************************************
*                                    *
* >--- <job-number> ---- DP ----->#  *
*                         --         *
**************************************
```

The DP input message allows the system operator to initiate a memory dump during a program's execution, and then abort that program.

The input of the DP message signals the MCP to halt program execution, dump memory out to disk, and abort the program as though a DM message had been entered immediately followed by a DS message.

Example:

```
1347 DP
```

DR DT (Change MCP Date) ($)

```
**********
*   DR   *
*   DT   *
**********
```

Format:

```
*************************************
*                                   *
*  >--- DR ------- <mm/dd/yy> --->#  *
*    |  --  |                       *
*    |      |                       *
*    |- DT -|                       *
*       --                          *
*************************************
```

The DR, DT input message allows the system operator to change the current date maintained by the MCP. For inquiry about the current date and time, the TD message is used (see TD in this section).

The MCP will accept only valid dates. The month entry must be between one and twelve, the day must be between one and thirty-one, and the year must be valid numeric digits. An invalid date or alphabetic input will produce an error message.

With the TABS option set, the date cannot be changed if there is a job in the schedule or in the mix.

DS (Discontinue Program)

```
**********
*   DS   *
**********
```

Format:

```
***************************************
*                                     *
* >--- <job-number> --- DS ------># *
*                       --            *
***************************************
```

The DS input message permits the system operator to discontinue the execution of a program.

The DS message may be entered at any time after the BOJ and prior to EOJ.

The DS message signals the MCP to stop the program's execution and return the memory the program occupied to the system. Any files not previously entered into the disk directory are lost and the disk area occupied is returned to the disk available table. All other files are closed.

DU (DUMP) (#)

```
**********
*   DU   *
**********
```

Format:

```
**********************************************************************
*                                                                    *
* >--- DUMP -------------------------------------------------->>     *
*      --     |-- FROM <disk-pack-id> -->|                           *
*                   --                                               *
*                                                                    *
*  >>-- TO <library-tape-id> ---- =/= ----------------- ; -># *
*       --                     |                    |                *
*                              |<------- , ---------------|          *
*                              |-- <file-name> --------->|           *
*                              |-- <mfid>/=  ----------->|           *
*                              |-- <mfid>/<file-name> -->|           *
*                                                                    *
**********************************************************************
```

The DUMP statement will cause one or more disk files to be placed on a library tape. The file is not removed from disk by the dump. The =/= option causes every file on the specified disk to be dumped. The <mfid>/= option causes every file with the specified family-name to be dumped. A maximum of 2248 files can handled with one DUMP statement.

Example:

```
?DU TO SYS.TAPE
PAYROLL/=,
ACCPAY/=,
MASTER/FILE;

?DU FROM USER TO BACKUP=/=;
```

DY (DYNAMIC)

```
**********
*   DY   *
**********
```

Format:

```
*****************************************************************
*                                                               *
*  >--- DY --- <job number> --------------------------------># *
*       --                   |                              |   *
*                            |-- <control attributes> -->|      *
*                                                               *
*****************************************************************
```

The DY input message allows modification of the working copy of a program that is already in the mix or scheduled for execution.

With one exception, DY will accept any statement that can be made with a MO (MODIFY), and only the working copy of the program will be altered. If the control attribute, however, involves FI (FILE) modifications, the user cannot modify it if the file is currently open. The file must have been closed with RELEASE, LOCK, PURGE, or REMOVE before dynamic modifications are allowed.

ED (Eliminate Pseudo Deck) ($)

```
***********
*   ED    *
***********
```

Format:

```
***************************************
*                                     *
* ---- ED ------ <integer> ------->#  *
*      --                             *
***************************************
```

The ED input message allows the system operator to eliminate a deck from a pseudo reader. This is equivalent to flushing the reader and then performing an RY message.

The deck will be eliminated from the pseudo reader and from the disk directory by the ED message.

EM (ELOG Message) ($)

```
***********
*   EM    *
***********
```

Format:

```
**************************************
*                                    *
* ---- EM --- <input-message> ---->#  *
*      --                            *
**************************************
```

The EM input message allows the operator to place a message into the ELOG.

The input-message starts in the first position after the EM on the input line and continues until the END OF MESSAGE is pressed.

ET (ELOG Transfer) ($)

```
**********
*   ET   *
**********
```

Format:

```
************************************
*                                  *
* ---- ET ---------------------->| *
*      --                          *
************************************
```

The ET input message transfers the information in the file SYSTEM/ELOG to the file ELOG/#integer. The program SYSTEM/ELOGOUT is then executed label equating ELOG/#integer and prints the file.

EX (EXECUTE) (#)

```
**********
*   EX   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- EXECUTE -- <program-name> ----------------------------># *
*     --                        |                          |    *
*                               |-- <control-attributes> ->|    *
*                                                               *
*****************************************************************
```

The EXECUTE statement instructs the MCP to call a program from the library for subsequent execution. EX must be the first statement in a set of control statements pertaining to the execution of a program.

If the program referenced in the EXECUTE statement resides on a removable disk cartridge or pack, the disk-pack-id must be part of the program name in order for the MCP to locate the correct file. If a disk-pack-id is omitted, the file is presumed to reside on system disk.

Sample Card Deck:

```
? EX TEST
? DATA <file-identifier>
 ---data cards---
? END
```

The program named TEST is called out of the library on system disk and executed. The execution is from cards and the sample presumes that a card file is expected as input.

FM (Response to Special Forms) ($)

```
**********
*   FM   *
**********
```

Format:

```
************************************************************
*                                                          *
* >--- <job-number> --- FM -- <unit-mnemonic> --------->#  *
*                       --                                 *
************************************************************
```

The FM input message is a response to the "SPECIAL FORMS REQUIRED" message.

The unit-mnemonic designates which unit is to be assigned to the file.

The message

<program-name> = <job-number> SPECIAL FORMS REQUIRED FOR <file-id>

is displayed on the console printer requiring that a FM message be submitted by the system operator before the file can be opened. Should FM specify a backup device, tape or disk, a backup file will be created.

Example:

```
93 FM LPA
94 FM DPB
```

FN (Display Internal File Name)

```
**********
*   FN   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >--- FN -- <program-name> -- <external-file-id> ----------># *
*      --                                                       *
*****************************************************************
```

The FN input message allows the system operator to display the internal file names of an object program.

The MCP lists on the console printer all the internal-file-names of the object program which have the specified external-file-identifier in the following format:

Example:

```
FN = <internal-file-identifier-1>
FN = <internal-file-identifier-2>
FN = ...
```

FR (Final Reel of Unlabeled Tape File)

```
**********
*   FR   *
**********
```

Format:

```
***************************************
*                                     *
*   >-- <job-number> -- FR ------->#  *
*                        --           *
***************************************
```

The FR input message gives the operator the ability to notify the MCP that the last reel of an unlabeled tape file has completed processing, and there are no more input reels to be read.

The FR message is a response to the message:

<job-number> NO FILE

This message is the result of an unlabeled tape file reaching the End-of-Reel; the FR message notifies the program that the file has reached EOF.

The FR message is also allowed labeled tape files in order to signal EOF without reading all of the reels of the file.

The FR message must be used with paper input files to signal EOF after all reels have been processed.

FS (Force from Schedule)

```
**********
*   FS   *
**********
```

Format:

```
***************************************
*                                     *
*  >--- FS ------ <job-number> --->#  *
*       --       |               |    *
*                |----- = ----->|     *
***************************************
```

The FS input message is used to force jobs from the WAITING SCHEDULE into the ACTIVE SCHEDULE.

The Equal sign option will force all jobs into the ACTIVE SCHEDULE.

The HS message is used to place a job in the WAITING SCHEDULE.

NOTE

The WAITING SCHEDULE is a schedule of jobs that are "waiting" to be placed in the ACTIVE SCHEDULE. For example, an EXECUTE with the attribute THEN or AFTER.NUMBER would place the program in the WAITING SCHEDULE.

The ACTIVE SCHEDULE are those jobs that have satisfied all the requirements for execution and are only waiting for memory space to run.

In order for a program to be in the "mix", it must have gone to BOJ.

FW (File Waiting)

```
**********
*   FW   *
**********
```

Format:

```
************************************
*                                  *
*   >--- FW ------------------->#  *
*        --                        *
************************************
```

The FW input message is a unique HOST/RJE message. When HOST/RJE receives an input stream, that stream is preceded by a control card section which is passed by the program to the MCP. If this control string contains a command which causes a job to be scheduled and it also contains a ?DATA card, the MCP puts the job in the waiting schedule in a special status. If it were put in the active schedule, it would hang waiting for the card file. The MCP sends the schedule message back to HOST/RJE with an indicator saying that an FW is required when the card file has been built.

GO (Resume Stopped Program)

```
***********
*   GO    *
***********
```

Format:

```
*****************************************
*                                       *
*  >--- <job-number> -- GO --------->#  *
*                       --              *
*****************************************
```

The GO input message is used by the system operator to request resumption of a program that has been stopped (ST message).

GT (Enable Trace)

```
**********
*   GT   *
**********
```

Format:

```
*********************************************************
*                                                       *
* >--- &lt;job-number&gt; --- GT --- &lt;integer&gt; ----------->| *
*                        --                             *
*********************************************************
```

The GT input message is used by the system operator to enable the tracing of normal-state programs. It places an integer into the program's run-structure nucleus that the interpreter can interrogate to determine what instructions are to be traced. The program must be using a trace version of its interpreter.

The integer can be:

```
0 = No trace
1 = Trace branch opcodes
2 = Trace store opcodes
4 = All other opcodes
```

or any sum of the above.

Example:

```
1111 GT 7
```

HS (Hold in Waiting Schedule)

```
**********
*   HS   *
**********
```

Format:

```
*******************************************
*                                         *
* >--- HS -------- <job-number> ------->#  *
*      --       |               |          *
*               |------- = ----->|          *
*******************************************
```

The HS input message will allow the system operator to place a HOLD on a specific job(s), thereby temporarily removing them from the Active Schedule.

The equal sign (=) option will place all jobs in the Active Schedule into the Waiting Schedule.

A job-number is assigned when a program is scheduled by the MCP.

A job that has been placed in the Waiting Schedule by a HS message will remain in the Waiting Schedule until FS-ed.

<u>HW (Hold in Waiting Schedule until Job EOJ)</u>

```
***********
*   HW    *
***********
```

Format:

```
*********************************************************
*                                                       *
* >--- HW --- <job-number1> ---- <job-number2> ------# *
*      --  |                 |                          *
*          |------ = ------->|                          *
*********************************************************
```

The HW input message allows the system operator to designate that certain jobs are to be placed in the Waiting Schedule, awaiting the EOJ of another job (by job-number).

The equal sign (=) option will place all jobs in the Active Schedule into the Waiting Schedule, and mark them as waiting for the completion of job-number-2.

A job that has been placed in the Waiting Schedule by a HW message will remain in the Waiting Schedule until job-number-2 reaches EOJ or until FS-ed by the operator.

IC (Interpreter Count) ($)

```
**********
*   IC   *
**********
```

Format:

```
*********************************************************
*                                                       *
* >--- IC ------------------------------------------># *
*      --          |                 |                  *
*                  |--- <integer> -->|                  *
*                                                       *
*********************************************************
```

The IC command is used to change the number of entries allocated in the MCP's interpreter dictionary. The maximum number of unique interpreters which can be associated with running jobs is bound at Clear/Start by the number of entries allocated by the system initializer (SYSTEM/INIT). The default set at COLDSTART is six (one each for GISMO, the micro-MCP and the SDL interpreter plus three for normal-state jobs).

The IC message allows the operator to change the number of slots to be allocated at the next (and subsequent) Clear/Starts. If <integer> is not given, the IC command will display the current Interpreter Count. If integer is specified, the field NUMBER.DIC.ENTTIES in COLD.START.VARIABLES (both memory and disk copies) will be changed to <integer>. <Integer> must be greater than or equal to 3 and less than or equal to 31.

Note: IC requires a null mix and schedule if <integer> is specified. Also, a Clear/Start is required following an IC which specifies <integer>.

Examples:

```
IC

IC 8
```

IL (Ignore Label)

```
**********
*   IL   *
**********
```

Format:

```
*********************************************************
*                                                       *
* >--- <job-number> -- IL ---- <unit-mnemonic> ----->#  *
*                      --    |                  |       *
*                            |-- <# integer> -->|       *
*                                                       *
*********************************************************
```

The IL input message allows the system operator to ignore the label on the file mounted on the designated unit.

The job-number must be used to identify the program. In a multiprogamming environment there may be more than one "NO FILE" condition at a time.

The IL message may be used in response to the following messages:

NO FILE ...

DUPLICATE INPUT FILE ...

<file-identifer> NOT IN DIRECTORY

It is assumed that the system operator knows that the file on the unit selected is the file needed regardless of the original file-identifier's location. If the unit-mnemonic specifies a disk, the directory on that drive will be searched for the required file-identifier. The #integer option is used to designate a pseudo- reader (by number) as the input drive.

KA (Analyze Disk Directory) ($)

