REMark

Issue 19 • August 1981

Official magazine for users of Heath computer equipment.

on the cover . . . .

Ontario Place in Toronto, Canada

Photo by Gerry Kabelman

on the stack

>CAT

"REMark" is a HUG membership magazine published ten times yearly. A subscription cannot be purchased separately without membership. the following rates apply.

	U.S. Domestic	Canada & Mexico	International
Initial	$18	$20 US FUNDS	$28
Renewal	$15	$17 US FUNDS	$22

Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

Back issues are available at $2.50 plus 10% handling and shipping. Requests for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Send payment to:

Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

HUG Manager and Editor ..... Bob Ellerton
Assistant Editor and
Software Developer ..... Patrick Swayne
HUG Secretary ..... Nancy Strunk
Software Developer ..... Gerry Kabelman
HUG BB ..... Terry Jensen

REMark

SUBMIT Update

In an Editor's note to the article"Another Pointer to the Type-Ahead Buffer" in REMark #13, I explained how to make the old (HDOS 1.6) version of SUBMIT (from 885-1060) work under HDOS 2.0. Because that explanation caused some confusion, I am re-presenting here how to upgrade the old SUBMIT. HUG is currently selling the updated version of SUBMIT, but if your version will not work under HDOS 2.0, you can update it as follows.

First, you will need to put the following files on the disk you are working with: SUBMIT.ASM, HDOS.ACM, ASCII.ACM, CONSL.ACM, and TTIO.ACM. Use your editor to delete lines from SUBMIT.ASM as indicated in Figure 1 following this article. Also change the line MVI M,CR to MVI M,NL. Then use your editor to enter the listing from Figure 2 as STUFF.ACM, putting it on the same disk as the files listed above. Assemble SUBMIT.ASM and you will have your new SUBMIT. It will work on either HDOS 1.6 or 2.0.

PS:

Figure 1. Modifications to SUBMIT.ASM

```
                 TITLE   'LIMITED HDOS SUBMIT CAPABILITY'
                 XTEXT   HDOS
                 XTEXT   ASCII
                 XTEXT   CONSL
                 ORG     USERFWA

Delete  [HDOS.04 EQU     1                  WE'RE USING 50.04.00 VER 1.5

         PROGRAM DB      377Q,000Q          MACHINE CODE ID, TYPE
                 DW      PROG1              WHERE IT STARTS IN MEMORY
                 DW      256                HOW LONG IT IS
                 DW      PROG1              WHERE (PC) SHOULD START

        [PROG1   CALL    SETUP
        |        LXI     H,TABLE
        |        MVI     B,0
        |CHRCNT  EQU     *-1                # CHARS TO STUFF
        |STUFF   MOV     A,M
        |        INX     H
Delete  |        PUSH    H
        |        CALL    PUSH
        |        POP     H
        |        DCR     B
        |        JNZ     STUFF
        |        XRA     A
        [        SCALL   .EXIT              AND LET IT TAKE OVER
                 XTEXT   STUFF
Delete  [TABLE   DS      110
        [        DB      0
```

Delete all bracketed lines.

