RSX-11D SPEC

TO: RSX-11D Distribution

FROM: H. Krejci

DATE: 21 Jun 72

SUBJ: THE MONITOR CONSOLE INTERFACE

DOC: 130-101-044-00

The material included in this functional specification, including but not limited to, instruction times and operating speeds is for information purposes only. All such material is subject to change without notice. Concequently DEC makes no claim and shall not be liable for its accuracy.

Unless specified otherwise, the terms "RSX" and "RSX-11" imply "RSX-11D".

THE MONITOR CONSOLE

The operator interface with an RSX-11D system is via a TTY Terminal[1], and is accomplished by mechanism called the MCR (for Monitor Console Routine), and the TTY Terminal is called the MCR Terminal, or the MCR TTY.

The MCR is implemented as a set of Tasks. One is a part of the resident executive, and is called the MCR Dispatch Task. The others are called MCR Function Tasks. The Dispatch Task reads a line of input and REQUESTs the appropriate Function Task which performs the actual MCR Function.

The Dispatch Task is imbedded within the executive, and the MCR Function Tasks normally all share a core partition

[1] A TTY Terminal may be an LA30, VT05, KSR-33, or KSR-35.

dedicated to MCR Functions; however, they may be installed to run in any partition.

The name of the MCR Dispatch Task is ",.;MCR". The name of an MCR Function Task is three dots followed by the first three characters of the name of the MCR function.

Any TTY Terminal can be used as the "MCR Terminal". However, since MCR Functions are powerful, and therefore generally dangerous, the use of the MCR is restricted to those who know a six-character password. The password is specified at System Configuration (SGEN).

MCR Input is initiated by typing a ↑C[2] on a TTY Terminal. If the TTY is the "MCR Terminal", an "MCR>" prompting symbol will be output, and MCR command information will be accepted. If the TTY is not the MCR Terminal, a "PASSWORD:" interrogation will be output, and if the correct password responce is typed, the TTY will become the MCR Terminal.

The MCR is not intended for multi-terminal use, and the ability to move the MCR Terminal by typing ↑C and a password is to provide a means of retaining a console interface when an MCR Terminal goes down. Hence, while an MCR Function is in progress, a ↑C Input on other than the MCR Terminal is ignored.

During lengthly MCR output, a ↑S[3] typein on the MCR TTY will cause the output to be suppressed.

By convention, MCR Function Tasks output to either LUN-2 or LUN-3 (of Task ",.;MCR). All MCR error messages and short outputs are written to LUN-2. Lengthly MCR output is written to LUN-3. Both LUN-2 and LUN-3 of the MCR Dispatch Task (,.;MCR) are normally assigned to a TTY, however, the lengthly MCR output may be directed to a printer without affecting normal operations by REASSIGNing LUN-3. Output for the operator's attention from other Tasks is also written to MCR's LUN-2, e.g., A message from an I/O Handler Task. A user Task may also take advantage of this convention.

---

[2] The symbol "↑C" is used to represent the character input by typing "C" while the "CTRL" key is depressed.

---

[3] The symbol "↑S" is used to represent the character input by typing "S" while the "CTRL" key is depressed.

NOTE: Since normal RSX Tasks are used to implement MCR Functions, special purpose functions to provide added flexibility or convinence (viz., an MCR Function to change the password) for a particular application or installation may be easily developed and added.

The following is a description of the MCR Functions provided. The syntax is defined in modified BNF using the following conventions and definitions:

Angle brackets delimit metalinguistic variables,
Quote marks delimit a character string,
A slash (/) indicates alternation (OR),
No operator indicates concatenation,
Parens indicate factoring,
'$' indicates any number [including zero] of,
'NUL' indicates the empty set,
<BC> ::= SPACE [break character],
<NBC> = Non-Break Character,
<CR> = Carriage Return,
<AM> = ALTMODE
<LT> ::= <CR>/<AM>; [line terminator],
<NTC> = Non-line terminator,

<DIGIT> ::= "0"/"1"/"2"/"3"/"4"/"5"/"6"/"7"/"8"/"9";
<LETTER> ::= "A"/"B"/"C" ;.;.;.;.;.; "X"/"Y"/"Z";
<TIME> ::= <HOURS> ":" <MINUTES> ":" <SECONDS>;
<DATE> ::= <MONTH> "/" <DAY> "/" <YEAR>;
<DELTA TIME> ::= <DECIMAL VALUE> ("H"/"M"/"S"/"T");

