the digital group
po box 6528 denver, colorado 80206 (303) 777-7133

HMON

296-088

THE DIGITAL GROUP

HARDWARE MONITOR VER 1.0

PREFACE

This hardware monitor (HMON) was designed originally to aid in the development of the DIGITAL GROUP double density controller. Its purpose was to aid the designer in testing and debugging every portion of the controller circuitry. During development we realized that this design "tool" could be put to use by other people in building and debugging other DIGITAL GROUP products. It is to this aim that we are releasing HMON. Bear in mind that HMON is NOT the ultimate in hardware monitors, but a collection of functions we deemed necessary to adequately test the controller under construction. To this end, some functions will not seem useful to the user. But as the user becomes comfortable with the "language" he will find uses for most of the functions. It is assumed that the user is familiar with assembly language programming, the Suding Operating system and a basic knowledge of the disk controller in use.

Larry Williams

TABLE OF CONTENTS

TABLE OF CONTENTS

HARDWARE MONITOR VER 1.0

INTRODUCTION

HMON is a hardware monitor suited to the testing of memory and peripheral equipment for the DIGITAL GROUP Z80 System. The basic system is supplied on cassette, although the user may want to save the system on either digital cassette or disk at a later date.

The monitor is an interpreter in the strict sense. It executes commands by reading source code typed in by the user. No attempt is made to parse or atomize the code. Therefore, no realtime applications faster than milliseconds were intended.

HMON will allow the user to inspect/modify any memory location or input/output port. Command chains may be written that allow the user to cause specific events to occur at a given rates. This allows external test equipment to "sync" to desired signals, speeding up debugging time.

HMON is supplied in two versions. Version (/1) is intended for use with the DIGITAL GROUP Single Density Disk Controller. The second version (/2) is intended for use with the new Single/Double Density Disk Controller. Although either version may be used without a controller present, the user should exersize caution that no disk associated functions are executed.

One function should be mentioned now is the HELp function. This function displays all of HMON's functions in their shortest form. This function will aid the user should he forget the short form of a function.

CHAPTER 1

OPERATING THE SYSTEM

The Monitor requires 9500 bytes of continuous memory starting at page 1. Of this 9500 bytes, the first 6800 are the Monitor code, while the rest is temporary storage and buffers. The FORmat function requires additional memory for its operation. Consult the explanation of the FORmat function for specific memory requirements.

Starting address for execution is 005 000. This is the standard Suding Operating system startup location. The actual Monitor has two different Entry Points. The first Entry Point initializes all Disk Drives attached to the system. This Entry Point (OPTION 5) should be used only with a functioning Disk Controller. Option 5 vectors to location 011 000. The other Entry Point does no Disk initialization. This Entry Point (OPTION 6) should be used when the user wishes to keep all internal Macro's and Disk Status intact. Option 6 vectors to location 011 007. Option 6 also has the feature that it disables all disk functions that could cause problems when an untested Controller board is present.

The prompt character (indicating user command input mode) is either the period '.' or the colon ':'. The period indicates the system is in the split octal mode (default on Entry or re-entry). The colon indicates that the system is in one of the other output modes (decimal, octal or hex).

Command execution is in one of two different fashions. The user may operate in a direct mode or in a stored program mode. The direct mode is utilized by typing in the command string directly. The stored program mode is utilized by executing the LOAd, LISt, or MACro run functions. There are eight Macro instructions, which consist of up to 255 Bytes of stored ASCII text containing monitor instructions. These may be executed independently or chained together, providing elementary programibility.

Commands may be strung together (up to 255 characters). They are separated by the colon delimiter. The multiple command feature permits the user to rapidly execute a series of functions. In addition, it greatly enhances the power of the stored program (Macro) mode.

1.1 OPERATION OF THE INPUT TEXT BUFFER

The input text buffer is a 255 Byte block of memory designed to hold the input data while it is being edited from the keyboard. This permits the user to set up the contents of a line completely before it is processed by the system. The user may type up to 255 characters into the input buffer. If this number is exceeded, each character not loaded is echoed along with a

"bell" character (Control-G). The number 255 represents the actual total number of EDITED characters and does not include deleted characters. During input, the user may backspace one character in one of three ways:

(1). Using the Control-H key. This causes a cursor reverse movement in many video and printing terminals.

