

**HP 3000 SERIES II/III  
COMPUTER SYSTEMS  
MANUAL OF STAND-ALONE DIAGNOSTICS**

**STAND-ALONE  
SLEUTH DIAGNOSTIC**

Diagnostic No. D411A



### **NOTICE**

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

# LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

Second Edition . . . . . Aug 1978

# PRINTING HISTORY

New editions are printed when extensive changes are made to the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The edition printing date on the title page and back cover of the manual changes only when a new edition is published. When the manual is reprinted, the current update, if any, is incorporated into the manual. No new content is added to the manual; neither the printing date on the title page nor back cover, nor the edition number change. The date of the incorporated printing is added to the back cover.

The software product part number printed beside the date indicates the version and update level of the software product at the time the manual edition or update was issued. Many product updates and fixes do not require manual changes, and conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

First Edition .....	Jun 1976
1st Update .....	April 1977
2nd Update .....	Aug 1977
Second Edition .....	Aug 1978

# CONVENTIONS USED IN THIS MANUAL

NOTATION	DESCRIPTION
[ ]	An element inside brackets is <i>optional</i> . Several elements stacked inside a pair of brackets means the user may select any one or none of these elements.  Example: $\left[ \begin{array}{c} A \\ B \end{array} \right]$ user may select A or B or neither
{ }	When several elements are stacked within braces the user <b>must</b> select one of these elements.  Example: $\left\{ \begin{array}{c} A \\ B \\ C \end{array} \right\}$ user must select A or B or C.
italics	Lowercase italics denote a parameter which must be replaced by a user-supplied variable.  Example: CALL <i>name</i> <i>name</i> one to 15 alphanumeric characters.
underlining	Dialogue: Where it is necessary to distinguish user input from computer output, the input is underlined.  Example: NEW NAME? <u>ALPHA1</u>
superscript C	Control characters are indicated by a superscript C  Example: Y <sup>C</sup>
<i>return</i>	<i>return</i> in italics indicates a carriage return
<i>linefeed</i>	<i>linefeed</i> in italics indicates a linefeed
...	A horizontal ellipsis indicates that a previous bracketed element may be repeated, or that elements have been omitted.

# CONTENTS

Section I	Page
<b>GENERAL INFORMATION</b>	
Introduction . . . . .	1-1
SLEUTH Features . . . . .	1-1
Hardware Required . . . . .	1-1
Software Required . . . . .	1-1

Section II	
<b>PRINCIPLES OF OPERATION</b>	
Introduction . . . . .	2-1
Logical Unit . . . . .	2-1
Error Print Limit . . . . .	2-1
Power Fail . . . . .	2-1
Variables . . . . .	2-2
Editing . . . . .	2-3
Line Numbering . . . . .	2-3
Add Operation . . . . .	2-4
Replace Operation . . . . .	2-4
Delete Operation . . . . .	2-5
Insert Operation . . . . .	2-5
Renumbering . . . . .	2-6
Proceed Mode . . . . .	2-7
Switch Registers . . . . .	2-9

Section III	Page
<b>OPERATING PROCEDURES</b>	
Introduction . . . . .	3-1
SLEUTH Tape . . . . .	3-1
MAKT and BA . . . . .	3-1
Operating Instructions . . . . .	3-2

Section IV	
<b>ERROR MESSAGES</b>	
Introduction . . . . .	4-1
Diagnostic Messages . . . . .	4-1
Error Messages . . . . .	4-2
Program Error Messages . . . . .	4-5

Section V	
<b>SLEUTH COMMANDS</b>	
Index of Commands . . . . .	5-1
Introduction . . . . .	5-1
SLEUTH Commands . . . . .	5-3

# ILLUSTRATIONS

Figure	Page
2-1 Generalized Command Execution Flow . . .	2-8

# TABLES

Table	Page	Table	Page
4-1 Diagnostic Messages . . . . .	4-1	5-0 Index of SLEUTH Commands . . . . .	5-0
4-2 Error Messages . . . . .	4-4	5-1 Elements of Command Format . . . . .	5-3
4-3 Program Error Messages . . . . .	4-7		





# GENERAL INFORMATION

SECTION

I

## 1-1. INTRODUCTION

The purpose of this manual is to assist the user in understanding the operation of SLEUTH. How the user applies SLEUTH to problem solving depends on the nature of the problem and the creativity of the user.

SLEUTH is a stand-alone diagnostic program written in Systems Program Language (SPL). The purpose of SLEUTH is to enable the user to generate unique test programs that may be used to isolate problems in the I/O section of an HP 3000 Series II Computer System and the peripheral devices.

## 1-2. SLEUTH FEATURES

- Common, powerful interactive language for all devices
- Ability to run up to eight devices of various types concurrently
- Ability to write and execute SIO programs for any device
- Ability to store and restore user programs on magnetic tape
- Ability to edit user programs
- SLEUTH format is similar to BASIC language format.

## 1-3. HARDWARE REQUIRED

SLEUTH can be run on an HP 3000 Series II Computer System with the following minimum equipment:

- Central Processor Unit (CPU) with at least 64K of memory
- Tape Drive Unit for loading magnetic tape
- System console device with interface PCA.

## 1-4. SOFTWARE REQUIRED

The SLEUTH stand-alone diagnostic magnetic tape is prepared using the Stand-Alone Diagnostic Program (SDUPII), Diagnostic No. D417. No additional software is required to load and execute SLEUTH.



# PRINCIPLES OF OPERATION

SECTION

II

## 2-1. INTRODUCTION

This section contains the principles of SLEUTH operation. Section III contains operating procedures utilizing these principles of operation.

## 2-2. LOGICAL UNIT

The computer system peripheral devices must be identified to the computer as to type and logical location. The devices are identified, or declared to the computer, with the Device (DEV) command (see page 5-28). The DEV command refers to the devices as logical units and each unit is given a logical unit number (LUN). SLEUTH is capable of handling up to eight logical units simultaneously.

## 2-3. ERROR PRINT LIMIT

Each logical unit has an error print limit in the range, 0 to 999, which is set by the user with the DEV command. This limit is the number of printing errors allowed. If a device exceeds its error limit, any commands referenced to that device are not executed. This allows the user to run multiple devices and continue testing others if one should exceed its error limit.

## 2-4. POWER FAIL

SLEUTH has power-fail auto restart capabilities; therefore the panel switch for restart should be enabled while running SLEUTH. If a power-fail is encountered, the message

```
D8 **POWER-FAIL**
```

is printed on the console. SLEUTH waits 50 seconds to allow discs and other devices to become ready. Program execution continues at the point at which it was interrupted.

## 2-5. VARIABLES

SLEUTH has variable manipulating capabilities. Variables may be placed in many of the command parameters, allowing these parameters to be dynamically changed during program execution.

All parameters, except the following, can be variables

- Step numbers and step number parameters
- Device declaration parameters (DEV)
- Dump parameters (DUMP)
- Define buffer parameters (DB)
- LUN and BUF parameters
- SIO PROG parameters (SIO, NAME)
- SIO order pairs or their parameters
- Parameters that demand a specific character.

Bounds checking and arithmetic overflow are not provided for any variable reference.

The following example illustrates how variables and variable manipulating commands can be used.

Example:

```
> 10 DEV 0,4,15,100,0  
> 10 RAND I  
> 20 LET A=I MOD 410  
> 30 SEEK 0,A,0,0  
> 40 LOOP 10,500  
> 50 END  
> 60 RUN  
> 60
```

## 2-6. EDITING

The user has an editing capability inherent in SLEUTH. Statement numbers control the sequences of the user's program. Lines may be added to, replaced in, deleted from, and inserted in a program.

## 2-7. LINE NUMBERING

A SLEUTH command is usually preceded by a line number. The number is an integer between 1 and 999. The line number indicates the order of execution. Commands are ordered by SLEUTH from the lowest to the highest line number. This order is maintained by SLEUTH, therefore, it is not necessary for the user to enter commands in the order of execution so long as the line numbers are in order.

Example:

```
> 10 NOPR
> 30 ES
> 20 PR
> 40 SS
> 50 PE
}
RUNS AS
{
> 10 NOPR
> 20 PR
> 30 ES
> 40 SS
> 50 PE
```

SLEUTH is normally in an automatic numbering mode wherein the line numbers are automatically supplied. When a program is edited the automatic numbering feature is cancelled. The user must then supply the line number or use the AUTO command to resume automatic numbering.

Example:

```
> 10 NOPR
> 20
```

In the above example of automatic numbering the computer supplied the SLEUTH prompt and the first line number, >10; the user entered the NOPR command and the computer responded with a carriage return and line feed signifying acceptance of line 10. The computer then prints the prompt and the second line number, >20, and halts to await the next user input.

Example:

```
> 10 NOPR
> 20 SS
> 30 SS
> 40 PR
> 50 PE
> 60 30 PR (Edit command to replace line 30 also terminates automatic numbering.)
> 60 AUTO (User supplies line and AUTO. The system resumes automatic numbering.)
> 60
```

SLEUTH commands are divided into three subsets (refer to Section V) and only those commands that execute when RUN is entered (the third subset) require line numbers. Line numbers are optional for all other commands. For example, the AUTO commands in the above and following examples can also be entered as >AUTO.

## 2-8. ADD OPERATION

New lines may be added to a program any time since it is the line number that determines the sequence of operation.

Example:

```
> 10 NOPR
> 20 PR
> 30 ES
> 40
```

## 2-9. REPLACE OPERATION

A previously entered line may be replaced by another line.

### NOTE

When a program is edited, automatic numbering is cancelled. The user must supply the line number or use the AUTO command.

Example:

```
> 10 NOPR
> 20 SS
> 30 ES
> 40 PR
> 50 PE
> 60 30 PR (Replaces line 30 and cancels automatic numbering.)
> 60 AUTO (User supplies line number and AUTO command.)
> 60 DUMP P (Dumps program.)
  10 NOPR
  20 SS
  30 PR
  40 PR
  50 PE
> 60
```

## 2-10. DELETE OPERATION

A previously written line may be deleted from the program.

Example:

```
> 10 NOPR
> 20 SS
> 30 PR
> 40 ES
> 50 20      (Deletes line 20 and cancels automatic numbering.)
> 50 AUTO    (User supplies line number and AUTO command.)
> 50 DUMP P  (Dumps program.)
  10 NOPR
  30 PR
  40 ES
> 50
```

## 2-11. INSERT OPERATION

New lines may be inserted between existing lines by using intermediate line numbers.

Example:

```
> 10 NOPR
> 20 SS
> 30 ES
> 40 15 PR   (Inserts line 15 and cancels automatic numbering.)
> 40 AUTO    (User supplies line number and AUTO command.)
> 40 DUMP P  (Dumps program.)
  10 NOPR
  15 PR
  20 SS
  30 ES
> 40
```

## 2-12. RENUMBERING

If many edits are performed on the program, or if automatic numbering is not used, statement numbers may become disorganized as shown in the first example below. The renumber command (REN) may be used to renumber the program. If the maximum step number (999) is exceeded while in the automatic numbering mode, SLEUTH displays a message and automatically renumbers the program as shown in the second example.

Whenever a SLEUTH program is renumbered, all statements containing references to statement numbers are automatically updated to reflect the new statement numbers.

Examples:

```
> 10 NOPR
> 20 11 PR
> 12 GET A
> 18 IF A <> 10 THEN 25
> 15 SS
> 21 PE
> 25 LOOP 12,10
> 33 GO 21,0,0
> 37 GOTO 12
> 38 AUTO
> 47 REN (Renumber.)
>100 DUMP P (Dump program.)
    10 NOPR
    20 PR
    30 GET A
    40 SS
    50 IF A <> 10 THEN 70 (Statement number updated.)
    60 PE
    70 LOOP 30,10 (Statement number updated.)
    80 GO 60,0,0 (Statement number updated.)
    90 GOTO 30 (Statement number updated.)
>100
```

The above example shows how the Renumber command may be used to renumber the program. The next example illustrates automatic renumbering of the same program.

```
> 10 AUTO 992
>992 NOPR
>993 PR
>994 GET A
>995 SS
>996 IF A <> 10 THEN 998
>997 PE
>998 LOOP 994,10
>999 GO 997,0,0
MAX STEP# EXCEEDED. PROGRAM RENUMBERED.
> 90 DUMP P
    10 NOPR
    20 PR
    30 GET A
    40 SS
    50 IF A <> 10 THEN 70
    60 PE
    70 LOOP 30,10
    80 GO 60,0,0
> 90
```



## 2-13. PROCEED MODE

SLEUTH can operate multiple devices simultaneously. This mode of operation is termed PROCEED MODE. Proceed mode allows the user to test and simulate conditions that would otherwise occur only under software system use.

### CAUTION

When operating in PROCEED MODE the user must take care not to generate conditions which would exceed the operating bandwidth of the SIO multiplexer.

Proceed Mode is a privileged mode, and executes as follows (see figure 2-1):

- a. SIO operations are issued to all devices indicated until:
  - (1) A command is issued to a device that has a previous operation pending.
  - (2) A command is issued to a DRT that has a previous operation pending (that is, multiple units on one controller).
- b. Direct I/O operations that do not require an interrupt to complete are issued regardless of any pending operations.
- c. Proceed Mode can be turned on or off at any point in the user's program using the Proceed command (PROC).
- d. If an operation is encountered that will not execute asynchronously, the command is executed in its entirety before proceeding to the next operation. Any previous operations pending are completed, and the results of that operation are evaluated.

Commands that do *not* operate asynchronously are:

FMT	Format
TDIL	Terminal Data Interface Loop
WD	Write Data Asynchronous MUX and Reader/Punch
RP	Ripple Print for Lineprinters and Asynchronous MUX
RD	Read Data for Asynchronous MUX
XDUI	Universal Interface Data Test

- e. When operating in proceed mode, more than one device should not read into the same buffer, as the contents of that buffer will be unpredictable. It should also be noted that a compare buffer operation (CB) may begin before the read is completed, thereby producing erroneous compare buffer errors. These conditions should be recognized or avoided.
- f. When operating in the proceed mode, if the user has defined his own SIO programs, more than one device should not execute the same SIO program.

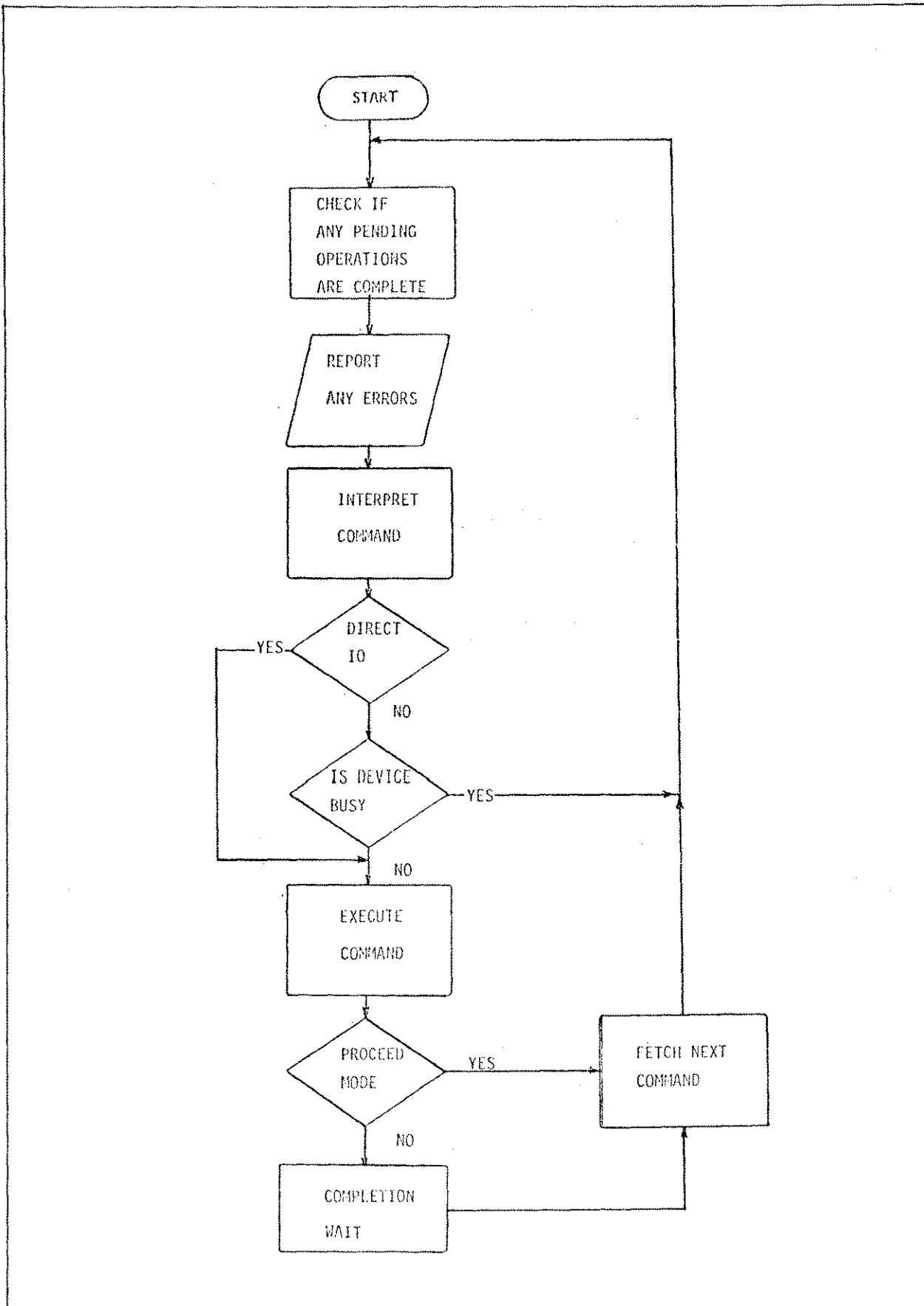


Figure 2-1. Generalized Command Execution Flow

## 2-14. SWITCH REGISTERS

In some cases, the front panel switch register can be used to supplement normal SLEUTH operations. The active switches and their respective operations are as follows:

- Switch 0 Must be set to enable the switch register. If Switch 0 is clear, no other switches are examined.
- Switch 7 Enable/disable (1/0) status checking. Duplicates functions of the Enable Status and Suppress Status commands. Each SLEUTH program begins with status checking enabled.
- Switch 11 Disable/enable (1/0) printing of all error messages. Duplicates functions of the No Print and Print commands. Each SLEUTH program begins with printing enabled.
- Switch 12 Enable/disable (1/0) SIO Program dump on error. Used only to disable (or possibly to re-enable) the SIO Program portion of an error message. Each SLEUTH program begins with SIO Dump enabled and can only be disabled by setting switch 12 to 0.
- Switch 13 When set, aborts execution of current SLEUTH program and returns SLEUTH to command input phase. Unlike entering control A which is unpredictable, the switch register is read between commands and after each cylinder of a Format, Verify, or Verify Immediate command. The maximum response time should be 1.0 second.

### NOTE

If the program is restarted, the Increment/Decrement commands (DS, IR, IS, IT, and IW) and the Loop command retain their current status and will continue where they left off. To reinitialize these commands, re-enter them.

- Switch 14 Pause on error. Halts computer (Halt %16) after printing an execution-time error message. Duplicates functions of the Pause On Error command. Each SLEUTH program begins with no pause on error.

### NOTE

If no print is in effect (Switch 11 set or No Print Command executed), Switch 14 settings are ignored.

- Switch 15 Switch output to line printer. If a line printer is configured into the SLEUTH system, forces all execution-time error messages to the printer. Overrides the Switch Output command. If Switch 15 is clear, the Switch Output command is executed.



# OPERATING PROCEDURES

SECTION

III

## 3-1. INTRODUCTION

This section contains procedures for operating an HP 3000 Series II Computer System and exercising the peripheral devices using SLEUTH. It is assumed that the user of SLEUTH has a working knowledge of the computer system.

## 3-2. SLEUTH TAPE

The SLEUTH tape is an I/O stand-alone diagnostic tape. The step-by-step procedure for constructing such a tape is contained in the *HP 3000 Series II Computer System, Stand-alone Diagnostic Program (SDUP)*, Diagnostic No. D417, HP part number 03000-90125.

The SLEUTH source program is found in

FIELD . SUPPORT, HP32230

RUN SDUP. HP32230 . SUPPORT; NOPRIV - (follow programmed console instructions to create a stand-alone diagnostic tape).

## 3-3. MAKT AND BA

To make an operating diagnostic tape from the SLEUTH tape constructed in paragraph 3-2, refer to the MAKT (Make Test Tape) command and the BA (Batch) command.

### 3-4. OPERATING INSTRUCTIONS

To cold-load the SLEUTH tape, proceed as follows:

- a. Mount cold loadable diagnostic tape containing SLEUTH. Press ON-LINE UNIT 0.
- b. Set Switch Register to %003006. Simultaneously depress

ENABLE  
LOAD

A short length of tape should load. The CPU will be in halt mode.

- c. Set the Switch Register to the diagnostic number of SLEUTH.
- d. Press RUN.

The tape should load and rewind.

- e. Depress carriage-return on the console; the following message will be printed.

```
D1 SLEUTH 3000 (HP D411A.XX.YY)
(c) COPYRIGHT HEWLETT-PACKARD COMPANY 1978
> 10.
```

- f. Any valid SLEUTH command may now be entered.

#### NOTE

Typing control A on the console aborts any operation and places SLEUTH in the input mode. The user's program and data are intact.

#### CAUTION

Control A will not always break SLEUTH out of a loop. If another character other than control A is typed in, any subsequent control A will have no effect.

# ERROR MESSAGES

SECTION

IV

## 4-1. INTRODUCTION

This section lists, explains, and shows examples of the various messages a user may encounter while running SLEUTH. There are three types of SLEUTH messages:

- DIAGNOSTIC MESSAGES (DXX)
- ERROR MESSAGES (EXX)
- PROGRAM ERROR MESSAGES (PEXX)

The messages have an alpha/numeric designation; i.e., DXX for diagnostic message, EXX for error message, and PEXX for program error message. Following the designator is a self-explanatory message.

## 4-2 DIAGNOSTIC MESSAGES

There are 11 diagnostic messages, D1 through D11, that alert the operator to miscellaneous conditions that could terminate or otherwise affect a program. See table 4-1 for a list of diagnostic messages.

Table 4-1. Diagnostic Messages

D1	SLEUTH 3000 (HP D411A.XX.YY) (XX = current update; YY = current fix)
D2	PASS COUNTER EQUALS XXXXX
D3	LAST TIO STATUS IS X XXX XXX XXX XXX XXX
D4	LAST CCG STATUS IS X XXX XXX XXX XXX XXX
D5	LAST SIO PROGRAM EXECUTED IS
D6	SIO PROGRAM POINTER EQUALS XXXXXX
D7	NO CURRENT SIO PROGRAM
D8	**POWER FAIL**
D9	LAST SYNDROME IS
D10	REQUESTED STATUS IS
D11	SECTOR ADDRESS IS XXXXXX

### 4-3. ERROR MESSAGES

When SLEUTH detects an error during interpretation or execution of a command, it suppresses execution of that command and prints one or more messages describing the error. There are 15 different error messages contained in SLEUTH. See table 4-2 for a list of error messages.

The following are examples of error messages:

Example 1:

```
> 10 DEV 0,10,6,10,1,2400
> 10 RP 0,72
> 20 END
> 30 RUN
```

```
E2 RP FAILED IN STEP 10
STATUS IS   0 100 110 100 000 000
SHOULD BE  0 100 110 000 DDD DDD
```

The above example is an error from Ripple Print on the Async MUX. The status indicates a break character was sensed.

Example 2:

```
> 10 DEV 0,8,14,10,0
> 10 RC 0
> 20 END
> 30 RUN
```

```
E2 RC FAILED IN STEP 10
STATUS IS   1 111 000 111 111 010
SHOULD BE  1 111 000 011 111 000
CYLINDER = 0,SECTOR = 0,HEAD = 0
D5 LAST SIO PROGRAM EXECUTED IS
  35740: 40000 10000 34000 71400
D6 SIO PROGRAM POINTER EQUALS 35744
> 30
```

The above example indicates pack change status was encountered while doing a recalibrate operation on an HP 2888A Disc.



Example 3:

```
> 10 DEV 0,7,14,3,1
> 10 DB AA,128,0
> 10 SEEK 0,200,5,3
> 20 RD 0,AA,202,5,3
> 30 END
> 40 RUN
```

```
E2 RD FAILED IN STEP 20
STATUS IS   1 111 000 000 111 001
SHOULD BE  1 111 000 000 000 001
CYLINDER = 200,SECTOR = 3,HEAD = 5
D5 LAST SIO PROGRAM EXECUTED IS
 35740:  41312  50503  77600  36307  34000  177777
D6 SIO PROGRAM POINTER EQUALS  35744
> 40
```

The above example attempts a read operation from a cylinder over which the heads are not positioned.