```
**********
*   KA   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- KA -------------------------------------->>               *
*         |                       |                             *
*         |-- <disk-pack-id>/ -->|                              *
*                                                               *
*     >>------------ =/= ------------------------------------>#  *
*         |                                          |          *
*         |-- DISKAVL/ ------------------------------>|          *
*         |                                          |          *
*         |-- <mfid>/ -------------------------------->|         *
*                             |-- <file-name> -------->|         *
*                             |------  =  ------------>|         *
*                                                               *
*****************************************************************
```

The KA input message allows the system operator to analyze the contents of a disk directory, including file area assignments. Incorrect syntax, e.g., KA =/-, is reported as an error on the console printer or CRT (SPO).

Inclusion of the disk-pack-id causes the MCP to list the requested information for the specified user disk pack or disk cartridge; otherwise, system disk is assumed.

Some examples of KA syntax are as follows:

KA =/= displays available disk areas and all files. If KI.KO is set, then all 540 bits of each file header will be displayed.

KA DSKAVL/ displays available disk areas. If the KI.KO flag (hex 3993) is set, then absolute addresses and contents of every sector of the available table, as well as the temporary table are displayed.

KA <mfid>/=	displays all files with the family name <mfid>, as well as any single-name files called <mfid>. If KI.KO is set, then all 540 bits of each file header are displayed. If DBUG is set, then the pack-information table is also displayed if a user pack is specified. A file header being in core will have both disk and core copies displayed.
KA A/B	displays information for the specified file only.
KA A	displays information for the specified file only.

KA will recover from and report on I/O errors on the disk available tables, file headers, and disk directories. This option is not available for user packs.

KB (Print SPOLOG) (S)

```
**********
*   KB   *
**********
```

FORMAT:

```
**************************************************************
*                                                            *
*  >--- KB ------- <number> --------------------------------># *
*       --  |                                   |            *
*           |-- <user-option> ----------------->|            *
*                              |-- <number> -->|             *
*                                                            *
**************************************************************
```

KB allows the user to print the SPOLOG on the printer and to control the format of the display for both CRT SPO and TTY SPO.

<n> — functions for both CRT and TTY SPO, simulating the output operation on the line printer if it is available and if KB LP ON has been specified. For CRT applications, <n> must have a value of 0-3 (See SCROLLING below); otherwise <n> ranges from zero to 199 and re-outputs all SPO messages stored in the latest <n> sectors on the SPO queue.

DEF[AULT] — sets all options to standard format: KB SW 39; KB W 39; KB L 22; KB LP OFF; KB SUP; KB DIR 0; and KB TIME OFF.

DIRN <n> — will cause the CRT SPO to display messages in reverse chronological order (default) if <n> = 0 or in normal chronological order if <n> = 1. Normal chronological order means that the most recent SPO messages will be displayed in the lower right-hand corner of the screen and previous messages will appear above the most recent one. Reverse chronological order means that the most recent message will be displayed in the upper left-hand corner (excluding the two-line input area) and older messages will appear below the more recent ones.

INP.LINES<n> — may be used, for CRT SPO only, to change the size of the input area reserved at the top of the screen. The default size of the input area is two lines. Using this message, the operator may set <n>, the number of lines to any value between two and sixteen included.

Changing the size of the input area will have a resultant effect on the size of the output area. Increasing the size of the input area will decrease the size of the output area, proportionately.

On the CRT SPO, the input area remains displayed on the screen and in the memory of the CRT itself until a different message is typed in. Therefore, if the operator has made an error in a long input message, after being told of the error by the MCP, he may not need to retype the entire message. It may be possible to simply move the cursor to the bad spot in the message by using the control keys provided on the keyboard for movement up, down, back, or ahead. After the error has been corrected, the message can be retransmitted in its error-free form. This option is not applicable to all messages.

L — sets the number of lines of SPO output on CRT SPO, for bot one-column and two-column display.

LP ON(OFF) — applies to both TTY and CRT SPO. "ON" will divert all output messages to an available line printer until the printer is grabbed by a user program or an MCP control function such as KA or KP. "OFF" will restore normal TTY or CRT SPO display. Default is "OFF".

SPOQ — is functional only if KI.KO is ON. It is equivalent to a KC on the 200 sectors of the SPO.Q on system disk.

SUP[PRESS] — will suppress all messages on the screen except SPO input and SPO output.

TIME ON(OFF) — functions for both TTY and CRT, to display with each message its "time stamp". The time of the occurrence of the event will be displayed in front of the message itself, and is always stamped on each message stored in the SPO Queue, regardless of whether the option is set or not. Time stamp display will reduce confusion when one is scrolling the CRT back and forth.

TRANSL 1[0] — allows the TRANSLATE option for the CRT SPO to be set or reset. Setting the translate option to "1" means that upper-case SPO input (lower-case letters locked out) is returned (displayed) in lower-case letters, except for "L" which could possibly be confused with "1". "0" means that no translation of alphabetic characters will take place and is the default option set at Clear/Start time.

Translation will impose some compute overhead, and the response times of the CRT SPO will seem longer.

UNS[SUPPRESS] — will display all messages, including ZIP, QUEUE, control cards from card readers and pseudoreaders, etc.

W[IDTH] <n> — functions for both TTY and CRT SPO to modify the maximum number of characters per line to <n> characters. The range of <n> is 20 to 72 characters for TTY SPO, and 39 or 80 for CRT SPO, depending upon the default setting at Clear/Start time.

SCROLLING

Scrolling allows the operator to review the history of the system's operation without requesting a hard copy of the SPO output. It is, in effect, an "instant replay" of the lines beyond the top of the screen. Scrolling is directed by the following options:

<ETX> — will cause the SPO to return to the latest screenful of messages, thereby providing an escape from the scrolling mode. Other means of escaping the scrolling mode are MX, WY, or leaving the SPO in RCV. When the SPO is in RCV, the MCP will automatically

take it out of scrolling mode and display current information.

0<ETX> same as <ETX> above.

1<ETX> causes the CRT screen to be scrolled backwards one full screen. When the previous screen has been displayed, entering another 1<ETX> will repeat the process. There is no limit to the number of times the operator may enter 1<ETX>; however, each screen will be less and less current, and eventually the SPO Queue will "wrap around" since the file is circular.

2<ETX> will cause the SPO to scroll forward by one disk sector of information. One disk sector may contain several lines of information; thus, the command means "scroll forward for a few lines".

3<ETX> will cause the SPO to scroll backward for a few lines to enable the user to salvage a message which may be split between two screens.

KC KP (Print Disk Segments) (S)

```
**********
*   KC   *
*   KP   *
**********
```

Format:

```
*******************************************************************
*                                                                 *
* >---- KC -------- <file-id> --------------------------------># *
*     |  --  |  |                         | |               |     *
*     |      |  |-- DKx --- <integer1> ->| |- <integer3> ->|      *
*     |- KP -|  |          |- <integer2> ->|                      *
*        --     |                          |                      *
*               |-- <unit-mnemonic> ---->|                        *
*                                                                 *
*******************************************************************
```

The KC or KP message provides a means for the system operator to print selected disk files or segments of a disk on the line printer.

The printout created by the KP message is in HEXADECIMAL format.

The print created by the KC message is in CHARACTER format.

The file-identifier option will print a file by that name. The DKx option is used for the Head-per-Track disk. Integer-1 is required with Head-per-Track disk and designates the electronics unit. In this context, Integer-1 must be a non-zero value.

Integer-2 is used to specify the disk address from which printing is to begin.

Integer-3 is used to specify the number of segments to print beginning either from the first segment of a file or the address specified by integer-2. If omitted, number of segments printed is one. KC and KP can recover from, and report on, I/O errors.

Examples:

```
KP A/B 10..................Print 10 segments of file
                           A/B

KP A/B.....................Print 1  segment  of file
                           A/B

KP CCC/X/..................Print 1 segment of file A
                           on pack CCC

KP DPA @5C@ 15.............Print   15  segments from
                           HEX LOC5C

KP DKA  1 200 10...........Print 10 segments on EU 1
                           from DECIMAL LOC 200
```

KI KO (Read and Write Disk) ($)

```
(DEBUG Only)
**********
*   KI   *
*   KO   *
**********
```

Format:

```
*******************************************************************
*                                                                 *
* >-- KI -- <unit -- <sector --- <bit -- <length ------------># *
*    -- |  mnemonic> address>   offset>  in bits> |-- H ->||     *
*       |                                         |-- A ->||     *
*       |                                                 |      *
*       |-- MEM --- <core ----- <length ----------------->|      *
*       |   ---      address>    in bits>                 |      *
*       |                                                 |      *
*       |-- AD ----- SPO.SQ------------------------------>|      *
*                                                                 *
*                    ***             ***          ***             *
*                                                                 *
* >-- KO -- <unit -- <sector --- <bit --- <literal> --------># *
*      -- | mnemonic>   address>   offset>                |      *
*         |                                               |      *
*         |                                               |      *
*         |-- MEM --- <core ----------------------------->|      *
*             ---      address>                                   *
*                                                                 *
*******************************************************************
```

The functions of KI and KO are defined as follows:

KI	reads from and displays a disk sector, starting from the (bit-offset)-th bit of a specified sector (sector address) of a particular disk (unit mnemonic). If H or A is specified, the bits read are displayed in HEX or EBCDIC, respectively; otherwise, they are displayed in ones and zeros.
KI MEM	displays a specified number of bits in memory, starting with the bit at address = bit-offset. It is displayed in HEX as a binary number.

KI AD SPO.SQ	displays the address of some vital fields of the MCP. At present, only the SPO.SQ globals are displayed.
KO	writes the specified literal on disk (unit mnemonic) or in memory (MEM) at the specified location. It should be exercised with care, since it is a brute-force way to change data.

KT (Disk Trace) ($)

```
                                                     (DEBUG Only)
                                                      **********
                                                      *  KT    *
                                                      **********
```

Format:

```
*****************************************************************
*                                                               *
*  Format 1:                                                    *
*                                                               *
*                                                               *
*  KT -- NAME ----------- <file-id> ----------------------->#   *
*  --    ----    |- = -|                  |- , -|      |  |     *
*                                                      |  |     *
*  |<--------------------------------------------------|  |     *
*  |----------------------------------------------------->|     *
*    |-[1]-- RECORD.SIZE -------------- <integer> -->|          *
*    |       -----------     |- = -|                 |          *
*    |                                               |          *
*    |-[1]-- RECORDS.BLOCK ------------ <integer> -->|          *
*    |       -------------     |- = -|               |          *
*    |                                               |          *
*    |-[1]-- PHYSICAL.RECORDS --------- <integer> -->|          *
*    |       ----------------    |- = -|             |          *
*    |                                               |          *
*    |------ BUFFERS ------------------ <integer> -->|          *
*    |       -------         |- = -|                 |          *
*    |                                               |          *
*    |------- MASK ------- <six-digit hex literal> ->|          *
*             ----   |- = -|                                    *
*                                                               *
*****************************************************************
```

Note: The above attributes may be specified in any order.

```
*************************************************************
*                                                           *
*  Format 2:                                                *
*                                                           *
* KT -- MCP ----- NAME ----- <file-id> -------------------->#  *
* --    ---       ----                           |   |      *
*                                                |   |      *
*   |<-------------------------------------------|   |      *
*   |----------------------------------------------->|      *
*      |                    |                               *
*      |-- MEM.TRACE ------>|                               *
*          ---------                                        *
*                                                           *
*  Format 3:                                                *
*                                                           *
* KT --- CLOSE ------------------------------------------->#  *
* --     -----                                              *
*                                                           *
*************************************************************
```

KT opens a trace file with the name specified by the user. Unless the user requires a trace file with other than default parameters, Format 2 should be used. If Format 1 is used, record size, blocking factor, and the number of records required must be specified in integers greater than zero. The number of buffers defaults to one and all mask bits are set. Format 3, which applies to both options, is used to close the trace file.

All formats of the KT command require a special MCP and special firmware.

Format 2 (KT MCP <filename>) is the default method of opening a trace file. Users must supply only the filename if this form is used. The attributes of the trace file opened Format 2 are:

RECORD.SIZE = 50 characters
RECORDS.BLCCK = 10
BUFFERS = 2
PHYSICAL.RECORDS = 5000
MASK = @80C000@

If MEM.TRACE is specified, a summary of the current memory configuration is dumped to the file for use by the MEMORY/MAP analyzer.

Format 3 closes the trace file, and any new messages after the close are ignored. If the trace file exceeds the size of PHYSICAL.RECORDS, it is closed automatically.

See Appendix A for sample traces, and more detailed instructions in the use of the trace facility.

LC (Load Cassette) ($)

LC (Load Cassette) ($)

```
**********
*   LC   *
**********
```

Format:

```
*************************************************************
*                                                           *
* >-- LC ---------------------------- =/= -------------># *
*     --                 !                          !       *
*                        !-- <mfid>/= -------------->!      *
*                        !-- <mfid>/file-id -------->!      *
*                                                           *
*************************************************************
```

The LC message is used to load system programs (compilers, interpreters, object code, system software) from a cassette in the console cassette reader to disk with appropriate additions in the disk directory.

The LC message cannot be used to load a freestanding program that does not execute under the control of the MCP.

The LC message calls the program SYSTEM/LOAD.CAS which loads the files.

LD (Pseudo Load)

```
**********
*   LD   *
**********
```

Format:

```
****************************************
*                                      *
* >-- LD --------------------------->#  *
*     --       |             |         *
*              |-- <string> -->|       *
*                                      *
****************************************
```

The LD input message is used to initiate the building of pseudo card deck(s) on disk to be processed by pseudo readers.

The LD message executes SYSTEM/LDCONTRL, an MCP utility program, which looks for a "?DATA CTLDCK" control statement that initiates the read.

The card deck's "file-id" is assigned by a "?DATA file-id" control statement preceding the data deck to be read. Each data deck that is loaded will be numbered consecutively along with its file-id which is used in opening the pseudo card files.

Terminating the LD function requires a "?END CTLDCK" control statement immediately following the last data deck that is to be read.

Users can create pseudodecks out of disk data files by file-equating CARD.IN (the internal name of SYSTEM/LDCONTRL's input file) to their disk files (See Example 2). Input statements must be terminated by either a semicolon (;) or a per cent sign (%) if the file is a sequenced disk file. If not, SYSTEM/LDCNTRL will scan the sequence number and errors will be reported.

Example:

The following example demonstrates how compile decks and a data deck can be loaded as pseudo card files to be used by pseudo readers.