(2). Using the underscore key. This is most commonly used on TELETYPE terminals, the DIGITAL GROUP TVC64 terminal and other terminals not equipped with a reversible cursor. It is also known as back-arrow. On the TELETYPE Model 33, this is the shift-O.

(3). Using the rubout key. This key causes the character lost (the most recent previously entered character) to be echoed back enclosed by slant-bars.

The user may wish to cancel the whole line and start again. There are two keys to accomplish this function.

(1). The "@" key. This key deletes the line and permits the user to re-start.

(2). The Control-X key. This prints a backslash and deletes the entire line, permitting the user to re-start.

The entire input line may be reviewed by striking the Escape key. This is the Control-Shift-K on some terminals. The entire line as is presently entered is typed back, stopping with the last character entered. The user may then continue typing where previously left off. The text is printed back with an extra space at the beginning, to line up the text as entered with the line printed back (to compensate for the prompt character). This feature is especially useful on terminals having no cursor backspace capability.

Example:

:abcdeEDC (three rubouts then the escape are entered) the system responds by typing:

AB (note the automatic lower to upper case conversion)

1.2 KEYBOARD SYSTEM CONTROL

During the execution of all system functions (except as noted) ,the system may be controlled with the following three keys:

(1). The Control-C key. Immediately cease execution and return to the

command input mode. The system responds by typing:

<STOP>

:

and the system is ready to accept commands again.

(2). The Control-S key. The system suspends execution of the current function and waits for the Control-Q key to be depressed. The user may now review the data on the screen or perform tests on the hardware.

(3). The Control-Q key. The system resumes execution of the function previously suspended by the Control-S key.

1.3 NUMERIC INPUT DATA FORMAT

There are four modes to enter data into the system. These are: DECimal, OCTal, SPLit-octal and HEXidecimal. The user may input data in any mode at any time. For example, the address of a DEPosit function may be in split octal while the data is hex.

(1). DECimal. Decimal numbers are entered into the system with a decimal point PRECEEDING the number. The maximum value for decimal entry is .65535.

(2). OCTal. Octal input is the default input mode with the largest value being 177777.

(3). SPLit-octal. Split-octal is entered with a slant-bar separating the Byte pairs. The largest value for Split-octal is 377/377. Note that if the constant to be entered is less than 400 octal, the format for octal and split octal are the same.

For example, to represent the value 377 base 8 the following are the same:

0/377
/377
and
377

(4). HEXidecimal. Hexidecimal numbers are entered with an 'H' PRECEEDING the number. The largest Hexidecimal value is HFFFF.

1.4 THE DISK BUFFER

The Monitor uses a disk buffer that varies in size with the density of the disk in use. The size of this buffer is either 128 or 256 bytes. The 128 byte buffer is used for all single density operations, while the 256 byte buffer is used in all double density operations. The 128 byte buffer is actually the bottom half of the 256 byte buffer. There exists a problem in displaying the entire 256 byte buffer. Data cannot be formatted on the screen without confusion. Therefore, for display purposes, the 256 byte buffer is treated with three separate functions, DMB, DMT and GED. These functions work on different portions of the entire 256 byte buffer.

1.5 COMMAND FORMAT

All commands are entered using one standard format:

OPCODE - <operand>[,<additional operands>]

Where any operand inclosed by <..> IS required and any optional operands are inclosed by [....].

All opcodes are three letter mnemonics representing the desired function. Only the first three letters of the command are used. Additional letters may be entered, but the system will ignore them. Thus, the following are equivalent:

```
EXA-6,.100
exa-6,.100
EXAMINE-6,.100
EXA - 6,   .100
```

Note that spaces are ignored entirely. They may be inserted at will to improve the clarity of the entries. Operands are always numeric (except for the ASL and MES function). These are numbers expressed in any valid input mode. Note that functions that require multiple operands may be any combination of modes and not restricted to any one mode. Leading zero characters are ignored. An omitted operand is treated in two different ways. It may be taken to mean a zero by some functions or it may be taken to mean "use the previously designated operand". See specific functions for how each treats missing operands. Some functions require no operands at all. With these functions, data within the operand field, if any, is ignored. The following are equivalent:

```
Hexidecimal
HEX-3,4,,4
```

Here, as is the case with each command, unneeded operands are simply ignored.