Figure 2. STUFF.ACM

```
* STUFF.ACM

* THIS ROUTINE IS A MODIFICATION OF THE PROGRAM
* BY JAY H. GOLD THAT APPEARED IN REMARK #13.

* OFFSETS TO POINTER LOCATIONS

S.DLINK EQU     040346A
O.LC    EQU     2                   LINE COUNTER
```

```
O.QTPT  EQU     6               QUEUE TAIL POINTER
O.QHPT  EQU     8               QUEUE HEAD POINTER
O.BSTPT EQU     10              BUFFER START POINTER
O.BENPT EQU     12              BUFFER END POINTER

PROG1   EQU     *
        LHLD    S.DLINK
        XCHG                    HIGHDAT IN DE
        LXI     H,O.LC
        DAD     D
        SHLD    LC              STORE COUNTER IN LC
        LXI     H,O.QTPT
        DAD     D
        SHLD    QTPT            STORE TAIL ADDRESS IN QTPT
        LXI     H,O.QHPT
        DAD     D
        SHLD    QHPT            STORE HEAD ADDRESS IN QHPT
        LXI     H,O.BSTPT
        DAD     D
        SHLD    BSTPT           STORE START POINTER
        LXI     H,O.BENPT
        DAD     D
        SHLD    BENPT           STORE END POINTER

* PUT COMMANDS INTO TYPE-AHEAD BUFFER

        LXI     H,TABLE
MOVEM   MOV     A,M             THIS IS NEW STUFF
        INX     H               FROM REMARK 13
        STA     SAVCHAR
        CALL    PUTIN
        LDA     CHRCNT
        DCR     A
        STA     CHRCNT
        JNZ     MOVEM
        SCALL   .EXIT           RETURN TO HDOS

* INSERT BYTES INTO THE BUFFER

PUTIN   PUSH    H               SAVE TABLE POINTER
        LHLD    QTPT            ADDR OF TAIL POINTER
        CALL    $HLIHL          HL NOW POINTS TO TAIL BYTE
        LDA     SAVCHAR         GET BYTE TO PUT IN BUFFER
        MOV     M,A             PUT BYTE IN QUEUE
        CPI     12Q             NEW LINE?
        CZ      INCLP           INC LINE COUNTER IF NEW LINE
        INX     H               INC TO NEXT POSITION IN QUEUE
        XCHG                    PUT ADDR IN DE
        LHLD    BENPT
        CALL    $HLIHL          HL POINTS TO END OF BUFFER
        CALL    $CDEHL          COMPARE END AND POINTER
        CZ      SETHEAD         IF AT END, SET TO START
        LHLD    QTPT            UPDATE TAIL POINTER
        MOV     M,E
        INX     H
        MOV     M,D
        POP     H               RESTORE POINTER
        RET

* GET START OF BUFFER IN DE

SETHEAD LHLD    BSTPT
        CALL    $HLIHL          BUFFER START IN HL
        XCHG                    IN DE
        RET

* INCREMENT LINE POINTER
```

Vectored to page 31

Recovering a Deleted File

By Donald Harton
2313 Covered Bridge Garth
Baltimore, MD 21234

Many of us have said oops, or words to that effect, after a delete command was given when it was found that the wrong file or disk was commanded to be deleted. The following procedure presents a method to recover the file if subsequent SAVE operations have not been performed on the disk. Note that this procedure is only good for single drive systems if the file DUMP.ABS (from HUG P/N 885-1062) is on the disk when the delete is performed since an attempted ONECOPY to the disk will most likely use the sectors made free by the delete.

The procedure to recover a DELETED file,if the disk has not been written to since the delete was performed,is applicable to Version 2.0 but can be applied to other versions with the proper sector numbers.

HDOS deletes the first letter of the name of the file in the system file DIRECT.SYS when a delete is performed. Additionally, the system file GRT.SYS is modified to free the previously used sectors for use.

PROCEDURE:

1. Find the deleted file with the missing first letter by using the program DUMP.ABS, from the HUG Disk VIII (P/N 885-1062), to dump the file DIRECT.SYS on the disk with the deleted file. The deleted letter on the name will be replaced with an FF Hex. Change the FF back to the Hex value of the original letter.

2. Each file in the directory consists of 23 bytes of information with the first byte being the first letter of the name of the file on the disk. Starting with the replaced letter being the first byte, make note of the values at the 17th and 18th bytes of this portion of the directory. The 17th byte is the address of the beginning of the file GRT.SYS (Group Reservation Table) and the 18th byte is the last group location. The GRT.SYS file is used by HDOS to determine used and unused Groups. A Group is equal to two sectors.