Example 4:

```
> 10 DEV 0,5,14,10,0
> 10 DB AA,128,0
> 10 CLR 0,AA
> 20 END
> 30 RUN
```

```
E2 CLR FAILED IN STEP 10
STATUS IS   1 111 000 001 001 000
SHOULD BE  1 111 000 000 000 000
CYLINDER = 0,SECTOR = 0,HEAD = 0
D5 LAST SIO PROGRAM EXECUTED IS
 35740:  40000  0 77600  36307  34000  177777
D6 SIO PROGRAM POINTER EQUALS 35744
> 30
```

The above example has encountered a cyclic error while executing a cold load read.

Example 5:

```
> 10 DEV 0,7,24,10,0
> 10 DB AA,1,W
> 1 (%52525)
> 10 DB BB,1,0
> 10 MC 0
> 20 WIO 0,%52525
> 30 RIO 0,BB
> 40 CB 0,AA,BB,1
> 50 END
> 60 RUN
```

E1 RIO FAILED IN STEP 30 CONDITION CODE = CCG

E2 CB FAILED IN STEP 40

DATA WORD 0 IS 0 000 000 000 000 001

SHOULD BE 0 101 010 101 010 101

> 60 RUN

The above example shows that an RIO operation failed as the device was busy. The subsequent compare buffer operation then also failed.

Table 4-2. Error Messages

E1	XXXX FAILED IN STEP XXX CONDITION CODE = XXX
E2	XXXX FAILED IN STEP XXX
E3	STATUS IS X XXX XXX XXX XXX XXX
E4	SHOULD BE X XXX XXX XXX XXX XXX
E5	CYLINDER = XXX,SECTOR = XX,HEAD = XX
E6	DATA WORD XXXX IS X XXX XXX XXX XXX XXX
E7	SHOULD BE X XXX XXX XXX XXX XXX
E8	TRACK = XXX,ARC = XX
E9	MISSING INTERRUPT IN STEP XXX
E10	UNEXPECTED INTERRUPT FROM DEVICE XXX IN STEP XXX
E11	ADDRESS READ IS XXXXXX, XXXXXX
E12	CHANNEL IS XX SHOULD BE XX
E13	ERROR COUNT ON LOGICAL UNIT X EXPIRED
E14	BUFFER IS NOT CORRECTABLE
E15	RIO STATUS IS X XXX XXX XXX XXX XXX

#### 4-4. PROGRAM ERROR MESSAGES

There are 32 program error messages contained in SLEUTH. The program error messages suppress program execution and print program error messages describing the program mistake. See table 4-3 for a list of program error messages. In addition to the error message, an arrow (↑) or caret (^) (depending on the type of terminal used) will point to the column position of the detected error.

#### NOTE

If you enter control X, SLEUTH deletes the line and prints three exclamation points (!!!) followed by a carriage return and linefeed. If the next line contains an error, the arrow or caret will point to where the erroneous parameter would have been if control X had not been entered.

Example 1:

```
> 10 DUMP G
      ^
PE1 INCORRECT PARAMETER
> 10
```

The above example shows that an incorrect parameter was encountered after column 10.

Example 2:

```
> 10 DEV 0,6,18,5,0
> 10 RS 0
      ^
PE9 DEVICE TYPE INCOMPATIBILITY
>10
```

The above example indicates an operation was attempted which the device cannot perform, i.e, random seek to a mag tape.

Example 3:

```
> 10 DEV 15,6,18,10,0
      ^
PE8 INVALID LOGICAL DEVICE
> 10
```

The above example indicates a logical unit was declared out of range.

Example 4:

```
> 10 LOOP 10,999999
      ^
PE16 NUMERIC VALUE OUT OF RANGE
> 10
```

The above example attempts using a numeric value which is out of range.

Example 5:

```
> 10 DB AA,9000,R
      ^
PE7 MAXIMUM BUFFER AREA DEFINED
> 10
```

The above example tries to define too large a buffer.

Example 6:

```
> 10 CAB 0
      ^
PE0 ILLEGAL INSTRUCTION
> 10
```

The above example uses an illegal instruction.

Example 7:

```
> 10 LOOP 1000,5
      ^
PE2 STEP # OUT OF RANGE
> 10
```

The above example attempts to loop to a step which is out of range.

Table 4-3. Program Error Messages

PE0	ILLEGAL INSTRUCTION
PE1	INCORRECT PARAMETER
PE2	STEP # OUT OF RANGE
PE3	END NOT FOUND
PE4	BRANCH TARGET NOT FOUND
PE5	ATTEMPTED DELETION OF ITEM REFERENCED IN STEP XXX
PE6	UNDEFINED BUFFER OR SIO PROGRAM
PE7	MAXIMUM BUFFER AREA DEFINED
PE8	INVALID LOGICAL DEVICE
PE9	DEVICE TYPE INCOMPATIBILITY
PE10	ILLEGAL DRT
PE11	ILLEGAL DEVICE TYPE
PE12	ILLEGAL UNIT NUMBER
PE13	ILLEGAL BAUD RATE
PE14	UNDEFINED SIO PROGRAM
PE15	EXPECTING OCTAL VALUE
PE16	NUMERIC VALUE OUT OF RANGE
PE17	ILLEGAL CYLINDER
PE18	ILLEGAL HEAD
PE19	ILLEGAL SECTOR
PE20	INVALID DELETE
PE21	INVALID REPETITION FACTOR
PE22	INVALID DATA
PE23	SIO AREA FULL, ENTER ENDS
PE24	PROGRAM OVERFLOW
PE25	ILLEGAL COMMAND IN SIO STATE
PE26	XXXX FAILED
PE27	NOT ON TAPE
PE28	MISSING PARENTHESIS
PE29	UNRESOLVED ____ FOR ____ LOOP IN STEP XXX
PE30	SUBSCRIPT OUT OF BOUNDS IN STEP XXX
PE31	MISSING STEP#

Table 5-0. Index of Commands and Mnemonics

INDEX OF COMMANDS			
Command Name and Mnemonic	Page No.	Command Name and Mnemonic	Page No.
Access Buffer (ACB)	5-4	Print (PR)	5-73
Address Record (AR)	5-5	Proceed (PROC)	5-74
ASCII Conversion (ASC)	5-6	Put (PUT)	5-75
Automatic Numbering Resumed (AUTO)	5-7		
Backspace File (BSF)	5-10	Randomize (RAND)	5-76
Backspace Record (BSR)	5-11	Random Seek (RS)	5-77
Batch (BA)	5-8	Read (READ)	5-78
Bump Pass Counter (BUMP)	5-12	Read Address (RA)	5-79
		Read Address Immediate (RAI)	5-80
Change Buffer (CHB)	5-13	Read Clock (RCLK)	5-81
Clear (CL)	5-14	Read Data (RD)	5-82
Cold Load Read (CLR)	5-15	Read Data Immediate (RDI)	5-86
Compare Address (CA)	5-16	Read Full Sector (RFS)	5-87
Compare Buffer (CB)	5-17	Read Full Sector Immediate (RFSI)	5-88
Configure (CONF)	5-18	Read Mask (RMSK)	5-89
Control (CONT)	5-20	Read Next Full Sector (RNFS)	5-90
Control I/O (CIO)	5-21	Read Next Full Sector Immediate (RNFI)	5-91
Correct Buffer (CORB)	5-22	Read Rec with CRCC (RDC)	5-92
Cyclic Check (CC)	5-23	Read with Offset (RWO)	5-93
Cyclic Check Immediate (CCI)	5-24	Read with Offset Immediate (RWOI)	5-94
		Read without Verify (RWV)	5-95
Decremental Seek (DS)	5-25	Read without Verify Immediate (RWVI)	5-96
Define Buffer (DB)	5-26	Recalibrate (RC)	5-97
Delay (DELY)	5-27	Renumber (REN)	5-98
Device (DEV)	5-28	Request Disc Address (RCS)	5-99
Display (DISP)	5-29	Request Sector Address (RSA)	5-100
Dump (DUMP)	5-30	Request Status (RST)	5-101
		Request Syndrome (RYSN)	5-103
Enable Status (ES)	5-34	Return Residue (RRES)	5-105
End (END)	5-35	Rewind (REW)	5-106
End SIO (ENDS)	5-36	Rewind and Reset (RST)	5-107
Erase Program (EP)	5-37	RIO (RIO)	5-108
Expected Status (ESTA)	5-38	Ripple (RP)	5-109
		Run (RUN)	5-110
Flag Track Defective (FTD)	5-39	Seek (SEEK)	5-111
Flag Track Defective Immediate (FTDI)	5-40	Seek Read Data (SKRD)	5-112
For (FOR)	5-41	Seek Write Data (SRWD)	5-113
Format (FMT)	5-42	Select Unit (SELU)	5-115
Forward Space File (FSF)	5-43	Sense (SENS)	5-116
Forward Space Record (FSR)	5-44	Set Bank (SBNK)	5-117
		Set Clock (SCLK)	5-118
Gap (GAP)	5-45	Set Enable/Disable Interrupts (SEQ)	5-119
Get (GET)	5-46	Set File Mask (SFM)	5-120
Go (GO)	5-47	Set Jumpers (SETJ)	5-121
Go To (GOTO)	5-48	Set Mask (SMSK)	5-122
		SIN (SIN)	5-123
Halt (HALT)	5-49	SIO (SIO)	5-124
		Skip Address Read (SA)	5-126
If (IF)	5-50	Skip Address Read Immediate (SAI)	5-127
Incremental Read (IR)	5-51	Status Dump (STAT)	5-128
Incremental Seek (IS)	5-52	Suppress Status (SS)	5-129
Incremental Track (IT)	5-53	Switch Output (SOUT)	5-130
Incremental Write (IW)	5-54		
Initialize Data (ID)	5-55	Terminal Data Interface Loop (TDIL)	5-131
Initialize Data Immediate (IDI)	5-57	Test I/O (TIO)	5-132
Interrupt (INT)	5-58		
		Universal Interface Data Transfer (XDUI)	5-133
Jump (JUMP)	5-59		
		Verify (VER)	5-134
Let (LET)	5-60	Verify Immediate (VERI)	5-135
List (LIST)	5-62		
Load TIO Register (LTIO)	5-63	WIO (WIO)	5-136
Loop (LOOP)	5-64	Write (WRIT)	5-137
		Write Address (WA)	5-138
Make Test Tape (MAKT)	5-65	Write Address Immediate (WAI)	5-139
Master Clear (MC)	5-66	Write Data (WD)	5-140
		Write Data Immediate (WDI)	5-145
Name SIO Program (NAME)	5-67	Write File Mark (WFM)	5-146
Next (NEXT)	5-68	Write Full Sector (WFS)	5-147
No Print (NOPR)	5-69	Write Full Sector Immediate (WFSI)	5-148
		Write Record with Zero Parity (WRZ)	5-149
Pack Certification (PCT)	5-70		
Pause On Error (PE)	5-71		
Poll (POLL)	5-72	Zero Buffer (ZBUF)	5-150

# SLEUTH COMMANDS

SECTION

V

## 5-1. INDEX OF COMMANDS

Table 5-0 is an index of SLEUTH commands and mnemonics. The commands are listed in alphabetical order according to formal name. The referenced page number is that page on which the command is described in detail.

## 5-2. INTRODUCTION

The SLEUTH command repertoire contains many different types of commands. The extensive set of commands gives the user a great range of creative flexibility in the check-out exercising of equipment. The types of SLEUTH commands are:

- Program control which provides the general functions necessary to operate SLEUTH.
- Direct I/O device commands which allow the user to perform direct I/O operations on the interfaces under test.
- SIO commands which allow the user to build and execute his own SIO programs.
- Specialized device commands designed to operate on specific devices.
- Commands which allow the user to check the results of an operation.

The set of SLEUTH commands are divided into three subsets. One subset of commands instruct the computer system to perform certain control, or utility, functions, and to initiate action immediately upon receipt of the command. These commands are executed in the input phase of SLEUTH; they are not part of the user's program. The commands in this subset are:

AUTO	Automatic Numbering
BA	Batch
CONF	Configure
DB	Define Buffer
DEV	Device Declaration
DUMP	Dump
EP	Erase Program
MAKT	Make Test Tape
NAME	Name SIO Program
REN	Renumber Program
RUN	Run Program

The second subset of commands are SIO order pair commands used for creating user SIO programs. The SIO order pair commands are:

CONT	Control order pair
ENDS	End order pair
INT	Interrupt order pair
JUMP	Jump order pair
READ	Read order pair
RRES	Return Residue order pair
SBNK	Set Bank order pair
SENS	Sense order pair
WRIT	Write order pair

The remaining 109 commands are used to write SLEUTH programs that will be subsequently executed. Each command performs a particular function. Each command entered becomes part of the current program and is kept until explicitly deleted or the user logs off the system. Programs may also be saved in one of the system libraries for future use.



### 5-3. SLEUTH COMMANDS

The 129 commands available to the user of SLEUTH are listed, and explained in the following format (see table 5-1):

- FORMAL NAME      Formal command name
- SYNTAX            Recognized command with parameter list
- PARAMETERS       Parameter descriptions
- OPERATION        Functional command descriptions
- DELAY TIME        Default interrupt time out
- EXAMPLE           Example and explanation

Table 5-1. Elements of Command Format

• COMMAND NAME:	is shown in CAPITAL LETTERS.
• SYNTAX:	is shown in CAPITAL LETTERS.
• PARAMETERS:	are shown in CAPITAL LETTERS IN REGULAR TYPE when they are literal information that must always enter exactly as shown;  are shown in <i>lower-case italics</i> when they are variable parameters to be replaced by user supplied information.
• OPTIONAL PARAMETERS:	[ A ] means "A" may be included.  $\left[ \begin{array}{c} A \\ B \end{array} \right]$ means "A" or "B" may be included.  $\left\{ \begin{array}{c} A \\ B \end{array} \right\}$ means "A" or "B" <i>must</i> be included.  [ A ] [ B ] means "A" or "B" may be included in any order.
• USER INPUT:	is underlined where necessary for clarity.
• NUMERICAL VALUE:	is assumed to be decimal except when preceded by percent sign (%) indicating the number to be octal.

# ACCESS BUFFER

# ACB

Accesses single word element of a buffer  
or a user defined SIO program.

FORMAL NAME: ACCESS BUFFER

SYNTAX:

$> \text{ACB} \left\{ \begin{array}{l} \text{buf}(\text{index}) = \text{primary} \\ \text{variable} = \text{buf}(\text{index}) \end{array} \right\}$
--

PARAMETERS:

buf

Two letter name of a buffer or SIO program.

index

Indicates element to be accessed. First element of *buf* is 0. May be a constant or a variable.

primary

May be a constant or a variable.

OPERATION:

ACB accesses single word elements of a buffer or a user defined SIO program.

EXAMPLE:

```
> 10 DB AA, 128, R
> 10 ACB A= AA(1)
> 20 END
> 30 RUN
> 30
```

The above example places the value in element 1 of buffer AA into the variable A.

# AR

# ADDRESS RECORD

Sets logical address into 13037 disc controller.

**FORMAL NAME:** ADDRESS RECORD

**SYNTAX:**

```
> AR lun[,cylinder,head,sector]
```

**PARAMETERS:**

lun Logical unit number. Must be 7905/7920/7925 disc.

cylinder

head

sector

} Disc parameters.

**DELAY TIME:**

Default interrupt time-out = 300 milliseconds.

**OPERATION:**

Sets logical address specified in *cylinder*, *head*, and *sector* into the 13037 disc controller.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 AR 0,5,2,3
> 20 RDA
> 30 DISP 0,D
> 40 END
> 50 RUN
D11 ADDRESS READ IS 5,1003
> 50
```

The above example uses the Address Record (AR) command to set the logical disc address in the 13037 disc controller. The Request Disc Address (RDA) command and the Display (DISP) command display the address.

# ASCII CONVERSION

# ASC

Converts a variable to a decimal or octal ASCII string.

FORMAL NAME: ASCII CONVERSION

SYNTAX: `ASC variable, buf (index) [,8]`

PARAMETERS:

<u>variable</u>	Quantity to be converted.
<u>buf</u>	Buffer name.
<u>index</u>	Leftmost byte element of result field in <i>buf</i> . May be a constant or a variable.

### NOTE

Elements are numbered beginning with 1.

[,8] If specified, octal conversion is performed and a leading “%” is supplied. If omitted, decimal conversion occurs.

OPERATION: The contents of *variable* are converted to octal or decimal ASCII and the result is stored in *buf*. The length of the result field is 6 bytes (decimal) or 8 bytes (octal). In either case, the result is right-justified in the field. The field is stored beginning at element *index*.

### NOTE

Elements are numbered beginning with 1.

EXAMPLE:

```

> 10 DB AA,40,0
> 10 LET A=78
> 20 ASC A,AA(10)
> 30 ASC A,AA(30),8
> 40 END
> 50 RUN
> 50 DUMP B,AA

```

0	0	0	0	40	20040	20067	34000
0	0	0	0	0	0	40	20040
20045	30461	33000	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

The above example demonstrates the conversion of the decimal value 78 (% 116) into its equivalent decimal and octal ASCII strings.

# AUTO

# AUTOMATIC NUMBERING RESUMED

Provides automatic line numbering.

**FORMAL NAME:** AUTOMATIC NUMBERING RESUMED

**SYNTAX:**

```
> AUTO [step n]
```

**PARAMETERS:**

step n

Line number, larger than any line number yet existing in the program, at which automatic numbering is to be resumed.

**OPERATION:**

Automatic numbering mode is normal in SLEUTH, but is terminated whenever an edit command (add, replace, delete, and insert) is used. The AUTO command reenters the automatic numbering mode at step n.

**EXAMPLE:**

```
> 10 NOPR  
> 20 PR  
> 30 SS  
> 40 25 HALT  
> 40 AUTO  
> 40
```

In the above example, line 40 inserts line 25. The AUTO command is used to resume automatic numbering.

```
> 10 SS  
> 20 ES  
> 30 PR  
> 40 NOPR  
> 50 33 HALT  
> 25 AUTO 60  
> 60
```

The above example inserts line 33 and resumes automatic numbering at line 60.

# BATCH

# BA

FORMAL NAME: BATCH

SYNTAX:

```
> BA [ record ]  
      [ E ]
```

PARAMETERS: record Number from 1 to 999 (decimal) indicating which test on the tape is to be restored.

E Indicates tape was made with the MPE Editor and contains ASCII SLEUTH source statements.

OPERATION: BATCH will batch in tests from magnetic tape. The tape must be mounted on DRT 6 unit 0. If the record parameter is specified the test indicated will be restored only. Record is a decimal number between 1 and 999 which will indicate which test on the tape to be restored. If record is not specified the entire tape will be batched in and each test run in succession until end of file. (See MAK T command.) The "E" parameter indicates the tape was made with the MPE Editor and contains ASCII SLEUTH source statements. The file equation used to make this tape must be as follows:

```
FILE T;DEV=TAPE;REC=-80,1,F,ASCII  
/K *T << EDITOR COMMAND >>
```

EXAMPLES:

```
FILE T; DEV=TAPE;REC=-80,1,F,ASCII  
:EDITOR  
  
HP32201A.3.01 EDIT/3000 MON, OCT 14, 1974, 3:21 PM  
/SET SHORT  
/A  
  
1 10 DEV 0,5,13,100,0  
2 10 DB AA,128,W  
3 5(%125252),5(%52525),5(%177777),5(0)  
4 10 DB BB,128,0  
5 10 RC 0  
6 20 RS 0  
7 30 WDI 0,AA  
8 40 RDI 0,BB  
9 50 CB 0,AA,BB,3  
10 60 LOOP 20,100  
11 70 END  
12 //  
  
/K *T  
/
```

The above example indicates how to make an edit tape using the online text editor. This tape can be interpreted using the "BA E" command. For long term compatibility it is suggested that online editor tape be used for saving SLEUTH programs.

**BATCH EXAMPLE:**

```

> 10 BA E
> 10 AUTO
> 80 DUMP P
  10 RC          0
  20 RS          0
  30 WDI        0,    AA
  40 RDI        0,    BB
  50 CB         0,    AA,    BB,    3
  60 LOOP       20,    100,    0,    0
  70 END
> 80 DUMP B
   DB          AA,    128
   DB          BB,    128
> 80
  
```

The above example demonstrates how to batch in the tape created in the previous example. The declarations are displayed using Dump command.

**BATCH EXAMPLE:**

```

> 10 DEV 0,5,14,100,0
> 10 DB AA,2944,R
> 10 DB BB,2944,0
> 10 NOPR
> 20 RC 0
> 30 RC 0
> 40 PR
> 50 IT 0
> 60 WDI 0,AA
> 70 CCI 0,23
> 80 ZBUF BB
> 90 RDI 0,BB
> 100 CB 0,AA,BB,3
> 110 LOOP 50,8119
> 120 END
> 130 MAKT N (makes a new test tape)
> 130 EP
> 10 BA 1 (restores test number 1)
> 130 DUMP P
  10 NOPR
  20 RC          0
  30 RC          0
  40 PR
  50 IT          0,    0,    0,    0
  60 WDI        0,    AA
  70 CCI        0,    23
  80 ZBUF       BB
  90 RDI        0,    BB
  100 CB        0,    AA,    BB,    3
  110 LOOP      50,    8119,    0    0
  120 END
> 130
  
```

The above example demonstrates how to create and restore tests generated under SLEUTH.

# BACKSPACE FILE

# BSF

Issues an SIO backspace file to a magnetic tape unit.

**FORMAL NAME:** BACKSPACE FILE

**SYNTAX:**

```
> BSF lun
```

**PARAMETER:**

> lun

Logical unit number. Must be a magnetic tape unit.

**OPERATION:**

Backspace file will issue an SIO BSF to a specified magnetic tape device.

**DELAY TIME:**

The default interrupt time out is 30 seconds.

**EXAMPLE:**

```
> 10 DEV 0,6,18,30,0  
> 10 GAP 0  
> 20 WFM 0  
> 30 LOOP 10,10  
> 40 BSF 0  
> 50 LOOP 40,9  
> 60 REW 0  
> 70 END  
> 80 RUN  
> 80
```

The above example demonstrates how a backspace file command might be utilized in a user program. Eleven file marks are written on the tape then the tape unit is backspaced to 10 file marks.



# BSR

# BACKSPACE RECORD

Issues an SIO BSR to a magnetic tape unit.

**FORMAL NAME:** BACKSPACE RECORD

**SYNTAX:**

```
> BSR lun
```

**PARAMETER:** lun Logical unit number. Must be a magnetic tape unit.

**OPERATION:** Backspace record will issue an SIO BSR to a specified magnetic tape device.

**DELAY TIME:** The default interrupt time out is 10 seconds.

**EXAMPLE:**

```
> 10 DEV 1,6,18,10,1  
> 10 DB AA,128,R  
> 10 WD 1,AA  
> 20 LOOP 10,10  
> 30 BSR 1  
> 40 LOOP 30,9  
> 50 REW 1  
> 60 END  
> 70 RUN  
> 70
```

The above example writes 128 word records, then the tape unit is backspaced through 10 of these.

# BUMP PASS COUNTER

# BUMP

Increments a pass counter.

**FORMAL NAME:** BUMP PASS COUNTER

**SYNTAX:**

```
> BUMP [P]
```

**PARAMETER:**

P

If P parameter is specified, the pass counter will be printed each time BUMP is executed.

**OPERATION:**

The BUMP command will increment a pass counter each time the command is executed. The pass counter can be displayed with a DUMP C command (see DUMP command).

**EXAMPLE:**

```
> 10 DEV 0,5,13,50,0
> 10 FMT 0
> 20 BUMP P
> 30 LOOP 10,5
> 40 END
> 50 RUN
D2 PASS COUNTER EQUALS 1
D2 PASS COUNTER EQUALS 2
D2 PASS COUNTER EQUALS 3
D2 PASS COUNTER EQUALS 4
D2 PASS COUNTER EQUALS 5
D2 PASS COUNTER EQUALS 6
> 50
```

The above example shows the pass counter being printed each time the BUMP P command is executed.

# CHB

# CHANGE BUFFER

Change buffer contents.

FORMAL NAME: CHANGE BUFFER

SYNTAX:

```
> CHB buf, type
```

PARAMETERS:

buf

Buffer to be changed.

type

Type of change.

TYPE	FUNCTION
A	Fill with address
R	Randomize
I	Increment
D	Decrement
C	Complement
S	Circular left shift (shifts bits within each element)
W	Circular word left shift (shifts words within buffer)
Value	Fill with a numeric value

OPERATION:

The CHB command will change the contents of the specified buffer.

EXAMPLE:

```
> 10 DEV 0,5,17,100,0  
> 10 DB AA,4096,R  
> 10 FOR I=0 TO 511  
> 20 DB BB,4096,0  
> 20 WD 0,AA,1,0  
> 30 RD 0,BB,1,0  
> 40 NEXT I  
> 50 CHB AA,R  
> 60 LOOP 10,1000  
> 70 END  
> 80 RUN  
> 80
```

The above example uses the Change Buffer command to randomize the data buffer.

# CLEAR

# CL

FORMAL NAME: CLEAR

SYNTAX:

```
> CL lun
```

PARAMETER

lun

Logical unit number. Must be a 7905/7920/7925 disc.

OPERATION:

Clear transmits the clear function to the disc drive, clears status, and disconnects all drives.

DELAY TIME:

The default interrupt timeout is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 CL 0  
> 20 END  
> 30 RUN  
> 30
```

The above example issues a Clear command to the 7905 disc.