1.6 ERROR CONDITIONS

All errors are FATAL to execution, that is, when the system encounters an error, the system will immediatelly cease execution and generate the appropriate error message. If the error occurs during the execution of a Macro instruction, the Macro is NOT lost. If the error occurs during the execution directly from the input buffer, the input buffer's contents are lost and must be retyped.

SYSTEM ERROR CONDITIONS

<STOP> : Acknowledges that the Control-C key was depressed.

<ILLEGAL FORMAT> : A command was entered that had fewer than three characters, or contained non-alpha characters. (Only A-Z or a-z are permitted in the OPCODE field)

<ILLEGAL COMMAND>: Attempt to execute a command that is not implemented.

<MISSING OPERAND(S)>: Ommission of one or more operands from the operand field. These are required for proper command operation.

<BUFFER OVERFLOW>: A constant value, in the operand field, had a value greater than can be expressed in 16 bits.

<OPERAND TOO LARGE>: A constant value, in the operand field was too large for a given operation. An example of this is an attempt to deposit 400 Octal into a memory location, 400 Octal is not defined in an eight bit field.

<INVALID CONSTANT>: A constant in the operand field could not be interpreted properly, such as 888 in octal or .45FE in decimal.

<SUBCODE OVERFLOW>: More than 64 decimal Bytes were entered (attempted) into the SUBroutine command.

<NO MACRO>: Attempt made to run an undefined MACro.

DISK ERRORS

<RECORD NOT FOUND>: After 'N' retrys, the desired sector could not be found.

<BAD UNIT NUMBER>: Unit number exceeding the maximum allowable by the controller in use.

<UNIT NOT PRESENT>: Valid unit number that the controller said was not installed.

<ID FIELD ERROR>: The requested sector had a error in the ID field or a seek command did not verify.

<INVALID SECTOR>: Sector number requested does not exist on the drive selected.

<INVALID TRACK>: Track number requested does not exist on the drive selected.

<SEEK ERROR>: A seek error occurred on the last disk operation.

<CRC ERROR>: A data field CRC error was detected on the last disk read operation. Note that the data requested was transferred to the location specified.

<DATA LATE>: The last sector read or write failed to transfer data at the requested rate.

CHAPTER 2

COMMAND DEFINITIONS

2.1 ASD ASCII DUMP

The AScii Dump function permits the user to display the contents of memory as actual ASCII characters. Data not printable as an ASCII character,is displayed as a period '.'. The command is functional in all input modes.

ASD-<Start address>,<End address>[,Bytes per line]

The parameters are handled exactly as the DuMP function would handle them. Each character is separated by a space. There is no indication that D7 (MSB) is set high or low within the text. Lower case is displayed properly on terminals so equipped. If no operands are supplied, the system will dump the contents of the entire disk buffer (all 256 loactions) in ASCII.

For example, after executing the ASL example outlined under the ASCII LOAD function, the following command is entered:

:ASD-100/000,100/002

The system responds with:

040000 A B C

2.2 ASL ASCII LOAD

The AScii Load function permits the user to load straight, printable ASCII text into the system's memory. The text has D7 set low. The system will respond after writing memory with the last byte written.

ASL-<Start address>

The system initially responds with an astrisk '*' indicating that it is ready for the ASCII text input. After the text has been entered and RETURN hit, the system responds by printing:

LAST ADDRESS WRITTEN = XXXXX

where XXXXX is the last address actually modified. It corresponds with the last character on the input line. The system input buffer is used for data entry. As a result, several characteristics are defined:

(1). The text may contain ONLY printable characters. No control characters may be entered (including carriage return/line feed) and in addition, the

underscore (or back-arrow) and rubout, as well as '@' may not be imbedded in the text. The comma ',' IS permitted.

(2). The text may be a maximum 255 characters long.

(3). All of the input editing features found in the system are available.

(4). The parity bit (D7) will be set low for each byte written.