```
        -----------------------------------------
        !
        !                 ?   DATA CTLDCK
        !                --
CONTROL !     DECK       ! ?   COMPILE program-name COBOL SYNTAX
DECK    !     A          ! ?   DATA CARDS
        !                !
        !                !     data deck
        !                ! ?   END
        !                --
        !
        !                --
        !     DECK       ! ?   COMPILE program-name FORTRAN
        !     B          ! ?   DATA CARDS
        !                !
        !                !     data deck
        !                ! ?   END
        !                --
        !
        !                --
        !     DECK       ! ?   DATA file-id
        !     C          !
        !                !     data deck
        !                !
        !                ! ?   END
        !                --
        !                 ?   ENDCTL
        ---------------------------------------------
```

Example 1

```
-------------------------------------------
!
!             --
!   DECK      ! ? EX SYCOPY SW=1            %
!   A         ! ? DA CARDS                  %
!             ! ? FI TAPE.1 NAM DIFFNAME;
!             ! ? END                       %
!             --
!
!
!             --
!   DECK      ! ? EX CHECK/LOAD.DUMP AFTER SYCOPY  %
!   B         ! ? FI T NAM DIFFNAME;
!             ! ? FI LINE NO HAR REP 2;
!             ! ? END                       %
!             --
!               ? ENDCTL
!
-------------------------------------------
```

Example 2

LJ (Emulators Only) ($)

```
**********
*   LJ   *
**********
```

Format:

```
************************************
*                                  *
* >------------ LJ ------------->| *
*               --                 *
************************************
```

The LJ input message is used to load a value into COLDSTART VARIABLES that is accessed by the SYSTEM/INIT program. The value loaded is the B1800/B1700 Micro Emulator key. The integer may contain any value from zero to 15. The value is determined by the binary equivalents of:

1 = SYSTEM/INIT to load B1800/B1700 Emulator
2 = Emulate
4 = Emulator to do its own I/O
8 = Load first 3KB of Emulator into Control Memory

LG LN (Transfer and Print Log) ($)

```
**********
*   LG   *
*   LN   *
**********
```

Format:

```
****************************************
*                                      *
* >----- LG ---------------------->#  *
*    |    --     |  |-- SPO --|        *
*    |           |      ---            *
*    |-- LN --|                        *
*         --                           *
****************************************
```

The LG, LN input message allows the system operator to transfer and print the log. The log files are numbered sequentially starting with LOG/#000001. The program SYSTEM/LOGOUT (TABS/LOGOUT if TABS is set) is executed, performing the necessary necessary file equate to print the log. The program SYSTEM/LOGOUT (or TABS/LOGOUT) must be in the disk directory in order for the MCP to accept the message.

If the SPO option is used, the new log file is SPOLOG/#000001 and the program is SYSTEM/SPOLOGOUT.

LO (LOAD) (#)

```
**********
*   LO   *
**********
```

Format:

```
**********************************************************************
*                                                                    *
* >-- LOAD --------------------------------------------------- ; ->#  *
*     --     |- TO <disk-pack-id> -->|     |               |         *
*               --                         |               |         *
* <----------------------------------------|               |         *
* --> FROM <library-tape-id> --- =/= ------------------->|          *
*     ----                     |<-----  ,  ---------------|          *
*                              |- <mfid> ---------------->|          *
*                                         |- /= ---------->|          *
*                                         |- /<filename> ->|          *
*                                                                    *
**********************************************************************
```

The LO (LOAD) statement will cause a file or files on a library tape to be placed on disk. If the file is already on disk, the old file will be removed.

The =/= option causes every file on the tape to be loaded, the <mfid>/= option is used to load every file with a specified family-name.

Example:

```
LO FROM SYSTEM COBOL,
RPG, BASIC, (SYSTEM)/=;
```

LP (Lock Protection) (#)

```
***********
*   LP    *
***********
```

Format:

```
*****************************************
*                                       *
* >-- <job-number> ---- LP ---------># *
*                    |         |        *
*                    |-- LP-  -->|      *
*****************************************
```

The LP input message is used to prevent certain SPO messages from interfering with the execution of the program. A protection bit is set in the Run Structure Nucleus and, with this bit set, messages such as DS, DP, GT, ST, and SW will not affect the run structure. The bit can be set at execute time by including the PROTECTED (or PT) attribute in the control string. The bit can be set while the program is running through <job-number>LP. It is reset through <job-number>LP-.

LS (Log SPO) (S)

```
**********
*   LS   *
**********
```

Format:

```
************************************
*                                  *
* >----------- LS --------------># *
*              --                  *
************************************
```

The LS command causes a boolean to be set and carried with a control string and eventually a job. The function of the LS boolean is to cause all control messages (both input and output) to be inserted in the control queue (Note: a QU command is required in the control string prior to the LS command). In addition, the LS boolean has the effect of bypassing the local SPO except for error messages which require operator intervention, or if the RMSG option in the MCP is set. For example, the control string "QU X/Y LS EX DMPALL" would cause the following messages to be inserted into the control queue X/Y:

a. The special schedule record message.

b. A message containing the actual control string.

c. The special BOJ message.

d. The actual SPO BOJ message.

e. The normal DMPALL display to the SPO.

f. The ACCEPT message from DMPALL.

g. Any input to DMPALL, and subsequent ACCEPTs from DMPALL.

h. The special EOJ message.

i. The actual SPO EOJ message.

Note: A job which has been spawned and has its LS boolean set may not spawn a job with the LS command.

LT (Load Translator) ($)

```
**********
*   LT   *
**********
```

Format:

```
***********************************************************************
*                                                                     *
* >-- LT --- <unit-mnemonic> ------------------------------------->$ *
*     --                      |                           |          *
*                             |-- <train-id> ---------|              *
*                                                                     *
***********************************************************************
```

The LT input message is used to load specific translate tables into the printer controls (Printer-Control-5 and Printer-Control-2). The syntax of LT is different for each of the two controls: the B1247 control (Printer-Control-2) requires a switch setting on the printer itself and B1247-4 (Printer-Control-5) requires train identification, either by number or name. LT may be used to load a table into the control without Clear/Starting. The translate tables for Printer-Control-2 are contained in a file called SYSTEM/PRINTCHAIN, also located on system disk.

Which printer control is being used on a particular system can be determined from an ELOG listing for the system since DEVICE ID for printer controls is specified on the printout. The translate tables for Printer-Control-5 are located in a data file on system disk called SYSTEM/TRAINTABLE that is created by the SYSTEM/BUILDTRAIN program.

Print-Control-5 (PC5) acknowledges train printers and has a buffering capability for storing translate tables for individual trains as used.

The printer must be ready and not in use by any program when a new translate table is to be loaded.

B1247 PRINTER CONTROL (DEVICE ID = 3103)

If this control is used (not allowed for the 1100 LPM printers), the file SYSTEM/PRINTCHAIN will be built by the MCP on system disk. A translate table will be loaded from this file into the control based upon the setting of the Train Selector Switch on the printer, as shown in the following table:

SWITCH	CHARACTER SET
1	64-character EBCDIC
2	48-character EBCDIC
3	16-character EBCDIC
4	96-character EBCDIC
5	48-character FORTRAN
6	48-character B300/B500
7	48-character RPG
8	Undefined

The MCP will load the required table, according to the switch setting, the first time the printer goes ready following a Clear/Start. If it is necessary to change the character set later, all that is required is to mount the new train module in the printer, select the proper switch setting, make the printer ready, and enter the LT message to alert the MCP to the presence of the new character set.

Example:

```
LT LPA
```

TRANSLATE TABLES (B1247-4 CONTROL)

If this control is used, the file SYSTEM/TRAINTABLE must be present on system disk. This file contains the translate tables for all "known" character sets (See the tables below). The MCP will select the specified translate table from the file and load it into the printer control.

For 400/750 LPM printers connected to the B1274-4 control, the train selector switch is ignored. The MCP will display

"LT" REQUIRED FOR <UNIT-MNEMONIC>

on the SPO when the printer goes ready after Clear/Start. It will load a translate table only when ordered to do so through LT. Once loaded, the MCP will not change the translate table until told otherwise.

The first time that the 1100 LPM printer goes ready after Clear/Start, or any time the train module is changed, the MCP will automatically load the translate table specified in the train module.

The following tables show the various character sets contained in the SYSTEM/TRAINTABLE file, along with their associated id numbers and names. In LT syntax, the train may be identified by either the ID NUMBER or the ID NAME.

*******400/750 LPM PRINTERS*******

ID NUMBER	ID NAME	CHARACTER SET
130	FORTRAN48	48-character FORTRAN
131	B300/B500.48	48-character B300/B500
132	EBCDIC3.48	48-character EBCDIC-3
140	RPG48	48-character RPB
144	EBCDIC96	96-character EBCDIC
254	EBCDIC3.16	16-character EBCDIC-3
255	EBCDIC3.64	64-character EBCDIC-3

***********1100 LPM PRINTER********

ID NUMBER	ID NAME	CHARACTER SET
001	EBCDIC18	18-character EBCDIC
002	FORTRAN48	48-character FORTRAN
003	B300.B50048	48-character B300/B500
004	EBCDIC48	48-character EBCDIC
005	EBCDIC72	72-character EBCDIC
012	BCL72	72-character BCL
015	ASCII72	72-character ASCII
016	EBCDIC96	96-character EBCDIC
018	EBCDIC.A72	72-character ALPHA/EBCDIC
019	EBCDIC.N72	72-character NUMERIC/EBCDIC
020	RPG48	48-character RPG
021	OCR.A72	72-character OCR-A

MH (Modify Header) (#)

```
**********
*   MH   *
**********
```

Format:

```
****************************************************************
*                                                              *
* >-- MH --- <file-name> --- <attribute> --------->>           *
*     --                                                       *
*                                                              *
*     >> --- <control variable> ------------------------->#    *
*                                                              *
****************************************************************
```

The MH input message is used to change security attributes in the disk file header. Disk files created under the MCP-supported security mechanism contain bits in the disk file header that prevents access of the file under improper conditions, i.e., those which violate file security. This command allows the user to modify two bits: DFH.PROTECTION and DFH.PROTECTION.IO.

DFH.PROTECTION and DFH.PROTECTION.IO are set to PUBLIC and INPUT OUTPUT respectively unless a program has been executed under a usercode, in which case, the security mechanism is implemented and files are locked into the directory with the protection bits set.

File attributes can be set or reset through the following attributes:

ATTRIBUTE	ABBREVIATION	CONTROL VARIABLE
PROTECTION	PTN	DEFAULT PUBLIC PRIVATE
PROTECT.IO	PIO	I.O INPUT OUTPUT
USER.COUNT	USR	<integer>

where integer is the USER.COUNT desired.

Note: DBUG must be set for USR.

ML (Mix Limit) ($)

```
**********
*   ML   *
**********
```

Format:

```
*****************************************
*                                       *
* >----- ML -----------------------># *
*        --    |                 |      *
*              |--- &lt;integer&gt; -->|      *
*                                       *
*****************************************
```

The ML input message allows the operator to specify the maximum number of jobs allowed in the mix at a priority of eight or less. ML without an integer displays the current limit. ML with an integer (1-63) changes the limit. COLDSTART sets the limit at 63.

ML is used in a multiprogramming environment to limit the number of lower-priority jobs that may run in the mix. When the mix limit has been reached, high-priority jobs (with a priority of nine or greater) are allowed into the active schedule regardless of the current mix limit. Lower-priority jobs, however, are automatically put into the waiting schedule and not allowed to actively execute on the system until another program(s) goes to EOJ and reduces the number of active jobs to less than the maximum mix limit.

MM (S)