ENTER TIME

The ENTER TIME MCR Function is used to set the system clock and calendar. One line of input of the following syntax is used:

SYNTAX ::= "ETI"$<NBC> <BC> <TIME> (<BC><DATE>/NUL) <LT>;

Examples:

```
MCR,ETIME 7:55:00 12/14/70
MCR,ETI 10:30:00
```

TIME

The TIME MCR Function outputs, on LUN-2, the Time of day. One line of input of the following syntax is used:

SYNTAX ::= "TIM"$<NTC> <LT>;

Example:

```
MCR,TIME
 07:55:05
```

DATE

The DATE MCR Function outputs, on LUN-2, the Date and Time. One line of input of the following syntax is used:

SYNTAX ::= "DAT"$<NTC> <LT>;

Example:

```
MCR,DATE
 2/15/72  10:23:15
```

TASK LIST

The TASK-LIST MCR Function outputs, on LUN=3, a description of each Installed Task. The description consists of: Task name, Default Partition name, Default Priority number (decimal), Size of Initial load image (octal), and Disk Address (octal). Additional information is also included when appropriate, viz., FIXED-IN-MEMORY, DISABLED. One line of input of the following syntax is used:

SYNTAX := "TAS"S<NTC> <LT>;

Example:

```
MCR,TASK LIST
 BILL    COMM    150 60031 DF 0-006400
 JOE     COMM    150 10374 DF 0-024000
 LP,,,   INOUT   255 00130 DK 0-007400
 SCAN    CNTRL   200 10374 DK 0-013400
 ;;,SCH  MCRPAR  250 00317 DK 0-017400
 ;;,REQ  MCRPAR  019 00203 DK 0-013000
```

PARTITION DEFINITIONS

The PARTITION DEFINITIONS MCR Function outputs, on LUN-3, a description of each Memory Partition in the system. The description consists of: Partition Name, Partition Base (octal address), and Partition Size (octal). One line of input of the following syntax is used:

SYNTAX ::= "PAR"$<NTC> <LT>;

Example:

```
MCR,PARTITIONS
  MCR     100000,3000
```

COMMON BLOCK DEFINITIONS

The COMMON BLOCK DEFINITIONS MCR Function outputs, on LUN-3, a description of each Global Common Block. The description consists of: Common Block Name, Common Block Base (octal address), and Block size (octal). One line of input of the following syntax is used:

SYNTAX ::= "COM"$<NTC> <LT>;

Example:

```
MCR,COMMON BLOCKS
COMBLK  300000,200
```

DEVICES

The DEVICES MCR Function outputs, on LUN=3, the symbolic names of all device-units known to the system. Each device-unit that is serviced by a RESIDENT Handler Task is flagged by "**". One line of input of the following syntax is used:

SYNTAX ::= "DEV"$<NIC> <LT>

Example:

```
MCR>DEVICES
 DK    **
 DT1
 DT2
 DT3
 LP    **
 TT0   **
 TT1   **
```

ASSIGNMENTS

The ASSIGNMENTS MCR Function outputs, on LUN=3, a list of Physical Device-units and corresponding Logical Unit Numbers for an indicated Task or Tasks. Only devices to which assignments have been made are listed. If no Task name is provides, the assignments for all Tasks will be output. Note -- The assignments listed are from the Task's disk image, and NOT from memory of currently executing Tasks. One line of input of the following syntax is used:

SYNTAX ::= "ASS"$<NBC> $(<BC><TASK NAME>) <LT>

Example:

```
MCR>ASSIGNMENTS JOE,PETE
***** JOE
 LP    3
 TT3   4,5,7
 DK    6
***** PETE
 TT0   1,2,3,4,5
 CR    7
```

REQUEST

The REQUEST MCR Function allows the console operator to request the immediate (contingent upon priority and memory availability) execution of Tasks, i.e., it allows the operator to issue REQUEST Directives. A Priority and a Partition may be specified. One line of the following syntax is used:

```
SYNTAX ::= "REQ"$<NBC> <RC> <REQUEST REQUEST>;
   <REQUEST REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / ","<REQUEST REQUEST> /
      "/"<OPTION><SUFFIX>;
         <OPTION> ::= "PRI="<PRIORITY NUMBER> /
         "PAR="<PARTITION NAME> / "INS="<DEV-UNIT>;
```