For example, if the user types:

ASL-100/000 (cr)

*ABC (cr)

The system responds with:

LAST ADDRESS WRITTEN = 040002

If the user then types:

DMP-100/000,100/002

The system will respond with:

040000 101 102 103

2.3 CAL EXTERNAL SUBROUTINE CALL

The CALL function permits the user to exit the main system and execute a machine language subroutine located elsewhere in memory. This external subroutine may be terminated by a RETURN instruction and the system will continue execution with the next function.

CAL-<Start address>

The operand, 'Start address', represents the beginning address of the subroutine. Upon entry the contents of the registers are not known. The user may destroy all registers. There exists two special cases of the CALL function. These are the RET and SUB functions to be explained later.

2.4 CHA CHARACTER INPUT

The CHAracter function permits the user to read data from the system keyboard in a manner similar to the INPut function. The CHAracter function for DIGITAL GROUP standard keyboards is logically equivalent to the

following:

WAI-0,200:INP-0

with the following exceptions:

(1). After the character is brought in, parity is stripped, and the input response message 'PORT XXX = YYY' is not printed.

(2). The Control S and Q functions are disabled, and only the Control-C remains functional.

If the user attempts to use the WAI...INP approach to obtaining data from the keyboard, approximately every other character typed WILL NOT be acknowledged, as they are trapped by the Control C/S/Q functions. Thus, for example, the user may implement a simple echo routine by typing the following code:

CHA:OUT-0:OUT-0,0:CON

2.5 CLE CLEAR MEMORY

The CLEar memory function permits the user to initialize a block of memory to a given arbitrary value.

CLE-<Start address>,<End address>[,value]

A memory field beginning at 'Start address', and ending with 'End address', is preset to a number corresponding to 'value' . If the optional 'value' is omitted, the memory field is set to a value of zero.

2.6 CON CONTINUE

The CONtinue function is used in conjunction with multiple statement per line expressions. It causes the system to continue execution at the beginning of the present line.

CON

2.7 DEL DELAY

The DELay function generates a predictable time delay. It is to be used in conjunction with other statements to create a time spacing between

functions.

DEL-[Delay]

The operand 'Delay', may be omitted. If this is done, the system will recall the most recent previous delay function parameter and use that. This feature permits a large number of identical DELay functions to be placed within system source code without the need for retyping the parameter each time. The operand in the data field is interpreted as a delay in milliseconds.

For example:

DEL-1750

Here, 1750 Octal= 1000 decimal. This will generate a one second delay. If, after this statement has been performed, the user desires to generate another one second delay, the following is entered:

DEL

This will again generate a one second delay. The DELay function is accurate down to delays of approximately 35 milliseconds. Below this value, the inter-instruction processing time becomes significant and the delay value will limit to whatever processing time is present. The DELay function is designed generally for delays of 500 milliseconds to 100 seconds. The delay function WILL respond to all Control keys during execution so that long delays may be aborted.

2.8 DEP DEPOSIT

The DEPosit function permits the user to load various sequential memory locations with a set of arbitrary Byte values. Note that the memory locations must be sequential.

DEP-<Start address>,<Byte>[[,Byte]...]

For example:

DEP-340/033,377,.300,HFF

Should the user desire to load a large number of sequential memory locations with a single, arbitrary Byte value, it is recommended that the CLEar function be used. This function is also useful to fill the disk buffer with small amounts of data or to change system parameters as listed in the Appendix on system buffers and variables.

2.9 DEC DECIMAL OUTPUT

The DECimal function instructs the system to output all numerical data in decimal format. This is useful when the user needs decimal numbers to calculate with.

DEC

2.10 DMB DUMP BOTTOM BUFFER

This function will dump the first 128 locations of the disk buffer. It is a short form of the DMP function and is useful when the user needs to look at the contents of a specific sector of data from a disk drive.

DMB

When the user needs to look at the last 128 bytes of the disk buffer use the DMT function.

2.11 DMT DUMP TOP BUFFER

This function acts the same as the DMB function except it displays the top 128 bytes of the disk buffer.

DMT

Note that the DMT command is only meaningful after reading a 256 Byte sector.

2.12 DMP DUMP MEMORY

The DuMP memory function permits the user to view a large number of sequential memory locations, in a more compact format than is possible by the use of the EXAmine function.