3. Use DUMP to view Track 14, Sector 8 (GRT.SYS)

   A. Starting at the address from byte 17 above, view backwards until that address value is found in the field.

   B. Change the value at that address to the value found at the address from byte 18 above.

   C. Change the value at the address from byte 18 to 00.

4. You MUST EXIT DUMP with a CTRL-B to prevent corruption of the directory. This step is necessary regardless of the particular drive that the disk being modified is located.

As with any new procedure, it is highly recommended that you perform this technique on a scratch disk prior to using it on a working disk since experimentation will allow familiarity with the formats used in the system files DIRECT.SYS and GRT.SYS. Changing of the wrong address could result in lost data on the disk.

EXAMPLE PROBLEM:

Let us assume that the file OOPS.BAS was deleted unintentionally and we desire to recover the program. The disk containing DUMP.ABS must be mounted on the other drive if it is not on the disk with the deleted OOPS.BAS. Using DUMP in the FILE MODE, DUMP DIRECT.SYS and pause with CTRL-C at each sector until you see .OPS....BAS.... (NOTE: The first "O" from OOPS is missing) in the ASCII chart on the right of the screen. Make note of the track and the sector since we will have to come back to change the directory in the DISK MODE of DUMP. Now, the DISK MODE of DUMP should be used to display that track and sector on the screen. The portion of the directory being viewed would appear as follows:

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF

0080: FF 4F 50 53 00 00 00 00 42 41 53 00 00 03 00 00  .OPS....BAS.....
0090: 50 64 and so forth                                Pd and so forth
```

The exact locations will not be as shown but the pattern will be the same. Now, change the FF located at the address 80 to 4F to return the first letter of the original name ("O" in OOPS.BAS) and make note of the values at the address 90 and 91. These two addresses are byte 18 and byte 19 of this file name.

Write the change to the disk and when prompted for TRACK and SECTOR, type in 14 and 8. The sector you are viewing is GRT.SYS. A portion of the sector might look like this:

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

0040: 4C 43 44 45 41 47 48 49 00 46 00 50 4D 4E 4F 00
0050: 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60
0060: 61 62 63 64 65
```

Starting at address 50 (byte 17), view backwards through the file until you come to the number 50. In this case it is located at address 4B. Change the value at this address to 65 which is the value presently located at address 64 (byte 18). Now, change the value located at address 64 to 00. This indicates to HDOS the end of that filename.

The Directory has now been restored and the data previously located in the file OOPS.BAS in now restored and can be run.

Exit DUMP with a CTRL-B regardless of the drive being used to make the changes.

SINGLE DRIVE SYSTEM PROCEDURE

Users' of systems with a single drive may recover from a deleted file even if the program DUMP.ABS is not on the disk to be recovered. This is accomplished with an undocumented HDOS feature that allows SY0: to be reset to allow the program DUMP.ABS to be loaded into memory. Briefly, the procedure is as follows:

A. At the command level prompt (>) type in 'SET HDOS STAND-ALONE' (the ' should not be typed, but is used in this article to indicate the commands to be typed).

B. After the warning message is printed and the prompt (>) is displayed, type ' RESET SY0:'.

C. When prompted, replace the disk in SY0: with the disk containing the program DUMP.ABS.

D. At the prompt, type 'DUMP'.

E. After the program loads in from the disk, replace the disk now in the drive with the disk containing the deleted file.

F. When asked for the drive, type '0'.

G. When asked for the track, type '13'.

H. When asked for the sector, type '2'.

You are now looking at the first sector of the DIRECT.SYS file. You should now continue with steps F, G, and H outlined previously (adding one to the sector each time) until the deleted file is located. Once you have located the deleted file, follow the procedure beginning with step 1 omitting the dump of DIRECT.SYS since this has been accomplished.