# CLR

# COLD LOAD READ

Cold load read to moving head disc.

FORMAL NAME: COLD LOAD READ

SYNTAX:

```
> CLR lun,buf
```

PARAMETERS:

lun

Logical unit number. Must be a moving head disc, unit 0.

buf

Buffer into which data is read.

OPERATION:

The Cold Load Read command will issue a cold load read to the specified moving head disc, which must be unit 0 of its DRT. The read will take place at cylinder 0, head 0, sector 0. The information will be read into *buf*. The buffer's length will determine the word count of the read.

## CAUTION

The buffer length must not exceed the length of a physical track, or an End of Cylinder error will occur.

DELAY TIME:

The interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,7,14,25,0  
> 10 DB AA,128,0  
> 10 SEEK 0  
> 20 CLR 0,AA  
> 30 END  
> 40 RUN  
> 40
```

The above example executes a cold load read operation at cylinder 0, head 0, sector 0, and reads into buffer AA.

# COMPARE ADDRESS

# CA

Compares disc address and read command address.

**FORMAL NAME:** COMPARE ADDRESS

**SYNTAX:**

```
> CA lun
```

**PARAMETER:**

lun

Logical device number. Must be an ISS disc.

**OPERATION:**

The Compare Address command compares the cylinder and head address of the internal disc address to the address read from a Read Address command. (See RA command.)

**EXAMPLE:**

```
> 10 DEV 0,7,14,100,0
> 10 IT 0
> 20 WAI 0
> 30 CCI 0,23
> 40 RAI 0
> 50 CA 0
> 60 LOOP 10,8119
> 70 END
> 80 RUN
> 80
```

The above example writes addresses, performs a cyclic check, then reads and compares a singular address of each track.

# CB

# COMPARE BUFFER

Compares contents of two buffers.

FORMAL NAME: COMPARE BUFFER

SYNTAX:

```
> CB lun,buf 1,buf 2,errcount[,maxcount]
```

PARAMETERS:

<u>lun</u>	Logical unit number of device being tested.
<u>buf 1,buf 2</u>	Buffers of which the contents are to be word by word compared.
<u>errcount</u>	Maximum number of errors to be displayed for each execution. May be a variable.
<u>maxcount</u>	Maximum number of words to be compared. Default is the length of the shorter buffer; <u>buf1</u> or <u>buf2</u> . May be a variable.

OPERATION:

The Compare Buffer command will compare word by word each element of buffers 1 and 2.

EXAMPLE:

```
> 10 DEV 0,5,13,25,0
> 10 DB AA,3072,W
> 128 (%125252),128(%52525),128(%177777),128(0),128(%22222),128(%11111)
> 10 DB BB,3072,0
> 10 IT 0
> 20 WDI 0,AA
> 30 CCI 0,24
> 40 ZBUF BB
> 50 RDI 0,BB
> 60 CB 0,AA,BB,3
> 70 LOOP 10,811
> 80 END
> 90 RUN
> 90
```

The above example indicates how a compare buffer operation may be used to help evaluate the results of an operation. One track of information is written on a 7900 disc at one time then each track is cyclic checked, the data read and compared until the entire disc is checked.

# CONFIGURE

Executes a system I/O configuration test.

# CONF

FORMAL NAME: CONFIGURE

SYNTAX:

```
> CONF
```

OPERATION:

Configure will execute an I/O configuration test on the system. The active DRT's will be listed with the device status. The operator must then key in the device type. The type codes may be obtained by the "DUMP T" command. If the test is successful a system configuration table will be produced. If unsuccessful, an appropriate error message is printed, however, the command halts only if the switch register is set for "Halt on Error." All peripheral devices should be placed online and made ready, the timer switch enabled, and the freeze if interrupt switch disabled before entering this command.

## NOTE

I/O hardware problems, such as improper polling of devices, has caused the CPU to go into an infinite loop or halt with an I/O instruction in the CIR. The faulty device number may be found in the TOS registers.

## CONFIGURATOR

### HALT

### MEANING

0	I/O COMMAND REJECTED (CC=CCL).
1	UNEXPECTED INTERRUPT. RA = DEVICE NUMBER.
2	I/O COMMAND REJECTED (CC=CCG).
3	INTERRUPT FROM UNIDENTIFIABLE DEVICE. RA = DEVICE NUMBER.
4	DEVICE INTERRUPTED MORE THAN ONCE PER TEST. RA = DEVICE NUMBER.
6	SIO MUX HAS TWO NON-ZERO RAMS.
%10	INTERRUPTED BY DECLARED NON- INTERRUPT DEVICE.
%11	UNEXPECTED MODULE VIOLATION. RA = PARAMETER.
%12	SIO PROGRAM DID NOT COMPLETE. RA = DEVICE STATUS, RB = DEVICE NUMBER.
%13	DRT CONTAINS WRONG SIO PROGRAM POINTER. RA = SIO POINTER, RB = EXPECTED POINTER, RC = DEVICE NUMBER
%14	SIO MUX ADDRESS RAM WAS WRONG VALUE



**EXAMPLE:**

```
> 10 CONF
DRT STATUS TYPE
3 54600 8
4 40000 1
5 150000 14
6 110016 18
7 140000 23
8 42000 6
9 53400 7
```

THE CPU IS CONFIGURED FOR 64K OF MEMORY

DRT	TYPE	INT	MASK	SIO	DATA	MUX
3	8	0	E	NO	—	—
4	1	NO	—	NO	—	—
5	14	3	E	YES	1	4
6	18	4	E	YES	4	4
7	23	5	E	YES	8	4
8	6	1	E	NO	—	—
9	7	2	E	NO	—	—

```
> 10
```

The above example shows a typical system configuration output. The DRT, type interrupt priority, interrupt mask, SIO service request, and multiplexer channel are listed for each DRT on the system.

# CONTROL

Places a control order in the order address pair of an SIO chain.

# CONT

**FORMAL NAME:** CONTROL

**SYNTAX:**

```
> CONT word1, word2
```

**PARAMETERS:** word1 Specifies the remaining 12 bit field in the IOCW.  
word2 The IOAW of the order pair.

**OPERATION:** Control will place a control order in the order address pair of an SIO chain. (See NAME command and SIO commands.) Control is an SIO order, acceptable only while entering a user-defined SIO program (i.e., between the NAME and ENDS commands).

**EXAMPLE:**

```
> 10 DEV 0,6,18,10,3
> 10 NAME SS
> 10 CONT 0,%1400
> 10 CONT 0,%1411
> 10 ENDS I
> 10 SIO 0,SS,1,5,%114016,%1100
> 20 END
> 30 RUN
> 30
```

The above example demonstrates how control double word SIO orders may be placed into an SIO program. The program issues a rewind-reset command to magnetic tape.

# CIO

# CONTROL I/O

Issues a CIO to specified device.

**FORMAL NAME:** CONTROL I/O

**SYNTAX:**

```
> CIO lun, controlword
```

**PARAMETERS:**

lun

Logical unit number.

controlword

Control I/O word. May be a variable.

**OPERATION:**

CIO will issue a Control I/O command of the control specified in the control word parameter to the indicated unit.

**EXAMPLE:**

```
> 10 DEV 0,5,13,20,0  
> 10 CIO 0,%40000  
> 20 END  
> 30 RUN  
> 30
```

The above example issues a Control I/O command to the device specified. The control word is %40000.

# CORRECT BUFFER

# CORB

Corrects a specified data buffer.

FORMAL NAME: CORRECT BUFFER

SYNTAX:

> CORB lun,buf

PARAMETERS:

lun Logical unit number. Must be a 7905/7920/7925 disc.

buf Buffer to be corrected.

OPERATION:

Correct Buffer will correct the data buffer specified in the BUF parameter according to the last syndrome requested for the logical unit designated.

EXAMPLE:

```

>10 DUMP P
>10 SS
>20 LET B=%177776
>30 LET C=%100000
>40 CHB AA,%177777
>50 WDI 0, AA, %2
>60 SEEK 0, 0, 0
>70 FOR J = 3 TO 128
>80 FOR I = 0 TO 15
>90 RFSI 0, CC
>100 ACB A, =, CC, (, J, )
>110 LET A=(A AND B) OR C
>120 ACB CC ( J ) = A
>130 WFSI 0, CC
>140 RDI 0, BB, %2
>150 RSYN 0
>160 CORB 0, BB
>170 CB 0, AA, BB, 3
>180 LET B=B CSL(1)
>190 LET C=C CSL(1)
>200 NEXT I
>210 CHB AA,%177777
>220 WDI 0, AA, %2
>240 NEXT J
>250 END
>260 RUN
>260

```

Cyclic check a moving head disc.

**FORMAL NAME:** CYCLIC CHECK

**SYNTAX:**

```
> CC lun,secount[,cylinder,head,sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7900/ISS disc.

secount

Number of sectors to be checked. May be a variable.

cylinder

head

sector

Disc parameters. The heads are assumed to be positioned at the correct address. The *cylinder,head,sector* parameters (default 0,0,0) are used to update the internal disc address.

**OPERATION:**

Cyclic Check will perform a cyclic check operation on a moving head disc.

**DELAY TIME:**

The default interrupt time out is 500 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,8,14,5,2
> 10 DB AA,2944,%125252
> 10 SEEK 0,250,0,0
> 20 WD 0,A,250,0,0
> 30 CC 0,23,250,0,0
> 40 END
> 50 RUN
> 50
```

The above example writes one track of data on an ISS disc at cylinder 250, head 0, then cyclic checks that track.

# CYCLIC CHECK IMMEDIATE

# CCI

Cyclic checks a moving head disc.

**FORMAL NAME:** CYCLIC CHECK IMMEDIATE

**SYNTAX:**

```
> CCI lun,secout
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7900/ISS disc.

secout

Number of sectors to be checked. May be a variable.

**OPERATION:**

Cyclic Check Immediate will do a cyclic check on a moving head disc. The heads are assumed to be positioned at the correct address.

**DELAY TIME:**

The Default Interrupt time out is 500 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,50,0
> 10 DB AA,128,R
> 10 RS 0
> 20 WDI 0,AA
> 30 CCI 0,1
> 40 LOOP 10,10
> 50 END
> 60 RUN
> 60
```

The above example performs a seek to random locations of a 7900 disc, writes one sector of information, then cyclic checks that sector.

# DS

# DECREMENTAL SEEK

Seeks a location then decrements by one.

**FORMAL NAME:** DECREMENTAL SEEK

**SYNTAX:**

```
> DS lun [, cylinder, head, sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be a moving head disc.

cylinder

head

sector

} Disc parameters indicating location of initial seek.

**OPERATION:**

Decremental Seek will do an initial seek to the location specified in the cylinder, head, and sector parameters, default is 0,0,0. Each time the instruction is executed the cylinder will be decremented by 1 until 0 then it will seek to the maximum cylinder. This command updates the internal disc address.

## NOTE

Multiple DS commands do not interact with each other (i.e., a second DS command will not update an address controlled by the first).

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,20,0  
> 10 IS 0  
> 20 DS 0  
> 30 LOOP 10,5000  
> 40 END  
> 50 RUN  
> 50
```

The above example indicates how a Decremental Seek operation may be used.

## CAUTION

The initial cylinder address is set only when the program is first run. Thereafter, the cylinder address is only decremented (or reset to the maximum cylinder address if previously 0) even if the program is run many times. This also applies if the program is aborted by the use of control A or by front panel switch 13, for example, when the sample program above is run the second time, the starting cylinder address for the DS command will be:

$$203 - (5001 \text{ MOD } 203) = 203 - 129 = 74$$

To re-initialize the address, the command must be re-entered.

# DEFINE BUFFER

# DB

Defines buffer.

**FORMAL NAME:** DEFINE BUFFER

**SYNTAX:**

```
> DB name,length,datatype
```

**PARAMETERS:**

name

Two letter buffer name.

length

Number of words allocated to buffer. Length cannot exceed 8,192 words. Total length of all buffers cannot exceed 16,333 words.

datatype

The type of data to be contained in buffer:

S - String Data  
W - Word Data  
B - Byte Data  
R - Random Data

*value* - The buffer is filled with the numeric value.

S, W and B type buffers will have repetition factors. If a repetition factor is larger than a buffer it will be truncated. If smaller, the repetition will be repeated until the buffer is full.

**OPERATION:**

Define a buffer.

**EXAMPLES:**

```
> 10 DB AA,100,W  
> 50 (%12345),50(%54321)
```

The above example will fill a 100 word buffer with 50 words of the octal pattern 12345 and 50 words of the octal pattern 54321.

```
> 10 DB BB,120,S  
> THIS IS AN EXAMPLE
```

The above example will fill a 120 word buffer with the string "THIS IS AN EXAMPLE". If repeating the string does not exactly fill the buffer the last repetition of the string will truncate the remainder.

```
> 10 DB AS,4000,R
```

The above example will fill a 4000 word buffer named AS with random data.

```
> 10 DB CC,250,B  
> 100(%123), 100(%321), 50(%177)
```

The above example will fill a 250 word buffer with 100 bytes of 123, 100 bytes of 321, 50 bytes of 177 and will repeat the repetition of each field of bytes until the buffer is filled.

```
> 10 DB DD,20,%125252
```

The above example will fill a 20 word buffer with the octal pattern 125252.

To delete a buffer simply redefine it. If the buffer has been referenced in the current SLEUTH program, the delete will fail and an error message (PE5) will be displayed.

```
> 10 DB DD  
> 10 DB DD,30,%52525
```

The above example deletes the buffer, then redefines it.



# DELY

# DELAY

Alters the interrupt time out.

**FORMAL NAME:** DELAY

**SYNTAX:**

```
> DELY increment
```

**PARAMETER:**

*increment*

Indicates number of 100 millisecond quanta to be allocated to the time out. May be a constant or a variable.

**OPERATION:**

Delay will alter the interrupt time out of the next command which utilizes interrupt time out.

**EXAMPLE:**

```
> 10 DEV 0,6,18,5,0  
> 10 DELY 100  
> 20 REW 0  
> 30 END  
> 40 RUN  
> 40
```

In the above example the Delay command was used to change the default interrupt time out for the rewind operation to 10 seconds.

# DEVICE

Defines device.

# DEV

FORMAL NAME: DEVICE

SYNTAX: > DEV lun,drt,type,errs,unit [,baud]

PARAMETERS:

<u>lun</u>	Logical unit number; 0 to 7.
<u>drt</u>	DRT number of the devices.
<u>type</u>	Coded number identifying type of device.
<u>errs</u>	Maximum error count the device is allowed; 0 to 999.
<u>unit</u>	Device unit number; 0 to 31.
[ <u>baud</u> ]	Baud rate, 75 to 9600 baud, optional.

OPERATION: The Device command allows the user to define the characteristics of a particular device and to assign an error count and logical unit number to that device.

TYPE	
CODES	CORRESPONDING DEVICE
1	SIO MUX CHANNEL
2	CARD READER
3	SYNC SINGLE LINE
5	2607A/13A/17A/18A PRINTER
6	ASYNC MUX CHANNEL
7	TERMINAL CONTROLLER
8	SYSTEM CLOCK
9	SEL CHAN MAINT CARD
10	READER/PUNCH
11	7925 DISC
12	7920 DISC
13	7900 DISC
14	ISS MOVING HEAD DISC
15	7905 DISC
16	256 TRACK F.H. DISC
17	512 TRACK F.H. DISC
18	800 CPI MAG TAPE
19	1600 CPI MAG TAPE
20	PAPER TAPE READER
21	PAPER TAPE PUNCH
22	PLOTTER
23	2610A/14A LINE PRINTER
24	SPECIAL INTERFACE (any device not having special SLEUTH commands)
25	UNIVERSAL INTERFACE

EXAMPLE:

```

> 10 DEV 0,%11,6,10,2,2400 << ASYNC MUX AT 2400 BAUD >>
> 10 DEV 1,5,14,3,0 << ISS DISC >>
> 10 DEV 2,6,18,5,1 << 800 CPI MAG TAPE >>
> 10 DEV 0,10,13,2,0 << REDEFINES LUN 0 AS 7900 DISC >>

```

For further reference see DUMP T and DUMP L commands.

# DISP

# DISPLAY

Displays data.

FORMAL NAME: DISPLAY

SYNTAX:

```
> DISP lun,type
```

OPERATION:

Display will display the item specified in the *type* parameter for the *lun* indicated. *Lun* must be a 7905/7920/7925 disc.

Type	Function
Y	Last Syndrome
D	Disc Address
S	Sector Address
R	Requested Status

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0
> 10 RS 0
> 20 RQST 0
> 30 DISP 0,R
> 40 END
> 50 RUN
D10 REQUESTED STATUS IS
  0 001 111 100 000 000
  0 000 010 000 101 000
> 50
```

# DUMP

Displays data.

# DUMP

FORMAL NAME: DUMP

SYNTAX:

```
> DUMP type [;FAST]
```

PARAMETERS: type

<u>T</u>	Type Code.
<u>P</u> [ <u>firststep</u> / <u>laststep</u> ]	Program dump will be between the step numbers designated.
<u>C</u>	Pass counter dump.
<u>L</u>	Logical unit table dump.
<u>V</u> [ , <u>var</u> ]	Variable Dump. Dumps current value of all variables. If designated, the contents of a single variable may be displayed.
<u>B</u> [ , <u>buf</u> ]	Buffer Dump. Dumps the name and size of all declared buffers. If specified, the contents of the buffer indicated will be displayed.
<u>S</u> [ , <u>name</u> ]	SIO program dump. Dumps the name of all declared SIO programs. If specified, the contents of the SIO program will be displayed.
<u>;<u>FAST</u></u>	If ;FAST is designated the item will be listed on the lineprinter. Otherwise the console.

OPERATION: Dump will list the item specified in the TYPE parameter.

EXAMPLES:

```
> 10 DEV 0,5,14,20,0
> 10 RS 0
> 20 LOOP 10,50
> 30 RC 0
> 40 SEEK 0,405,0,0
> 50 SEEK 0
> 60 LOOP 40,400
> 70 END
> 80 DUMP P
  10 RS      0
  20 LOOP   10,    50,    0,    0
  30 RC      0
  40 SEEK   0    405,    0,    0
  50 SEEK   0      0,    0,    0
  60 LOOP   40    500,    0,    0
  70 END
> 80
```

In the above example a program is declared and then displayed using the DUMP P command.

# DUMP

# DUMP

Continued

## EXAMPLES:

```
> 10 NAME SS
> 10 CONT 0,0
> 10 WRIT BB
> 10 ENDS,I
> 10 NAME SI
> 10 CONT 0,0
> 10 READ CC
> 10 ENDS
> 10 NAME FF
> 10 SENS
> 10 ENDS,I
> 10 DUMP S
    NAME      SS
    NAME      SI
    NAME      FF
> 10
```

The above example builds 3 SIO programs. The DUMP S command is used to display the declarations.

```
> 10 DUMP S, SS
40763:      40000      0  64060  41770  34000  17777
> 10
```

The above example uses the DUMP S command to display the contents of a SIO program.

```
> 10 DUMP B,CC
45710:      1      1      1      1      1
> 10
```

The above example uses the DUMP B command to display the contents of a buffer.

```
> 10 DUMP C
D2 PASS COUNTER      0
> 10
```

The above example shows how to display the contents of the pass counter.

# DUMP

Continued

# DUMP

## EXAMPLES:

```
> 10 LET A=50
> 20 LET B=A*2
> 30 END
> 40 RUN
> 40 DUMP V
      A = 50
      B = 100
      C = 0
      D = 0
      E = 0
      F = 0
      G = 0
      H = 0
      I = 0
      J = 0
      K = 0
      L = 0
      M = 0
      N = 0
      O = 0
      P = 0
      Q = 0
      R = 0
      S = 0
      T = 0
      U = 0
      V = 0
      W = 0
      X = 0
      Y = 0
      Z = 0
> 40
```

The above example uses the DUMP V command to display, in decimal, the contents of each variable.

```
> 10 DEV 0,5,15,100,0
> 10 DEV 1,7,14,100,0
> 10 DEV 2,6,18,50,1
> 10 DEV 3,12,6,105,2400
> 10 DUMP L
      LUN#   DEV#   TYPE   ERR#   UNIT#   BAUD
          0       5      15     100     0        0
          1       7      14     100     0        0
          2       6      18     50      1        0
          3      12       6     10      5      2400
          0       0       0       0       0        0
          0       0       0       0       0        0
          0       0       0       0       0        0
          0       0       0       0       0        0
> 10
```

The above is an example of logical unit table dump.

# DUMP

# DUMP

Continued

## EXAMPLES:

```
> 10 DB AA,128,R
> 10 DB BB,2000,%125252
> 10 DB CC,5,1
> 10 DUMP B
      DB AA,          128
      DB BB,          2000
      DB CC,           5
> 10
```

The above example defines 3 buffers. The DUMP B command is used to display the declarations.

```
> 10 DUMPT
TYPE CODE    CORRESPONDING DEVICE
  1          SIO MUX CHANNEL
  2          CARD READER
  3          SYNC SINGLE LINE
  5          2607A/13A/17A/18A PRINTER
  6          ASYNC MUX CHANNEL
  7          TERMINAL CONTROLLER
  8          SYSTEM CLOCK
  9          SEL CHAN MAINT CARD
 10          READER PUNCH
 11          7925 DISC
 12          7920 DISC
 13          7900 DISC
 14          ISS MOVING HEAD DISC
 15          7905 DISC
 16          256 TRACK F.H. DISC
 17          512 TRACK F.H. DISC
 18          800 CPI MAG TAPE
 19          1600 CPI MAG TAPE
 20          PAPER TAPE READER
 21          PAPER TAPE PUNCH
 22          PLOTTER
 23          2610A/14A LINE PRINTER
 24          SPECIAL INTERFACE (any device not having special
      SLEUTH commands)
> 10 25      UNIVERSAL INTERFACE
```

The above is an example of a type code dump.

# ENABLE STATUS

# ES

Enables automatic checking of device status.

FORMAL NAME:       ENABLE STATUS

SYNTAX:

```
> ES
```

OPERATION:

Enable Status will enable automatic checking of device status unless overridden by the setting of the B register switch 7. (See SS command.)

EXAMPLE:

```
> 10 DEV 0,5,13,5,0
> 10 DB AA,128,0
> 10 SEEK 0,100,0,0
> 20 SS
> 30 RD 0,AA,200,0,0
> 40 ES
> 50 RD 0,AA,200,0,0
> 60 END
> 70 RUN
```

```
E2 RD FAILED IN STEP 50
STATUS IS           1 001 000 000 111 000
SHOULD BE           1 001 000 D00 000 D00
CYLINDER = 100,SECTOR= 0,HEAD= 0
D5 LAST SIO PROGRAM EXECUTED IS
  36140:   40310   50000   77600   36507   34000 177777
D6 SIO PROGRAM POINTER EQUALS 36144
> 70
```

The above example suppresses status then executes a read from a cylinder over which the heads are not positioned. The status is then enabled using the ES command and the read is attempted again producing heads mispositioned status.



# END

# END

Terminates program.

**FORMAL NAME:** END

**SYNTAX:**

```
> END
```

**OPERATION:**

End terminates the program and must be the last statement of the program. Any command having step numbers greater than End will not be executed (and will be lost at run time).

## NOTE

Execution of the program will not occur until the Run command is given.

**EXAMPLE:**

```
> 10  
> 20  
> 30  
> 40  
> 50 END  
> 60
```

} Any SLEUTH  
statements

The above example shows that a program must be terminated with an End command.

# END SIO

Places an end order in an SIO chain.

# ENDS

**FORMAL NAME:** END SIO

**SYNTAX:**

```
> ENDS[,I]
```

**PARAMETER:** I If specified, the end will be with interrupt.

**OPERATION:** End SIO will place an End order in an SIO chain. End SIO also terminates entering SIO orders until another NAME command is encountered. (See NAME command and SIO orders.) ENDS is an SIO order acceptable only while entering a user-defined SIO program (i.e., following a NAME command).

**EXAMPLE:**

```
> 10 DEV 0,127,23,10,0
> 10 NAME FF
> 10 CONT 123,345
> 10 SENS
> 10 CONT 1,2
> 10 ENDS
> 10 DUMP S,FF
35502: 40123 345 50000 177777 40001 2 30000 177777
> 10 NAME CC
> 10 RRES
> 10 ENDS,I
> 10 DUMP S,CC
35512: 10000 177777 34000 177777
> 10
```

The above example uses both end orders, with and without interrupt. The SIO programs themselves perform no particular functions on any specific device but are used here only to show how SIO programs might be constructed containing end orders.

# EP

# ERASE PROGRAM

Erases program.

**FORMAL NAME:** ERASE PROGRAM

**SYNTAX:**

```
> EP [S]
```

**PARAMETER:**

S

The logical unit declarations, data and user SIO programs are saved.

**OPERATION:**

Erase Program will eliminate the users program, data, logical unit declarations and user SIO programs.