DMP-<Start address>,<End address>[,Bytes per line]

The memory dump starts with the 'Start address', and ends with the 'End address', and has an optional Bytes per line parameter. If this parameter is omitted, or set to zero, eight Bytes per line are generated. Otherwise, the user may print any number from one to 255 bytes per line. for example:

DMP-2,11,3

generates the following:

```
000002    061 000 002
000005    041 000 340
000010    030 003
```

2.13 ECH ECHO CHARACTER

The ECHo function permits operation with half-duplex terminals. By typng ECH-0, the user places the system in a no-echo mode, for operation with a half-duplex terminal. By typing ECH-1, the system places itself in a full duplex mode.

2.14 ERA ERASE SCREEN

The erase screen function enables the user to format the screen with data starting at the top without sending multiple carriage returns.

ERA

2.15 EXA EXAMINE MEMORY

The EXAmine function permits the user to view various memory locations. Multiple, arbitrary locations may be examined by placing a string of addresses in the operand field.

EXA-<Address>[[,Address]...]

For example:

EXA-3,4,,5

Performs an examination of locations 000003, 000004,000000, and 000005. The resultant output will be (in Octal mode):

```
000003 = 000
000004 = 377
000000 = 061
000005 = 303
```

Should the user desire to view a large number of SEQUENTIAL memory

locations, it is recommended that the DMP function be used.

2.16 FOR FORMAT A DISKETTE

The FORmat Diskette function allows the user to format any variety of diskette. The FORmat function checks the drive type and prompts the user prior to actually formatting the diskette.

FOR-<Drive>

Where 'Drive' is any valid drive number (0 through 3).

If the user types:

FOR-1

The system responds with:

DOUBLE DENSITY STANDARD -SINGLE SIDED-
THIS OK? (Y/N)

The user can terminate the FORmat by typing N, or proceed by typing Y.

--WORKING--

--DONE--

The Format function has to preformat an entire track in memory prior to writing this to the diskette. For each different drive type, a different ammount of memory is required. The maximum memory required by the FORmat function is 8500 more bytes for formatting a Double Density Standard Drive. The FORmat function uses the Disk Buffer and the Macro buffer in addition to the extra memory required.

The user should be aware then, that the FORmat function DESTROYS THE DISK BUFFER AND ALL MACROS RESIDENT at the time of the formatting.

2.17 GED GET AND DISPLAY BUFFER (BOTTOM)

The GEt and Display buffer function is a short form of the GET function. This function allows the user to get a sector from the selected disk and display the bottom half of the disk buffer.

GED-<Track>,<Sector>

2.18 GET GET TRACK AND SECTOR

The GET function permits the user to retrieve 128/256 Bytes of data from the selected drive.

GET-<Track>,<Sector>[,Start address]

Where, 'Track' is any valid Track for the unit selected and 'Sector' is any valid Sector for the unit selected. The 'Start address' is optional and, if used, the system will place the data starting at 'Start address'. If the 'Start address' is omitted, the system will place the data in the disk buffer. There is internal retry logic, so that if the drive and controller are experiencing difficulty in obtaining the desired Track and Sector, retrys will be made before the system gives up. In addition, if the Head position has gotten lost, the system will return to a known position (Track 0) before attempting to seek to the desired Track and Sector.

2.19 HOM HOME CURSOR (DIGITAL GROUP TVC64)

The HOMe function aids the user in formatting the output to any desired line. Note that no screen erase is performed.

HOM

2.20 HEL HELP

The HELp function aids the user in remembering the entire set of three letter functions available.

HEL

This function erases the screen and then displays all of the present system functions.

2.21 HEX HEXIDECIMAL OUTPUT

The HEXidecimal function instructs the system to output all numerical data in Hexidecimal format.

HEX

2.22 INP INPUT DATA

The INPut function is similar in many ways to the EXAmine function. It permits the user to view the contents of any Input Port. Multiple, arbitrary ports may be viewed by placing their port numbers sequentially within the operand field. Each time an INPut instruction is executed, the byte value of the data at that particular input port is stored internally. This value may be recalled with a subsequent OUTput instruction.

INP-<Port>[[,Port]...]