EOF

Menu-Driven Demo Program

By Gene Sevin
7534 Westlake Terrace
Bethesda, MD 20034

The programming technique illustrated in the following Benton Harbor BASIC program may be of interest to other BASIC novices who like to use Menu-Driven selections for program options. Further, it avoids the rather inelegant procedure of "ENTER (1) FOR ..., etc." by highlighting each menu item in reverse video and using the function keys to select available options for the designated item. Beginners also may appreciate examples of cursor addressing, 25th line labels, and use of the HDOS Type Ahead Buffer to avoid the Carriage Return.

```
00010 REM ***************************************************************
00020 REM *                                                             *
00030 REM *                    MENU-DRIVEN DEMO PROGRAM                 *
00040 REM *                                                             *
00050 REM *       THIS PROGRAM DEMONSTRATES A TECHNIQUE IN E.B.H.       *
00060 REM *       BASIC FOR SELECTING MENU-DRIVEN PROGRAM OPTIONS       *
00070 REM *       UTILIZING  H-19 GRAPHICS, FUNCTION KEYS, CURSOR       *
00080 REM *       ADDRESSING,  THE  25TH LINE,  AND THE HDOS TYPE       *
00090 REM *       AHEAD  BUFFER.  THE USER STEPS THROUGH THE MENU       *
00100 REM *       ITEMS BY MEANS OF THE  SPACE  BAR OR THE RETURN       *
00110 REM *       KEY, AND THEN  MAKES THE  SELECTION  USING  THE       *
00120 REM *       DESIGNATED FUNCTION KEYS.                             *
00130 REM *                                                             *
00140 REM ***************************************************************
00150 E$=CHR$(27)                  :REM =ESCAPE
00160 E1$=E$+"E"                   :REM =ERASE PAGE
00170 P$=E$+"p"                    :REM =ENTER REVERSE VIDEO
00180 Q$=E$+"q"                    :REM =EXIT REVERSE VIDEO
00190 Y$=E$+"Y"                    :REM =ADDRESS CURSOR
00200 L$=E$+"l"                    :REM =ERASE LINE
00210 X1$=E$+"x1"                  :REM =ENABLE 25TH LINE
00220 Y1$=E$+"y1"                  :REM =DISABLE 25TH LINE
00230 E3$=CHR$(7)                  :REM =BELL
00240 E4$=E$+"x5"                  :REM =CURSOR OFF
00250 E5$=E$+"y5"                  :REM =CURSOR ON
00260 E6$=E$+"o"                   :REM =ERASE TO CURSOR
00270 E7$=E$+"A"                   :REM =CURSOR UP
00280 DEF FN C$(R0,C0)=Y$+CHR$(31+R0)+CHR$(31+C0)
00290 DEF FN P(X)=PEEK(X)+256*(PEEK(X+1))
00300 Z-FN P(8422)+6               :REM =HDOS BUFFER QUE TAIL POINTER
```

```
01000 :
01010 REM ************************+++ MENU +++****************************
01020 :
01030 GOSUB 2000                   :REM READ DEMO MENU ITEMS
01040 T$="MENU"                    :REM SCREEN #1 TITLE
01050 C=INT(35-LEN(T$)/2)
01060 PRINT E3$;E4$;E1$;X1$;      :REM MENU LABELS
01070 PRINT FN C$(25,7);L$;P$;"MENU";Q$;
01080 PRINT FN C$(25,63);P$;"END";Q$
01090 PRINT FN C$(6,C+4);T$
01100 FOR I=1 TO N
01110 PRINT FN C$(7+I,C);M$(I)
01120 NEXT I
01130 :
01140 REM ******************+++ MENU SELECTION +++************************
01150 :
01160 FOR I=1 TO N
01170 PRINT FN C$(7+I,C);P$;M$(I);Q$                        :REM HI-LITE SELECTION
01180 GOSUB 4000                                            :REM KEY INTERPRETER
01190 ON S GOTO 1210,3000,1240
01200 REM                                                   :REM NEXT END RUN
01210 PRINT FN C$(7+I,C);M$(I)                              :REM REMOVE HI-LITE
01220 NEXT I
01230 GOTO 1160
01240 :
01250 REM ***********************+++ RUN +++*******************************
01260 :
01270 PRINT E3$;E1$;X1$;FN C$(25,7);L$;P$;"RUN ";Q$
01280 PRINT FN C$(25,63);P$;"END";Q$
01290 PRINT FN C$(10,30);"RUNNING SELECTION #";I
01300 GOSUB 4000                                  :REM KEY INTERPRETER
01310 ON S GOTO 1330,3000,1060
01320 REM                                                   :REM ERROR END MENU
01330 GOTO 1300
02000 :
02010 REM ********************+++ DEMO MENU ITEMS +++**********************
02020 :
02030 N=10 : DIM M$(10)
02040 FOR I=1 TO N
02050 M$="SELECTION #"+STR$(I)
02060 M$(I)=LEFT$(M$,LEN(M$)-1)
02070 NEXT I
02080 RETURN
03000 :
03010 REM ***********************+++ END +++*******************************
03020 :
03030 PRINT FN C$(20,2);P$;"ARE YOU SURE";Q$;
03040 LINE INPUT ;A$
03050 IF LEFT$(A$,1)<>"Y" GOTO 3080
03060 PRINT Y1$;E1$;E5$                                     :REM RESTORE SCREEN
03070 GOTO 9999
03080 PRINT FN C$(20,2);L$
03090 GOTO 1060
04000 :
04010 REM *****************+++ FUNCTION KEY INTERPRETER +++****************
04020 :
04030 IF PEEK(Z)=PEEK(Z+2) THEN 4030
04040 POKE (Z+2),PEEK(Z)
04050 Z0=FN P(Z)-2
04060 IF PEEK(Z0)<>27  THEN S=1 : RETURN
04070 IF PEEK(Z0+1)=82 THEN S=2 : RETURN
04080 IF PEEK(Z0+1)=83 THEN S=3 : RETURN
04090 GOTO 4030
09999 END
```