Possible error messages:

```
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- TASK ALREADY (OR STILL) ACTIVE
*** <TASK NAME> -- NODE FOR ATL UNAVAILABLE
*** <TASK NAME> -- TASK NOT INSTALLABLE
```

Examples:

```
MCR,REQUEST SCAN
MCR,REQ SCAN/PAR=XYZ/PRI=58
MCR,REQ JOE,PETE/PRI=10,BILL/PAR=BBBB/PRI=500,SCAN
```

EXECUTE

The EXECUTE MCR Function allows the console operator to request the immediate (contingent upon priority) execution of tasks if, and only if, memory is available, i.e., it allows the operator to issue EXECUTE Directives. A Priority and a Partition may be specified. One line of the following syntax is used:

```
SYNTAX ::= "EXE"$<NBC> <BC> <EXECUTE REQUEST>;
   <EXECUTE REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / ","<EXECUTE REQUEST> /
      "/"<OPTION><SUFFIX>;
         <OPTION> ::= "PRI="<PRIORITY NUMBER> /
         "PAR="<PARTITION NAME>;
```

Possible error messages:

```
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- TASK ALREADY (OR STILL) ACTIVE
*** <TASK NAME> -- NODE FOR ATL UNAVAILABLE
*** <TASK NAME> -- MEMORY UNAVAILABLE
```

Examples:

```
MCR,EXECUTE SCAN
MCR,EXE SCAN/PRI=100/PAR=XXX
MCR,EXE SCAN/PRI=25,JOE,PETE,BILL/PAR=ZZZ,PRI=99
```

SCHEDULE

The SCHEDULE MCR Function allows the console operator to schedule Tasks in terms of "absolute time of day" with the option to specify Periodic Rescheduling, Run Priority and Memory Partition, i.e., It allows the operator to issue SCHEDULE directives, One line of the following syntax is used:

```
  SYNTAX ::= "SCH"$<NBC> <BC> <SCHEDULE REQUEST>;
     <SCHEDULE REQUEST> ::= <TASK NAME> <BC> <TIME>
 <SUFFIX>;
        <SUFFIX> ::= <LT> / ","<SCHEDULE REQUEST> /
        "/"<OPTION><SUFFIX>;
           <OPTION> ::= "RSI="<RESCHEDULE INTERVAL> /
           "PRI="<PRIORITY NUMBER> /
           "PAR="<PARTITION NAME>;
              <RESCHEDULE INTERVAL> ::= <DELTA TIME>;
              <PRIORITY NUMBER> ::= <DECIMAL VALUE>;
```

Possible error messages:

```
  *** <TASK NAME> -- SYNTAX ERROR
  *** <TASK NAME> -- TASK NOT IN SYSTEM
  *** <TASK NAME> -- TASK DISABLED
  *** <TASK NAME> -- NODE FOR CLOCK QUEUE UNAVAILABLE
```

Example:

```
  MCR,SCHEDULE JOE 12:23:15
  MCR,SCH JOE 12:00:00/RSI=6M/PRI=99/PAR=XYZ,BILL 13:07:00
  MCR,SCH SCAN 23:30:00/RSI=10M/PRI=200
  MCR,SCH JOE 1:00:00,PETE 1:30:00,BILL 2:00:00
```

RUN

The RUN MCR Function allows the console operator to schedule Tasks in terms of "delta time from now" with the option to specify Periodic Rescheduling, Run Priority and Memory Partition, i.e., It allows the operator to issue RUN Directives. One line of the following syntax is used:

```
SYNTAX ::= "RUN"$<NBC> <BC> <RUN REQUEST>;
   <RUN REQUEST> ::= <TASK NAME> <BC> <DELTA TIME>
<SUFFIX>;
      <SUFFIX> ::= <LT> / ","<RUN REQUEST> /
      "/"<OPTION><SUFFIX>;
         <OPTION> ::= (same as SCHEDULE)
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- NODE FOR CLOCK QUEUE UNAVAILABLE
```