For example:

INP-3,4,,377

will generate the following output:

```
PORT 003  = 000
PORT 004  = 376
PORT 000  = 104
PORT 377  = 377
```

Upon completion of this processing, 377, which was read from port 377, remains in the internal input holding register. There exists a conflict between the keyboard control structure (the Control-C/S/Q) and the use of the INP function with this port. This is because the keyboard port is scanned frequently looking for the above control characters. Thus, it is not possible to obtain a character from the system input device by the use of the WAI...INP method. For this purpose, a special command, CHAracter, has been defined.

2.23 LIS LIST MACRO INSTRUCTION

The LISt macro instruction function permits the user to review the contents of any one macro instruction or optionally all eight macro instructions.

LIS-[Macro]

If the operand macro number is omitted, all eight macro instructions will be listed, with their text proceeded by the macro number followed by a colon. For example:

LIS-3

might produce the following output:

3: INP-50,51,52,53,54:OUT-0:GED-0,1:ERA:CON

2.24 LOA LOAD MACRO INSTRUCTION

The LOAd macro instruction function permits the user to enter text into any of the eight macro instructions (0-7). The text entered is placed in the macro instruction block, overwriting the previous macro instruction, if any. Note that the LOAd function must exist either by itself or at the end of a line of code, as it uses the entire direct input buffer for input editing.

LOA-[Macro]

For example, to load the macro numbered 3, type the following:

LOA-3

will cause the system to type:

ENTER MACRO INSTRUCTION
>GET-0,1:STA:PUT-0,1:SEL-1:GET-0,1:PUT-0,1:STA:CON

at this point the user types in the text of the macro instruction. The macro instruction is NOT executed at this time. If no operand field is present the system will load macro zero. See also the MACro run function.

2.25 MAC MACRO RUN

The MACro run function starts execution of a set of system commands beginning at the start of the text corresponding to the selected macro number. It is, in effect, an unconditional branch.

MAC-[Macro]

Note that this instruction may be placed within macro instructions, permitting the instructions to be chained together. This allows for some elementary programability. If the operand is omitted, macro zero will be executed.

2.26 MES MESSAGE

The MESsage function allows the user to comment macros when desired. This is usefull when the user is performing a series of diagnostics upon the selected disk.

MES-[ASCII Text]

The 'ASCII Text' may contain any valid ASCII printable character except the colon ":". The colon is the command separator and cannot be included in the text. Also, the slant-bar is used to represent the carriage return code. For example:

MES-// Thru with test 7//:MAC-4

will produce:
(cr)
(cr)
Thru with test 7 (cr)
(cr)

and then proceed to macro four. Note that if the operand is omitted the system will produce a simple carriage return.

2.27 NEW INITIALIZE STORAGE

The NEW function clears the subroutine and macro buffers.

NEW

Note that this function has NO effect on the Disk Log Table.

2.28 NEX SET-NEXT LOOP

The NEXt function is used in conjunction with the SET function. It permits a CONtinue function to be executed if the internal loop counter holds a non-zero value. The internal loop counter is decremented by one for each pass through the loop. If the internal loop counter is zero, the NEXt instruction will be skipped.

NEX

For example, if the previous SET instruction placed the loop counter at four, and the following statement is entered:

OUT-0,301:OUT-0,0:NEX

the following processing will take place:

The ASCII character "A" will be put to the TVC64 exactly four times.

2.29 NUL NULLS AFTER CR/LF

The NULl function permits the user to set the number of null characters to be sent after a carriage return/line feed operation. This accomodates slow printers and fast video devices. If the operand is omitted, zero nulls are printed. The software driver implemented for the TVC64 has no internal delay loops to slow down the scrolling. Therefore, this function is the only way to slow down output to the TVC64.

For example:

NUL-4

sets the number of nulls as described to four.

NUL

Would set the number of nulls to zero.

2.30 OCT OCTAL OUTPUT

The OCTal function instructs the system to output all numerical data in STRAIGHT OCTAL format.

OCT

2.31 ONE WRITE ONES TO DISK

The ONE function writes all one bits(2F) on the selected disk. This function expects that the desired Track is under the head prior to the command.

ONE

Note that this function WILL DESTROY all data on this Track. The ONE function enables the user to perform head symmetry adjustments on some drives.

2.32 OUT OUTPUT DATA

The OUTput function permits the user to send data to an arbitrary set of output ports. The operand field contains a port number, followed by a byte value, followed by a port number, byte value, and so on, with the operand data occurring in pairs.