EDITORS NOTE: The example above shows the use of special features and the ability to select program options. It does, however, display some problems in that it allows any key pushed to be displayed when selecting the menu choice. Additionally,

Vectored to page 32

Basic Printer Information for the Hobbyist

By Robert G. Traub
9731-154 Street
Edmonton, Alberta Canada
T5P 2G4

Printers vary greatly in their functions and features. There are a great number of types to choose from and the choice can be almost impossible. Some very basic points about printers may assist in the selection of a printer for personal use.

The first point to cover is the columns. What is a column? Some printers will print 132 columns, some only 40 columns and some just about everything in between and more. A column is the space occupied by a single character or letter. Consider first, printers with fixed pitch. Fixed pitch means that each character occupies the same amount of space on the page. The common fixed pitch is 10 characters per inch. This standard pitch would allow 85 columns across an 8 1/2 inch wide paper if no room were allowed for margins. That is 10 characters per inch times 8.5 inches equals 85 columns. If the page were allowed to have margins of 5/8 inch on each side, that would leave 7.2 inches. At 10 characters per inch, that would give us 72 columns, and this is the standard print page for TTY type printers (and others). If the printer were to allow 132 columns, then the paper would have to be at least 10 characters per inch divided into 132 columns or a paper width of 13.2 inches. If margins were to be included, that would bring the width of the paper to 15 7/8 inches. Therefore, a common 132 column page would be 15 7/8 inches wide by 11 inches long. Some fixed pitch printers offer the ability to select the pitch at which the characters will be fixed. Common values are 12 characters per inch and 13.5 characters per inch. A bit of math would soon tell us that a printer with a fixed pitch of 13.5 characters per inch could print a standard page with margins with up to 96 characters or columns, while a printer with a fixed pitch of 12 characters per inch could print 87 columns in the same space. The 13.5 character per inch pinters compress the characters much closer together and may be a bit harder to read if not a good quality print head.