Example:

```
MCR,RUN JOE 15M
MCR,RUN JOE 15M/RSI=90S
MCR,RUN JOE 15M/RSI=90S/PRI=150/PAR=XYZ
MCR,RUN JOE 15M,PETE 20M,BILL 25M
```

SYNC

The SYNC MCR Function allows the console operator to schedule Tasks in terms of "delta time from clock unit synchronization" with the option to specify Periodic Rescheduling, Run Priority and Memory Partition, i.e., it allows the operator to issue SYNC Directives, One line of the following syntax is used:

```
SYNTAX ::= "SYN"S<NBC> <BC> <SYNC REQUEST>;
   <SYNC REQUEST> ::= <TASK NAME> <BC>
   <DELTA TIME> <BC> <SYNC UNIT> <SUFFIX>;
      <SYNC UNIT> ::= "H" / "M" / "S";
      <SUFFIX> ::= <LT> / ","<SYNC REQUEST> /
      "/"<OPTION><SUFFIX>;
         <OPTION> ::= (same as SCHEDULE)
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- NODE FOR CLOCK QUEUE UNAVAILABLE
```

Examples:

```
MCR,SYNCHRONIZE JOE H 10M
MCR,SYNC JOE H 10M/PRI=10,PETE M 0S/RSI=15S/PRI=90
```

CANCEL

The CANCEL MCR Function allows the console operator to cancel periodic rescheduling of Tasks, i.e., it allows the operator to issue CANCEL Directives. One line of the following syntax is used:

```
SYNTAX ::= "CAN"$<NBC> <BC> <CANCEL REQUEST>;
   <CANCEL REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / ","<CANCEL REQUEST>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- NODE FOR CLOCK QUEUE UNAVAILABLE
```

Examples:

```
MCR,CANCEL JOE
MCR,CAN JOE,BILL,PETE
```

ABORT

The ABORT MCR Function allows the console operator to terminate the execution of an indicated Task. One line of input of the following syntax is used.

```
SYNTAX ::= "ABO"$<NBC> <BC> <TASK NAME> <LT>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK NOT ACTIVE
*** <TASK NAME> --
*** <TASK NAME> --
*** <TASK NAME> --
```

Example

```
MCR,ABORT SCAN
```

RESUME

The RESUME MCR Function allows the console operator to cause the resumption of SUSPENDED Tasks, i.e., It allows the operator to issue RESUME Directives, One line of the following syntax is used:

```
SYNTAX ::= "RES"S<NBC> <BC> <RESUME REQUEST>;
   <RESUME REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / <RESUME REQUEST>!
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK NOT SUSPENDED
```

Examples:

```
MCR,RESUME JOE
MCR,RES JOE,BILL,PETE
```

FIX IN MEMORY

The FIX-IN-MEMORY MCR Function allows the console operator to cause Tasks to become memory resident but not active. I.e., It allows the operator to issue FIX Directives. One line of the following syntax is used:

```
SYNTAX ::= "FIX"$<NBC> <BC> <FIX REQUEST>;
   <FIX REQUEST> ::= <TASK NAME><SUFFIX>;
      <SUFFIX> ::= <LT> / <FIX REQUEST>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK DISABLED
*** <TASK NAME> -- TASK ACTIVE
*** <TASK NAME> -- TASK ALREADY FIXED
*** <TASK NAME> -- NODE FOR FIXING UNAVAILABLE
```

Examples:

```
MCR,FIX JOE
MCR,FIX JOE,BILL
```

UNFIX

The UNFIX MCR Function allows the operator to free "FIXED" Tasks from memory. I.e., It allows the operator to issue UNFIX Directives. One line of the following syntax is used:

```
SYNTAX ::= "UNF"$<NBC> <BC> <UNFIX REQUEST>;
   <UNFIX REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / <UNFIX REQUEST>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK NOT FIXED
```

Examples:

```
MCR,UNFIX JOE
MCR,UNF BILL,PETE
```

DISABLE

The DISABLE MCR Function allows the console operator to effectively delete Tasks from the system without actually removing them, i.e., It allows the operator to issue DISABLE Directives. One line of input of the following syntax is used:

```
SYNTAX ::= "DIS"$<NBC> <BC> <DISABLE REQUEST>;
   <DISABLE REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / ","<DISABLE REQUEST>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- ALREADY DISABLED
```