OUT-<PORT>,<VALUE>[[,PORT][,VALUE]...]

Note that if the data expression is omitted, the system will recall the most recent results of the INPut or CHAracter function, and use that data for the data segment of the OUTput operand field. For example:

OUT-3,4,5

causes the following action: Data 004 is sent to output port 3. The most recent result of an INPut or CHAracter function is then sent to port 5. Note that multiple port/data pairs may exist, but when the data experssion is omitted (in order that the internal INPut register be recalled), there obviously can exist no further port/data pairs.

2.33 PIN PING DIAGNOSTIC

The PINg function is a disk diagnostic. Its function is to check out the track seek and settle logic. It performs a read of every sector twice. The order is as follows with tr= Track, se= Sector, n= last Track, m=last Sector:

[tr=0,se=1],[tr=n,se=1],[tr=0,se=2],[tr=n,se=2],...

[tr=1,se=m],[tr=n-1,se=m]...[tr=n,se=m],[tr=0,se=m]

This is a non-destructive read process. NOTE: It is not recommended that this test be run for long periods as it could cause STEPPER MOTOR FAILURE due to excessive heat buildup. Expect this test to take up to 15 minutes to run.

2.34 PUT PUT SECTOR

The PUT function is the complement of the GET function. It places 128/256 bytes of data on an arbitrary track and sector of the selected drive. Again if the optional 'address' is missing, the system will use the contents of the disk buffer for the data.

PUT-<Track>,<Sector>[,address]

Internal retry logic is similar, except that the data field is not reread when write operation is being performed.

2.35 RAT STEP RATE

The RATe function sets the step rate of the selected drive. This is the bottom three bits of the disk command word. See the Appendix on step values for specific rates.

RAT-<number>

2.36 RES RESET DISK LOG TABLE

The RESet function clears all entrys to the Disk Log Table to zero. Note that the Disk Log Table is NOT affected by the NEW function. See also the STAtus function.

RES

2.37 RET RETURN TO OPSYS

The RETurn functions purpose is to exit the system and return to the Suding Operating System.

RET

2.38 RND RANDOM READ/WRITE

The RaNDom read/write function is a disk diagnostic. Its purpose is to check the controller and drive read/write electronics. It performs 100 Decimal random seek/read/write operations.

RND

NOTE: It is not recommended that this test be run for long periods as it could cause STEPPER MOTOR FAILURE due to excessive heat buildup. This test could take up to 5 minutes to run.

2.39 SEL SELECT DRIVE

The SElect drive function permits the user to operate the system on more than one drive. The initialize routine must have been executed for this function to operate properly, that is, the standard entry point (OPTION 5) must have been entered before this function is used. Drives are numbered from 0 to 3.

SEL-<Drive number>

2.40 SET SET LOOP COUNTER

The SET loop counter function allows the user to initialize the internal loop counter, to be used in conjunction with the NEXt function.

SET-[Loop Count]

The operand, 'Loop Count', may be omitted. If this is done, the system will recall the most recent previous loop count and substitute that for the missing operand. For example:

SET-6

will cause the next statement line terminated with the NEXt function to be executed only six times before skipping the NEXt function. After the loop has been executed, the internal counter will hold a value of zero. Then, executing a simple:

SET

function, causes the internal loop counter to be restored to a value of six. It is not recommended that the SET function occur in the same line as a NEXt function (where the SET occurs BEFORE the NEXt) as this will generally cause an infinate loop. It is acceptable to place the SET function AFTER a NEXt function in order to initialize the loop counter in preparation for the following command statement line.

2.41 SPL SPLIT/OCTAL MODE

The SPLit/octal function instructs the system to output all numerical data in split/octal format.

2.42 STA DISK STATUS DISPLAY

The STAtus function permits the user to view the Disk Log Table. This table reflects all activity related to the performance of the disk drives since the last RESet function was performed or the option 5 entry point was executed.

STA

2.43 SUB SUBROUTINE CALL

The SUBroutine function consists of two subfunctions. These are selected depending upon whether or not there is an operand field.

SUB-[data][[,data]]

The SUBroutine function with no data causes the system to CALl the subroutine buffer area. The SUBroutine function with data in the operand field causes the data to be placed in the subroutine buffer. An automatic Return instruction is inserted into the buffer after the last data element has been entered. The SUBroutine buffer is 64 bytes long.