```
**********
*   MM   *
**********
```

Format:

```
**********************************************************
*                                                        *
* >--- MM --- THRASH ---- ON ---------------------->#   *
*      --  |  ---      |          |     |                *
*          |           |-- OFF -->|     |                *
*          |                            |                *
*          |-- O.T --- <decimal> ----->|                *
*                                                        *
**********************************************************
```

The MM message controls thrashing detection if the THR option is set (See Table of Options (TO) for a definition of this MCP option). When thrashing occurs, the schedule is not automatically restarted until some job goes to EOJ and the message SYSTEM IS THRASHING, SCHEDULE STOPPED appears on the SPO. MM allows the following optional settings:

MM THRASH ON

MM THRASH OFF — When THRASH is on, the thrashing message will be redisplayed on the SPO once per N.SECOND as long as thrashing continues. This is the default value. When THRASH is off, the thrashing warning message is displayed once each time a job is added to or removed from the mix as long as thrashing continues.

MM O.T <decimal #> — Sets the overlay target to <decimal #> which may vary between 4 and 31. The default value is 8. These values denote the number of overlays per time interval. One time interval, in this instance, is equivalent to 800 milliseconds. Higher values will make the thrashing detector less sensitive and lower values will make it more sensitive.

MO (MODIFY)

```
**********
*   MO   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- MODIFY ---- &lt;program-name&gt; -- &lt;control-attribute&gt; ----># *
*     --                                                        *
*****************************************************************
```

The MODIFY message allows the user to permanently change attributes within a program. It has the same syntax as the EXECUTE statement but does not execute the program.

Example:

```
?MO A/B PR 6
```

The above statement will permanently change the priority of program A/B to six.

The MODIFY statement can be used to change the following attributes:

CHARGE	MEMORY
DYNAMIC.SPACES	PRIORITY
FILE	SCHEDULE.PRIORITY
FREEZE	UNFREEZE
INTERPRETER	VIRTUAL.DISK
INTRINSIC.NAME	
INTRINSIC.DIRECTORY	

MP (Memory Priority)

```
**********
*   MP   *
**********
```

FORMAT:

```
****************************************************************
*                                                              *
* >-- <job-number> --- MP -------------- <integer> -------->#  *
*                      --    |         |                       *
*                            |-- = -->|                        *
*                                                              *
****************************************************************
```

The MP input message reports the memory priority of the specified job or sets the memory priority of the specified job to the value specified by <integer> if the MPRI option is set. The MPRI option is part of the Prioritized-Memory system discussed in Section V of this document and defined in the system options listed under the TO message.

The default value of <integer> is 4 and the range is 0-15.

MR (Close Output File with Purge)

```
***********
*   MR    *
***********
```

Format:

```
*************************************
*                                   *
* >--- <job-number> ---- MR ------># *
*                        --         *
*************************************
```

The MR input message gives the system operator the ability in a duplicate file situation to save the old file by purging the newly created file.

MU (List Multipack File Tables) (S)

```
**********
*   MU   *
**********
```

FORMAT:

```
*************************************
*                                   *
*  >---------- MU ----------->#     *
*              --                   *
*************************************
```

The MU input message gives the operator the ability to interrogate the MCP's multipack file table which contains all multipack files that have been entered in the table since the last Clear/Start or RT message. If present, their absolute addresses on the system disk will be displayed if the DBUG option is set.

MX (Display MIX)

```
***********
*   MX    *
***********
```

Format:

```
*************************************
*                                   *
*  >------------ MX ------------># *
*                --                 *
*************************************
```

The MX input message allows a system operator to request that the MCP display on the console printer all the programs, and their job numbers, currently executing in the mix.

The MX response lists the memory and processes priority, program-names and job numbers of all programs currently running.

Example:

```
MX

 <program-name> = <job-number> PP=<integer> MP=<integer>

 <program-name> = <job-number> PP=<integer> MP=<integer>

 END MX
```

NC (Memory Chip Table) ($)

```
**********
*   NC   *
**********
```

Format:

```
**************************************
*                                    *
* >-- NC ------------------------->| *
*     --    |                 |      *
*           |--- <integer> -->|      *
*                                    *
**************************************
```

The NC input message allows the system operator to control how the table of failing memory chips is maintained.

If the integer is omitted, the current size of the table is reported. The value reported will be the size of the table following any subsequent Clear/Start. That value can be changed by using the NC <integer> option of the command.

The integer has the following meaning:

0 — reports the current size of the table and sets the size following the next Clear/Start to the default size (one per 16KB).

1-255 — specifies the number of locations to be used following the next Clear/Start. Integers greater than 255 exceed the maximum allowed for this command and default to 255 locations.

On machines without error-correcting memories, NC has no effect.

NT (Disable Trace)

```
 (DEBUG Only)
 **********
 *   NT   *
 **********
```

Format:

```
*****************************************
*                                       *
* >-- <job-number> -- NT ------------># *
*                     --                *
*****************************************
```

The NT input message allows the operator to terminate program traces. The job-number of the program being traced is required. The MCP, upon receipt of this command, sets the trace integer in the program's Run Structure Nucleus to zero, and the trace is discontinued.

OF (Optional File Response)

```
***********
*   OF    *
***********
```

Format:

```
***************************************
*                                     *
* >--- <job-number> -- OF --------># *
*                      --             *
***************************************
```

The OF input message is used in response to the NO FILE message. It informs the MCP that the specified file is "optional" and can be bypassed.

The OF message indicates that the file being requested is to be bypassed for this execution. Usage is restricted for input files that have been declared or label-equated (FILE control word) as OPTIONAL.

OK (Continue Processing)

```
**********
*   OK   *
**********
```

Format:

```
****************************************
*                                      *
*  >--- <job-number> -- OK --------># *
*                       --             *
****************************************
```

The OK message is used by the system operator to direct the MCP to attempt to continue processing a program marked as WAITING.

The OK message should only be given after the necessary action has been taken to correct the problem that caused the program to be placed in WAITING status.

Examples:

<job-specifier> DUPLICATE INPUT FILES...

<job-specifier> DUPLICATE FILE ON DISK...

<job-specifier> NO DISK...

<job-specifier> NO MEMORY...

<job-specifier> FILE <file-identifier> NOT PRESENT

If the corrective action is not taken before the OK message is entered, the original output message is repeated.

OL (Display Peripheral Status)

```
**********
*   OL   *
**********
```

Format:

```
*******************************************
*                                         *
*  >--- OL ----- <unit-mnemonic> ----->#  *
*       --    |-- <unit-type-code> ->|    *
*             |----- PSR ----------->|    *
*                                         *
*******************************************
```

The OL input message allows the system operator to interrogate the status of the system's peripheral units.

The unit-mnemonic option displays the status of a specific unit and, for disk devices, also reports on how many users are active on that unit.

The unit-type-code option displays the status of all peripherals of the same type. Unit-type-codes are the first two characters of a unit mnemonic.

Any invalid type unit used in the OL message will cause the MCP to display the following message:

NULL <unit-type-code> TABLE

The PSR option is used to interrogate the status of the pseudo readers on the system.

Examples:

```
CDA NOT READY

MTC UNLABELED

DPA LABELED "USER" #123456

MTA LABELED "MASTER" [123456]
```

OU (Specify Output Device)

```
***********
*   OU    *
***********
```

Format:

```
*******************************************************
*                                                     *
* >-- <job-number> --- OU ---- <unit-mnemonic> --->#  *
*                      --                             *
*******************************************************
```

The OU input message is a response to direct an output file to a specified output device.

Example:

```
4 OU DPC
```

The OU is normally used in response to the "PUNCH RQD..." message to direct the file to backup.

INITIAL LETTERS P-R

PB (Print/Punch Backup)

```
**********
*   PB   *
**********
```

Format:

Format 1:

```
--------------------- PB -------------------------------->>
                   |   --  |                           |
                   |       |--/1\-- <unit-mnemonic> -->|
--- <us/pw> ------>|       |--/1\-- <unit-name>/ ----->|

     >>----- =/= ------------------------------------------------#
        |--- PRT/= --------------->|  |--- SAVE --->|
        |--- PRN/= --------------->|  |--- LABEL -->|
        |--- PCH/= --------------->|  |--- LABELS ->|
        |------------- <number> -->|
        | |-- # -->|               |
        | |-- % -->|               |
        |                          |
        |-- <name> --------------->|
                    |-- /= ----->|
                    |- /<name> ->|
```

Format 1 is used to print or punch a number of backup files in their entirety. When the =/= option is used, all backup files existing on the designated disk or tape at the time the message is entered are printed or punched. If both printer and punch backup files exist on the disk, two copies of SYSTEM/BACKUP are executed; one copy handles printer files, and the second copy handles punch files. The PRT/= and PCH/= options cause the printing or punching of all printer and punch files, respectively, that exist on the designated unit. PRN/= is an acceptable equivalent of the PRT/= option.

Note: When format 1 is used, no options may be included.

If specified, the unit-mnemonic must be a tape (MT) or disk (DC, DK, or DP) device, and must indicate to the MCP the location of the requested backup file or files. If the unit-mnemonic is omitted, the default backup designation (BD) disk will be assumed.

Ordinarily, backup files are removed after they are printed or punched. Where user-named backup file (U.N NAM <name>) are concerned, however, a special condition exists: namely, they are not removed after they are printed or punched out unless the default condition is overridden. The default (no removal) can be changed by setting program switch two (SW2) to a non-zero value. When program switch two is set to a non-zero value (SW2=1), backup files are removed after printing/punching. These conditions also apply when AUTOBACKUP is set.

Format 2:

```
------------------ PB -------------------------- <number> -->>
              !    --  !                      !
              !        !-- <unit-mnemonic> -->!
--- <us/pw> -->!

    >>------------------------------------------------------->>
          !-- <output-device> -->!

    >>----------------------------------------------------------#
       !<---------------------------------------------------!
       !-- SAVE ------------------------------------------->!
       !-- COPIES ---------------- <integer> -------------->!
       !            !-- = -->!                              !
       !                                                    !
       !---- SINGLE --------------------------------------->!
       !  !- DOUBLE -->!                                    !
       !                                                    !
       !-- RECORD --- <integer-1> ------------------------->!
       !                           !              !         !
       !                           !-- <integer-2> -->!     !
       !                                                    !
       !-- KEY --- <compiler-name> --------->>              !
               !                      !                     !
               !- <column> <length> -->!                    !
                                                            !
                                                            !
         >>--- RANGE <string-1> <string-2> ---------------->!
            !                                         !
            !-- EQUAL <string-3> ---------------------!
```

Format 2 causes the printing or punching of one printer or punch backup file, as specified by the integer. When this format is used, options can be used to control the output and action taken by SYSTEM/BACKUP. The options, in alphabetical order, are defined as:

COPIES

Causes SYSTEM/BACKUP to produce <integer> copies of the specified backup file. One copy is the default if this option is not specified.

DOUBLE

Causes SYSTEM/BACKUP to double-space the entire printer listing, overriding any carriage control specified in the backup file.

KEY

Allows specification of a range of records to be printed or punched. A detailed description of the syntax is given next. All records in the file will be printed or punched if this option is omitted.

Use of the KEY option allows specification of a range of records to be printed or punched according to information within the records themselves (e.g., a sequence number). The portion of each record to be compared may be specified, as well as the information that will start and stop the output.

The <compiler-name> option causes automatic generation of the proper column number and length pair that corresponds to the sequence number field of the output listing produced by the specified compiler. The permissible compiler-names that can be used are: BASIC, COBOL, DASDL, FORTRAN, MIL, NDL, RPG, SDL and UPL.

The <column> specifies the beginning column number of the subfield to be used for the compare argument, and <length> specifies a length which must be greater than 0 and less than 10.

The RANGE and EQUAL parameters specify the argument to which the subfield in each record is to be compared, and the action to be taken when a "true" comparison is detected. The strings can be either an integer(s) or an alphanumeric literal(s) enclosed within quote marks. When the comparison arguments are of different lengths, an integer string is either left-truncated or left zero-filled to the same length as the subfield; an alphanumeric literal is either right truncated or right space-filled to the same length as the subfield.

If RANGE is specified, printing or punching begins when an exact comparison is made between the subfield and <string-1>. The printing and punching continues until an exact comparison is made between the subfield and <string-2>, or until the end-of-file is reached, whichever occurs first.

If EQUAL is specified, printing or punching begins when an exact comparison is made between the subfield and <string-3>, and will continue until end-of-file is reached.

If <string-1> is equal to <string-2>, the entire backup file will be searched. Every record in which the designated subfield matches <string-1> is printed or punched.

Since the specified comparisons require an exact match between the string and the subfield, no sequential ordering of the backup file is assumed.

NOTE: If both the RECORD option and the KEY option are specified in the same statement, the comparisons specified by the KEY option will be made only within the range of records specified by the RECORD options. This defines the intersection of the two specifications.

<u>RECORD</u>

Allows specification of a range of records to be printed or punched. Output will begin with the physical record specified by <integer-1> (the first record in the backup file is record number 1) and continue until the physical record specified by <integer-2>. If <integer-2> is omitted, end-of-file is assumed as the terminator. All records in the file are printed or punched if this option is omitted.

SAVE

Causes SYSTEM/BACKUP to leave the backup file on disk when the file is closed. The file will be removed from disk if this option is omitted. Tape backup files are always closed with release, so specification of SAVE is unnecessary.

SINGLE

Causes SYSTEM/BACKUP to single-space the entire printer listing, overriding any carriage control specified in the backup file.

<unit-mnemonic>

If this option is included, it must specify a printer (LP) or card punch (CD or CP), and must be the first option specified following the backup file number. It will cause SYSTEM/BACKUP to direct its output to the designated unit, if that unit is available.

AUTO BACKUP

When Auto Backup is active, a copy of SYSTEM/BACKUP for each designated printer is executed. The names of the existing backup files are entered in a community queue file, AUTOPRINT. The names are read and written to OUTFILE's FPB. The individual file processing is the same as with a simple PB (PB 123). The backup files are removed after printing.

The AUTOBACKUP mechanism may be overridden for a particular backupfile during execution or compilation by specifying NO AUTOPRINT or (NO ATP). NO ATP sets a bit in the FPB called FPB.AUTOPRINT which, when set, prevents AUTOBACKUP from retrieving the specified backup file.

The FPB.AUTOPRINT bit may be set or reset when using file equation. By default, FPB.AUTOPRINT is reset. If the AUTOPRINT bit has been set, ATP resets the bit (allowing AUTOBACKUP to retrieve the file). Note: The AUTOPRINT bit may be dynamically modified (DY-ed) while the file is open.

The syntax for AUTOPRINT is defined as:

```
>-- FILE -- <filename> ------------------ AUTOPRINT --------->#
    --                 |-- NO -->|       |--- ATP ----->|
```

Examples:

```
PB 125

PB 17 LPA SAVE

PB DCC 4 RECORD 5

PB MTA =/=

PB 3 KEY COBOL RANGE 123 567

PB 2 KEY 7 6 EQUAL "ABC"

PB 53 RECORD 1 100 DOUBLE SAVE
```

PD (Print Directory) (#)

```
**********
*   PD   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- PD ---------------------------  =/= -------------------># *
*        |-- <disk-pack-id> -->| |-- <file-name> ---------->|   *
*                                |-- <mfid> --------------->|   *
*                                |-- <mfid>/<file-name> -->|    *
*                                                               *
*****************************************************************
```

The PD input message allows a system operator to request a list of all files on a disk directory or to interrogate a disk directory for a specific file(s).

The =/= format gives a complete listing of files in a specified or system disk directory. For partial or specific listings, the family-name or file-identifier format message is used.

If the file-identifier is not present in the disk directory the MCP will respond with the message: <file-identifier> NOT IN DIRECTORY

Examples:

Does a file named COBOLZ reside on the system pack?

Request: PD COBOLZ

Response: PD = COBOLZ (affirmative response)

What files reside on the system pack?

Request: PD =/=

Response: PD = <file-identifier-1>

PD = <file-identifier-2>

PD = ...

Does a family-name PAYROLL with a file-identifier QUARTERLY reside on a removable pack called MASTER?

Request: PD MASTER/PAYROLL/QUARTERLY

Response PD = MASTER/PAYROLL/QUARTERLY

Do the files ALPHA, BETA, CHARLIE, reside on the system pack?

Request: PD ALPHA, BETA, CHARLIE

Response: PD = ALPHA

PD = BETA

CHARLIE NOT IN DIRECTORY

PG (Purge) ($)

```
**********
*   PG   *
**********
```

Format:

```
***********************************************************
*                                                         *
* >-- PG --- <unit-mnemonic> -------------------------># *
*                             |-- <serial-number> --|     *
*                                                         *
***********************************************************
```

The PG message permits the system operator to purge a removable disk cartridge, disk pack, or magnetic tape.

A disk cartridge/pack that is purged will be marked as UNRESTRICTED with its disk pack-id remaining unchanged.

The serial-number is required when purging a disk, and must be a six-digit number matching the serial number of the pack being purged.

Magnetic tape must have a write ring in place in order to be PURGED.

The serial-number is not used when purging a tape, and the serial number in the tape label will not be changed. To assign or change a tape serial number, use the SN message.

Examples:

```
PG DPA 000456

PG MTC, MTD
```

PM (Print Memory Dump)

```
**********
*   PM   *
**********
```

Format:

```
**********************************************
*                                            *
* >--- PM --------------------------------># *
*            |                     |         *
*            |-- <integer> --------->|       *
*                          |-SAVE-|          *
*                          |- NC -|          *
*                                            *
**********************************************
```

The PM input message allows a system operator to print the entire contents of memory or single program dump file.

A PM by itself will cause the execution of the MCPII/ANALYZER program which will analyze and print the contents of SYSTEM/DUMPFILE. (System Memory)

The "integer" option will cause the execution of the DUMP/ANALYZER program which will analyze and print the contents of DUMPFILE/integer. (Program Memory)

The programs DUMP/ANALYZER and MCPII/ANALYZER (MCPII) must be located on systems disk to perform a PM message.

The SAVE option will cause the DUMP/ANALYZER to leave the specified DUMPFILE on disk at EOJ; without this option, the DUMPFILE will be removed from disk.

The NC option informs DUMP/ANALYZER that the file to be analyzed contains the dump of a network controller.

PO (Power Off) ($)

```
***********
*   PO    *
***********
```

Format:

```
***********************************************
*                                             *
* >--- PO ----- <unit-mnemonic> ----------->| *
*                                             *
***********************************************
```

The PO input message informs the MCP that a removable disk pack or cartridge is to be removed from the system.

A system pack may not be powered off.

A PO message entered for a unit currently being used will cause the MCP to display the following message: <unit-mnemonic> HAS <integer> USERS

A PO message entered for a unit that is not currently in use will cause the message: <unit-mnemonic> MAY NOW BE POWERED DOWN

The PO message may be used on a multipack file base pack if there are no single-pack files in use at the time of the request.

The PO message may be used for any Model-5 tape drives attached to a MTC-4. In which case, the tape will be unloaded.

PP (Processor Priority)

```
**********
*   PP   *
**********
```

Format:

```
************************************************************
*                                                          *
*  >-- <job-number> -- PP ------------- <integer> ------># *
*                         |         |                      *
*                         |-- = -->|                       *
*                                                          *
************************************************************
```

The PP input message sets the processor priority of the specified job to the integer selected only if the MPRI option is set. The MPRI option is part of the Prioritized-Memory management system discussed more fully in Section V of this document. The default value of integer is 4 and the range is 0-15.

PR (Change Priority)

```
**********
*   PR   *
**********
```

Format:

```
**********************************************************
*                                                        *
* >-- <job-number> -- PR ------------- <integer> ----->| *
*                     --  |        |                     *
*                         |-- = -->|                     *
*                                                        *
**********************************************************
```

The PR input message allows the system operator to change or set the processor and memory priority of a program that is executing in a multiprogramming environment.

The default value of integer is 4 and the range of values is 0-15.

PS (Prod Schedule)

```
***********
*   PS    *
***********
```

Format:

```
*************************************
*                                   *
* >------------- PS ----------->#   *
*                --                 *
*************************************
```

The PS input message gives the system operator the ability to request that the MCP attempt to execute the "top" entry in the ACTIVE SCHEDULE.

The normal function of the MCP checks the ACTIVE SCHEDULE at each EOJ or when a program is scheduled. The PS message will cause the MCP to check the ACTIVE SCHEDULE when the message is entered.

QC (Quit Controller) ($) ********** * QC * **********

Format:

```
*************************************
*                                   *
* >------------- QC ------------>| *
*                --                 *
*************************************
```

The QC input message allows the system operator to bring the NDL-generated Network Controller to End-of-Job.

There can be only one NDL-generated Network Controller executing on the system. If any additional Network Controllers are executed the following message will be output:

NETWORK CONTROLLER ALREADY RUNNING

After entering the QC message and all activity in the NDL system has stopped, the Network Controller issues STOP codes to all attached stations and then goes to End-of-Job.

If a station for any reason is unable to receive its output messages, the Network Controller waits indefinitely.

With a MCS in the system, the QC message is invalid and its function should be performed within the MCS.

QF (Query File) (#)

```
**********
*   QF   *
**********
```

Format:

```
****************************************************************
*                                                              *
*  >-- QF --- <program-id> --- <control-attribute-id> ----->#  *
*      --                                                      *
****************************************************************
```

The QF input message allows the system operator to interrogate a program on disk for the status of its control attributes.

Examples:

```
QF A/B CG

QF A/B FILE LINE BACKUP
```

QP (Query Program)

```
***********
*   QP    *
***********
```

Format:

```
******************************************************************
*                                                                *
*  >-- QP --- <job-number> --- <control-attribute-id> ----->|  *
*      --                                                        *
******************************************************************
```

The QP input message allows the system operator to interrogate a program while running on the system for the status of its control attributes.

Examples:

QP 14 PRIORITY

QP 15 CHARGE FREEZE

QU (QUeue)

```
**********
*   QU   *
**********
```

Format:

```
*****************************************************
*                                                   *
* >--- QUEUE -- <name-1> ---------------------->| *
*      --                |                |         *
*                        |-- /<name-2> ----|         *
*                                                   *
*****************************************************
```

The QU input message allows the user to designate a previously opened queue file as a control queue. The control queue provides a mechanism whereby the MCP can communicate to a controlling job on behalf of a spawned job, or in response to other control card commands which the controlling job may wish to receive. For example, the control string "QU X/Y EX DMPALL" will cause the MCP to insert into queue X/Y the following messages:

a. A special schedule record message.

b. A special BOJ message.

c. Eventually, a special EOJ message.

The QU command may also be used to obtain MCP responses to normal SPO input messages. For example, the string "QU X/Y WY" would cause the standard output from the WY command to appear in queue X/Y.

RB RF (Remove Backup Files) (#)

```
**********
*   RB   *
*   RF   *
**********
```

Format:

```
****************************************************************
*                                                              *
*  >--- RB ----------------------------  =/= ----------->#  *
*    |  --  | |-- <disk-pack-id> -->| |<--- , ---------|      *
*    |      |                        |-- <integer> -->|      *
*    |- RF -|                        |-- PRT/= ------>|      *
*       --                           |-- PRN/= ------>|      *
*                                    |-- PCH/= ------>|      *
*                                    |-- DMP/= ------>|      *
*                                                              *
****************************************************************
```

The RB or RF input message gives the system operator the ability to remove backup or dump files on disk.

The integer will remove the backup file (print, punch, or dump) specified by the integer.

The PRT/=, PCH/=, and PCH/= options will remove backup files of that respective type. PRN is equivalent to PRT.

The =/= option will remove all backup files from a specified or a system disk.

The disk-pack-identifier specifies that the backup files to be removed are on the designated removable disk.

RC (Recover Database) ($)

```
**********
*   RC   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >--- RC --- <database-name> ------------------->>             *
*      --                                                       *
*                                                               *
*   >>------------------------------------------------------>#  *
*      |                     |       |                |         *
*      |-- ON -- <pack id> -->|       |<------  ,  ----|         *
*                                    |                |         *
*                                    |-- <file-id> -->|         *
*                                                               *
*****************************************************************
```

The RC message zips the execution of the data base recovery program, identifying the name of the data base that is to be recovered. The type of recovery to be done is defined by the information in the data base dictionary on the status of the data base.

RC allows an optional <pack-id> for those databases where the dictionary resides on a user disk as well as an optional file-list to indicate dump recovery of the specified data base files only. The <file-identifier>s are put into a parameter file similar to a LOAD.DUMP file. The address of this file is passed to the recovery program and SWitch 4 is programatticaily set to one (SW4=1) to indicate recovery of a partial database. There is no file list for Clear/Start, abort recovery, or full-dump recovery.

RD (Remove Pseudo Card Files) ($)

```
***********
*   RD    *
***********
```

Format:

```
****************************************
*                                      *
* >--- RD ---------- &lt;integer&gt; ---->#  *
*      --           |---- =/= ---->|   *
*                                      *
****************************************
```

The RD input message allows the system operator to remove pseudo card files from disk.

RE (REmove) (#)

```
**********
*   RE   *
**********
```

Format:

```
*****************************************************************
*                                                               *
*         |<--------------------,----------------------|        *
*         |                                            |        *
* >-- RE ---------------------- <file-id> ------------------># *
*     --  |-- <pack-id> ->| |--- <mfid>/= ------------->|       *
*                           |--- <mfid>/<family-name> ->|       *
*                                                               *
*****************************************************************
```

The REMOVE input message is used to delete specified file(s) from the disk directory, making the file space available to the MCP. RE may delete any number of files; however, any statement affecting more than one file must have the file-identifiers separated by commas. Once a file has been removed, there is no means of recovering it.

The "/=" form will delete the main directory entry and in turn all the files in its subdirectory. If the file identifier referenced is located on a removable disk pack, the diskpack-id must precede the file identifier in order for the MCP to locate the correct file. When the diskpack-id is not included, the MCP assumes that the file resides on system disk.

RE may be continued to additional cards (lines) with the last "remove" terminated by a semicolon.

Example:

```
Card          CRT SPO             TTY SPO
----          -------             -------

? RE A/A      RE A/A, X,Y, Z/Z/Z  RE A/A, X,Y, Z/Z/Z;
,X,Y,
Z/Z/Z;
```

RH (Reset Halts) ($)

```
(DEBUG Only)
**********
*   RH   *
**********
```

Format:

```
*************************************
*                                   *
* >-- RH ------- S ------------># *
*             |-- M --|             *
*                                   *
*************************************
```

The RH input message allows the operator to reset the CONDITIONAL.HALT flags for the SMCP if the S-option is set and the flags for the Micro-MCP if the M-option is set. The command also requires that the DBUG option is on (set to one).

RL (Relabel User Pack) ($)

```
**********
*   RL   *
**********
```

Format:

```
**********************************************************************
*                                                                    *
* >-- RL --- <unit-mnemonic> --- <disk-pack-id> ---->#               *
*     --                        |------ R ---------|                 *
*                               |------ U ---------|                 *
*                                                                    *
**********************************************************************
```

The RL input message gives the operator the ability to change the disk-pack-id and/or the type of user pack.

RM (Remove Duplicate Disk File)

```
**********
*   RM   *
**********
```

Format:

```
***************************************
*                                     *
* >--- <job-number> -- RM -------->|  *
*                      --             *
***************************************
```

The RM input message allows the system operator to remove a disk file from the disk directory in response to a DUPLICATE FILE ON DISK message.

The DUPLICATE FILE message is a result of a program trying to close a disk output file with the same name as a file already in the directory. This causes the program to go into a wait state. The RM message will remove the old file, close the new file, enter it in the directory, and continue processing.

Example:

```
421 RM
```

RN (Assign Pseudo Readers) ($)

```
**********
*   RN   *
**********
```

Format:

```
**************************************
*                                    *
* >--- RN ------ <integer> ------># *
*      --                            *
**************************************
```

The RN message is used by the system operator to assign a specific number of pseudo card readers.

The RN message can be entered either before or after the creation of pseudo files.

It is the responsibility of the operator to determine the optimum number of pseudo readers in relation to the number of pseudo files to be processed.

By entering RN 0(zero) "all" pseudo card readers will be closed as soon as they are finished processing the file that they are presently reading.

The pseudo card readers may also be closed by performing a Clear/Start.

RO (Reset Option) (S)

```
***********
*   RO    *
***********
```

Format:

```
*******************************************
*                                         *
*               |<------  ,  ---------|   *
*               |                     |   *
* >-- RO -------- <option> ----------->|  *
*     --                                  *
*                                         *
*******************************************
```

The RO message allows the system operator to reset the options used to direct or control some of the MCP functions.

The MCP replies with a verification that the option has been reset after each RO input message.

LOG, CHRG, and SPOL options cannot be reset. The MCP message LOG LOCKED CHRG LOCKED, or SPOL LOCKED will be displayed when an attempt has been made to reset these options.

The TO message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

TIME and DATE options cannot be reset if TABS is set.

Examples:

```
RO LIB

   LIB=0

RO DATE, TIME

   DATE=0
   TIME=0
```

RP (Ready and Purge) ($)

```
**********
*   RP   *
**********
```

Format:

```
**********************************************
*                                            *
*          |<--------- , ----------|         *
*          |                       |         *
* --- RP ----- &lt;unit-mnemonic&gt; ------------># *
*                                            *
**********************************************
```

The RP message entered by the system operator will set a tape unit in READY status and PURGE the tape.

The RP message can be used for tape only.

Examples:

RP MTA

RP MTC, MTD

RS (Remove Jobs from Schedule)

```
**********
*   RS   *
**********
```

Format:

```
*********************************************
*                                           *
*  >-- RS ---------- =/= -------------->#   *
*      --     |                     |       *
*             |<----- , ----------|        *
*             |-- <job-number> -->|         *
*                                           *
*********************************************
```

The RS input message will allow the system operator to remove a job from the schedule prior to its being entered in the MIX for execution.

The RS message can remove one or more jobs from the schedule.

The schedule number is the number assigned to the job by the MCP when it is entered into the schedule.

The job-number will be displayed by the MCP when the job is entered into the schedule if the SCHM option is set. The WS message will display the jobs in the schedule together with their job-numbers.

The "=/=" option will remove all jobs from the schedule.

If the requested program(s) are not in the schedule, the MCP will notify the operator that an invalid request has been entered.

Example:

```
RS  33, 34, 35, 36

#33 RS-ED
#34 RS-ED
#35 RS-ED

36 NULL SCHEDULE      (job 36 not in schedule)
```

RT (Remove Multipack File Table) ($)

```
**********
*   RT   *
**********
```

Format:

```
************************************
*                                  *
* >-- RT --- &lt;file-id&gt; --------># *
*     --                           *
*                                  *
************************************
```

The RT input message allows the operator to remove an entry from the multipack file table before Clear/Starting.

Examples:

```
RT USER/A/B

RT BASEPACK/MASTER/
```

RV (Pack Override) ($)

```
**********
*   RV   *
**********
```

Format:

```
***********************************************
*                                             *
* >--- US <usercode/password> --- RV ------># *
*                                             *
***********************************************
```

The RV message is used to reset the pack override bit for individual entries in the (SYSTEM)/USERCODE file. It will reset the override bit for all occurrences of that usercode.

The override bit is automatically set ON when a user (usercode/password) signs on and the default pack, specified in the (SYSTEM)/USERCODE file, is not on-line. RV is used to reset (i.e., return to default) the override condition.

If the override bit is set for a particular usercode/password, an OV, indicating that the bit has been set, precedes the usercode-index entry in a listing of the (SYSTEM)/USERCODE file.

RX (Remove eXcept) ($)

```
**********
*   RX   *
**********
```

Format:

```
*****************************************************************************
*                                                                           *
* >-- RX --- <unit-mnemonic> ---------------------->>                       *
*          |               |                                                *
*          |- <pack-id> ------>|                                            *
*                                                                           *
*                                                                           *
*       |<-------- , -------------|                                         *
*       |                         |                                         *
*    >>------ <file-id> -------------------------------------->#           *
*       |---- <mfid>/<file-id> -->|  |             |                        *
*       |---- <mfid>/= ---------->|  |-- ? END -->|                         *
*                                                                           *
*****************************************************************************
```

All files on the specified disk will be removed unless the USER.COUNT is greater than zero, the file's permanent type equals 3F3, or the file name is included in the name list in the RX message. Any I/O error will abort the operation.

Note: RX must terminate with a semicolon or an ?END card. If a card reader exception occurs, e.g., the reader jams, RX can be stopped by input of a card with illegal syntax, e.g., RX A/B/C/D, thus forcing RX to abort. If the MCP reads an ETX before a semicolon or an ?END card, it returns an error message and terminates the command.

Note: The current release of MCPII allows the pack's serial number to be specified; in future releases it will be required.

RY (Ready Peripheral) (S)

```
**********
*   RY   *
**********
```

Format:

```
***********************************************
*                                             *
*          |<------  ,  ------------|         *
*          |                        |         *
* >-- RY --- <unit-mnemonic> -------------->| *
*                                             *
*                                             *
***********************************************
```

The RY input message allows the system operator to ready a peripheral unit and make it available to the MCP.

Any number of units may be made ready with one RY message.

When a removable disk cartridge is attached to Disk Cartridge Control-1, the MCP must be notified of its presence with the RY message.

If the designated unit is not in use and is in the remote status, the RY message causes all exception flags maintained by the MCP for the specified unit to be reset. After the unit has been made ready, the MCP attempts to read a file label (input devices only).

Examples:

```
RY DPB
RY MTC
RY LPA, LPB
```

INITIAL LETTERS S-Z

SD (Assign Additional System Drives) ($)

```
**********
*   SD   *
**********
```

Format:

```
***************************************************************
*                                                             *
*  >-- SD --- <unit-mnemonic> --- <serial-number> --->|       *
*      --                                                     *
***************************************************************
```

The SD input message gives the system operator the ability to assign additional system drives for the MCP.

The SD message, after verification of the serial-number, will PURGE the pack, and add it to the system packs already on the system.

At COLDSTART, there is only one system drive, so additional drives may be added by the SD message. Once a system has been added to the system, it "cannot" be removed without performing a COLDSTART.

The following message is displayed when the new system drive is linked to the system.

<unit-mnemonic> IS NOW A SYSTEM PACK--CLEAR START REQUIRED

Note: SD only applies to removable packs and cartridges. All system drives must be of the same device type.

SE (Switch Enable) ($)

```
**********
*   SE   *
**********
```

Format:

```
*****************************************************
*                                                   *
*  >-- SE ---- &lt;integer&gt; ----------------------->| *
*      --                                           *
*****************************************************
```

The value of the integer specifies the programs that are allowed to sense the Data Entry Switches, as follows:

Value of Integer	Program
0	Disable Switches
1	MCP
2	Normal-state Programs
4	Interpreters
8	GISMO

The integer may also be any sum of the above values, in which case multiple classes of programs are allowed to sense the Data Entry Switches.

Only SDL/UPL and MIL programs contain the source language constructs required to sens the Data Entry Switches. In both languages the CONSOLE SWITCHES construct is used, and provides the 24-bit image of the Data Entry Switches.

Examples:

SE 2

SE 10

SH (Set Halts) ($)

(DEBUG Only)

```
**********
*   SH   *
**********
```

Format:

```
*************************************************
*                                               *
*  >-- SH -------------------- = ------------># *
*      --     |-- S -->| |-- <integer> -->|     *
*             |-- M -->|                        *
*                                               *
*************************************************
```

The SH input message allows the operator to force the SMCP (S) or the MMCP (M) to halt at CONDITIONAL.HALTs coded into the MCP. The integer option requests a specific halt and must be a six-digit hex number, e.g., @393240@.

The "=" option causes the MCP to halt at all CONDITIONAL.HALTs.

Note: DBUG must be set.

SL (Set LOG) ($)

```
**********
*   SL   *
**********
```

Format:

```
*******************************************************************
*                                                                 *
*  >-- SL -------------- <integer-1> ------------------------>#  *
*          |          |            |                       |      *
*          |-- SPO -->|            |-- <integer-2> -->|      *
*                                                                 *
*******************************************************************
```

The SL input message gives the operator the ability to set the LOG or SPOL(OG) options, and allocate the area required.

The integer-1 entry is the size of each area to be assigned to the LOG and cannot be less than 100 or greater than 1000 disk segments.

The integer-2 entry is optional and is the maximum number of area desired. It must be between 2 and 105, inclusive. Default is 25.

The MCP responds with the following message when an SL message has been entered:

LOG SPOLOG] NOW SET-CLEAR START REQUIRED

If there is not sufficient disk space for the first area of the log, the following message will be displayed:

NO SPACE TO BUILD LOG SPOLOG]

If integer-1 is zero and TABS is not set, the LOG SPOL] option will be reset and the log will be transferred (as though a TL message had been entered). A new SYSTEM/LOG SYSTEM/SPOLOG] is not created.

Examples:

```
SL 1000
SL 250 5
SL 0
SL SPO 100 105
```

SM (Set Database Parameters) ($)

```
***********
*   SM    *
***********
```

Format:

```
*****************************************************************
*                                                               *
* >--- SM --- <data-base-name> ---------------------------->>   *
*      --                      |-- ON -- <pack-id> -->|         *
*                                                               *
*   >>------------------------------------------------------->#  *
*      | |-- SYNCPOINT ----->|   |-- <value> -->|    |          *
*      | |-- CONTROLPOINT -->|                        |          *
*      |                                              |          *
*      |-- AUDIT ---------------------------------->|          *
*                        |-- SET ----->|                        *
*                        |-- RESET --->|                        *
*                                                               *
*****************************************************************
```

The SM input message allows the operator to interrogate or set the database parameters, but only when the database is not active. The pack option is required if the database dictionary resides on a user pack. If a <value> is specified for CONTROLPOINT and SYNCPOINT or if SET or RESET is specified for AUDIT, the parameter is updated in the dictionary; otherwise the parameter is interrogated and displayed. Appropriate ranges are checked.

SN (Assign a Tape Serial Number) ($)