**EXAMPLE:**

```
> 10 DEV 0,6,18,100,0
> 10 DB AA,8000,R
> 10 WD 0,AA
> 20 REW 0
> 30 END
> 40 RUN
> 40 EP
> 10
```

In the above example, the EP command erases all entries and recycles the line count.

# EXPECTED STATUS

# ESTA

Changes expected status of next command  
which utilizes status checking.

**FORMAL NAME:** EXPECTED STATUS

**SYNTAX:**

```
> ESTA status,mask
```

**PARAMETERS:**

status

Expected Status. May be a constant or a variable.

mask

A word of don't care bits. A 1 bit in the mask corresponds to a don't care bit in the *status*. May be a constant or a variable.

**OPERATION:**

This command changes the expected status of the next command which utilizes status checking.

**EXAMPLE:**

```
> 10 DEV 05,13,100,0  
> 10 ESTA %100000,7  
> 20 SEEK 0  
> 30 END  
> 40 RUN
```

```
E2 SEEK FAILED IN STEP 20  
STATUS IS      1 001 000 011 111 001  
SHOULD BE      1 000 000 000 000 DDD  
CYLINDER = 0,SECTOR= 0,HEAD= 0  
D5 LAST SIO PROGRAM EXECUTED IS  
75660:  40000  20000  30000  11000  
D6 SIO PROGRAM POINTER EQUALS 75664
```

The above example changes the expected status of a Seek command.

# FTD

# FLAG TRACK DEFECTIVE

Flags a defective disc track.

**FORMAL NAME:** FLAG TRACK DEFECTIVE

**SYNTAX:**

```
> FTD lun[,cylinder,head,sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be a moving head disc.

cylinder

head

sector

} Disc parameters indicating the address written in the address field of the first sector transferred. (Default is 0,0,0.) Subsequent sectors get updated addresses. The heads should be previously positioned at the proper cylinder and head address (using the SEEK command) before the FTD command is executed.

**OPERATION:**

Flag Track Defective will flag a single track defective or (in the case of a 7900) one cylinder (48 sectors) will be flagged.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,20,0  
> 10 SEEK 0,100,0,0  
> 20 FTD 0,100,0,0  
> 30 END  
> 40 RUN  
> 40
```

The above example flags that a track is defective starting at cylinder 100 on a 7900 disc.

# FLAG TRACK DEFECTIVE IMMEDIATE      FTDI

Flags a defective disc.

**FORMAL NAME:**      FLAG TRACK DEFECTIVE IMMEDIATE

**SYNTAX:**

```
> FTDI lun
```

**PARAMETER:**      lun                      Logical unit number. Must be a moving head disc.

**OPERATION:**      Flag Track Defective Immediate will flag a single track defective or (in the case of the 7900) one cylinder (48 sectors) will be flagged. The heads are assumed to be positioned at the correct address. The contents of the internal disc address will be written in the address field of the first sector transferred. Subsequent sectors get updated addresses.

**DELAY TIME:**      Default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,100,0  
> 10 SEEK 0,50,0,0  
> 20 FTDI 0  
> 30 END  
> 40 RUN  
> 40
```

The above example uses Flag Track Defective Immediate command to flag that a track is defective on a 7900 Disc.

# FOR

# FOR

FOR-NEXT loop.

**FORMAL NAME:** FOR

**SYNTAX:**

```
> FOR simple variable = initial value TO final value
```

**PARAMETERS:**

<u>simple variable</u>	A letter (A to Z).
<u>initial value</u>	A letter (A-Z) or a constant.
<u>final value</u>	A letter (A-Z) or a constant.

**OPERATION:**

FOR allows repeated execution of any number of steps between the FOR and NEXT commands. Each FOR command must have a NEXT command designating the same variable in the program. FOR command statements may be nested to 26 (maximum number of variables) levels.

The first execution of the For command statement initializes the *simple variable* to the *initial value*. Each execution of the Next command increments the *simple variable*, by one, until it exceeds the *final value*. If the *final value* is less than the *initial value*, no error message is displayed and the steps between the For and Next commands are executed once.

**EXAMPLE:**

```
> 10 DEV 0,5,17,100,0  
> 10 DB AA,4096,R  
> 10 DB BB,4096,0  
> 10 FOR I=0 TO 511  
> 20 WD 0,AA,I,0  
> 30 RD 0,BB,I,0  
> 40 CB 0,AA,BB,3  
> 50 NEXT I  
> 60 END  
> 70 RUN  
> 70
```

The above example utilizes a For command loop to execute a sequence of commands. The variable I, is used within the loop as the track address parameter for reads and writes on a fixed head disc.

# FORMAT

# FMT

Formats a moving head disc.

FORMAL NAME:       FORMAT

SYNTAX:

> <u>FMT lun</u> [ <u>L</u> ] <u>U</u> (7900 disc)	Format 1
> <u>FMT lun</u> [ <u>retry</u> ] (7905,7920,7925 disc)	2
> <u>FMT lun</u> (ISS disc)	3

PARAMETERS:

lun                   Logical unit number. Must be a moving head disc.

L                    7900 lower platter only. }  
U                    7900 upper platter only. } Default = both platters.

retry                Error retry count, 0-15. May  
be a constant or a variable.  
Default is 0.

OPERATION:

Format will format a moving head disc. Format will perform a cyclic check or verify of the sectors written. If the *lun* is an ISS disc a read and compare of address will also be done.

EXAMPLES:

```
> 10 DEV 0,5,13,10,0
> 10 FMT 0
> 20 END
> 30 RUN
> 30
```

The above example will format both platters of a 7900 disc.

```
> 10 DEV 0,5,13,10,0
> 10 FMT 0,L
> 20 END
> 30 RUN
> 30
```

The above example will format the lower platter of a 7900 disc.

```
> 10 DEV 0,5,14,100,0
> 10 NOPR
> 20 RC 0
> 30 RC 0
> 40 PR
> 50 FMT 0
> 60 END
> 70 RUN
> 70
```

The above example formats an ISS disc. The recalibrations will clear pack change status from changing packs and initial turn on.



# FSF

# FORWARD SPACE FILE

Issues a forward space file.

**FORMAL NAME:** FORWARD SPACE FILE

**SYNTAX:**

```
> FSF lun
```

**PARAMETER:**

lun

Logical unit number. Must be a magnetic tape device.

**OPERATION:**

Forward Space File issues a forward space file SIO program to the tape device specified.

**DELAY TIME:**

The default interrupt delay time is 30 seconds.

**EXAMPLE:**

```
> 10 DEV 0,6,18,20,0  
> 10 DB AA,4000,R  
> 10 WD 0,AA  
> 20 WFM 0  
> 30 LOOP 10,10  
> 40 REW 0  
> 50 FSF 0  
> 60 LOOP 50,9  
> 70 END  
> 80 RUN  
> 80
```

The above example writes 11 records of random data with a file mark after each, rewinds the tape, then forward spaces to 10 of them.

# FORWARD SPACE RECORD

# FSR

Issues a forward space record.

**FORMAL NAME:** FORWARD SPACE RECORD

**SYNTAX:**

```
> FSR lun
```

**PARAMETER:** lun Logical unit number. Must be a magnetic tape device.

**OPERATION:** Forward Space Record will issue a forward space record SIO program to the specified magnetic tape device.

**DELAY TIME:** The default interrupt time out is 10 seconds.

**EXAMPLE:**

```
> 10 DEV 0,6,18,30,0  
> 10 DB AA,8000,R  
> 10 WD 0,AA  
> 20 LOOP 10,10  
> 30 REW 0,  
> 40 FSR 0  
> 50 LOOP 40,8  
> 60 REW 0  
> 70 END  
> 80 RUN  
> 80
```

The above example writes 11 records of random data on tape, rewinds the tape, then forward spaces to 9 of them.

# GAP

# GAP

Erases magnetic tape.

**FORMAL NAME:** GAP

**SYNTAX:**

```
> GAP lun
```

**PARAMETER:** lun Logical unit number. Must be a magnetic tape device.

**OPERATION:** Gap will write a gap on the magnetic tape of the unit specified.

**DELAY TIME:** The default interrupt time out is 600 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,6,18,10,0  
> 10 GAP 0  
> 20 END  
> 30 RUN  
> 30
```

The above example erases a 3 inch portion of magnetic tape using a Gap command.

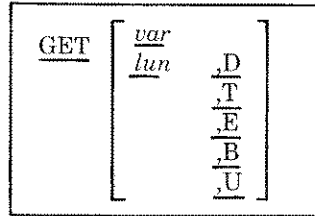
# GET

Obtains information from the operator.

# GET

FORMAL NAME: GET

SYNTAX:



PARAMETERS:

The parameter may be a single variable as designated in the *var* parameter or it may be a parameter of the device command. The *lun* parameter designates which logical unit the request is made for.

- D - DRT number
- T - Type code
- E - Error count
- B - Baud Rate
- U - Unit number

OPERATION:

Get will read from the console the parameter indicated.

EXAMPLE:

```
> 10 PUT "ENTER THE DRT NUMBER"
> 20 GET 0,D
> 30 PUT "NUMBER OF ERRORS ?"
> 40 GET 0,E
> 50 PUT "NUMBER OF PASSES ?"
> 60 GET A
> 70 FOR I= 1 TO A
> 80 PUT "EXAMPLE"
> 90 NEXT I
> 100 END
> 110 RUN
ENTER THE DRT NUMBER
6
NUMBER OF ERRORS ?
10
NUMBER OF PASSES ?
3
EXAMPLE
EXAMPLE
EXAMPLE
> 110
```

The above example shows how the Get command may be used to dynamically obtain information from the operator.

# GO

# GO

Conditionally transfers control to line specified.

FORMAL NAME: GO

SYNTAX:

```
GO stepn,status,mask
```

PARAMETERS:

step n

Program will branch to step n.

status

Termination target. May be a constant or a variable.

mask

A constant or a variable (usually an octal word) designating which bits are not to be evaluated. A 1 bit corresponds to a don't care condition.

OPERATION:

Go will branch to the step specified in the *stepn* parameter until the last TIO or interrupt status is equal to the octal status parameter, in which case the branch will be defeated.

EXAMPLE:

```
> 10 DEV 0,6,18,100,0  
> 10 DB AA,4096,%177777  
> 10 NAME GG  
> 10 CONT 0,0  
> 10 CONT 0,4  
> 10 WRIT AA,C  
> 10 JUMP *-2  
> 10 ENDS  
> 10 SIO 0,GG,0,2,0,%177777  
> 20 TIO 0,  
> 30 GO 20,%2000,%175777  
> 40 MCO  
> 50 REW 0  
> 60 END  
> 70 RUN  
> 70
```

The above example writes the entire tape until the end of the tape with ones a pattern. This example could be used as a skew check. Tape errors are ignored.

# GOTO

Unconditionally transfers control to line specified.

# GOTO

FORMAL NAME: GOTO

SYNTAX:

```
> GOTO stepn
```

PARAMETERS:

stepn

Program will branch to *stepn*.

OPERATION:

GOTO will branch to the step specified in *stepn* parameter (unconditional branch).

EXAMPLE:

```
> 10 LETA = 0  
> 20 GOTO 30  
> 30 LETA = A+1  
> 40 LIST A  
> 50 LOOP 30,10  
> 60 END  
> 70 RUN
```

Program will list 0 through 10 on system console.

# HALT

# HALT

Performs a halt %17.

**FORMAL NAME:** HALT

**SYNTAX:**

```
> HALT
```

**OPERATION:** Halt will perform a halt %17. Pressing run will restart program execution.

**EXAMPLE:**

```
> 10 HALT  
> 20 END  
> 30 RUN  
> 30
```

The above program executes a halt %17.

# IF

Conditional transfer.

# IF

FORMAL NAME: IF

SYNTAX:

```
> IF < pri > < relop > < pri > THEN < step >
```

PARAMETERS:

< pri > ::= < var > | < constant >  
< var > ::= A|B|C|. . .|Z  
< constant > ::= < digit > . . .  
< relop > ::= =|>|<|<>|>=|<=  
< step > ::= target line number.

OPERATION:

The IF statement allows conditional transfer of control based on the relation between two primaries. If the condition is met, control is transferred to the step specified in the *step* parameter.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 LET A=5  
> 20 SEEK 0,A,0,0  
> 30 LET A=A+5  
> 40 IF A<410 THEN 20  
> 50 END  
> 60 RUN  
> 60
```

The above example issues 5 cylinder increment seeks on a 7905 disc. The IF command is used to check the bounds of the cylinder address in the variable A.



FORMAL NAME: INCREMENTAL READ

SYNTAX:

```
> IR lun,buff [track,arc]
```

PARAMETERS:

<u>lun</u>	Logical unit number. Must be a fixed head disc.
<u>buff</u>	Buffer into which data is read.
<u>track</u>	} Disc parameters. Default = 0,0
<u>arc</u>	

OPERATION:

Incremental Read will do a read from the track and arc specified into a buffer. The read will continue until the buffer is filled or 128 words have been transferred. Each time this command is encountered the arc address will be incremented by one until arc 31 has been accessed. The next increment will proceed to arc 0 of the next track. When the last track has been completely processed, the command continues at track 0, arc 0.

#### NOTE

Multiple IR commands do not interact with each other (i.e., a second IR command will not update an address controlled by the first).

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,17,100,0
> 10 DB AA,128,R
> 10 DB BB,128,0
> 10 IW 0,AA,0,0
> 20 ZBUF BB
> 30 IR 0,BB,0,0
> 40 CB 0,AA,BB,3
> 50 LOOP 10,16383
> 60 END
> 70 RUN
> 70
```

The above example shows how Incremental Read may be used to test a fixed head disc. One sector at a time is written then read and compared.

#### CAUTION

The initial track and arc address is set only when the program is first run. Thereafter, the addresses are only incremented according to the description above; even if the program is run many times. For example, if the last execution of the IR command was to track 24, arc 15, the next time the program is run, the first execution will be to track 24, arc 16. This also applies if the program is aborted by the use of control A or by front panel switch 13. To re-initialize the address, the command must be re-entered.

# INCREMENTAL SEEK

# IS

FORMAL NAME: INCREMENTAL SEEK

SYNTAX:

```
> IS lun[cylinder, head, sector]
```

PARAMETERS:

lun Logical unit number. Must be a moving head disc.

cylinder  
head  
sector } Disc parameters.

OPERATION:

Incremental Seek will do an initial seek to the address specified in the cylinder, head and sector parameters. Each time this command is executed it will increment the cylinder address. If the cylinder address exceeds the largest address defined for the particular disc, the command will seek to 0 and start over. This command updates the internal disc address. Default *cylinder*, *head*, and *sector* is 0,0,0.

## NOTE

Multiple IS commands do not interact with each other (i.e., a second IS command will not update an address controlled by the first).

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,13,100,0  
> 10 IS 0  
> 20 LOOP 10,2000  
> 30 END  
> 40 RUN  
> 40
```

The above example causes a 7900 disc to execute 1 cylinder Increment Seeks.

## CAUTION

The initial cylinder address is set only when the program is first run. Thereafter, the cylinder address is only incremented (or reset to 0 if the largest cylinder address is exceeded) even if the program is run many times. This also applies if the program is aborted by the use of control A or by front panel switch 13. For example, when the sample program above is run the second time, the starting cylinder address will be:

2001	Number of times IS was executed.
MOD	
203	Number of cylinders in a 7900.
= 174	Cylinder address at start of second run.

To re-initialize the address, the command must be re-entered.

# IT

# INCREMENTAL TRACK

FORMAL NAME: INCREMENTAL TRACK

SYNTAX: > IT lun[,cylinder,head,sector]

PARAMETERS: lun Logical unit number. Must be a moving head disc.

cylinder  
head  
sector } Disc parameters.

OPERATION: Incremental Track will do an initial seek to the location specified in the cylinder, head and sector parameters. Thereafter, each time it is encountered, it will increment the head address by 1 until the last head has been accessed and then the next increment will proceed to head 0 of the next cylinder. If the next cylinder exceeds the maximum cylinder address for the drive specified by lun, the command will seek to 0 and start over. This command updates the Internal Disc Address.

### NOTE

Multiple IT commands do not interact with each other (i.e., a second IT command will not update an address controlled by the first.

DRIVE	DEVICE TYPE	LAST HEAD	LAST CYLINDER
7900	13	3	203
ISS	14	19	405
7905	15	2	410
7920	12	4	822
7925	11	8	822

DELAY TIME: The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,7,14,100,0
> 10 DB AA,2944,W
> 128 (%125252),128(%52525),128(%177777),128(0),128(%22222),128(%11111)
> 10 DB BB,2944,0
> 10 IT 0
> 20 WDI 0,AA
> 30 CCI 0,23
> 40 ZBUF BB
> 50 RDI 0,BB
> 60 CB 0,AA,BB,5
> 70 LOOP 10,8119
> 80 END
> 90 RUN
> 90
```

The above example indicates how Increment Track may be used to create tests on an ISS disc. Each track is written with the patterns in buffer AA, cyclic checked then the data is read and compared.

### CAUTION

The initial cylinder and head address is set only when the program is first run. Thereafter, the addresses are only incremented according to the description above, even if the program is run many times. For example, if the last execution of an IT command in a program was to cylinder 35, head 1, the first execution the next time the program is run will be to cylinder 35, head 2. This also applies if the program is aborted by the use of control A or by front panel switch 13. To re-initialize the address, the instruction must be re-entered.

# INCREMENTAL WRITE

# IW

FORMAL NAME: INCREMENTAL WRITE

SYNTAX:

```
> IW lun, buff, {track, arc}
```

PARAMETERS:

- lun Logical unit number. Must be a fixed head disc.
- buff Buffer from which data is written.
- track } Disc parameters. Default = 0,0.
- arc }

OPERATION:

Incremental Write will do a write to the *track* and *arc* specified. The write will continue until the buffer is exhausted or the word count reaches 128. Each time this command is encountered the arc address will be incremented by one until arc 31 has been accessed. The next increment will proceed to arc 0 of the next track. When the last track has been completely processed, the command continues at track 0, arc 0.

### NOTE

Multiple IW commands do not interact with each other (i.e., a second IW command will not update an address controlled by the first).

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,17,50,0  
> 10 DB AA,128,%666666  
> 10 DB BB,128,%177777  
> 10 DB CC,128,%125252  
> 10 DB DD,128%52525  
> 10 DB EE,128,0  
> 10 IW 0,AA,0,0  
> 20 IR 0,EE,0,0  
> 30 CB 0,AA,EE,3  
> 40 IW 0,BB,0,0  
> 50 IR 0,EE,0,0  
> 60 CB 0,BB,EE,3  
> 70 IW 0,CC,0,0  
> 80 IR 0,EE,0,0  
> 90 CB 0,CC,EE,3  
> 100 IW 0,DD,0,0  
> 110 IR 0,EE,0,0  
> 120 CB 0,DD,EE,3  
> 130 LOOP 10,16383  
> 140 END  
> 150 RUN  
> 150
```

The above example uses Incremental Write to check a fixed head disc. Four different patterns are written and each pattern is read and checked.

### CAUTION

The initial track and arc address is set only when the program is first run. Thereafter, the addresses are only incremented according to the description above, even if the program is run many times. For example, if the last execution of the IW command was to track 24, arc 15, the first execution the next time the program is run will be to track 24, arc 16. This also applies if the program is aborted by the use of control A or by front panel switch 13. To re-initialize the address, the instruction must be re-entered.

# ID

# INITIALIZE DATA

Initializes device

FORMAL NAME: INITIALIZE DATA

SYNTAX:

		Format
>	<u>ID lun,buf</u> [,cylinder,head,sector]	1
>	<u>ID lun,buf</u> [,mask[,flag[,cylinder,head,sector]]]	2

PARAMETERS:

lun Logical unit number. Must be a 7900 (format 1) or 7905/7920/7925 (format 2) disc.

buf Buffer from which data is written.

cylinder  
head  
sector } Disc parameters indicating the address written in the address field of the first sector transferred. (Default is 0,0,0.) Subsequent sectors get updated addresses. The heads should be previously positioned at the proper cylinder and head address (using the SEEK command) before the ID command is executed.

flag Flags the track as:  
S - Spare  
P - Protected  
D - Defective  
N - Non-flagged (default)

mask Loads file mask on controller. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented at End-of-Cylinder; otherwise, it will be incremented.
13	Allow automatic spare track access.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

Default mask is cylinder mode, no retries. May be a constant or a variable. Only bit 8-15 are used.

# INITIALIZE DATA

# ID

Continued

**OPERATION:** Initialize Data will perform an initialize operation on a 7900, 7905, 7920, or 7925 disc. The initialize operation will begin at the cylinder, head, and sector designated in the most recent seek-type command (e.g., SEEK, IS, DS) and will continue until the word count of the buffer is exhausted.

**DELAY TIME:** The default interrupt time out is 500 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,6144,0
> 10 SEEK 0,10,0,0
> 20 ID 0,AA,3,D,400,0,0
> 30 SEEK 0,400,0,0
> 40 ID 0,AA,3,S,10,0,0
> 50 SEEK 0,10,0,0
> 60 RDI 0,AA,7
> 70 LOOP 60,10
> 80 END
> 90 RUN
> 90
```

The above example seeks to cylinder 10, head 0 of a 7905 disc, initializes the track defective with an alternate cylinder head address of 400/0, then seeks to cylinder 400, head 0, and initializes the track spare with a cylinder head address of 10/0. The program then seeks back to physical address 10/0 and attempts a read operation with sparing enabled. The disc will then automatically seek to logical address 10/0 (physical cylinder/head 400/0) and read.

```
> 10 DEV 0,4,13,100,0
> 10 DB AA,3072,0
> 10 SEEK 0,20,0,0
> 20 ID 0,AA,20,0,0
> 30 END
> 40 RUN
```

The above example seeks to cylinder 20, head 0 of a 7900 disc and initializes the entire track.

# IDI

# INITIALIZE DATA IMMEDIATE

FORMAL NAME: INITIALIZE DATA IMMEDIATE

SYNTAX:

> <u>IDI</u> <i>lun,buf</i>	Format
> <u>IDI</u> <i>lun,buf[,mask[,flag]]</i>	1
	2

PARAMETERS:

lun Logical unit number. Must be a 7900 (format 1) or 7905/7920/7925 (format 2) disc.

buf Buffer from which data is written. Buffer length determines word count of the write.

flag Flags the track as:  
S - Spare  
P - Protected  
D - Defective  
N - Non-flagged (default)

mask Loads file mask on controller. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented at End-of-Cylinder; otherwise, it will be incremented.
13	Allow automatic spare track access.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, end-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

Default mask is cylinder mode, no retries. Mask may be a constant or a variable. Only bits 8-15 are used.

OPERATION:

Initialize Data Immediate will perform an initialize operation on a 7900 or 7905/7920/7925 disc. The heads are assumed to be properly positioned at the correct address. The internal disc address is the address written in the address field of the first sector transferred. Subsequent sectors get updated addresses.

DELAY TIME:

The default interrupt time out is 500 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,6144,0
> 10 IS 0
> 20 IDI 0,AA
> 30 IS 0,0,1,0
> 40 IDI 0,AA
> 50 LOOP 10,410
> 60 END
> 70 RUN
> 70
```

The above example formats the upper cartridge on a 7905 disc.

# INTERRUPT

# INT

Places an interrupt order in an SIO chain.

**FORMAL NAME:** INTERRUPT

**SYNTAX:**

```
> INT
```

**OPERATION:**

Interrupt will place an interrupt order in an SIO chain (see SIO orders and NAME command). Interrupt is an SIO order, acceptable only while entering a user-defined SIO program (i.e., between the NAME and ENDS commands).

**EXAMPLE:**

```
> 10 DEV 0,6,24,10,0  
> 10 NAME IN  
> 10 INT  
> 10 ENDS  
> 10 SIO 0,IN,1,2,%100000,%77777  
> 20 END  
> 30 RUN  
> 30
```

The above example uses an Interrupt order to generate an interrupt from magnetic tape.



# JUMP

# JUMP

Places a jump order in an SIO chain

FORMAL NAME: JUMP

SYNTAX:

```
JUMP address,C
```

PARAMETERS: address Self-relative address which to jump.

C Specifies a conditional jump.

OPERATION: Jump will place a Jump order in an SIO chain. (See NAME command and SIO orders.) Jump is an SIO order, acceptable only while entering a user-defined SIO program (i.e., between the NAME and ENDS commands).

EXAMPLE:

```
> 10 DEV 0,5,13,10,0
> 10 DB AA,3072,%177777
> 10 NAME DF
> 10 CONT 0,%100000
> 10 WRIT AA
> 10 JUMP *-4
> 10 ENDS
> 10 SEEK 0
> 20 SIO 0,DF,0,2,0,%177777
> 30 HALT
> 40 END
> 50 RUN
> 50
```

The above example shows how a Jump command may be used. The program causes the continual transfer of data to a 7900 disc. The CPU halts but the I/O system remains active.