For example, if the user wishes to send a single letter "A" to the TVC64 the following SUBroutine could be entered:

SUB-076,301,323,0,227,323,0 (cr)

For this Subroutine to be executed the user need only type:

SUB (cr)

2.44 TRK SEEK TO TRACK

The seek to TRacK function permits the user to get the head of the selected drive over some specified Track. This may be useful performing a "C.E. Alignment" or "Symmetry Adjust" function. Once there, this function performs repeated seeks to assure that the head stays loaded. To exit this function the Control C key should be used. If there is no usefull data on the diskette in use, be sure to set the RATe function for no verify.

TRK-<Track>

'Track' is any valid Track number for the selected drive.

2.45 TRY SET RETRY COUNT

The TRY function sets the internal disk error retry count to a specific value.

TRY-<number>

Where 'number' is any valid number up to 255 decimal. Note that the controller retrys five times by itself so, the number entered by the TRY function should be multiplied by five to derive the actual number of attempts. If a seek error is encountered and a restore (or recalibrate) is performed, the system will only retry 'number' times. The RETRY count is initially set to 2 on entry to the monitor.

2.46 VER VERIFY DISKETTE

The VERify function allows the user to check a diskette to see if all the sectors can be read. The monitor will stop on the first sector that causes errors in excess of what the present retry count specifies.

VER

2.47 WAI WAIT FOR DATA

The WAIt function permits the system to pause until a given condition becomes true. The system remains under keyboard control. The Control S/Q/C functions perform while the system is WAIting.

WAI-<Port>,<AND>[,XOR]

The 'Port' is the port number where the data is to be read, the 'AND' is the mask to be anded to the data, and the 'XOR' is the mask to be XOR'd with the data. If the result of this operation is non-zero the WAIting is over. If not, the system will attempt another input from 'Port' and perform the masking all over again. Example:

WAI-54,200,200

Will wait for the top bit (7) to go to a zero before exiting

2.48 ZER WRITE ZEROS TO DISK

The ZERo function writes an all zero data pattern on the selected drive. The function expects that the desired Track is under the head prior to the command.

ZER

Note that this function WILL DESTROY all data on this Track. The ZERo function enables the user to perform head symmetry adjustment on some drives.

APPENDIX A

HOW TO LOAD THE SYSTEM

The system is provided on an 1100 Baud Suding audio cassette. The cassette is loaded as any standard DIGITAL GROUP cassette would be. The first program on the tape is the version using the Single/Double Density Controller routines. The second program on the tape is the version using the Single Density Controller routines. Depressing option 2 will write the entire system to audio cassette. It is recommended that this be done to backup the system tape. The system also expects to see an EPROM at location zero that contains the screen erase feature at location 000 346. The starting address of the load is 001 000. The ending location is 034 000. Start execution address is 005 000.

APPENDIX D

VARIABLE STORAGE AREAS (HMON/2)

The following addresses may be usefull to the user.

LOCATION	LABEL	COMMENT
033 151	POSIT	CURRENT TVC64 COLUMN
026 355	CUN	CURRENT DISK UNIT
026 356	DS0	DEVICE 0 PARAMETER TABLE
026 364	DS1	DEVICE 1 PARAMETER TABLE
026 372	DS2	DEVICE 2 PARAMETER TABLE
027 000	DS3	DEVICE 3 PARAMETER TABLE
027 006	CUNPTR	CURRENT UNIT NUMBER ADDRESS POINTER
027 010	RDWR	READ/WRITE FLAG
027 011	RTRY	CURRENT RETRY COUNT
033 172	DISK	CURRENT SELECTED DISK
033 153	IMS	ONE MILLISECOND CONSTANT
033 335	STACK	STACK AREA (98 BYTES DOWN)
033 363	INPBUF	INPUT BUFFER (256 BYTES UP)
035 063	DBUFF	DISK BUFFER (256 BYTES UP)
034 363	SUBCOD	SUBROUTINE BUFFER (64 BYTES UP)
036 063	MACROS	START MACRO BUFFER (2K BYTES UP)
046 063	FREE	FREE MEMORY STARTS HERE