Examples:

```
MCR,DISABLE SCAN
MCR,DIS JOE,PETE,BILL
```

ENABLE

The ENABLE MCR Function allows the console operator to nullify the effect of DISABLE Directives, i.e., It allows the operator to issue ENABLE Directives. One line of the following syntax is used:

```
SYNTAX ::= "ENA"$<NBC> <BC> <ENABLE REQUEST>;
   <ENABLE REQUEST> ::= <TASK NAME> <SUFFIX>;
      <SUFFIX> ::= <LT> / ","<ENABLE REQUEST>;
```

Possible error messages:

```
*** <TASK NAME> -- SYNTAX ERROR
*** <TASK NAME> -- TASK NOT IN SYSTEM
*** <TASK NAME> -- TASK NOT DISABLED
```

Examples:

```
MCR,ENABLE SCAN
MCR,ENA JOE,PETE,BILL
```

REASSIGN

The REASSIGN MCR Function allows the operator to de-assign Logical Unit Numbers (LUN's) from one physical device-unit, and assign them to another. The reassignments are performed on indicated Task's disk image, and do not affect a memory resident Task. One line of input of the following syntax is used:

```
SYNTAX ::= "REA"$<NBC> <BC> <TASK NAME> <BC>
<NEW DEVICE> "←" <OLD DEVICE> <BC> <LUN LIST>;
   <OLD DEVICE> ::= <DEVICE-UNIT SYMBOL>;
   <NEW DEVICE> ::= <DEVICE-UNIT SYMBOL>;
      <DEVICE-UNIT SYMBOL> ::= <LETTER><LETTER>
      (<OCTAL VALUE> / NUL);
   <LUN LIST> ::= <LUN> $(","<LUN>) <LT>;
      <LUN> ::= <DECIMAL VALUE>;
```

Possible error messages:

```
*** SYNTAX ERROR
*** TASK NOT IN SYSTEM
*** LUN <LUN> OUT OF RANGE
*** <NEW DEVICE-UNIT> NOT KNOWN TO SYSTEM
*** <OLD DEVICE-UNIT> NOT ASSIGNED AS INDICATED
```

Possible warning messages:

```
*** <NEW DEVICE> HANDLER TASK NOT RESIDENT
```

Examples:

```
MCR, REASSIGN JOE TT0 TT1
MCR,REA BILL TT2LP 2,3,4,5
```

REDIRECT

The REDIRECT MCR function allows the console operator to redirect all requests from one physical device-unit to another. One line of command input of the following format is used:

```
SYNTAX ::= "RED"$<NBC> <BC>
<NEW DEVICE> "=" <OLD DEVICE> <LT>;
   <OLD DEVICE> ::= <DEVICE-UNIT SYMBOL>;
   <NEW DEVICE> ::= <DEVICE-UNIT SYMBOL>;
      <DEVICE-UNIT SYMBOL> ::= <LETTER><LETTER>
      (<OCTAL VALUE> / NUL);
```

Possible error messages:

```
*** <DEVICE-UNIT SYMBOL> -- SYNTAX ERROR
*** <DEVICE-UNIT SYMBOL> -- DEVICE NOT KNOWN TO SYSTEM
*** <NEW DEVICE> NOT KNOWN TO SYSTEM
```

Possible warning messages:

```
*** <NEW DEVICE> HANDLER NOT RESIDENT
```

Example:

```
MCR,REDIRECT TT3TT6
```

LOAD

The LOAD MCR Function allows the console operator to cause an Indicated I/O Handler Task to become resident in memory and ready for service, One line of input of the following syntax is used:

```
SYNTAX ::= "LOA"$<NBC> <BC> <DEVICE SYMBOL> <LT>;
   <DEVICE SYMBOL> ::= <LETTER><LETTER>;
```