The next thing to consider is the type of print. One common type is the dot-matrix print. This type of printer comes in many dot-matrix forms; some may be 5 by 7, some 7 by 9, some 9 by 9 and some even greater. The better quality print will be produced by larger matrix numbers such as 9 by 9. The least expensive of the dot-matrix printers will generally have a standard 5 by 7 matrix print head. This type of dot- matrix printer is satisfactory for general use, but is not intended for word processing or "letter quality" print as it does not have descenders. A descender is the tail of lower case characters such as "p","q","j", "y" etc. Note that the tail of these characters will extend below the base line on a normal typewriter quality printer. On dot-matrix printers, this is not always available, and never on a 5 by 7 dot-matrix. As the number of the matrix increases, so does the price and overall general quality of the printer. The very elaborate dot-matrix printers that are available can rival almost any type of print, but are very expensive and therefore not generally appropriate for hobby applications. Printers that offer fully formed characters as found on a typewriter are best for word processing at a more reasonable cost. Some of the cheaper printers, whether dot-matrix or full formed character type, do not offer lower case characters; again, this may or may not be important to the user. Each printer must be studied in order to determine if it offers lower case characters, descenders, and other special features such as graphics that would be of interest to the user.

This brings us to the question of friction feed or tractor feed. In the case of friction feed, the paper is held in place by a small (one or two) roller that presses against the printer's platen. This is fine in most cases where each line is advanced one at a time by a carriage return, line feed combination, but if the lines were to be advanced an inch at a time by a sudden command, as is the case with the form feed character, the paper would "slip" as the platen first starts its' fast advance. To overcome this problem, the tractor feed type of paper advance system can be used. With this type of printer option, paper can be advance rapidly with the assurance that the paper will start at the same line position on each page or form. One other type of paper feed system is the pin feed; this system is used on TTY printers to ensure that forms such as telegrams will always line up properly. Another feature offered by the tractor feed or pin feed option is the assurance that the printed line is always horizontal with respect to the top and bottom edge of the paper. With friction feed systems, the page can slip slightly one way or the other and the print lines may be at a slight angle with respect to the top and bottom edge of the paper, so the user must be careful when putting paper in this type of feed system. As there are different systems that can be used to feed paper, the choice will depend on the type of work the printer will be required to do. If a lot of forms are going to be filled out, then of course the tractor feed option would be a good choice. If individual letters are the order of the day, then the standard friction feed type of paper advance will serve well.

Briefly we will take a look at the question of BAUD rate. The BAUD rate or just plain BAUD means "BITS PER SECOND". If the ASCII code were taken as a 10 unit code, then the BAUD rate of 1200 would transfer data at a rate of 1200 bits per second divided by 10 units per character for 120 characters or letters per second. If the ASCII code were to be considered an 8 unit code, then 1200 BAUD would represent a rate of 1200 bits per second divided by 8 units per character for a total of 150 characters per second. Many printers will accept data at a rather high BAUD rate, say 9600 BAUD; this is the rate at which the data is transferred into their buffers and not the rate that they will print. The printer may be only able to produce 150 characters per second on paper and therefore the printer's true BAUD rate is 1200 if an 8 bit or unit ASCII code is assumed. The BAUD rate or throughput is then the speed at which the printer can transfer data or information to paper. There are many reasons why faster speed is needed in some cases and not at all needed in other cases. Typical BAUD rates range from a slow 110 BAUD to a fast 9600 BAUD, but be sure to check if the BAUD rate is the rate that the printer will print characters or if it is the rate at which the host computer can send characters to its' internal storage area (buffer).

Some printers you hear about are called "LINE PRINTERS"; a line printer is a special type of printer that will not print each character as it is received, but rather will wait for a complete line and then print the entire line at once, a character at a time. The length of the line that will be printed is determined by the sending of the "RETURN" character, as a return signifies the end of that line in text. Line printers require special handling by the host computer and provisions must be made for "HANDSHAKING". Line printers have buffers to store the data in before it is printed and the handshaking is simply the printer's method of telling the computer when to send more data to the buffer and when to stop sending data as it cannot handle any more at the moment. Printers are generally slower than the host computer, although there are some very fast printers not generally used by the hobbyist.