```
> 10 DEV 0,12,4,10,0
> 10 DB AA,3072,R
> 10 DB SA,2,W (Address Record parameters)
I(100),I(0)
> 10 NAME DR (Start of SIO program)
> 10 CONT 0,%6000 (Address Record)
> 10 WRIT SA
> 10 CONT 0,%2400 (Read)
> 10 READ AA
> 10 JUMP *8,C (Repeat if error during transfer)
> 10 ENDS, I (End of SIO program)
> 10 SEEK 0,100,0,0
> 20 SFM 0,%62 (Set cylinder mode, allow three retries)
> 30 SIO 0,DR,1,2,0,0 (Execute the SIO program)
> 40 END
> 50 RUN
> 50
```

The above example shows the use of a Conditional Jump. The program seeks to cylinder 100, head 0, sector 0 of a 7920 disc and reads 24 sectors. If a data error is detected, the program will retry the transfer up to three times before reporting the error.

# LET

Assignment command.

# LET

FORMAL NAME: LET

SYNTAX:

```
> LET <var> = <expr>
```

PARAMETERS:

- <expr> ::= <primary> <op> <expr> | <uop> <expr> | TIO (<expr>)
- <primary> ::= <var> | <constant>
- <var> ::= A | B | C | . . . | Z
- <constant> ::= <digit> . . .
- <uop> ::= NOT
- <op> ::= LSL | LSR | ASL | ASR | CSL | CSR | \* | / | MOD | + | AND | XOR | OR

OPERATION:

Let allows the execution of arithmetic and logical operations on constants and variables. An expression defines a sequence of arithmetic or logical operations which result in a single value. A variable can be any single alpha digit from A to Z. A constant may be a decimal or octal value in the range of 0 to %177777. The reserved word TIO defines the last TIO status from any device. The order of precedence is:

- Rank 1: Any expression in parentheses. NOT (unary one's complement).
- Rank 2: All shift operators.
- Rank 3: Multiply, divide and modulo operators.
- Rank 4: Addition and subtraction operators.
- Rank 5: Logical and operator.
- Rank 6: Logical exclusive - or operator.
- Rank 7: Logical or operator.

EXAMPLE:

```
> 10 LET A=100
> 20 LET B=A*(A MOD 99)
> 30 LET C=B+1
> 40 LET D=B LSR(2)
> 50 END
> 60 RUN
> 60 DUMP V
```

**LET EXAMPLE:**

```
A = 100  
B = 100  
C = 101  
D = 25  
E = 0  
F = 0  
G = 0  
H = 0  
I = 0  
J = 0  
K = 0  
L = 0  
M = 0  
N = 0  
O = 0  
P = 0  
Q = 0  
R = 0  
S = 0  
T = 0  
U = 0  
V = 0  
W = 0  
X = 0  
Y = 0  
Z = 0
```

> 60

The above example shows the use of the Let command as an assignment operator.

# LIST

Display contents on console.

# LIST

**FORMAL NAME:** LIST

**SYNTAX:**

```
> LIST var[,base]
```

**PARAMETERS:**

var

A letter (A to Z).

base

Base (8 or 10) the output value will appear in. Default = 10.

**OPERATION:**

List will list on the console the contents of a designated variable.

**EXAMPLE:**

```
> 10 RAND I
> 20 LIST I
> 30 LIST I,8
> 40 END
> 50 RUN
20275
%47463
> 50
```

The above example generates a random value in the variable I. The List command is used to display the contents of the variable.

# LTIO

# LOAD TIO REGISTER

Loads TIO Register.

**FORMAL NAME:** LOAD TIO REGISTER

**SYNTAX:**

```
> LTIO lun,word
```

**PARAMETERS:**

lun

Logical unit number.

word

Word to be loaded into the TIO register. May be a constant or a variable.

**OPERATION:**

Load TIO Register will load bits 3-15 of the value in the WORD parameter into the TIO register of the 7905/7920/7925 interface.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 LTIO 0,%525
> 20 TIO 0
> 30 STAT I
> 40 END
> 50 RUN
D3 last TIO status is 1 000 000 101 010 101
> 50
```

The above example loads the TIO register on the 7905/7920/7925 interface. The TIO status is then obtained and displayed.

# LOOP

Unconditional branch.

# LOOP

**FORMAL NAME:** LOOP

**SYNTAX:**

```
> LOOP stepn,count
```

**PARAMETERS:**

stepn

Unconditional branch to step "n".

count

Number of times (decimal) execution will loop. May be a constant or a variable.

**OPERATION:**

Loop will perform an unconditional branch to the step number specified until this command has been executed the number of times specified in the *count* parameter.

**EXAMPLE:**

```
> 10 DEV 0,5,14,20,0  
> 10 DB AA,128,123  
> 10 RS 0  
> 20 WDI 0,AA  
> 30 LOOP 10,1000  
> 40 END  
> 50 RUN  
> 50
```

The above example uses the Loop command to execute a sequence of commands 1001 times.

## CAUTION

If the program is aborted by use of control A or by setting front panel switch 13, or if control transfers out of range of the Loop command before it is fully executed and the program is subsequently restarted with the RUN command, the Loop command will execute only the number of loops remaining at the time the command was last executed.

# MAKT

# MAKE TEST TAPE

Makes a test tape.

**FORMAL NAME:** MAKE TEST TAPE

**SYNTAX:**

```
> MAKT { A }  
          { N }
```

**PARAMETERS:**

<u>A</u>	Appends a tape onto an existing test tape.
<u>N</u>	New tape.

**OPERATION:** Make Test Tape will make a test tape. A test tape is a SLEUTH made tape consisting of user programs. The N parameter indicates this is a new tape. The A parameter will append a test onto an existing test tape. The current program will only be dumped if an END statement exists in the program. The tape unit must be mounted on DRT 6 unit 0. Only SLEUTH can interpret this tape. (See BA command.)

**EXAMPLE:**

```
> 10 DEV 0,5,13,50,0  
> 10 DB AA,128,%177777  
> 10 RS 0  
> 20 WDI 0,AA  
> 30 LOOP 10,10  
> 40 END  
> 50 MAKT N  
> 50
```

The above example will make a test tape of the above program.

## CAUTION

A test tape can only be interpreted by the same release of SLEUTH that created the tape. If a different version-update-fix of SLEUTH is anticipated, the user is advised to store his program in an MPE Editor file and then enter it with the BA E form of the BATCH command.

# MASTER CLEAR

Master Clears device.

# MC

**FORMAL NAME:** MASTER CLEAR

**SYNTAX:**

```
> MC lun
```

**PARAMETER:** lun Logical unit number.

**OPERATION:** Master Clear will Master Clear the *lun* specified.

**EXAMPLE:**

```
> 10 DEV 0,6,24,100,0  
> 10 MC 0  
> 20 END  
> 30 RUN  
> 30
```

The above example will send a Master Clear to the device.



# NAME

# NAME SIO PROGRAM

Names a program.

**FORMAL NAME:** NAME SIO PROGRAM

**SYNTAX:**

```
> NAME sioprogram
```

**PARAMETER:** sioprogram Two character name of an SIO program.

**OPERATION:** Name SIO Program will allocate space for an SIO program. Immediately following the Name command, only SIO orders will be acceptable until an ENDS order is entered.

**EXAMPLE:**

```
> 10 DEV 0,5,23,10,0
> 10 DB AA,100,S
> THIS IS A STRING BUFFER
> 10 NAME JJ
> 10 CONT %456,%123
> 10 SENS
> 10 INT
> 10 READ AA,C
> 10 READ AA
> 10 WRIT AA,C
> 10 WRIT AA
> 10 JUMP *-7
> 10 ENDS
> 10 DUMP S,JJ
35502: 40456 123 50000 177777 20000 17777 177634 36307
35512: 77634 36307 167634 36307 67634 36307 0 35511
35522: 30000 177777
> 10
```

The above example indicates how an SIO program may be defined using the Name SIO Program command. The example shows the various combinations of SIO order pairs.

# NEXT

Terminates a For loop.

# NEXT

**FORMAL NAME:** NEXT

**SYNTAX:**

```
> NEXT var
```

**PARAMETER:** var A letter (A to Z).

**OPERATION:** Next terminates a For loop. Each execution of the Next command increments the indicated variable by one. If the variable has not exceeded its maximum count control is transferred to the command immediately following the For command; otherwise, control is transferred to the command following NEXT. (See For command.)

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0  
> 10 FOR Z=0 TO 410  
> 20 SEEK 0,Z,0,0  
> 30 NEXT Z  
> 60 END  
> 50 RUN  
> 50
```

The above example used the Next command to terminate a For loop

# NOPR

# NO PRINT

Suppresses messages.

**FORMAL NAME:** NO PRINT

**SYNTAX:**

```
> NOPR
```

**OPERATION:** The No Print command will cause the suppression of all messages except User-SLEUTH dialogue. No Print also disables halt on error.

**EXAMPLE:**

```
> 10 DEV 0,14,14,5,0  
> 10 NOPR  
> 20 RC 0  
> 30 RC 0  
> 40 PR  
> 50 END  
> 60 RUN  
> 60
```

The above example uses the No Print command to turn off all messages except user dialogue.

# PACK CERTIFICATION

# PCT

Performs a pack certification test on disc.

**FORMAL NAME:** PACK CERTIFICATION

**SYNTAX:**

```
> PCT lun,pattern
```

**PARAMETERS:**

lun

Logical unit number. Must be an ISS disc.

pattern

Octal test pattern. May be a constant or a variable.

**OPERATION:**

Pack Certification will execute a PCT on the ISS disc indicated. One track, wherever the heads are positioned, will be certified using the octal pattern.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,7,14,30,0  
> 10 IT 0  
> 20 PCT 0,%125252  
> 30 LOOP 10,8119  
> 40 END  
> 50 RUN  
> 50
```

The above program performs a pack certification test on the entire disc pack using the octal pattern 125252.

# PE

# PAUSE ON ERROR

Pause on error.

**FORMAL NAME:** PAUSE ON ERROR

**SYNTAX:**

```
> PE
```

**OPERATION:**

Pause On Error will execute a halt %16 if an error is encountered. It will not be enacted if no print is in effect.

**EXAMPLE:**

```
> 10 DEV 0,5,13,5,0  
> 10 PE  
> 20 RS 0  
> 30 LOOP 20,9999  
> 40 END  
> 50 RUN  
> 50
```

The above program sets the Pause On Error option using the PE command.

# POLL

Resumes polling.

# POLL

**FORMAL NAME:** POLL

**SYNTAX:**

```
> POLL lun
```

**PARAMETER:** lun Logical unit number. Must be a 7905/7920/7925 disc.

**OPERATION:** Poll causes the 13037 disc controller to resume polling.

**DELAY TIME:** The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 POLL 0
> 20 END
> 30 RUN
> 30
```

The above example issues the Poll command to the 13037 disc controller.

# PR

# PRINT

Resume message printing.

**FORMAL NAME:** PRINT

**SYNTAX:**

```
> PR
```

**OPERATION:**

Print will cause the resumption of all message output that was suppressed by the NOPR command. Pause on error (if selected) is re-enabled.

**EXAMPLE:**

```
> 10 DEV 0,5,14,100,0  
> 10 NOPR  
> 20 RC 0  
> 30 RC 0  
> 40 PR  
> 50
```

The above example turns off the Print at step 10, performs two recalibrated operations on an ISS disc, then turns on the Print at step 40 using the PR command.

# PROCEED

# PROC

FORMAL NAME: PROCEED

SYNTAX:

```
> PROC |N|  
          |D|
```

PARAMETERS: [N] Cancels proceed mode.  
              [D] Causes direct I/O interference to each device executing in proceed mode.

OPERATION: Proceed will cause SLEUTH to go into the asynchronous mode of operation. PROC N should be placed before the last command to be executed in proceed mode.

EXAMPLES:

```
> 10 DEV 0,8,14,25,2  
> 10 DEV 1,6,19,25,0  
> 10 DB AA,8000,R  
> 10 NOPR  
> 20 RC 0  
> 30 RC 0  
> 40 PR  
> 50 PROC  
> 60 SEEK 0  
> 70 WD 1,AA  
> 80 WDI 0,AA  
> 90 LOOP 70,100  
> 100 REW 1  
> 110 END  
> 120 RUN  
> 120
```

The above example writes an 8000 word buffer of random data on mag tape and an ISS disc. The proceed mode is used to overlap the writes.

```
> 10 DEV 0,7,14,20,0  
> 10 DEV 1,8,14,20,0  
> 10 DB AA,8000,R  
> 10 NOPR  
> 20 RC 0  
> 30 RC 1  
> 40 RC 0  
> 50 RC 1  
> 60 PR  
> 70 PROC  
> 80 SEEK 0  
> 90 SEEK 1  
> 100 WDI 0,AA  
> 110 WDI 1,AA  
> 120 LOOP 100,100  
> 130 END  
> 140 RUN  
> 140
```

The above example writes an 8000 word buffer of random data on two ISS discs. The proceed mode is used to overlap the writes.



# PUT

# PUT

Prints message.

**FORMAL NAME:** PUT

**SYNTAX:**

```
> PUT "string"
```

**PARAMETER:** "string" 0-72 characters enclosed by quotation marks.

**OPERATION:** Put will print a message on the console. The message will be contained within quotes.

**EXAMPLE:**

```
> 10 PUT "THIS IS AN EXAMPLE"
> 20 END
> 30 RUN
THIS IS AN EXAMPLE
> 30
```

The above example utilizes the Put command to output a string on the console.

# RANDOMIZE

Generates a random number.

# RAND

FORMAL NAME: RANDOMIZE

SYNTAX:

```
> RAND var
```

PARAMETER: var A letter (A to Z).

OPERATION: Randomize generates a positive random number and places it in the variable designated.

## CAUTION

SLEUTH does not produce true random data. Bit 15 is always high.

EXAMPLE:

```
> 10 RAND A
> 20 RAND B
> 30 RAND C
> 40 END
> 50 RUN
> 50 DUMP V
    A = 19139
    B = 7209
    C = 16117
    D = 0
    E = 0
    F = 0
    G = 0
    H = 0
    I = 0
    J = 0
    K = 0
    L = 0
    M = 0
    N = 0
    O = 0
    P = 0
    Q = 0
    R = 0
    S = 0
    T = 0
    U = 0
    V = 0
    W = 0
    X = 0
    Y = 0
    Z = 0
> 50
```

The above example Randomizes 3 variables.

# RS

# RANDOM SEEK

Issues a random seek command to the disc.

**FORMAL NAME:** RANDOM SEEK

**SYNTAX:**

```
> RS lun
```

**PARAMETER:**

lun

Logical unit number. Must be a moving head disc.

**OPERATION:**

Random Seek will issue a Random Seek command to the *lun* designated. This instruction updates the internal disc address.

**DELAY TIME:**

The default Interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,7,14,10,0  
> 10 RS 0  
> 20 LOOP 10,5000  
> 30 END  
> 40 RUN  
> 40
```

The above example executes 5001 Random Seek operations on an ISS disc.

# READ

Places a read order in an SIO chain.

# READ

**FORMAL NAME:** READ

**SYNTAX:**

```
> READ buf[,C]
```

**PARAMETERS:**

buf

Buffer into which data is read. Buffer length (4k max.) determines word count of the read.

[C]

Specifies a chain read.

**OPERATION:**

Read will place a read order in an SIO chain.

**EXAMPLE:**

```
> 10 DEV 0,8,14,5,0
> 10 DB AA,128,%12345
> 10 DB XX,128,0
> 10 NAME RD
> 10 CONT 0,%50000
> 10 READ XX
> 10 ENDS,I
> 10 SEEK 0
> 20 WDI 0,AA
> 30 SIO 0,RD,1,3,%170000,0
> 40 CB 0,AA,XX,5
> 50 END
> 60 RUN
> 60
```

The above example uses the Read SIO order to read 1 sector from an ISS disc.

# RA

# READ ADDRESS

Performs a read address.

**FORMAL NAME:** READ ADDRESS

**SYNTAX:**

```
> RA [lun | cylinder,head,sector]
```

**PARAMETERS:**

lun Logical unit number. Must be an ISS disc.

cylinder }  
head } Disc parameters.  
sector }

**OPERATION:**

Read Address performs a Read Address operation on the *lun* indicated. The read will take place on the cylinder and head specified. The heads are assumed to be over the correct cylinder. Default *cylinder*, *head*, and *sector* is 0,0,0.

## NOTE

Only the cylinder and head address are valid on an RA.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> DEV 0,7,14,3,1  
> 10 SEEK 0,124,3,3  
> 20 RA 0,124,3,3  
> 30 CA 0  
> 40 END  
> 50 RUN  
> 50
```

The above example reads a singular address from cylinder 124, head 3. Then uses the CA command to check the cylinder and head of the Address Read.

# READ ADDRESS IMMEDIATE

# RAI

Performs a Read Address Immediate.

**FORMAL NAME:** READ ADDRESS IMMEDIATE

**SYNTAX:**

```
> RAI lun
```

**PARAMETER:**

lun

Logical unit number. Must be an ISS disc.

**OPERATION:**

Read Address Immediate performs a Read Address on an ISS disc. The Internal Disc Address will be used as the address where the operation will take place.

**DELAY TIME:**

The default Interrupt Time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,8,14,5,0
> 10 RS 0
> 20 RAI 0-
> 30 CA 0
> 40 LOOP 10,500
> 50 END
> 60 RUN
> 60
```

The above example randomly seeks and reads addresses. The cylinder and head portion of each address is then checked.

# RCLK

# READ CLOCK

Reads elapsed time.

FORMAL NAME: READ CLOCK

SYNTAX:

```
> RCLK var
```

PARAMETER: var A letter (A to Z).

OPERATION: Read Clock reads the process clock in the CPU into the variable designated.

EXAMPLE:

```
> 10 DEV 0,5,13,100,0
> 10 SCLK 0
> 20 RS 0
> 30 LOOP 20,100
> 40 RCLK A
> 50 LET A=A+A/1000
> 60 PUT "ELAPSED TIME IN MILLISECONDS IS"
> 70 LIST A
> 80 END
> 90 RUN
ELAPSED TIME IN MILLISECONDS IS
2668
> 90
```

The above example loads the process clock with 0, executes 101 random seeks then reads the process clock. Since the process clock counts at a rate of 1.001 milliseconds the value in the variable A is modified to reflect the true elapsed time in milliseconds.

# READ DATA

Reads devices.

# RD

**FORMAL NAME:** READ DATA

**SYNTAX:**

	Format
> <u>RD <i>lun,buf[,cylinder,head,sector]</i></u>	1
> <u>RD <i>lun,buf[,mask[,cylinder,head,sector]]</i></u>	2
> <u>RD <i>lun,buf[,track,arc]</i></u>	3
> <u>RD <i>lun,buf,mode</i></u>	4
> <u>RD <i>lun,buf</i></u>	5
> <u>RD <i>lun,buf,[mode[,hopper[,stacker]]]</i></u>	6

**OPERATION:**

Read Data will perform a read operation on the *lun* indicated. Due to the variance of readable devices, six discrete formats are required.

**Format 1:** Will be utilized with ISS and 7900 discs. The heads are assumed to be positioned at the correct address. The *cylinder, head, sector* parameters (default 0,0,0) are used to update the internal disc address.

**Format 2:** Will be used on the 7905/7920/7925 disc. The file mask on the controller will be set with the value in the *mask* parameter. Default is cylinder mode, no retries. The file mask may be a constant or a variable. See Set File Mask (SFM) command for the mask format. The *cylinder, head, sector* parameters will designate the starting point of the read. Default is 0,0,0. This command updates the internal disc address.

**Format 3:** Will be used with fixed head discs. The *track, arc* parameters will indicate the starting point of the read. Default is 0,0.

**Format 4:** Will be utilized with Card Readers, SEL CHAN MAINT Cards and Paper Tape Readers. The *mode* parameter will be one of the following:

- Paper Tape
  - W - Word Transfer Mode
  - B - Byte Transfer Mode
- Card Reader
  - A - ASCII
  - P - Packed Binary
  - C - Column Binary
- SEL CHAN MAINT Card
  - F - Fast Request Mode
  - S - Slow Request Mode

**Format 5:** Will be used with magnetic tape and devices attached to the Asynchronous Multiplexer.

If reading from devices attached to the Asynchronous Multiplexer a carriage return may terminate the read, otherwise the buffer's length determines the word count of the read.



**Format 6:** Will be used on the Reader/Interpreter/Punch. The *mode* parameter can be one of the following:

- C - Column Binary
- H - Hollerith

Default is Hollerith to ASCII conversion. The *hopper/stacker* parameters may be the following:

- P - Primary *hopper/stacker*
- S - Secondary *hopper/stacker*

Default *hopper/stacker* parameters are secondary.

**DELAY TIMES:**

The following are default interrupt times out:

<b>Device</b>	<b>Time Out</b>
Disc	300 Milliseconds
Magnetic Tape	5 Seconds
Asynchronous Multiplexer	50 Seconds/Per Character
Card Reader	3 Seconds
Paper Tape	30 Seconds
Reader/Punch	7 Seconds

# READ DATA

Continued

# RD

## EXAMPLES:

```
> 10 DEV 0,22,10,100,0
> 10 DB AA,40,0
> 10 DB BB,40,0
> 10 CHB AA,%30060
> 20 FOR I=0 TO 9
> 30 RD 0,BB
> 40 CB 0,AA,BB,40
> 50 CHB AA,I
> 60 FOR J=0 TO 7
> 70 CHB AA,S
> 80 NEXT J
> 90 CHB AA,I
> 100 NEXT I
> 110 END
> 120 RUN
> 120
```

The above example reads ten cards consisting of Hollerith data from a Reader/Punch.

```
> 10 DEV 07,14,30,0
> 10 DB AA,2944,R
> 10 DB BB,2944,0
> 20 WD 0,AA,150,0,0
> 30 RD 0,BB,150,0,0
> 40 CB 0,AA,BB,5
> 50 END
> 60 RUN
> 60
```

The above is an example of Read Data on an ISS disc. The program writes one track of random data at cylinder 150, head 0 then reads and checks the data.

```
> 10 DEV 0,6,18,3,0
> 10 DB AA,4000,R
> 10 DB BB,4000,0
> 10 WD 0,AA
> 20 REW 0
> 30 RD 0,BB
> 40 CB 0,AA,BB,3
> 50 REW 0
> 60 END
> 70 RUN
> 70
```

The above example indicates how a Read Data operation may be performed on magnetic tape. The program writes one 4000 word record on magnetic tape then reads and checks the data.

## EXAMPLES:

```

> 10 DEV 0,10,6,10,1,2400
> 10 DB AA,36,0
> 10 RD 0,AA,72
> 20 END
> 30 RUN
> 30 DUMP B,AA
36307: 52110 44523 20111 51440 40516 20105 54101 45520
36317: 46105 20117 43040 51104 20117 47040 52110 42440
36327: 40523 54516 41440 46525 54000 0 0 0
36347: 0 0 0 0 0 0 0 0
36347: 0 0 0 0 0
> 30

```

The above example reads characters which were typed in from a device on port 1 of the asynchronous multiplexor.

```

> 10 DEV 0,7,2,10,0
> 10 DB AA,60,0
> 10 RD 0,AA,P
> 20 END
> 30 RUN
> 30

```

The above example reads a packed binary card from a card reader into buffer AA.

```

> 10 DEV 0,7,21,15,0
> 10 DEV 1,8,20,15,0
> 10 DB AA,300,R
> 10 DB BB,300,0
> 10 WD 0,AA,B
> 20 END
> 30 RUN
> 30 10 RD 1,BB
> 15 CB 1,AA,BB,5
> 30 AUTO 30
> 30 RUN
> 30

```

The above example punches a 300 word buffer on paper tape, then the tape is placed on a tape reader, read, and the data is checked.

```

> 10 DEV 0,5,17,10,0
> 10 DB AA,128,0
> 10 RD 0,AA,5,3
> 20 END
> 30 RUN
> 30

```

The above program reads 128 words from a fixed head disc.

# READ DATA IMMEDIATE

# RDI

Reads device.

**FORMAL NAME:** READ DATA IMMEDIATE

**SYNTAX:**

```
> RDI lun,buf
> RDI lun,buf [,mask]
```

Format
1
2

**PARAMETERS:**

lun Logical unit number. Must be a moving head disc.

buf Buffer into which data is read. Length determines word count of read.

mask Loads file mask on controller. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

**OPERATION:**

Read Data Immediate will perform a read operation on a moving head disc. For a 7900 or ISS disc, the heads are assumed to be positioned at the correct address. For a 7905/7920/7925, the internal disc address will designate the starting point of the read. Format 1 will be used with 7900 and ISS discs. Format 2 will be used with the 7905/7920/7925 disc. The file mask on the controller will be loaded with the value specified in the *mask* parameter. (See Set File Mask (SFM) command for mask format.) Default mask is cylinder mode, no retries. *Mask* may be a constant or a variable.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,128,0
> 10 DB BB,128,R
> 10 RS 0
> 20 WDI 0,BB,7
> 30 RDI 0,AA,7
> 40 CB 0,BB,AA,3
> 50 LOOP 10,100
> 60 END
> 70 RUN
> 70
```

The above example randomly seeks writes and reads data on a 7905 disc. Sparing is enabled.

# RFS

# READ FULL SECTOR

Reads a full sector of a moving head disc.

**FORMAL NAME:** READ FULL SECTOR

**SYNTAX:**

```
> RFS lun,buf[,cylinder,head,sector]
```

**PARAMETERS:**

<u>lun</u>		Logical unit number. Must be a moving head disc.
<u>buf</u>		Buffer into which data is read. Buffer length determines word count of read.
<u>cylinder</u>	}	Disc parameters.
<u>head</u>		
<u>sector</u>		

**OPERATION:**

Read Full Sector executes a full sector read operation of a moving head disc. The heads are assumed to be positioned at the correct address. The *cylinder, head, sector* (default 0,0,0) parameters are used to update the internal disc address.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,25,0
> 10 DB AA, 131,%44444
> 10 DB BB, 131, 0
> 10 SEEK 0,150,0,0
> 20 WFS 0,AA,150,0,0
> 30 RFS 0,BB,150,0,0
> 40 CB 0,AA,BB,3
> 50 END
> 60 RUN
> 60
```

The above example uses Read Full Sector to read data from a 7900 disc.