Possible error messages:

```
*** <DEVICE SYMBOL> -- SYNTAX ERROR SYNTAX ERROR
*** <DEVICE SYMBOL> -- DEVICE NOT KNOWN TO SYSTEM
*** <DEVICE SYMBOL> -- HANDLER TASK NOT IN SYSTEM
*** <DEVICE SYMBOL> -- HANDLER TASK DISABLED
*** <DEVICE SYMBOL> -- INSUFFICENT MEMORY TO LOAD
```

Example:

```
MCR,LOAD LP
```

UNLOAD

The UNLOAD MCR Function allows the console operator to cause an Indicated I/O Handler Task to complete queued requests and leave memory, One line of input of the following syntax is used:

```
SYNTAX ::= "UNL"$<NBC> <BC> <DEVICE SYMBOL> <LT>;
   <DEVICE SYMBOL> ::= <LETTER><LETTER>;
```

Possible error messages:

```
*** <DEVICE SYMBOL> -- SYNTAX ERROR
*** <DEVICE SYMBOL> -- DEVICE NOT KNOWN TO SYSTEM
*** <DEVICE SYMBOL> --
```

Possible interogitives:

```
*** <DEVICE SYMBOL> -- ATTACHED TO TASK, STILL UNLOAD?
*** <DEVICE SYMBOL> -- OPEN FILE(S), STILL UNLOAD?
```

Example:

```
MCR,UNLOAD PP
```

SAVE

The Save MCR Function is used to record the core image of an RSX-11D system on the system disk such that a bootstrap can reload it and start up the system (warm start). The Function should only be requested when the system is quiescent, and is intended to provide "development plateaus", rather than "crash recoveries". One line of input of the following syntax is used:

```
SYNTAX ::= "SAV"$<NTC> <LT>;
```

Example:

```
MCR,SAVE
```

OPEN REGISTER

The Open MCR Function facilitates the "opening" of a memory or disk location (word) for examination and optional modification, and the "opening" of a related location.

The address of the first location to open is indicated in an initial input line of the following syntax:

```
SYNTAX ::= "OPE"$<NBC> <BC> <ADDRESS> $<BIAS> <LT>;
   <ADDRESS> ::= <DISK ADR> / <MEMORY ADR>;
   <LT> ::= <CR> / <AM>;
      <DISK ADR> ::= <DSK NAME> ":" <DSK UNIT>;
         <DSK NAME> ::= "DF" / "DK" / "DP" ;
         <DSK UNIT> ::= <NUMBER>;
      <MEMORY ADR> ::= <NUMBER>;
      <BIAS> ::= <NUMBER>;
         <NUMBER> ::= <SIGN><MAGNITUDE>;
            <SIGN> ::= "+" / "-" / NUL;
            <MAGNITUDE> ::= "D"<DECIMAL VALUE> /
            "B"<OCTAL VALUE> / <OCTAL VALUE>;
```

If the address is valid, the location's address and contents are output, to LUN=2, in octal on a line followed by a "/" prompting character. A response in the following syntax is expected:

```
SYNTAX ::= (<NUMBER> / NUL) <TERMINATOR>;
   <TERMINATOR> ::= <AM> / <CR> / "↑"<CR> / "@"<CR>;
```

If a number is input, that quantity is set in the open location, otherwise the location is unaltered.

If the response is terminated by an ALTMODE, no related location is opened, and the MCR Function has been completed.

If the response is terminated by a CAR RTN, the next sequential location is opened.

If the response is terminated by an UP-ARROW CAR RTN, the previous location is opened.

If the response is terminated by an ASTERISK CAR RTN, the location pointed to by the final contents (Kernel virtual) of the opened location is opened.

Whenever a location relative to an open location is opened, the newly opened location may also be modified, and used to determine another location to be opened.

LOG COMMENT

The Log MCR Function allows the console operator to type a comment on the MCR Terminal log. One line of input of the following syntax is used:

SYNTAX ::= "COM"$<NTC> <LT>;

There is no ".,.LOG" MCR Function Task, the Dispatch Task simply ignores the command input.

CLOSE MCR

The Close MCR Function provides a means of requiring the password to be input before another MCR command will be recognized. Thus, the operator can leave the MCR TTY without leaving the MCR "open". One line of command input of the following syntax is used:

SYNTAX ::= "CLO"$<NTC> <LT>;

There is no ".,.CLO" MCR Function Task. This function is performed by the Dispatch Task itself.