```
**********
*   SN   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- SN -- &lt;unit-mnemonic&gt; -- &lt;serial-number&gt; ------------># *
*     --                                    |- ASCII -|         *
*                                           |- ODD ---|         *
*                                                               *
*****************************************************************
```

The SN input message is used to initialize (and purge) a magnetic tape, assigning a volume serial number to the tape label.

The SN message initialize a magnetic tape by putting a scratch ANSI label on the tape. Any tape that does not contain a valid ANSI label cannot be purged (PG message). This includes both unlabeled tapes and those that have the Burroughs standard label.

The serial-number is normally numeric, but any alphabetic or numeric string up to six characters in length is allowed. This serial number is placed in the label, and remains in the label even when the tape is purged. The serial number can be explicitly changed by another SN message.

The ASCII option forces the MCP to translate all labels on the tape to ASCII before writing them.

The ODD option requests that the MCP write all volume and file labels in odd parity.

NOTE: MARK 6.0 MCPII does not recognize tapes generated with even parity labels. All tapes generated on MARK 6.1 which require MARK 6.0 accessibility should be created with odd parity labels.

Example:

```
SN MTA 123456
```

SO (Set Option) ($)

```
**********
*   SO   *
**********
```

Format:

```
*********************************************
*                                           *
*            |<------- , --------|          *
*            |                   |          *
*  >--- SO ----- <option-name> --------->#  *
*                                           *
*                                           *
*********************************************
```

The SO input message allows the system operator to set the options used to direct or control some of the MCP functions.

The MCP replies with a verification that the option has been set after each SO input message.

The LOG and SPOLOG options cannot be set with an SO message. The MCP message "LOG LOCKED" or "SPOL LOCKED" will be displayed when an attempt has been made to set these options with an SO message.

The TO input message may be entered to determine which options are set at any given time. The option indicator equals one when set and zero when reset. A complete list of the MCP options and their status will be displayed.

TIME and DATE will be set automatically when the TABS option is set. TABS cannot be set if LOG is not set.

SP (Change Schedule Priority)

```
**********
*   SP   *
**********
```

Format:

```
****************************************************
*                                                  *
*  >--- SP -- <job-number> --- <integer> ------->| *
*                                                  *
****************************************************
```

The SP input message provides a means for the system operator to change the schedule priority of a program currently in the schedule.

The schedule priority is separate from the job when it is in the mix.

The job-number will identify the program in the schedule that is to be affected by the SP message.

The integer in the SP message specifies the new priority that will be assigned to the program. Priorities may range from zero through fourteen, where zero is the lowest priority, fourteen is the highest priority, and four is the default.

To change the priority of a program in the schedule with a job-number of 33 to a priority of 7, for example, the following message would be used.

```
SP 33 7
```

This program would be selected from the schedule ahead of the programs with a lower priority.

The following message would be displayed after the MCP has rescheduled the priority of the job specified:

```
#33 SCHEDULE PRIORITY CHANGED TO 7
```

SQ (Squash Disk) ($)

```
**********
*   SQ   *
**********
```

Format:

```
**********************************************************
*                                                        *
*  --- SQ ---- <unit-mnemonic> ----------------------># *
*      --  |---- DKx ---------->| |-- <integer1> -->| | |  *
*                                                   | |  *
*  |<-----------------------------------------------| |  *
*  |------------------------------------------------->|  *
*                    |<-------------------------|        *
*                    |-- SIZE ----------------->|        *
*                    |-- STOP ----------------->|        *
*                    |-- TIL -- <integer2> ---->|        *
*                    |-- TRACE -- <integer> --->|        *
*                                                        *
**********************************************************
```

The SQ input message permits the system operator to initiate or terminate a "disk squash". When "squashing" a disk, the MCP attempts to move areas of data to numerically-lower disk addresses in order to alleviate disk checkerboarding.

The unit-mnemonic specified must be a disk device (DC, DK, or DP). If head-per-pack disk (DKx) is designated, integer-1 must be used to indicate the electronics unit (EU); integer-1 is not used with other types of disk.

A disk squash cannot be initiated if there are any jobs in the mix or in either schedule (WAITING or ACTIVE). Once the disk squash is initiated, only additional SQ input messages can be entered on the console printer. All card readers are inaccessible during the disk squash.

Both system and user disks can be squashed. With multiple system disks, only one drive or EU can be squashed at a time. The MCP automatically produces a KA listing of the specified disk both before and after the disk squash; therefore, a line printer must be available.

If the TIL option is specified, the MCP will terminate the disk squash as soon as integer-2 contiguous sectors have been made available. Otherwise, the squash will continue until normal termination or until explicitly stopped by the operator with the STOP option. The MCP displays the largest area currently available whenever interrogated by the SIZE option. "TRACE 1" prints tracing information on the printer, which may be helpful for post-mortem analysis, if the MCP fails.

When the disk squash is terminated, the MCP displays the size of the largest area as well as the total number of sectors available on the designated disk.

Examples:

```
SQ DPA

SQ DKB 1

SQ DCC TILL 5000

SQ DPB SIZE

SQ DCA STOP
```

If the disk is in bad shape, SQ may halt in the middle, and a memory dump at that point may be inadequate for post-mortem analysis, unless "TRACE 1" has been specified. It is advisable to run DISKMAP/UTILITY before issuing a SQ.

ST (Suspend Processing)

```
**********
*   ST   *
**********
```

Format:

```
*********************************************
*                                           *
*  --- <job-number> -- ST -------------->|  *
*                      --  |          |     *
*                          |-- EOJ -->|     *
*                                           *
*********************************************
```

The ST input message provides a means for the system operator to temporarily suspend the processing of a program in a multiprogramming environment. The job-number identifies the program to be suspended.

The MCP will not suspend the program until all I/O operations in progress for that program have been completed. When the MCP suspends a program, it is rolled-out to disk and the memory it was using is returned to the MCP for reallocation. A suspended program will retain the peripherals assigned to it.

The EOJ option stops the program indicated by <job-number> and automatically restarts it when another program goes to EOJ. ST EOJ can be used when the system gets into a thrashing state. The operator enters the ST EOJ message to stop jobs until the mix is not thrashing. These jobs will then be automatically resumed after other jobs go to EOJ. A <job-number>GO message will restart a stopped job.

To restart a program after it has been suspended, the GO message must be used. If for some reason all of the conditions necessary for the program to run are not met when the GO message is issued, the MCP will not restart the program.

Example:

```
823 ST

947 ST EOJ
```

SV (Save Peripheral Unit) ($)

```
**********
*   SV   *
**********
```

Format:

```
*********************************************
*                                           *
*            |<--------  ,  --------|       *
*            |                      |       *
*  >--- SV ----- <unit-mnemonic> ------->#  *
*                                           *
*                                           *
*********************************************
```

The SV message allows the system operator to make a peripheral unit inaccessible to the MCP until a Clear/Start operation occurs, or an RY input message is used to ready the unit.

Any number of peripheral units may be saved with one SV input message.

When the SV message is entered and the unit is not in use, the specified unit is marked SAVED and "unit-mnemonic SAVED" is displayed by the MCP.

If the unit is in use, the MCP will respond with "unit-mnemonic TO BE SAVED" and will save the unit as soon as it is no longer being used.

Example:

```
SV LPA, CRA
```

SW (Set Switch)

```
**********
*   SW   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- <job-number> -- SW --------  =  ----------- <value> --->#  *
*                    --  |-- <integer> ------->|                *
*                        |              |- = -| |               *
*                                                               *
*****************************************************************
```

The SW input message allows the system operator to set programmatic switches.

Program switches may also be set at schedule time by using the SWITCH control statement attribute.

The number must be a decimal digit from zero to nine (0-9) that references the switch to be set. To determine what switches are available, the specific language manual for the program for which the switches are being set must be referenced. If the "=" option is used, all ten switches are implied (40 bits of information).

The value is the value that the switch or switches will be assigned.

Examples:

```
25 SW1 = @F@

2 SW8 = 6

1473 SW  = @0123456789@
```

SZ (Session ZIP) (S)

```
***********
*   SZ    *
***********
```

Format:

```
****************************************
*                                      *
* >-- SZ ------ <integer> -------->| *
*     --                               *
****************************************
```

The SZ input message is mainly an RJE control card. It should not be invoked in the course of simple job spawning. The session number generated by an SZ command is carried with a "zipped" control string and applied to a job or mix of jobs depending on what other commands are in the control string. The primary function of the session number is to associate independent jobs into logically related groups. For example, HOST RJE assigns a session number to a physical site at log-on time and all jobs submitted from that site will contain identical session numbers, thus relating them to the originating remote site.

TD (Time and Date)

```
***********
*   TD    *
***********
```

Format:

```
*************************************
*                                   *
*  >--- TD ------------------->#    *
*       --                          *
*************************************
```

The TD input message allows the system operator to request that the MCP type the current values of the time and date.

The MCP displays the date and time in the following format:

```
DATE = mm/dd/yy   TIME = hh:mm:ss.t
```

Where:

```
hh - hours
mm - minutes
ss - seconds
 t - tenths of seconds
```

TG (Trace Gismo) ($)

(DEBUG Only)

```
**********
*   TG   *
**********
```

Format:

```
****************************************
*                                      *
* --- TG ---------------------->#      *
*     --    |-- <integer> -->|         *
*                                      *
****************************************
```

The TG input message allows the operator to set or interrogate a group of flags that instruct GISMO to trace various events on the system. TG, without parameters, will return the current settings of the trace flags. TG integer will set the flags.

The integer maps onto a 24-bit field used by GISMO to determine which functions will be placed in the trace table. If a bit is on, then the corresponding function is traced. The field is mapped as follows:

Bit	Function
0	Channel 0
1	Channel 1
	.
	.
	.
14	Channel 14
15	GISMO functions
16	Port activity
17	User activity
18	MMCP activity
19	SMCP activity

Note that the system must be Clear/Started with the debug versions of GISMO and MMCP in order to invoke the use of these parameters. The DBUG option must also be set.

TI (Time Interrogation)

```
**********
*   TI   *
**********
```

Format:

```
***************************************
*                                     *
*  >-- <job-number> --- TI ------>#   *
*                       --            *
***************************************
```

The TI input message allows the system operator to interrogate the MCP as to the amount of processor time the program has used up to the time the interrogation was made.

The job-number identifies the program for which the interrogation was requested.

The time is given in hours, minutes, seconds, and tenths of seconds.

Example:

```
84TI

COBOL: A/B = 84 CPU TIME = 00:03:15.7
```

TL (Transfer LOG) (S)

```
**********
*   TL   *
**********
```

Format:

```
*************************************
*                                   *
*  >--- TL -------------------->#   *
*       --    |-- SPO -->|          *
*                                   *
*************************************
```

The TL input message allows the system operator to transfer the information in the SYSTEM/LOG or SYSTEM/SPOLOG to LOG/#<integer> or SPOLOG/#<integer>. To print the LOG or SPOLOG, refer to the LG command.

TO (Display Options) ($)

```
**********
*   TO   *
**********
```

Format:

```
********************************************
*                                          *
* >-- TO -------------------------------># *
*     --    |                     |        *
*           |<-------  ,  --------|        *
*           |                     |        *
*           |-- <option-name> --->|        *
*                                          *
********************************************
```

The TO input message allows the system operator to interrogate the status of the MCP options.

The TO message entered by itself will display all of the options and their settings.

A value of zero (0) indicates a reset (off) condition; a value of one (1) indicates a set (on) condition.

The DBUG option allows the following messages to be used:

```
CN
CS
CT
DM-----(without a job-number)
RH
SH
TG
```

Example:

```
TO LOG, TIME

   LOG = 1

   TIME = 0
```

or:

	OPTION		DEFINITION: WHEN OPTION IS SET
TO	LOG	-	Creates a disk file recording SPO activity.
	CHRG	-	Requires jobs to be charged to "account numbers".
	LIB	-	Reports all file maintenance activity.
	OPEN	-	Reports all file opens.
	TERM	-	DSes any job that would normally get a DS or DP message.
	TIME	-	Requires time to be reset at Clear/Start.
	DATE	-	Requires date to be reset at Clear/Start.
	CLOS	-	Reports all file closes.
	PBT	-	Sends printer backup files to tape, by default.
	PBD	-	Sends printer backup files to disk, by default.
	BOJ	-	Reports beginning of job.
	EOJ	-	Reports end of job.
	SCHM	-	Reports all entries into the schedule.
	LAB	-	Reports all labels.
	RMOV	-	Removes duplicate disk files.
	DUMP	-	Creates a file called SYSTEM/DUMPFILE which allows memory dumps to be taken at Clear/Start.
	ZIP	-	Reports all ZIP communicates.
	MEM	-	Reports all NO MEM conditions.

DMS - Allows DMS to be run after the next Clear/Start. If this option is set, the initializer binds into GISMO the code that allows DMS activity.

TRMD - Same as TERM except does a DP instead of a DS. Takes precedence over TERM.

DBUG - Enables the messages listed above.

DISP - Reports Segment and Displacement on all DS-DP requests.

SPOL - Creates a disk file of SPO (TTY) activity.

RMSG - Forces all messages to the local SPO.

SQRM - Disables micro-coded QUEUE and REMOTE file handling.

BREL - Reports the release of backup files on the SPO. Users who do not want this information reported should not set this option.

TABS - Activates TABS system. The LOG option must be set also.

THR - Invokes Level Two of the new memory management system after the next Clear/Start. If the THR option is set, the initializer binds into GISMO the code to detect thrashing.

MPRI - Invokes Prioritized-Memory Level of the new memory management system after the next Clear/Start. If the MPRI option is set, the initializer binds into GISMO the code to detect thrashing as well as the prioritized "victim selector".

VLIO - Allows specification and display of activity on the IO subsystem.

VLCP - Displays CPU usage and overlay activity.

BRGR - Sets "bar graph" display, on B1800 only, to show the activities of the state lamp. If both VLIO and VLCP are reset, this option is also considered reset.

FLMP - Must be set at Clear/Start for console lamps.

TR (Time Change) ($)