# READ FULL SECTOR IMMEDIATE

# RFSI

Full sector read.

**FORMAL NAME:** READ FULL SECTOR IMMEDIATE

**SYNTAX:**

```
RFSI lun,buf
```

**PARAMETER:**

lun

Logical unit number. Must be a moving head disc.

buf

Buffer into which data is read. Buffer length determines word count of read.

**OPERATION:**

Read Full Sector Immediate performs a full sector read operation on a moving head disc. The heads are assumed to be positioned at the correct address.

The above example issues random seeks to a 7900 disc, writes and reads full sectors,

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,20,0  
> 10 DB AA,131,%12345  
> 10 DB BB,131,0  
> 10 RS 0  
> 20 WFSI 0,AA  
> 30 ZBUF BB  
> 40 RFSI 0,BB  
> 50 CB 0,AA,BB,5  
> 60 LOOP 10,200  
> 70 END  
> 80 RUN  
> 80
```

The above example issues random seeks to a 7900 disc, writes and reads full sectors, then compares the data.

# RMSK

# READ MASK

Read the mask register.

FORMAL NAME: READ MASK

SYNTAX:

```
> RMSK buf
```

PARAMETER:

*buf*

Buffer into which the data is read.

OPERATION:

RMSK will read the interrupt mask register into the buffer specified.

EXAMPLE:

```
> 10 DB AA,1,0
> 10 RMSK AA
> 20 END
> 30 RUN
> 30 DUMP B,AA
30600: 400
> 30
```

The above example reads the interrupt mask register into buffer AA and then displays the contents of the buffer.

# READ NEXT FULL SECTOR

# RNFS

Reads the next full sector.

**FORMAL NAME:** READ NEXT FULL SECTOR

**SYNTAX:**

```
> RNFS lun,buf [,cylinder,head,sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7900 disc.

buf

Buffer into which the data is read. Buffer length determines word count of the read.

cylinder

head

sector

} Disc parameters.

**OPERATION:**

Read Next Full Sector performs a Read Next Full Sector Operation on a 7900 disc. The heads are assumed to be positioned at the correct address. The *cylinder, head, sector* (0,0,0) parameters (default 0,0,0) are used to update the internal disc address.

**DELAY TIME:**

The default Interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,10,0
> 10 DB AA,3144,%125252
> 10 DB BB,128,0
> 10 SEEK 0
> 20 WFSI 0,AA
> 30 RNFS 0,BB
> 40 CB 0,AA,BB,3
> 50 END
> 60 RUN
> 60
```

The above example uses Read Next Full Sector to read the next available full sector from cylinder 0 head 0.



# RNFI READ NEXT FULL SECTOR IMMEDIATE

Reads the next full sector of a disc.

**FORMAL NAME:** READ NEXT FULL SECTOR IMMEDIATE

**SYNTAX:**

```
> RNFI lun, buf
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7900 disc.

buf

Buffer into which data is read.

**OPERATION:**

Read Next Full Sector Immediate executes a read next full sector operation on the 7900 disc. The buffer's length determines the word count of the read. The heads are assumed to be positioned at the correct address.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,13,10,0  
> 10 DB AA,3144,%12345  
> 10 DB BB,131,0  
> 10 SEEK 0,200,0,0  
> 20 WFSI 0,AA  
> 30 RNFI 0,BB  
> 40 CB 0,AA,BB,3  
> 50 END  
> 60 RUN  
> 60
```

The above example uses the Read Next Full Sector Immediate command to read one full sector of data from a disc pack. The sector read will be whichever sector the head happens to be over when the command is executed.

# READ RECORD WITH CRCC

# RDC

FORMAL NAME: READ RECORD WITH CRCC

SYNTAX: > RDC lun,buf

PARAMETERS: lun Logical unit number. Must be an 800 CPI magnetic tape unit.  
buf Buffer into which data is read.

OPERATION: Read Record with CRCC will issue a RDC SIO Program to the 800 CPI magnetic tape unit. The buffer length determines the length of the record read and should be large enough to hold the CRCC.

DELAY TIME: The default interrupt time out is 5 seconds.

EXAMPLE:

```
> 10 DEV 0,6,18,10,3
> 10 DB AA,10,%52525
< 10 DB BB,11,0
> 10 WD 0,AA
> 20 REW 0
> 30 RDC 0,BB
> 40 END
> 50 RUN
> 50 DUMP B,BB
36321: 52525 52525 52525 52525 52525 52525 52525 52525
36331: 52525 52525      4
> 50
```

The above example writes a 10 word buffer of the octal pattern 52525 then reads the record with CRC. The buffer dump shows the CRC character as the last word of the buffer.

# RWO

# READ WITH OFFSET

**FORMAL NAME:** READ WITH OFFSET

**SYNTAX:**

```
> RWO lun,buf,mask,offset[,cylinder,head,sector]
```

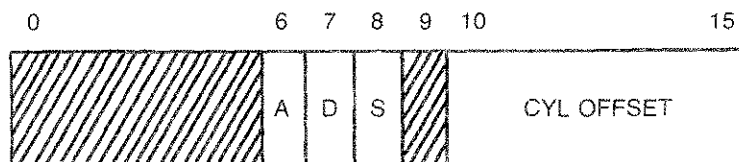
**PARAMETERS:**

- lun* Logical unit number. Must be a 7905/7920/7925 disc.
- buf* Buffer into which data is read. Buffer length determines word count of read.
- mask* Loads file mask on 13037 disc controller. May be a constant or a variable. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

*offset* Contains cylinder offset and the separator clock information. May be a constant or a variable.

OFFSET WORD FORMAT



A - Advance (+10 nsec)  
 D - Delay (-10 nsec)  
 S - Sign Bit (OFFSET)  
 CYL OFFSET - Cylinder Offset

} CLOCK OFFSET  
 } HEAD OFFSET

*cylinder*  
*head*  
*sector*

} Disc parameters; determine starting point of the read. Default is 0,0,0.

**OPERATION:**

Read With Offset executes like a normal read except an offset word is transmitted to the drive before executing. The clock offset advances or delays the separator read clock 10 nanoseconds with respect to the data transition. The head offset moves the heads off of track center by an amount proportional to the head offset.

**NOTE**

A head offset is cancelled whenever the operation completes (with or without error) or whenever the drive executes a seek.

The *cylinder, head, sector* parameters (default 0,0,0) determine the starting point of the read. This command updates the internal disc address.

**DELAY TIME:**

The default interrupt time out is 500 milliseconds.

# READ WITH OFFSET IMMEDIATE

# RWOI

**FORMAL NAME:** READ WITH OFFSET IMMEDIATE

**SYNTAX:** `> RWOI lun,buf,mask,offset`

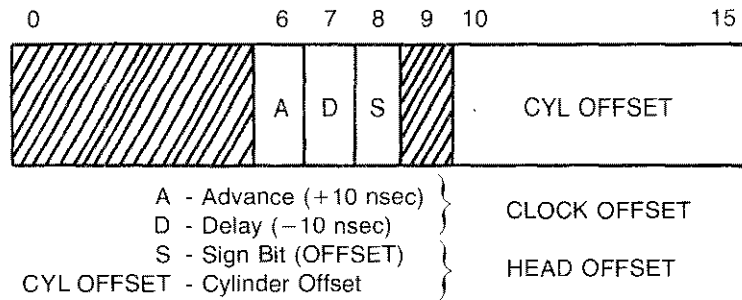
**PARAMETERS:**

- lun* Logical unit number. Must be a 7905/7920/7925 disc.
- buf* Buffer into which data is read. Buffer length determines word count of read.
- mask* Loads file mask on 13037 disc controller. May be a constant or a variable. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

*offset* Contains cylinder offset and the separator clock information. May be a constant or a variable.

OFFSET WORD FORMAT



**OPERATION:** Read With Offset Immediate executes like a normal read except an offset word is sent to the disc before executing. The clock offset advances or delays the separator read clock 10 nanoseconds with respect to the data transition. The head offset moves the heads off of track center by an amount proportional to the head offset.

## NOTE

A head offset is cancelled whenever the operation completes (with or without error) or whenever the drive executes a seek.

The heads are assumed to be positioned at the correct address.

**DELAY TIME:** The default interrupt time out is 500 milliseconds.

# RWV

# READ WITHOUT VERIFY

FORMAL NAME: READ WITHOUT VERIFY

SYNTAX:

```
> RWV lun,buf [,mask] ,cylinder,head,sector||
```

PARAMETERS:

- lun Logical unit number. Must be a 7905/7920/7925 disc.
- buf Buffer into which data is read. Buffer length determines word count of read.
- mask Loads file mask on 13037 disc controller. May be a constant or a variable. Default is cylinder mode, no retries. Mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

cylinder  
head  
sector } Disc parameters.

OPERATION:

Read Without Verify operates like a normal read operation but does not verify the preceding sector. The heads are assumed to be positioned at the correct address. No address checking or sparing operations occur unless a track boundary is crossed during the operation. The *cylinder*, *head*, *sector* parameters (default 0,0,0) are used to update the internal disc address.

DELAY TIME:

The default interrupt time out is 500 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 DB AA,6144,0  
> 10 FOR C=0, TO 410  
> 20 SEEK 0,C,0,0  
> 30 RWV 0,AA,2,C,0,0  
> 40 NEXT C  
> 50 END  
> 60 RUN  
> 60
```

The above example uses the Read Without Verify command to read one full surface from a 7905 disc.

# READ WITHOUT VERIFY IMMEDIATE RWVI

FORMAL NAME: READ WITHOUT VERIFY IMMEDIATE

SYNTAX:

```
> RWVI lun,buf [,mask]
```

PARAMETERS:

lun Logical unit number. Must be a 7905/7920/7925 disc.

buf Buffer into which data is read. Buffer length determines word count of read.

mask Loads file mask on 13037 disc controller. May be a constant or a variable. Default is cylinder mode, no retries. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

OPERATION:

Read Without Verify Immediate operates like a normal read operation but does not verify the preceding sector. No address checking or sparing operations occur unless a track boundary is crossed during the operation. The heads are assumed to be positioned at the correct address.

DELAY TIME:

The default interrupt time out is 500 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 DB AA,128,0  
> 10 RS 0  
> 20 RWVI 0,AA  
> 30 LOOP 10,100  
> 40 END  
> 50 RUN  
> 50
```

The above example randomly seeks and uses the Read Without Verify Immediate command to read one sector of information.

# RC

# RECALIBRATE

FORMAL NAME: RECALIBRATE

SYNTAX:

```
> RC lun
```

PARAMETER: lun Logical unit number. Must be a moving head disc.

OPERATION: Recalibrate performs a Recalibrate Operation on a moving head disc.

DELAY TIME: The default interrupt time out is 5 seconds.

EXAMPLE:

```
> 10 DEV 3,7,14,5,0  
> 10 SEEK 3,405,0,0  
> 20 RC 3  
> 30 LOOP 10,100  
> 40 END  
> 50 RUN  
> 50
```

The above example seek an ISS disc to cylinder 405 then recalibrates. The program is looped 100 times.

# RENUMBER

# REN

FORMAL NAME: RENUMBER

SYNTAX:

```
> REN
```

OPERATION:

Renumber will renumber the commands in the program area. The numbering increment will range between 10 and 2 depending on the size of the program. If a program is being entered and command numbers extend beyond 999, SLEUTH prints a message and automatically rennumbers the program. Renumber is also useful if many program edits are made.

EXAMPLE:

```
> 10 NOPR
> 20 PE
> 30 PR
> 40 5 SS
> 35 ES
> 40 AUTO 40
> 40 DUMP P
   5 SS
   10 NOPR
   20 PE
   30 PR
   35 ES
> 40 REN
> 60 DUMP P
   10 SS
   20 NOPR
   30 PE
   40 PR
   50 ES
> 60
```

The above example demonstrates how program renumbering may be accomplished. Edits were made to the program then the program was renumbered using the Renumber command.



# RDA

# REQUEST DISC ADDRESS

**FORMAL NAME:** REQUEST DISC ADDRESS

**SYNTAX:**

```
> RDA lun
```

**PARAMETER:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

**OPERATION:**

Request Disc Address will return a two word address that can be displayed using the DISP command. The RDA command may be used to determine where an error occurred during a verify or any other command which terminates with an error.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0
> 10 RS 0
> 20 RDA 0
> 30 DISP 0,D
> 40 END
> 50 RUN
```

```
D11 ADDRESS READ IS 66, 3
CYLINDER = 54, SECTOR = 3, HEAD = 0
> 50
```

The above example utilizes the Request Disc Address command to obtain the last address from the 13037 disc controller.

# REQUEST SECTOR ADDRESS

# RSA

**FORMAL NAME:** REQUEST SECTOR ADDRESS

**SYNTAX:**

```
> RSA lun
```

**PARAMETER:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

**OPERATION:**

Request Sector Address returns the logical sector address of the sector currently under the heads. This address may be displayed using the DISP command.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0  
> 10 RS 0  
> 20 RSA 0  
> 30 DISP 0,S  
> 40 END  
> 50 RUN
```

```
D11 SECTOR ADDRESS IS 20  
> 50
```

The above example uses Request Sector Address to obtain the current sector address.

# RQST

# REQUEST STATUS

FORMAL NAME: REQUEST STATUS

SYNTAX:

```
> RQST lun
```

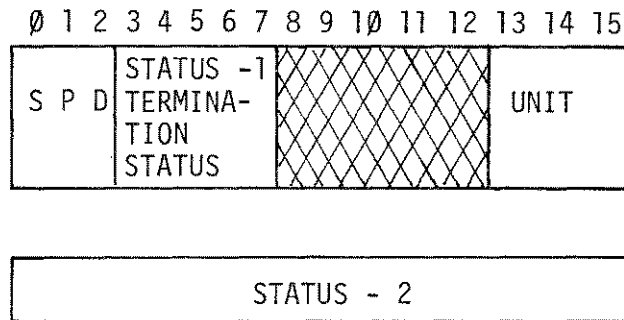
PARAMETER:

lun

Logical unit number. Must be a 7905/7920/7925 disc.

OPERATION:

Request Status will return two words of status from the 13037 disc controller. This status may be displayed using the DISP command. The format of the status words are as follows:



## STATUS - 1

The S,P, and D bits reflect the status of the track most recently accessed:

S = Spare  
P = Protected  
D = Defective

## EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 SEEK 0,10,0,0  
> 20 RQST 0  
> 30 DISP 0,R  
> 40 END  
> 50 RUN
```

```
D10 requested status is  
0 001 111 100 000 000  
0 000 010 000 100 000  
> 50
```

The above example uses the Request Status command to return two words of disc status.

## Encoded Termination Status (octal)

00 Normal completion  
01 Illegal opcode  
02 Wakeup  
07 Cylinder compare error  
10 Uncorrectable data error  
11 Head-sector compare error  
12 I/O program error

# REQUEST STATUS

# RQST

Continued

- 14 End of cylinder
- 16 Overrun
- 17 Possibly correctable data error
- 20 Illegal access to spare track
- 21 Defective track
- 22 Access not ready during data operation
- 23 Status - 2 error
- 26 Attempt to write on protected track
- 27 Unit unavailable
- 37 Drive attention

Status - 2 Format	Status - 2 Format
15 * Drive busy	9 Protected
14 * Drive not ready	8 Attention
13 * Seek check	7 0
12 First status	3-6 Drive Type
11 * Fault	0 Status - 2 error (true if any bit marked * is true)
10 Format	

Drive Type	Model
1	7920
2	7905
3	7925

# RSYN

# REQUEST SYNDROME

**FORMAL NAME:** REQUEST SYNDROME

**SYNTAX:**

```
> RSYN lun
```

**PARAMETER:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

**OPERATION:**

Request Syndrome obtains a 7 word syndrome from the 13037 disc controller. A Request Syndrome operation may be issued after any Read operation which terminates with a possibly correctable data error. The 7 words of information will be read into an internal buffer which may be displayed with a DISP command. The format of the syndrome returned is as follows:

	Word
status	1
cylinder	2
head   sector	3
displacement	4
PATT 1	5
PATT 2	6
PATT 3	7

# REQUEST SYNDROME

Continued

# RSYN

## EXAMPLE:

```
> 10 DEV 0,5,15,100,0
> 10 DEV AA,3072,%155555
> 10 DB BB,3072,0
> 10 IT 0
> 20 WDI 0,AA
> 30 RDI 0,BB
> 40 GO 10,%107400,0
> 50 RSYN 0
> 60 DISP 0,Y
> 70 END
> 80 RUN
```

E2 RDI failed in step 30

Status is 1 000 111 100 000 000

Should be 1 000 000 000 000 000

Cylinder = 379, Sector = 0, Head = 1

D5 last SIO program executed is

53017: 40000 7402 40000 6000 67776 101147 40000 2400

53027: 72000 62273 4000 53021 34000 177777

D6 SIO program pointer equals 53031

D9 Last syndrome is

7400

573

420

177774

0

0

1

> 80

The above example writes and reads data on a 7905 disc. If a possibly correctable data error is detected, the Syndrome is requested and displayed.

## CAUTION

Use of RSYN except when a possibly correctable data error is indicated will return incorrect information.

# RRES

# RETURN RESIDUE

**FORMAL NAME:** RETURN RESIDUE

**SYNTAX:**

```
> RRES
```

**OPERATION:**

Return Residue places a return residue order pair in an SIO chain. See NAME command and SIO orders.) Return Residue is an SIO order acceptable only while entering a user-defined program (i.e., between the NAME and ENDS commands).

**EXAMPLE:**

```
> 10 DEV 0,5,13,25,0  
> 10 DB AA,128,R  
> 10 NAME RR  
> 10 RRES  
> 10 ENDS,I  
> 10 RS 0  
> 20 WDI 0,AA  
> 30 SIO 0,RR,1,3,%110000,0  
> 40 STAT S  
> 50 END  
> 60 RUN
```

```
D5 LAST SIO PROGRAM EXECUTED IS  
35502: 10000 14000 34000 30000  
D6 SIO PROGRAM POINTER EQUALS 35506  
> 60
```

The above is an example of Return Residue. The Return Residue was executed after a write to a 7900 disc.

# REWIND

# REW

FORMAL NAME: REWIND

SYNTAX:

```
> REW lun
```

PARAMETER: lun Logical unit number. Must be a magnetic tape unit.

OPERATION: Rewind will issue a rewind command to the magnetic tape unit.

DELAY TIME: The default interrupt time out is 3 minutes.

EXAMPLE:

```
> 10 DEV 0,6,18,10,2  
> 10 DB AA,128,B  
> 10 (%123),10(%345),10(%377),10(0),10(5),10(4)  
> 10 WD 0,AA  
> 20 LOOP 10,10  
> 30 REW 0  
> 40 END  
> 50 RUN  
> 50
```

The above example writes eleven 128 word records of data then rewinds the tape unit using the REW command.



# RST

# REWIND AND RESET

**FORMAL NAME:** REWIND AND RESET

**SYNTAX:**

```
> RST lun
```

**PARAMETER:**

lun

Logical unit number. Must be a magnetic tape unit.

**OPERATION:**

Rewind Reset will issue a Rewind Reset command to the magnetic tape unit.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 2,6,18,33,1  
> 10 RST 2  
> 20 END  
> 30 RUN  
> 30
```

The above example places the tape unit offline by issuing a Rewind Reset command.

# RIO

# RIO

FORMAL NAME: RIO

SYNTAX:

```
> RIO lun,buf
```

PARAMETERS:

lun

Logical unit number.

buf

Buffer into which the data is read.

OPERATION:

RIO will issue a read IO to the *lun* specified. The word that is read will be placed in the first word of the buffer specified in the *buf* parameter.

EXAMPLE:

```
> 10 DEV 0,127,24,10,0
> 10 DB AA,1,W
> 10 (%123)
> 10 DB BB,1,0
> 10 MC 0
> 20 CIO 0,%377
> 30 CIO 0,%150000
> 40 CIO 0,%170000
> 50 WIO 0,%123
> 60 RIO 0, BB,
> 70 CB 0,AA,BB,1
> 80 END
> 90 RUN
> 90
```

The above example indicates how an RIO may be used to test a device. The device used was a synchronous single line controller.

# RP

# RIPPLE PRINT

**FORMAL NAME:** RIPPLE PRINT

**SYNTAX:**

```
> RP lun,linelength
```

**PARAMETERS:** lun Logical unit number. Must be a lineprinter or a device on the Asynchronous MUX.

linelength Number of columns defining the area of ripple print. May be a constant or a variable.

**OPERATION:** Ripple Print will write a ripple pattern on the LUN indicated.

**DELAY TIME:** The interrupt time out is as follows:

Asynchronous MUX - 500 milliseconds/character  
Lineprinter - 4 seconds/line

**EXAMPLES:**

```
> 10 DEV 0,8,23,100,0  
> 10 RP 0,132  
> 20 END  
> 30 RUN  
> 30
```

The above is an example of 132 column Ripple Print on the lineprinter.

```
> 10 DEV 0,8,6,100,2,110  
> 10 RP 0,72  
> 20 END  
> 30 RUN  
> 30
```

The above is an example of 72 column Ripple Print on the Asynchronous MUX at 110 baud rate.

# RUN

# RUN

FORMAL NAME: RUN

SYNTAX:

```
> RUN
```

OPERATION:

Run will cause the execution of the previously entered program. Before the program is executed SLEUTH ensures all LOOP, IF, GOTO and GO commands are satisfied. If not, the appropriate error message is printed and the program is not executed. The Run command sets SLEUTH to the Automatic Numbering mode.

EXAMPLE:

```
> 10  
> 20  
> 30  
> 40  
> 50  
> 60 END  
> 70 RUN  
> 70
```

} ANY SLEUTH  
STATEMENTS

The above example illustrates how an End command terminates entering commands to the program and how the Run command causes the program to execute.

# SEEK

# SEEK

**FORMAL NAME:** SEEK

**SYNTAX:** > SEEK *lun* [, *cylinder*, *head*, *sector*]

**PARAMETERS:** *lun* Logical unit number. Must be a moving head disc.

*cylinder* }  
*head* } Disc parameters.  
*sector* }

**OPERATION:** Seek will issue a Seek command to a moving head disc. The Seek command updates the internal disc address.

**DELAY TIME:** The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 4,7,14,12,0
> 10 SEEK 4
> 20 SEEK 4,200,5,23
> 30 SEEK 4,405,1,5
> 40 LOOP 10,300
> 50 END
> 60 RUN
> 60
```

The above program example executes Seek operations on an ISS disc. The first Seek is to cylinder 0, head 0, sector 0. The second Seek is to cylinder 200, head 5, sector 23. The last Seek is to cylinder 405, head 1, sector 5. The program then loops 300 times.

# SEEK READ DATA

# SKRD

FORMAL NAME: SEEK READ DATA

SYNTAX:

```
> SKRD lun,buf [,mask [,cylinder,head,sector]]
```

PARAMETERS:

lun Logical unit number. Must be a 7905/7920/7925 disc.

buf Buffer into which data is read. Buffer length determines word count of the read.

mask Loads file mask on controller. Default is cylinder mode, no retries. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

cylinder  
head  
sector } Disc parameter; indicate target address of seek-read operation. Default is 0,0,0.

OPERATION:

Seek Read Data will issue a seek and a read operation within the same SIO program to a 7905/7920/7925 disc.