One other thing you might run across is the term "BI-DIRECTIONAL" printer. What this means is that the print head will print a line from left to right across the paper, advance the line (line feed) and then print the next line from the right side back to the left. The BI-DIRECTIONAL printer requires fewer mechanical parts and movement than does the single direction types and this is one reason for the increased printer speeds. With the conventional type of printer, a carriage return is required in order to bring the print head back to the start of the next line. This takes time and the computers have to send the printer a pad or fill character on order to assure that the head has returned to the far left before it starts printing again. After many many line feeds and carriage returns, the amount of time wasted can be considerable. Therefore, the bi-directional printer, which does not return the head on every line, is capable of greater speeds (throughput). And, since there are fewer mechanical parts on the bi-directional printer to wear out, reliability is increased over the long run.

There are other things to consider in the purchase of a printer, such as whether a warrantee or service contract is available. Second hand equipment often does not come with a service contract. The application the printer will be used for will largely determine the quality of print required, but the cost could be the main consideration for the hobbyist. This article provides some basic information about printers. It is intended to help inform and not to suggest any one type of printer over another.

EOF

CONGRATULATIONS!
Bob Carson, Heath's Director of Customer Service, stumbled across the Heathkit Dungeon Master in Bob Wild's popular DND. This is the first reported "WIN" on HUG's disk (885-1095).

RDT in Review

Self-Relocating Debugging Tool
HUG P/N 885-1092
By Andy Dessler

This program, written by Pat Swayne, is an extremely useful debugging tool for assembly language programs. It is similiar in many respects to the program DBUG included on everyones distribution diskette for HDOS, but this program relocates itself to high memory so the programmer doesn't have to org his program to some ridiculously high number (like 060.000) to make room for the debugger. This in itself makes the RDT more useful than DBUG.

Most of the commands for RDT are similiar to the commands for DBUG, however, RDT has a few extended commands that make it very useful.

RDT allows a user to look at memory in either hex or octal (only these bases are allowed). RDT also allows the user to modify memory in hex, octal, or ASCII. The fact that the user can only display and modify in these bases is one of the few drawbacks that RDT has.

Unlike DBUG, RDT has a command that allows the user to search through portions of memory looking for occurances of one or two byte strings. Again, the user is limited to hex or octal bases. This command is fairly useful and helps locate sections of code that would otherwise take quite a bit of time to locate.

Also, RDT has a command that allows the user to fill entire blocks of memory with any given 8-bit number. This command is of dubious value.

Like DBUG, RDT has commands that allow the user to execute the program in memory while setting breakpoints (points at which the program execution stops and RDT is given control again). However, only two breakpoints are allowed under RDT, as compared to eight under DBUG (most of the time, however, only two are needed). Also, both programs have the power to display and alter the registers, although changing registers under DBUG is a little easier. The single step function under RDT is much more meaningful than the single step function under DBUG. Under RDT, the value of the registers are printed out after every step, while only the program counter (PC) value is printed in DBUG.

Two very interesting commands available under RDT are TSAVE AND TLOAD. These two commands, not available under DBUG, allow the user to load absolute binary programs from tape. I have not yet experimented with these commands because I have my H8-5 board in mothballs and don't want to put it back in my computer.

I have saved the most interesting command for last. The RDT has a mnemonic disassembler built in! This incredibly useful feature almost justifies the entire price of RDT. It allows the user to disassemble memory and examine patches made to the program or hand-entered code. It also allows the user to search for routines in code that otherwise would be difficult to find.

Although I have not had much time to play with it, the RDT seems to be a very useful tool for assembly language programmers. It is infinitely more useful than the DBUG program supplied on your HDOS Distribution Disk. Even better, HUG included the source code to the program with the disk.