```
***********
*   TR    *
***********
```

Format:

```
**************************************
*                                    *
*  >-- TR ----- <integer> -------->#  *
*      --                            *
**************************************
```

The TR message allows the system operator to change the current value of the time maintained by the MCP.

The time specified by the integer is designated according to a 24-hour clock, and must be four digits in length.

This message is not accepted by the MCP if the value of the integer is greater than 2400 hours or the value is not numeric.

Example:

TR 1919

Sets the time in the MCP to 19:19

Time cannot be reset, also, when the TABS option is set and there are jobs either executing or scheduled.

TS (Test Switches)

```
**********
*   TS   *
**********
```

Format:

```
***************************************
*                                     *
*  --- <job-number> --- TS ------>#   *
*                       --            *
***************************************
```

The TS input message allows the system operator to test the program switches set by the SW console message or the SWITCH control statement attribute.

The output of the TS message is in hexadecimal format.

Example:

```
94 TS

PAYROLL/103=94 SWITCHES=@0123456789@
```

UL (Assign Unlabeled File)

```
**********
*   UL   *
**********
```

Format:

```
*****************************************************************
*                                                               *
* >-- <job-number> -- UL --- <unit mnemonic> ---------------->| *
*                     --                     |-- integer -->|   *
*                                                               *
*****************************************************************
```

The UL message allows the system operator to designate the unit on which a particular unlabeled input file is located in response to a "FILE NOT PRESENT" message from the MCP.

The UL message is used only if the unit designated is to be acted on as an unlabeled file. The MCP assumes the file on the designated unit is the file requested by the program that caused the "FILE NOT PRESENT" message.

The job-number must be used to identify the program to which the file is to be assigned. If integer is used, the MCP spaces forward "integer" blocks prior to reading the first data block into the object program. Tape marks are read as blocks. This is done at the time the file OPEN is performed. Zero is treated as a null entry, i.e., <job-number> UL MTA 0 is treated as UL MTA.

Example:

A program with a job-number of 1 calling for an unlabeled input tape file could be assigned a tape on a unit with the unit-mnemonic of MTA with the following UL message:

    1 UL MTA

If the first three blocks on the tape are not desired, they can be skipped with the following UL message:

    1 UL MTA 3

UN (UNLOAD) (#)

```
**********
*   UN   *
**********
```

Format:

```
******************************************************************
*                                                                *
* >--- UNLOAD ------------------------------------------- ; -># *
*      --        |-- FROM <disk-pack-id> -->|  |         |       *
*                     --                       |         |       *
*  <-------------------------------------------|         |       *
*  --> TO <library-tape-id> -- =/= -------------------->|       *
*      --                   |<---------- , -------------|       *
*                           |- <mfid> ------------------|       *
*                                    |- /= ----------->|       *
*                                    |- /<filename> -->|       *
*                                                                *
******************************************************************
```

The UN (UNLOAD) statement will cause one or more disk files to be placed on a library tape. The file is removed from disk by this command. The =/= option causes every file on the specified disk is to be dumped. The family-name/= option causes every file with the specified family-name to be dumped. A maximum of 2248 files can be handled with one UNLOAD statement.

Example:

```
?UN TO SYS.TAPE
PAYROLL/=,
ACCPAY/=,
MASTER/FILE;

UN FROM USER TO BACKUP =/=
```

US (USER)

```
**********
*   US   *
**********
```

Format:

```
*********************************************************************
*                                                                   *
* >-- USER ----------- <usercode> ---------------------------->#    *
*     --   |        |             |<-----------------|              *
*          |-- = -->|             |-- /<password> -->|              *
*                                 |----- /= -------->|              *
*                                                                   *
*********************************************************************
```

The US input message provides a way of invoking the file security mechanism and its associated naming convention. The USER command causes the MCP to verify the usercode (and/or password) against the system's usercode/password file. The usercode is carried with the control string and is used to obtain the information to apply the RJE naming convention to any subsequent file name reference. For example, the string "US AL PD =/=" would generate a name in the PD of <PACK-ID>/(AL)/=, where the pack-id and the usercode (AL) were obtained from the system usercode file. In addition, the usercode index is stored in the Run Structure Nucleus of jobs zip-executed with a USER command and is used to apply the RJE naming convention to any files the job may open during the execution.

WD (Display MCP Date)

```
**********
*   WD   *
**********
```

Format:

```
************************************
*                                  *
* >--- WD --------------------># *
*      --                          *
************************************
```

The WD input message permits the system operator to request the current date used by the MCP.

WM (Display Current MCP and Interpreter)

```
**********
*   WM   *
**********
```

Format:

```
************************************
*                                  *
* >--- WM -------------------->#  *
*      --                          *
************************************
```

The WM input message allows the system operator to inquire which system software is currently being used since there can be more than one MCP, Interpreter, etc. residing on the system pack.

The reply to the WM message is in the following format:

```
MCP = <mcp-name><version-number> INTERP = <interpreter-name> GISMO =
<gismo-name> INIT = <initializer-name>  MICRO.MCP = <micro.mcp-name>
NETWORK.CONTROLLER = <controller-name> USERCODE.FILE= <usercode-
file-name>
```

GISMO also contains:

```
<segment-name-1>
<segment-name-2>
      .
      .
      .
<segment-name-n>
```

WS (Display Schedule)

```
**********
*   WS   *
**********
```

Format:

```
****************************************
*                                      *
*  >--- WS --- <job-number> -------->#  *
*       --   |----- = -------|          *
*                                      *
****************************************
```

The WS input message allows the system operator to interrogate what program or programs are currently in the schedule and their status.

The job-number is assigned by the MCP as the program is entered into the schedule.

The MCP response to the WS message gives the program-name, schedule number, memory required in KB's, program priority, and the length of time the program has been in the schedule.

Example:

```
WS 4

ALPHA = 4 NEEDS 8 KB PR = 4 IN FOR 00:08:37.4
```

WT (Display MCP Time)

```
***********
*   WT    *
***********
```

Format:

```
************************************
*                                  *
*  >--- WT ------------------->#   *
*       --                         *
************************************
```

The WT input message permits the system operator to request the current time used by the MCP. The reply is in the twenty-four hour clock format.

WW (List Contents of NAME TABLE) ($)

```
***********
*   WW    *
***********
```

Format:

```
*************************************
*                                   *
* >--- WW ------- =/= ----------->#  *
*      --       |-- N/= -->|        *
*               |-- G/= -->|        *
*               |-- I/= -->|        *
*               |-- M/= -->|        *
*               |-- S/= -->|        *
*               |-- MM/= ->|        *
*               |-- C/= -->|        *
*               |-- U/= -->|        *
*************************************
```

The WW input message gives the operator the ability to list the different types of system software/firmware in the NAME/TABLE. It lists any individual entry with its appropriate id, e.g., WWMX lists the name of the Experimental MCP. It lists group entries through the /= syntax, e.g., WWM/= lists all MCP entries: M, MT, and MX. WW=/= will list the entire NAME TABLE.

WY (Program Status Interrogation)

```
**********
*   WY   *
**********
```

Format:

```
******************************************
*                                        *
* >--------------------- WY ------->#   *
*     |-- <job-number> --|  --           *
*                                        *
******************************************
```

The WY message allows the system operator to check the current status of one program or all the programs in a multiprogramming mix of jobs.

The job-number identifies the program in the mix that is to be checked and its status displayed on the console printer (SPO). If the job-number is omitted, the MCP will display the status of every program in the mix.

If the program is waiting for some type of operator action, the alternatives available to the operator will be identified.

Examples:

```
421WY

    PAYROLL/PAY105=1 IL UL DS-NO FILE "PAYROLL/MASTER"

WY

    COBOL; LISTER=1 EXECUTING

    DMPALL=2 AX DS-WAITING FOR KEYBOARD INPUT

    USER/ACCPAY/=3 WAITING FOR I/O COMPLETE ON "CARDS" (CRA)
```

XC XD (Remove Segments) ($)

```
**********
*   XC   *
*   XD   *
**********
```

Format:

```
****************************************************************
*                                                              *
* >--- XC --- <unit-mnemonic> -----------------------------#   *
*   |- XD -|                                       |    |      *
*                                                  |    |      *
*  |<----------------------------------------------|    |      *
*  |------------------- <integer-2> --- <integer-3> -->|      *
*   |-- <integer-1> ->|                                        *
*                                                              *
****************************************************************
```

The XC and XD input message allow the removal of contiguous disk segments from the MCP tables of available disk space temporarily (XC) or permanently (XD).

The unit-mnemonic specified must be a disk device (DC, DK, or DP). If head-per-track disk (DKx) is specified, integer-1 must be used to indicate the electronic unit (EU). Integer-1 is not used when other types of disk are specified.

Integer-2 specifies the beginning segment address, and can be expressed in any format. If the operation is being performed on a disk cartridge (DCx), and the beginning segment address is not the address of the first segment in a track, the MCP will automatically adjust it backward to the beginning of the track, and will remove the entire track.

Integer-3 specifies the number of segments to be removed from use by the MCP. If the operation is being performed on a disk cartridge (DCx), the number of segments removed will always be a multiple of entire tracks. The MCP will make necessary adjustments, both in starting address and number of segments.

The disk space to be removed must be available in order to be removed; therefore, if any portion of the space is occupied by files or headers, for example, the MCP will reject the request with the message:

REQUESTED SEGMENTS NOT REMOVED-NOT AVAILABLE

The requested disk space is permanently removed from use by the XD message. To return the removed segments a disk initialization (for disk packs or cartridges) or COLDSTART (for head-per-track disk) is required.

The XC message temporarily removes the disk space from use. The disk space is returned at the next CLEAR/START or, for user packs or cartridges, when the disk is powered down (PO message).

Examples:

```
XC DKA 0 200 1000

DISK SPACE REMOVED FROM @EE00000C8@ THRU @EE00004AF@

XC DCC @46@ 30

DISK SPACE REMOVED FROM @EA2000040@ THRU @EA200007F@
```

XM (Remove Memory From System) ($)

```
**********
*   XM   *
**********
```

Format:

```
*************************************************************************
*                                                                       *
* >--- XM ---------------------------------------------------->#        *
*          |                                         | |                *
*          |--  -  ------------------------------->| |                  *
*                    |                             |   |                *
*                    |-- <begin-address> -->|          |                *
*                                                      |                *
*          |<------------------------------------------|                *
*          |                                           |                *
*          |-- <begin-address> -- <length> ------>|                     *
*                                                                       *
*************************************************************************
```

XM <begin address> <length> input message adds the address-length pair to the XM table on disk with a bit on to indicate that this XM will take effect at the next Clear/Start. The length is specified in bytes.

XM- <begin address> <length>] removes the entry from the XM table which has the same <begin address>. The <length>, if present, is ignored.

XM- removes all entries from the XM table.

XM lists all entries in the XM table including whether each entry is currently active, will be active on the next Clear/Start, or could not be activated on the last Clear/Start. Any temporary XM entered through the toggles at Clear/Start time will also be listed along with its temporary status.

No attempt is made to prevent the entry of overlapping or duplicate entries in the XM table. The advantage of this is that the user can reverse any entry he has made by simply entering the same entry preceded by a "-". If, instead, the entries in the XM table were combined to prevent any overlapping areas, a user wishing to reverse his last entry would have to carefully consider how to return the table to its prior state. In fact, unless the user listed the XM table before each new XM message it would sometimes be impossible to return the table to its prior state by inspection of its current state and the last XM message entered.

The XM table provides a complete list of all areas of memory which the initializer will attempt to remove from use at the next Clear/Start. Note, however, that the XM table may be incomplete in describing current memory since entries which have been deleted from the XM table may have been removed from use at the last Clear/Start.

ZQ (Zip Queue)

```
**********
*   ZQ   *
**********
```

Format:

```
**************************************************
*                                                *
*  >-- ZQ --- <name-1> ----------------------->| *
*      --              |<-----------------|      *
*                      |-- /<name-2> --->|       *
*                                                *
**************************************************
```

The ZQ message is similar to QU(EUE) except that the queue specified is used exclusively for scheduling messages and data-card messages. Where the control queue may contain many messages concerning jobs, MCP responses to SPO commands, etc., the ZIP QUEUE command will only contain schedule records of jobs zip-executed by the controlling program and/or the data-card label message(s) if a DATA control card was encountered in the zip string.

This effectively allows the controlling program to be immediately aware of the fact that a job has been scheduled without having to scan through the general control queue for pertinent messages. In general, the control queue is designed for general communication, while the zip queue is specifically to be used for job-spawning control.

INDEX