DELAY TIME:

The default interrupt time out is 500 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0  
> 10 DB AA,128,0  
> 10 SKRD 0,AA,2,10,1,2  
> 20 END  
> 30 RUN  
> 30
```

The above example uses the Seek Read Data command to read one sector of information on the 7905 disc.

# SKWD

# SEEK WRITE DATA

**FORMAL NAME:** SEEK WRITE DATA

**SYNTAX:**

```
> SKWD lun,buf [,mask [,cylinder,head,sector]]
```

**PARAMETERS:**

- lun* Logical unit number. Must be a 7905/7920/7925 disc.
- buf* Buffer from which data is written. Buffer length determines word count.
- mask* Loads file mask on controller. Default is cylinder mode, no retries. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

*cylinder*  
*head*  
*sector* } Disc parameter; indicate target address of seek-write operation. Default is 0,0,0.

**OPERATION:**

Seek Write Data will issue a seek and a write operation within the same SIO program to a 7905/7920/7925 disc.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

# SEEK WRITE DATA

Continued

# SKWD

## EXAMPLE:

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,4096,R
> 10 DB BB,4096,0
> 10 RAND I
> 20 LET A=I MOD 409
> 30 LET B=I MOD 3
> 40 LET C=I MOD 48
> 50 SKWD 0,AA,7,A,B,C
> 60 SKRD 0,BB,7,A,B,C
> 70 GO 100,%107400,0
> 80 RSYN 0
> 90 DISP 0,Y
> 100 CB 0,AA,BB,3
> 110 LOOP 10,100
> 120 END
> 130 RUN
> 130
```

The above example uses a Seek Write Data command to test a 7905 disc.



# SELU

# SELECT UNIT

**FORMAL NAME:** SELECT UNIT

**SYNTAX:**

```
> SELU lun, unit
```

**PARAMETERS:**

*lun*

Logical unit number. Must be a magnetic tape unit.

*unit*

Temporary unit select in the range of 0 to 3. Does not affect the logical unit number. The unit does not have to be on-line.

**OPERATION:**

Select Unit will select the unit specified in the *unit* parameter for the *lun* indicated. The *lun* must be defined as a mag tape.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 3,6,18,2,2  
> 10 SELU 3,1  
> 20 END  
> 30 RUN  
> 30
```

The above example issues a Select Unit SIO program to the magnetic tape unit. The selected unit is 1.

# SENSE

# SENS

FORMAL NAME: SENSE

SYNTAX:

```
> SENS
```

OPERATION:

Sense will place a Sense order pair in a SIO chain. (See NAME command and SIO orders.) Sense is an SIO order acceptable only while entering a user-defined SIO program (i.e., between the NAME and ENDS commands).

EXAMPLE:

```
> 10 DEV 0,5,13,23,0
> 10 NAME SN
> 10 SENS
> 10 ENDS,I
> 10 ENDS,I
> 10 SIO 0,SN,1,3,%110000,0
> 20 STAT S
> 30 END
> 40 RUN

D5 LAST SIO PROGRAM EXECUTED IS
  35502: 50000 10000 34000 30000
D6 SIO PROGRAM POINTER EQUALS 35506
> 40
```

The above program issues a Sense SIO order to the disc. The SIO program is then displayed.

# SBNK

# SET BANK

**FORMAL NAME:** SET BANK

**SYNTAX:**

```
> SBNK bank
```

**PARAMETER:**

*bank*

Specifies bank number. Must be  $\leq 15$ . All SLEUTH buffers are in bank 0. Use of any other bank requires the user to manage his own buffers.

**OPERATION:**

Set Bank places a set bank double word order pair in an SIO program. Set Bank is an SIO order acceptable only while entering user-defined SIO programs (i.e., between the NAME and ENDS commands).

**EXAMPLE:**

```
> 10 DB AA,128,0
> 10 NAME SB
> 10 SBNK 0
> 10 WRIT AA
> 10 ENDS,I
> 10 DUMP S,SB
75072: 14000 0 67600 77136 34000 177777
> 10
```

The above example uses the Set Bank command in the SIO program.

# SET CLOCK

# SCLK

FORMAL NAME: SET CLOCK

SYNTAX:

```
> SCLK value
```

PARAMETER:

value

Value loaded into the CPU process clock. May be a constant or a variable.

OPERATION:

Set Clock loads the process clock in the CPU with the indicated value or the value contained in a variable.

EXAMPLE:

```
> 10 SCLK 10  
> 20 RCLK A  
> 30 LIST A,8  
> 40 END  
> 50 RUN  
    %12  
> 50 EP,S  
> 10
```

The above example loads the process clock with a value of 10 using the Set Clock command.

# SED SET ENABLE/DISABLE INTERRUPTS

FORMAL NAME: SET ENABLE/DISABLE INTERRUPTS

SYNTAX:

```
> SED { 0 }  
          { 1 }
```

PARAMETERS: 0 Interrupt system is disabled.  
1 Interrupt system is enabled.

OPERATION: The SED command enables or disables the interrupt system (status register bit 1).

## CAUTION

**Disabling the interrupt system will produce erroneous results on most SLEUTH operations.**

EXAMPLE:

```
> 10 DEV 0,15,24,10,0  
> 10 SED 0  
> 20 MC 0  
> 30 CIO 0,%70350  
> 40 WIO 0,0  
> 50 CIO 0,%211  
> 60 WIO 0,%10000  
> 70 TIQ 0  
> 80 GO 60,%40,%177737  
> 90 MC 0  
> 100 SED 1  
> 110 END  
> 120 RUN  
> 120
```

The above example turns off the Interrupt System, then generates an interrupt from a clock console board. The interrupt condition is recognized with a test I/O. The condition is cleared and the interrupt system re-enabled.

# SET FILE MASK

# SFM

**FORMAL NAME:** SET FILE MASK

**SYNTAX:**

```
> SFM lun,mask
```

**PARAMETER:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

mask

Loads file mask on controller. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

**OPERATION:**

Set File Mask will set the file mask on the 13037 disc controller from bits 8-15 of the *mask* parameter. Parameter may be a constant or a variable.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0  
> 10 SFM 0,7  
> 20 END  
> 30 RUN  
> 30
```

The above example loads the file mask on the 7905 controller with the value 7.

# SETJ

# SET JUMPERS

**FORMAL NAME:** SET JUMPERS

**SYNTAX:**

```
> SETJ lun, jumpers
```

**PARAMETERS:**

lun

Logical unit number. Must be a Universal Interface with a test hood installed.

jumpers

Octal number between 0 and 1777 with each bit position corresponding to a jumper. 0 = reset, 1 = set. May be a constant or a variable. The jumper list is:

BIT	JUMPER
15	W1
14	W2
13	W3
12	W4
11	W5
10	W6
9	W7
8	W8
7	W9
6	W10

**OPERATION:**

Set Jumpers will Set Jumpers on the Universal Interface Test Hood.

# SET MASK

# SMSK

**FORMAL NAME:** SET MASK

**SYNTAX:**

```
> SMSK maskword
```

**PARAMETER:** *maskword* Octal word used to set the interrupt mask register.

**OPERATION:** Set Mask will set the interrupt mask register with the octal value designated in the *maskword* parameter. May be a constant or a variable.

**EXAMPLE:**

```
> 10 SMSK 0  
> 20 END  
> 30 RUN  
> 30
```

The above example issues a Set Mask with the pattern 0.



# SIN

# SIN

**FORMAL NAME:** SIN

**SYNTAX:**

```
> SIN lun
```

**PARAMETER:** lun Logical unit number.

**OPERATION:** SIN will issue a Set Interrupt Command to the specified unit.

**DELAY TIME:** The default interrupt time out is 200 milliseconds.

**EXAMPLE:**

```
> 10 DEV 4,5,24,12,0
> 10 SIN 4
> 20 END
> 30 RUN
> 30
```

The above example executes a direct set interrupt to the designated device.

# SIO

Issues SIO program to device.

# SIO

**FORMAL NAME:** SIO

**SYNTAX:**

```
> SIO lun,sioprogram,ints,time,status,mask
```

**PARAMETERS:**

<u>lun</u>	Logical unit number.
<u>sioprogram</u>	Two letter name of user's SIO program.
<u>ints</u> *	The number of interrupts which the SIO Program will generate.
<u>time</u> *	The amount of time allowed for the operation in 100 ms. increments.
<u>status</u> *	The expected interrupt status.
<u>mask</u> *	A mask of status don't care bits. 0 = CARE; 1 = DON'T CARE.

\*May be a constant or a variable.

**OPERATION:**

SIO will issue a user generated SIO Program to the logical unit specified in the *lun* parameter. SLEUTH will provide status checking, interrupt evaluation and appropriate error messages. Up to 16 SIO programs may be declared as long as the total length is less than 352 words. To delete as SIO program redefine it. If the SIO program has been referenced in the current SLEUTH program, the delete will fail and an error message (PE5) will be displayed.

**EXAMPLE:**

```
> 10 NAME SI  
> 10 SENS  
> 10 ENDS
```

- The above example defines an SIO program called SI.

```
> 10 NAME SI
```

The above example deletes the SIO program.

```

> 10 DEV 0,5,13,5,0
> 10 DEV 1,6,18,5,0
> 10 DB AA,4096,%177777
> 10 NAME XX
> 10 CONT 0,%100000
> 10 WRIT AA
> 10 JUMP *-4
> 10 ENDS
> 10 NAME YY
> 10 CONT 0,0
> 10 CONT 0,4
> 10 WRIT AA,C
> 10 JUMP *-2
> 10 ENDS
> 10 SEEK 0
> 20 SIO 0,XX,0,%177777,%177777
> 30 SIO 1,YY,0,5,%177777,%177777
> 40 TIO 1
> 50 GO 40,%2000,%175777
> 60 MC 1
> 70 REW 1
> 80 LOOP 30,3
> 90 MC 0
> 100 MC 1
> 110 END
> 120 RUN
> 120

```

The above example builds two SIO programs then issues one to disc and one to tape.

```

> 10 DEV 0,5,13,5,0
> 10 NAME SK
> 10 CONT,%100,%20000
> 10 ENDS
> 10 SIO 0,SK,1,3,%10370,0
> 20 END
> 30 RUN
> 30

```

The above example builds a seek SIO program and sends it to a disc.

# SKIP ADDRESS READ

# SA

FORMAL NAME: SKIP ADDRESS READ

SYNTAX:

```
> SA lun,buf [,cylinder,head,sector]
```

PARAMETERS:

<u>lun</u>	Logical unit number. Must be an ISS disc.
<u>buf</u>	Buffer into which the data is read. Buffer's length determines the word count of the read.
<u>cylinder</u>	} Disc parameters. Starting point of the read. The heads are assumed to be over the correct cylinder.
<u>head</u>	
<u>sector</u>	

OPERATION:

Skip Address Read will read the data field without checking the address field.

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,8,14,5,0
> 10 DB AA,2944,R
> 10 DB BB,2944,0
> 10 SEEK 0,400,0,0
> 20 WA 0,401,0,0
> 30 WD 0,AA,401,0,0
> 40 SA 0,BB,400,0,0
> 50 CB 0,AA,BB,5
> 60 END
> 70 RUN
> 70
```

The above program writes an invalid address at cylinder 400, then writes data on cylinder 400 (which now has the address of cylinder 401) then uses Skip Address Read to recover the data off that track.

# SAI

# SKIP ADDRESS READ IMMEDIATE

**FORMAL NAME:** SKIP ADDRESS READ IMMEDIATE

**SYNTAX:**

```
> SAI lun, buf
```

**PARAMETERS:**

lun

Logical unit number. Must be an ISS disc.

buf

Buffer into which data is read. Buffer length determines the word count of the read.

**OPERATION:**

Skip Address Read Immediate will read the data field without checking the address field. The starting point will be the address contained in the internal disc address.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,14,30,0  
> 10 DEV 1,6,18,30,0  
> 10 DB AA,2944,0  
> 10 IT 0  
> 20 SAI 0,AA  
> 30 WD 1,AA  
> 40 LOOP 10,19  
> 50 END  
> 60 RUN  
> 60
```

The above example reads one cylinder of an ISS disc and transfers it to magnetic tape using the Skip Address Read Immediate command. This operation could be used for data recovery, should the address fields become unreadable.

# STATUS DUMP

Prints status information from device.

# STAT

FORMAL NAME: STATUS DUMP

SYNTAX:

```
> STAT [ S ]  
          [ T ]  
          [ D ]
```

PARAMETERS: S - dump last SIO program  
T - dump last TIO status  
D - dump last CCG status

OPERATION: Status Dump will print status information obtained from the device regardless if an error has occurred or not.

## NOTE

If a SIO program dump is requested and the last SIO program is not available the following informative message will be printed:

D 7 NO CURRENT SIO PROGRAM

EXAMPLE:

```
> 10 DEV 0,127,24,3,0  
> 10 MC 0  
> 20 TIO 0  
> 30 STAT T  
> 40 END  
> 50 RUN  
  
D3 LAST TIO STATUS IS 1 100 101 011 100 000  
> 50
```

The above example obtains a TIO Status Dump using the STAT T command.

```
> 10 DEV 0,6,18,10,0  
> 10 WIO 0,123  
> 20 STAT D  
> 30 END  
> 40 RUN  
  
E1 WIO FAILED IN STEP 10 CONDITION CODE=CCG  
D4 LAST CCG STATUS IS 1 001 000 000 001 110  
> 40
```

The above example obtains the last CCG status using the STAT D command.

```
> 10 DEV 0,5,24,10,0  
> 10 NAME CA  
> 10 SENS  
> 10 ENDS.1  
> 10 SIO 0,CA,1,5,%177777,%177777  
> 20 STAT S  
> 30 END  
> 40 RUN  
  
D5 LAST SIO PROGRAM EXECUTED IS  
35702: 50000 10000 34000 30000  
D6 SIO PROGRAM POINTER EQUALS 35706  
> 40
```

The above example uses the STAT S command to display the last SIO program.

# SS

# SUPPRESS STATUS

Disables automatic check of status.

FORMAL NAME: SUPPRESS STATUS

SYNTAX:

```
> SS
```

OPERATION: Suppress Status will disable the automatic checking of status.

EXAMPLE:

```
> 10 DEV 0,5,13,10,0  
> 10 DB AA,128,0  
> 10 SEEK 0,100,0,0  
> 20 SS  
> 30 RD 0,AA,200,0,0  
> 40 END  
> 50 RUN  
> 50
```

The above example suppresses the status using the SS command then attempts an invalid read. The status is not checked.

# SWITCH OUTPUT

# SOUT

FORMAL NAME: SWITCH OUTPUT

SYNTAX:

```
> SOUT
```

OPERATION:

Switch Output will output error messages to the lineprinter or the system console. Initially error messages will be output to the system console. Each SOUT command will alternate the output device for error messages. This command will only be effective if the lineprinter's DRT number has been configured on the SLEUTH cold-load tape.

EXAMPLE:

```
> 10 DEV 0,11,5,100,0  
> 10 DB AA,66,S  
> ABC  
> 10 MC 0  
> 20 WD 0,AA,2  
> 30 SOUT  
> 40 LOOP 20,10  
> 50 END  
> 60 RUN  
> 60
```

The above example will alternate the output of error messages if errors are encountered.



# TDIL

# TERMINAL DATA INTERFACE LOOP

**FORMAL NAME:** TERMINAL DATA INTERFACE LOOP

**SYNTAX:**

```
> TDIL lun1,lun2,buf1,buf 2
```

**PARAMETERS:**

lun1 }  
lun2 }

Logical unit number. Must be asynchronous multiplexers.

buf1 }  
buf 2 }

Buffers between which data is transferred. Buffers must be of equal length.

**OPERATION:**

Terminal Data Interface Loop will transfer data from *lun1* to *lun2*, and from *buf1* to *buf2*. A test cable is installed between the *lun1* and *lun2* and data will be transferred one byte at a time until the buffers are exhausted.

**DELAY TIME:**

The default interrupt time out is 400 milliseconds/character.

**EXAMPLE:**

```
> 10 DEV 0,10,6,20,14,2400  
> 10 DEV 1,10,6,20,15,2400  
> 10 DB AA,4000,R  
> 10 DB BB,4000,0  
> 10 TDIL 0,1,AA,BB  
> 20 CB 0,AA,BB,5  
> 30 END  
> 40 RUN  
> 40
```

The above example shows a 4000 word buffer of random data being transmitted one byte at a time from port 14 to port 15 through a test loop using the TDIL command.

# TEST I/O

Issues TIO to device.

# TIO

**FORMAL NAME:** TEST I/O

**SYNTAX:**

```
> TIO lun
```

**PARAMETER:**

lun

Logical unit number.

**OPERATION:**

Test I/O will issue a Test I/O to the logical unit indicated. The TIO status may be displayed with a STAT T command.

**EXAMPLE:**

```
> 10 DEV 0,5,17,100,0
> 10 MC 0
> 20 TIO 0
> 30 STAT T
> 40 END
> 50 RUN

D3 LAST TIO STATUS IS 1 000 000 000 010 100
> 50
```

The above example issues a Test I/O to device number 5, and displays the status using a STAT T command.

# XDUI

# UNIVERSAL INTERFACE DATA TRANSFER

**FORMAL NAME:** UNIVERSAL INTERFACE DATA TRANSFER

**SYNTAX:**

```
> XDUI lun, buf1, buf2, [mode]
```

**PARAMETERS:**

<u>lun</u>	Logical unit number. Must be a Universal Interface with a test hood installed.
<u>buf1</u>	Buffer from which data is transferred.
<u>buf2</u>	Buffer to which data is transferred. Buffers must be of equal length.
<u>mode</u>	Specifies byte mode [ <u>B</u> ] or word mode [ <u>W</u> ]. Default is word mode.

**OPERATION:**

Universal Interface Data Transfer will execute a data transfer test on the indicated Universal Interface.

**DELAY TIME:**

The default interrupt time out per word is 100 milliseconds.

**EXAMPLE:**

```
> 10 DB AA,4000,R  
> 10 DB BB,4000,0  
> 10 DEV 0,9,25,100,0  
> 10 XDUI 0,AA,BB,B  
> 20 CB 0,AA,BB,5  
> 30 END  
> 40 RUN  
> 40
```

The above example transfers data through a Universal Interface from buffer AA to buffer BB in byte mode. The read buffer is then compared to the write buffer using the CB command.

# VERIFY

Verifies data.

# VER

**FORMAL NAME:** VERIFY

**SYNTAX:**

```
> VER lun,seccount [,cylinder,head,sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

seccount

Number of sectors to be verified. Must be consistent with file mask setting or errors will occur. May be a constant or a variable.

cylinder

head

sector

} Disc parameter indicating the starting address. Default is 0,0,0.

**OPERATION:**

The Verify command will verify the data on a number of sectors of a 7905/7920/7925 disc.

**DELAY TIME:**

The default interrupt time out is 2 seconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0  
> 10 SFM 0,7  
> 20 FOR I=0 TO 410  
> 30 SEEK 0,1,0,0  
> 40 VER 0,144,1,0,0  
> 50 NEXT I  
> 60 END  
> 70 RUN  
> 70
```

The above example verifies one 7905 cylinder at a time until the entire disc is checked.

# VERI

# VERIFY IMMEDIATE

Verifies data.

**FORMAL NAME:** VERIFY IMMEDIATE

**SYNTAX:**

```
> VERI lun,seccount
```

**PARAMETERS:**

lun

Logical unit number. Must be a 7905/7920/7925 disc.

seccount

Number of sectors to be verified. Must be consistent with file mask setting or errors will occur. May be a constant or a variable.

**OPERATION:**

Verify Immediate will verify the data on a number of sectors of a 7905/7920/7925 disc. The starting point will be the internal disc address.

**DELAY TIME:**

The default interrupt time out is 2 seconds.

**EXAMPLE:**

```
> 10 DEV 0,5,15,100,0  
> 10 DB AA,128,%155555  
> 10 RS 0  
> 20 WDI 0,AA  
> 30 VERI 0,1  
> 40 LOOP 10,500  
> 50 END  
> 60 RUN  
> 60
```

The above example seeks to random locations, writes, and verifies one sector.

# WIO

Issues a WIO to device.

# WIO

**FORMAL NAME:** WIO

**SYNTAX:**

```
> WIO lun,word
```

**PARAMETERS:**

lun

Logical unit number.

word

Word pattern to be written. May be a constant or a variable.

**OPERATION:**

WIO will issue a write I/O command of the word specified in the *word* parameter.

**EXAMPLE:**

```
> 10 DEV 0,15,24,5,0
> 10 DB AA,1,%12345
> 10 DB BB,1,0
> 10 MC 0
> 20 CIO 0,50
> 30 WIO 0,%12345
> 40 RIO 0,BB
> 50 CB 0,AA,BB,1
> 60 END
> 70 RUN
> 70
```

The above example selects the limit register of a clock console interface, issues a WIO of the pattern %12345, then it issues an RIO and compares the data read to the data written.



# WRITE ADDRESS

# WA

Writes one track of disc addresses on an ISS disc.

**FORMAL NAME:** WRITE ADDRESS

**SYNTAX:**

```
> WA lun [,cylinder,head,sector]
```

**PARAMETERS:**

lun

Logical unit number. Must be an ISS disc.

cylinder

head

sector

} Disc parameters specifying starting address regardless of actual position of heads.

**OPERATION:**

Write Address will write one track of disc addresses on an ISS disc.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,7,14,20,0  
> 10 SEEK 0,405,0,0  
> 20 WA 0,405,0,0  
> 30 END  
> 40 RUN  
> 40
```

The above example program performs a Write Address operation on one track starting at cylinder 405, head 0, sector 0.



# WAI

# WRITE ADDRESS IMMEDIATE

**FORMAL NAME:** WRITE ADDRESS IMMEDIATE

**SYNTAX:**

```
> WAI lun
```

**PARAMETER:**

lun

Logical unit number. Must be an ISS disc.

**OPERATION:**

Write Address Immediate will write one track of disc addresses on an ISS disc. The internal disc address will be used as the starting point.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,7,14,5,0  
> 10 IT 0  
> 20 WAI 0  
> 30 LOOP 10,8119  
> 40 END  
> 50 RUN  
> 50
```

The above example writes addresses on an ISS disc.

# WRITE DATA

# WD

Write data.

FORMAL NAME: WRITE DATA

SYNTAX:

	Format
> <u>WD lun,buf [,cylinder,head,sector]</u>	1
> <u>WD lun,buf [,mask [,cylinder,head,sector]]</u>	2
> <u>WD lun,buf [,track,arc]</u>	3
> <u>WD lun,buf,linelength</u>	4
> <u>WD lun,buf,mode,linelength</u>	5
> <u>WD lun,buf,mode</u>	6
> <u>WD lun,buf</u>	7
> <u>WD lun,buf [ mode[,print[,hopper[,stacker]]]]</u>	8

OPERATION:

Write Data will execute a Write operation on the *lun* indicated. Due to the variance of writable devices 8 discrete formats are required.

**Format 1:** Will be utilized with ISS and 7900 discs. The heads are assumed to be positioned at the correct address. The *cylinder, head, sector* (default 0,0,0) parameters are used to update the internal disc address.

**Format 2:** Will be used on 7905/7920/7925 discs. The file mask on the controller will be loaded with the value in the *mask* parameter. Default is cylinder mode, no retries. May be a constant or a variable.

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

The *cylinder, head, sector* parameters will designate the starting point of the write. Default is 0,0,0. The command updates the internal disc address.

**Format 3:** Will be used on fixed head discs. The *track, arc* parameters will indicate the starting point of the write. Default is 0,0.

**Format 4:** Will be utilized with terminal devices attached to the Asynchronous Multiplexer. The *linelength* parameter will indicate the length of each line. The parameter may be a constant or a variable.

**Format 5:** Will be used with lineprinters. The *mode* parameter will be the format character and will be one of the following:

OCTAL CODE	BIT							COMMAND
	9	10	11	12	13	14	15	
000	0	0	0	0	0	0	0	SUPPRESS SPACE
001	0	0	0	0	0	0	1	SINGLE SPACE
002	0	0	0	0	0	1	0	DOUBLE SPACE
⋮								⋮
077	0	1	1	1	1	1	1	63 SPACES
100	1	0	0	0	0	0	0	Chan 1 (Top of form)*
101	1	0	0	0	0	0	1	Chan 2 (Bottom of form)*
102	1	0	0	0	0	1	0	Chan 3 (Single space forms* step-over)
103	1	0	0	0	0	1	1	Chan 4 (Double space forms step-over)
104	1	0	0	0	1	0	0	Triple space forms step-over*
105	1	0	0	0	1	0	1	Next one-half page*
106	1	0	0	0	1	1	0	Next one-fourth page*
107	1	0	0	0	1	1	1	Next one-sixth page*

\*Assigned according to HP programming standards.

The *linelength* parameter will indicate the length of each line. Either parameter may be a constant or a variable.

**Format 6:** Will be used on the paper tape punch and SEL CHAN MAINT cards. The *mode* parameter will indicate the transfer mode and will be one of the following:

#### PAPER TAPE PUNCH

W - Word mode  
B - Byte mode

#### SEL CHAN MAINT CARD

F - Fast Request Mode  
S - Slow Request Mode

**Format 7:** Will be used with Mag Tape and the Plotter.

# WRITE DATA

Continued

# WD

**Format 8:** Will be used on the Reader/Punch. The *mode* parameter may be one of the following:

- C - Column Binary
- H - Hollerith

Default is Hollerith to ASCII conversion. The *print* parameter indicates the type of printing and punching to be performed and may be one of the following:

- P - Print and Punch Data
- N - No punching, print only
- S,buf* - Separate Print Data Supplied from a buffer designated in the *buf* parameter.

Default is Print/Punch Data. The *hopper/stacker* parameters indicate the source and destination of cards and may be one of the following:

- P - Primary Hopper/Stacker
- S - Secondary Hopper/Stacker

Default is Primary Hopper/Stacker.

**DELAY TIMES:** The following are default interrupt time outs:

DEVICE	TIME OUT
Async Mux	300 milliseconds
Discs	300 milliseconds
Tape	5 seconds
Plotter	2 minutes
Lineprinter	4 seconds
Paper Tape Punch	20 seconds
Reader/Punch	7 seconds

## EXAMPLES:

```

> 10 DEV 0,12,6,100,0,2400
> 10 DB AA,35,S
> X
> 10 LET A= 1
> 20 WD 0,AA,A
> 30 LET A=A+ 1
> 40 LOOP 20,68
> 50 LET A=A- 1
> 60 WD 0,AA,A
> 70 LOOP 50,67
> 80 LOOP 10,5
> 90 END
> 100 RUN
> 100

```

The above is an example of Write Data on the Async Mux. The line length in the variable A is modified each write.

```

> 10 DEV 0,11,5,100,0
> 10 DB AA,66,S
> E
> 10 WD 0,AA,1,132
> 20 LOOP 10,100
> 30 END
> 40 RUN
> 40

```

The above example writes on a lineprinter with single space with a line length of 132 characters.

```

> 10 DEV 0,7,21,15,0
> 10 DB AA,200,R
> 10 WD 0,AA,B
> 20 END
> 30 RUN
> 30

```

The above example writes a 200 word buffer of random data on a paper tape punch.

```

> 10 DEV 0,5,17,25,0
> 10 DB AA,4000,R
> 10 DB BB,4000,0
> 10 WD 0,AA,100,5
> 20 RD 0,BB,100,5
> 30 CB 0,AA,BB,4
> 40 END
> 50 RUN
> 50

```

The above example writes 4000 words of random data starting at track 100, sector 5, then reads and checks the data on a fixed head disc.

# WRITE DATA

Continued

# WD

## EXAMPLES:

```
> 10 DEV 0,6,18,35,0
> 10 DB AA,8000,R
> 10 WD 0,AA
> 20 LOOP 10,50
> 30 REW 0
> 40 END
> 50 RUN
> 50
```

The above example writes records of 8,000 random words of data 51 times on magnetic tape.

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,6144,R
> 10 FOR I=0 TO 410
> 20 SEEK 0,1,0,0
> 30 WD 0,AA,7,1,0,0
> 40 NEXT I
> 50 END
> 60 RUN
> 60
```

The above example fills one full surface of a 7905 disc with random data.

```
> 10 DEV 0,22,10,100,0
> 10 DB AA,40,0
> 10 CHB AA,%30060
> 20 FOR I=0 TO 9
> 30 WD 0,AA
> 40 CHB AA,I
> 50 FOR J=0 TO 7
> 60 CHB AA,S
> 70 NEXT J
> 80 CHB AA,I
> 90 NEXT I
> 100 END
> 110 RUN
> 110
```

The above example prints and punches 10 cards from the primary hopper on the Reader/Punch and places them in the primary stacker. Successive cards are filled with the digits 0-9.

# WDI

# WRITE DATA IMMEDIATE

Writes data.

FORMAL NAME: WRITE DATA IMMEDIATE

SYNTAX:

> <u>WDI <i>lun,buf</i></u>	Format	1
> <u>WDI <i>lun,buf</i> [,<i>mask</i>]</u>		2

PARAMETERS:

<u><i>lun</i></u>	Logical unit number. Must be a moving head disc.
<u><i>buf</i></u>	Buffer into which data will be written. Buffer length determines word count of the write.
<u><i>mask</i></u>	Loads file mask on 13037 disc controller. Default in cylinder mode. The mask bits are:

Bits	Function
8-11	Number of retries allowed.
12	Incremental/Decremental Seek. If set and bit 15 is a 1, the cylinder address will be decremented when End-of-Cylinder; otherwise, incremented.
13	Allow sparing.
14	Cylinder/Surface mode. If set, a cylinder consists of all available surfaces; End-of-Cylinder is set when the last sector of the last surface has been transferred. In surface mode, End-of-Cylinder is set when the last sector of any surface has been transferred.
15	Allow Incremental/Decremental seek.

OPERATION:

Write Data Immediate will write data on a moving head disc. Format 1 will be used on 7900 and ISS discs. Format 2 will be used on the 7905/7920/7925 disc. The internal disc address will designate the starting point of the write.

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 0,5,15,100,0
> 10 DB AA,128,%155555
> 10 RS 0
> 20 WDI 0,AA,7
> 30 LOOP 10,100
> 40 END
> 50 RUN
> 50
```

The above example randomly seeks and writes data on a 7905 disc. Sparing is enabled.

# WRITE FILE MARK

# WFM

Write a file mark on magnetic tape.

**FORMAL NAME:** WRITE FILE MARK

**SYNTAX:**

```
> WFM lun
```

**PARAMETER:** lun Logical unit number. Must be a magnetic tape unit.

**OPERATION:** Write File Mark will write a file mark on a magnetic tape.

**DELAY TIME:** The default interrupt time out is 200 milliseconds.

**EXAMPLE:**

```
> 10 DEV 1,6,18,3,0  
> 10 DB FF,6000,%22222  
> 10 WD 1,FF  
> 20 WFM 1  
> 30 REW 1  
> 40 FSF 1  
> 50 RST 1  
> 60 END  
> 70 RUN  
> 70
```

The above example writes a file mark on magnetic tape, rewinds, then forward spaces to the file mark.



# WFS

# WRITE FULL SECTOR

Performs a write full sector operation.

FORMAL NAME: WRITE FULL SECTOR

SYNTAX:

```
> WFS lun,buf [,cylinder,head,sector]
```

PARAMETERS:

<u>lun</u>		Logical unit number. Must be a moving head disc.
<u>buf</u>		Buffer from which data is written. Buffer length determines word count of the write.
<u>cylinder</u>	}	Disc parameters.
<u>head</u>		
<u>sector</u>		

OPERATION:

Write Full Sector executes a full sector write operation on a moving head disc. The heads are assumed to be positioned at the correct address. The default *cylinder, head, sector* is 0,0,0. This command updates the internal disc address.

DELAY TIME:

The default interrupt time out is 300 milliseconds.

EXAMPLE:

```
> 10 DEV 1,7,14,10,0  
> 10 DB AA,129,%125252  
> 10 DB BB,129,0  
> 10 SEEK 1,400,3,5  
> 20 WFS 1,AA,400,3,5  
> 30 RFS 1,BB,400,3,5  
> 40 CB 1,AA,BB,5  
> 50 END  
> 60 RUN  
> 60
```

The above example performs a single write full sector and a read full sector on an ISS disc. The buffers are then checked to verify the data.

## CAUTION

If full sector write operations are performed the disc should be reformatted afterwards.

# WRITE FULL SECTOR IMMEDIATE

# WFSI

Performs a full sector write operation.

**FORMAL NAME:** WRITE FULL SECTOR IMMEDIATE

**SYNTAX:**

```
> WFSI lun,buf
```

**PARAMETERS:**

lun

Logical unit number. Must be a moving head disc.

buf

Buffer from which data is written. Buffer length determines word count of the write.

**OPERATION:**

Write Full Sector Immediate will perform a full sector write operation on a moving head disc. The heads are assumed to be positioned at the correct address.

**DELAY TIME:**

The default interrupt time out is 300 milliseconds.

**EXAMPLE:**

```
> 10 DEV 0,5,14,30,0  
> 10 DB AA,2967,R  
> 10 DB BB,2967,0  
> 10 SEEK 0  
> 20 WFSI 0,AA  
> 30 RFS 0, BB  
> 40 CB 0,AA,BB,5  
> 50 END  
> 60 RUN  
> 60
```

The above example uses the Write Full Sector Immediate command to write one track of random data on an ISS Disc. The data is then read and compared.

## CAUTION

If full sector write operations are performed the disc should be reformatted afterwards.

# WRZ

# WRITE RECORD WITH ZERO PARITY

Writes a record with zero parity.

**FORMAL NAME:** WRITE RECORD WITH ZERO PARITY

**SYNTAX:**

```
> WRZ lun,buf
```

**PARAMETERS:**

lun

Logical unit number. Must be an 800 CPI magnetic tape unit.

buf

Buffer into which data is written. Buffer length determines word count of the write.

**OPERATION:**

Write Record With Zero Parity will write a record with zero parity on a magnetic tape.

**DELAY TIME:**

The default interrupt time out is 5 seconds.

**EXAMPLE:**

```
> 10 DEV 0,6,18,10,0  
> 10 DB AA,8000,R  
> 10 WRZ 0,AA  
> 20 REW 0  
> 30 END  
> 40 RUN  
> 40
```

The above example writes one record without parity using the WRZ command.

# ZERO BUFFER

Zero the buffer.

# ZBUF

FORMAL NAME: ZERO BUFFER

SYNTAX:

```
> ZBUF buf
```

PARAMETER: buf Buffer to be zeroed.

OPERATION: Zero Buffer will zero the indicated buffer.

EXAMPLE:

```
> 10 DEV 0,5,14,20,0  
> 10 DB AA,2944,%55555  
> 10 DB BB,2944,0  
> 10 IS 00,19,0  
> 20 WDI 0,AA  
> 30 ZBUF BB  
> 40 RDI 0,BB  
> 50 CB 0,AA,BB,5  
> 60 LOOP 10,404  
> 70 END  
> 80 RUN  
> 80
```

The above program writes data across the disc on head 19. Each track is read and compared. The ZBUF command is used to zero the read buffer before each read.

# REFERENCE TABLES

APPENDIX

A

Table A-1. Index of Commands and Mnemonics

INDEX OF COMMANDS AND MNEMONICS					
MNEMONIC	VARIABLE	COMMAND NAME	MNEMONIC	VARIABLE	COMMAND NAME
ACB	BUF(INDEX)=PRIMARY VARIABLE=BUF(INDEX)	ACCESS BUFFER	PUT	"STRING"	PUT
AR	LUN [,CYL,HD,SEC]	ADDRESS RECORD	RA	LUN[,CYL,HEAD,SEC]	READ ADDRESS
ASC	VARIABLE,BUF(INDEX){,8}	ASCII CONVERSION	RAI	LUN	RA IMMEDIATE
AUTO	{STEPN}	AUTO NUMBER STEPS	RAND	VAR	RANDOMIZE
BA	{RECORD/E}	BATCH IN TESTS	RC	LUN	RECALIBRATE
BSF	LUN	BACKSPACE FILE	RCLK	VAR	READ CLOCK
BSR	LUN	BACKSPACE RECORD	RD	LUN,BUF[,MODE,ETC.]	READ DATA
BUMP	{P}	BUMP PASS COUNT	RDA	LUN	REQUEST DISC ADDRESS
CA	LUN	COMPARE ADDRESS	RDC	LUN,BUFFER	READ RECORD W/CRC
CB	LUN,BUF1,BUF2,ERRCOUNT	COMPARE BUFFERS	RDI	LUN,BUFFER	RD IMMEDIATE
CC	LUN,SECCOUNT[,CYL,HEAD,SEC]	CYCLIC CHECK	READ	BUF[,C]	READ ORDER,SIO
CCI	LUN,SECCOUNT	CC IMMEDIATE	REN		RENUMBER PROGRAM
CHB	BUF,TYPE	CHANGE BUFFER	REW	LUN	REWIND MAG TAPE
CIO	LUN,CONTROLWORD	CONTROL I/O	RFS	LUN,BUF[,CYL,HEAD,SEC]	READ FULL SECTOR
CL	LUN	CLEAR	RFSI	LUN,BUF	RFS IMMEDIATE
CLR	LUN,BUFFER	COLD LOAD READ	RIO	LUN,BUF	READ I/O
CONF		CONFIGURE	RMSK	BUF	READ MASK
CONT	WORD1,WORD2	CONTROL ORDER,SIO	RNFI	LUN,BUF	RNFS IMMEDIATE
CORB	LUN,BUF	CORRECT BUFFER	RNFS	LUN,BUF[,CYL,HEAD,SEC]	READ NEXT FULL SECTOR
DB	NAME,LENGTH,DATA TYPE	DEFINE BUFFER	RP	LUN,LINELENGTH	RIPPLE PRINT
DELY	RESOLUTION	DELAY	RQST	LUN	REQUEST STATUS
DEV	LUN,DRT,TYP,ERR,UNIT[,BAUD]	DEVICE	RRES		RTN RESIDUE ORDER,SIO
DISP	LUN,TYPE	DISPLAY	RS	LUN	RANDOM SEEK
DS	LUN[,CYL,HEAD,SEC]	DECREMENTAL SEEK	RSA	LUN	REQUEST SECTOR ADDRESS
DUMP	PARAM[,PARAM]	DUMP QUANTITY	RST	LUN	REWIND & RESET
END		END COMMAND	RSYN	LUN	REQUEST SYNDROME
ENDS	{,}	END ORDER,SIO	RUN		RUN COMMANDS
EP	{S}	ERASE PROGRAM	RWO	LUN,BUF,MASK,OFFSET[,CYL,HD,SEC]	READ WITH OFFSET
ES		ENABLE STATUS	RWOI	LUN,BUF,MASK,OFFSET	READ WITH OFFSET IMMEDIATE
ESTA	STATUS,MASK	EXPECTED STATUS	RWV	LUN,BUF[[],MASK],CYL,HD,SEC)	READ WITHOUT VERIFY
FMT	LUN[,L/UI]	FORMAT	RWV1	LUN,BUF[,MASK]	READ WITHOUT VERIFY IMMEDIATE
FOR	<VAR>><PRI> TO <PRI>	FOR	SA	LUN,BUF[,CYL,HEAD,SEC]	SKIP ADDRESS READ
FSF	LUN	FORWARD SPACE FILE	SAI	LUN,BUF	SA IMMEDIATE
FSR	LUN	FORWARD SPACE RECORD	SBNK	BANK	SET BANK
FTD	LUN[,CYL,HEAD,SEC]	FLAG TRACK DEFECTIVE	SCLK	VALUE	SET CLOCK
FTDI	LUN	FTD IMMEDIATE	SED	0/1	SET ENABLE/DISABLE INT
GAP	LUN	GAP MAG TAPE	SEEK	LUN[,CYL,HEAD,SEC]	SEEK
GET	VAR	GET	SELU	LUN,UNIT	SELECT UNIT,MAG TAPE
GO	STEPN,EXPSTATUS,MASK	CONDITIONAL BRANCH	SENS		SENSE ORDER,SIO
GOTO	STEPN	UNCONDITIONAL BRANCH	SETJ	LUN,JUMPERS	SET JUMPERS UNIV IFACE
HALT		HALT %17	SFM	LUN,MASK	SET FILE MASK
ID	LUN,BUF[,CYL,HEAD,SEC]	INITIALIZE DATA	SIN	LUN	SET INTERRUPT
IDI	LUN,BUF	ID IMMEDIATE	SIO	LUN,PROG,INTS,TIME,EST,MASK	START I/O
IF	<PRI><RELOP><PRI> THEN <STEP>	IF	SKRD	LUN,BUF[[],MASK],CYL,HD,SEC)	SEEK READ DATA
INT		INTERRUPT ORDER, SIO	SKWD	LUN,BUF[[],MASK],CYL,HD,SEC)	SEEK WRITE DATA
IR	LUN,BUF,TRACK,ARC	INCREMENTAL READ	SMSK	MASKWORD	SET MASK
IS	LUN[,CYL,HEAD,SEC]	INCREMENTAL SEEK	SOUT		SWITCH OUTPUT
IT	LUN[,CYL,HEAD,SEC]	INCREMENTAL TRACK	SS		SUPPRESS STATUS
IW	LUN,BUF,TRACK,ARC	INCREMENTAL WRITE	STAT	S/T/D	STATUS DUMP
JUMP	ADDRESS[,C]	JUMP ORDER, SIO	TDIL	LUN1,LUN2,BUF1,BUF2	TERM DATA IFACE LOOP
LET	<VAR> = <EXPR>	LET	TIO	LUN	TEST I/O
LIST	VAR[,BASE]	LIST	VER	LUN,SECCOUNT[,CYL,HD,SEC]	VERIFY
LOOP	STEPN,TIMES	BRANCH	VERI	LUN,SECCOUNT	VERIFY IMMEDIATE
LTIO	LUN,WORD	LOAD TIO REGISTER	WA	LUN[,CYL,HEAD,SEC]	WRITE ADDRESS
MAKT	[A/N]	MAKE TEST TAPE	WAI	LUN	WA IMMEDIATE
MC	LUN	MASTER CLEAR	WD	LUN,BUF[,MODE ETC.]	WRITE DATA
NAME	SIOPROG	NAME SIO PROGRAM	WDI	LUN,BUF	WD IMMEDIATE
NEXT	VAR	NEXT	WFM	LUN	WRITE FILE MARK
NOPR		NO PRINT	WFS	LUN,BUF,CYL,HEAD,SECTOR	WRITE FULL SECTOR
PCT	LUN,PATTERN	PACK CERTIFICATION	WFSI	LUN,BUF	WFS IMMEDIATE
PE		PAUSE ON ERROR	WIO	LUN,WORD	WRITE I/O
POLL	LUN	RESUME POLLING	WRIT	BUF[,C]	WRITE ORDER, SIO
PR		PRINT	WR2	LUN,BUF	WRITE REC W/O PARITY
PROC	[N/D]	PROCEED	XDU1	LUN,BUF1,BUF2,MODE	XFR DATA UNIV. IFACE
			ZBUF	BUF	ZERO BUFFER

SWITCH REGISTER CONTROL	
Switch	
0	ENABLE SWITCH REGISTER
7	ENABLE STATUS CHECKING
11	NO PRINT
12	ENABLE SIO DUMP ON ERROR
13	ABORT EXECUTION OF CURRENT PROGRAM
14	PAUSE ON ERROR
15	SWITCH PRINTOUT TO ALTERNATE DEV

CODED HALTS	
Halt	Seg
XX	06 COLD LOAD HALT
15	20 INTERRUPT FROM SEG <>0,1,2 OR 5
16	20 ERROR HALT
17	20 COMMAND HALT

Table A-2 List of Commands by Device Types

GENERAL	ISS DISC	7800 DISC	MAG TAPE	DIRECT I/O	SIO ORDERS	7905/7920/7925 DISC
ACB LOOP	CA	CC	BSF	CIO	CONT	AR
ASC	CC	CCI	BSR	MC	ENDS	CL
AUTO MAKT	CCI	CLR	FSF	RIO	INT	CLR
BA MC	CLR	DS	FSR	RMSK	JUMP	DISP
BUMP NAME	DS	FMT	GAP	SED	READ	DS
CB NEXT	FMT	FTD	RD	SIN	RRES	FMT
CHS NOPR	FTD	FTDI	RDC	SIO	SBNK	FTD
CONF PE	FTDI	ID	REW	SMSK	SENS	FTDI
CORB PR	IS	IDI	RST	TIO	WRIT	ID
DB PROC	IT	IS	SELU	WIO		IDI
DELY PUT	PCT	IT	WD		P.T. PUNCH	IS
DEV RAND	RA	RC	WFM		WD	IT
DUMP RCKL	RAI	RD	WRZ	LINEPRINTER		LTIO
END REN	RC	RDI		RP		POLL
EP RUN	RD	RFS	F.H. DISC	WD		RC
ES	RDI	RFSI	IR		CARD READER	RD
ESTA SCLK	RFS	RNFI	IW		RD	RDA
FOR SOUT	RFSI	RNFS	RD	PLOTTER		RDI
GO SS	RS	RS	WD	WD		RFS
GET STAT	SA	SEEK				RFSI
HALT TAB	SAI	WD	P.T. READER			ROST
IF ZBUF	SEEK	WFS	RD			RS
LET	WA	WFSI		UNIVERSAL IF		RSA
LIST	WA1		ASYNC MUX	SETJ	SELECTOR CHANNEL TEST BOARD	R5YN
	WD		RD	XDU1	RD	RWO
	WFS		RP		WD	RWOI
			TDIL			RWV
			WD			RWVI
						SEEK
						SFM
						SKRD
						SKWD
						VER
						VER1
						WD
						WDI
						WFS
						WFSI

## INDEX OF COMMANDS

Command Name and Mnemonic	Page No.	Command Name and Mnemonic	Page No.
Access Buffer (ACB)	5-4	Print (PR)	5-73
Address Record (AR)	5-5	Proceed (PROC)	5-74
ASCII Conversion (ASC)	5-6	Put (PUT)	5-75
Automatic Numbering Resumed (AUTO)	5-7		
Backspace File (BSF)	5-10	Randomize (RAND)	5-76
Backspace Record (BSR)	5-11	Random Seek (RS)	5-77
Batch (BA)	5-8	Read (READ)	5-78
Bump Pass Counter (BUMP)	5-12	Read Address (RA)	5-79
		Read Address Immediate (RAI)	5-80
Change Buffer (CHB)	5-13	Read Clock (RCLK)	5-81
Clear (CL)	5-14	Read Data (RD)	5-82
Cold Load Read (CLR)	5-15	Read Data Immediate (RDI)	5-86
Compare Address (CA)	5-16	Read Full Sector (RFS)	5-87
Compare Buffer (CB)	5-17	Read Full Sector Immediate (RFSI)	5-88
Configure (CONF)	5-18	Read Mask (RMSK)	5-89
Control (CONT)	5-20	Read Next Full Sector (RNFS)	5-90
Control I/O (CIO)	5-21	Read Next Full Sector Immediate (RNFI)	5-91
Correct Buffer (CORB)	5-22	Read Rec with CRCC (RDC)	5-92
Cyclic Check (CC)	5-23	Read with Offset (RWO)	5-93
Cyclic Check Immediate (CCI)	5-24	Read with Offset Immediate (RWOI)	5-94
		Read without Verify (RWV)	5-95
Decremental Seek (DS)	5-25	Read without Verify Immediate (RWVI)	5-96
Define Buffer (DB)	5-26	Recalibrate (RC)	5-97
Delay (DELY)	5-27	Renumber (REN)	5-98
Device (DEV)	5-28	Request Disc Address (RCS)	5-99
Display (DISP)	5-29	Request Sector Address (RSA)	5-100
Dump (DUMP)	5-30	Request Status (RQST)	5-101
		Request Syndrome (RYSN)	5-103
Enable Status (ES)	5-34	Return Residue (RRES)	5-105
End (END)	5-35	Rewind (REW)	5-106
End SIO (ENDS)	5-36	Rewind and Reset (RST)	5-107
Erase Program (EP)	5-37	RIO (RIO)	5-108
Expected Status (ESTA)	5-38	Ripple (RP)	5-109
		Run (RUN)	5-110
Flag Track Defective (FTD)	5-39		
Flag Track Defective Immediate (FTDI)	5-40	Seek (SEEK)	5-111
For (FOR)	5-41	Seek Read Data (SKRD)	5-112
Format (FMT)	5-42	Seek Write Data (SRWD)	5-113
Forward Space File (FSF)	5-43	Select Unit (SELU)	5-115
Forward Space Record (FSR)	5-44	Sense (SENS)	5-116
		Set Bank (SBNK)	5-117
Gap (GAP)	5-45	Set Clock (SCLK)	5-118
Get (GET)	5-46	Set Enable/Disable Interrupts (SEQ)	5-119
Go (GO)	5-47	Set File Mask (SFM)	5-120
Go To (GOTO)	5-48	Set Jumpers (SETJ)	5-121
		Set Mask (SMSK)	5-122
Halt (HALT)	5-49	SIN (SIN)	5-123
		SIO (SIO)	5-124
If (IF)	5-50	Skip Address Read (SA)	5-126
Incremental Read (IR)	5-51	Skip Address Read Immediate (SAI)	5-127
Incremental Seek (IS)	5-52	Status Dump (STAT)	5-128
Incremental Track (IT)	5-53	Suppress Status (SS)	5-129
Incremental Write (IW)	5-54	Switch Output (SOUT)	5-130
Initialize Data (ID)	5-55		
Initialize Data Immediate (IDI)	5-57	Terminal Data Interface Loop (TDIL)	5-131
Interrupt (INT)	5-58	Test I/O (TIO)	5-132
Jump (JUMP)	5-59	Universal Interface Data Transfer (XDUI)	5-133
Let (LET)	5-60	Verify (VER)	5-134
List (LIST)	5-62	Verify Immediate (VERI)	5-135
Load TIO Register (LTIO)	5-63		
Loop (LOOP)	5-64	WIO (WIO)	5-136
		Write (WRIT)	5-137
Make Test Tape (MAKT)	5-65	Write Address (WA)	5-138
Master Clear (MC)	5-66	Write Address Immediate (WAI)	5-139
		Write Data (WD)	5-140
Name SIO Program (NAME)	5-67	Write Data Immediate (WDI)	5-145
Next (NEXT)	5-68	Write File Mark (WFM)	5-146
No Print (NOPR)	5-69	Write Full Sector (WFS)	5-147
		Write Full Sector Immediate (WFSI)	5-148
Pack Certification (PCT)	5-70	Write Record with Zero Parity (WRZ)	5-149
Pause On Error (PE)	5-71		
Poll (POLL)	5-72	Zero Buffer (ZBUF)	5-150