NOTE: I am really not very familiar with DBUG, so any mistakes that I have made in referring to the power of DBUG should be treated with kindness and sympathy. Also, there are many other commands that RDT has that I have not yet touched on.

EOF

PS from PS: This review was reprinted from <u>SCALL</u>, the newsletter of the Houston Heath Users' Group (HUGH). Mr. Dessler points out that the only bases available in RDT are hex and octal, but this is better than DBUG, which offers only octal and decimal. Decimal is practically useless for debugging, leaving DBUG with only one useful base.

One use for the ability to fill blocks of memory with a given number is to find out where a program is storing certain data. For example, if you are prying into someone else's editor for which you do not have the source, and you want to find out where it puts text in memory, just fill all free memory not used by the program with a known value. Then run the editor, enter some text, return to RDT, and you can check the memory you filled for any changes.

RDT offers other advantages not mentioned in the review, such as the ability to read from and write to ports, compare blocks of memory, move data in memory, add and subtract in hex or octal/split octal, convert between hex and octal/split octal, and send disassemblies and memory dumps to a printer.

Loosing Weight with HDOS 2.0

NOTE: This article presents techniques that can destroy files and disks if not carefully done. Use caution and back up everything before starting.

One of the few complaints I have heard about HDOS 2.0 is that it takes up too much space on system disks. In this article, I will show you how to accomplish the following:

1. Reduce the size of many system files by two sectors each.

2. Allow system files to be patched by the PATCH program without modifying it.

3. Remove the Lock flag from files that have it.

These items may not seem related to you at this point, but they are. To see how they are related, and how we are going to accomplish these things, I need to explain something to you called the "File History Record". This is an extra sector that is tacked onto the end of most HDOS system files, such as HDOS.SYS or BASIC.ABS. It contains a history of patches applied to that file, and is in the form of coded numbers, checksums, etc. It is this sector that causes the Patch program to ask for a "Patch ID" when you try to patch a system file. It is not loaded into memory when you run the program, and is not required for its operation. It is loaded when the file is copied by PIP or ONECOPY, so copying the file does not get rid of it. If a system file occupies an odd number of sectors on the disk, such as EDIT (17 sectors), it actually uses one more sector (on a standard 5-inch disk) because HDOS allocates space in 2-sector clusters. Therefore, if you can get rid of the File History Record on the end of EDIT, you free up two disk sectors.

The program REDUCE.ASM listed following this article can be used to remove File History Records from system files. It does this by simply loading the file into memory, deleting file from the disk, and writing the file back to the disk. Since it does not load the History Record, the file written back to disk does not contain it. To use it (after you have typed it in and assembled it), you must first remove the write protect flag from the file you wish to reduce. Then give the command SYn:REDUCE SYn:FNAME.EXT, where "n" is the appropriate drive number, and FNAME.EXT is the file name and extension of the file to be reduced. For example, if both EDIT and REDUCE are on your system disk, you would type REDUCE EDIT followed by a carriage return. The default extension for REDUCE is .ABS. The following files each contain an odd number of sectors, and using REDUCE on them results in a savings of two sectors per file:

EDIT.ABS
PATCH.ABS
INIT.ABS
SYSGEN.ABS
TEST47.ABS
ASM.ABS
XREF.ABS
DBUG.ABS

You can also use REDUCE on other system files with the .ABS extension, because after you do you will be able to patch them with unmodified PATCH.ABS.

CAUTION! Do not use REDUCE on any system file that does not have an .ABS extension (.SYS or .DVD) except the two mentioned below. To do so may ruin the file.

There are three other files that can be REDUCEd for 6 more sectors. These are PIP.ABS, HDOS.SYS, and HDOSOVL1.SYS. Each of these has the Lock flag set, which must be removed, and the latter two require other special procedures. The Lock flags can be removed using the DUMP program from HUG part no. 885-1062. The flags are physically located in the file DIRECT.SYS, which always starts at track 13, sector 2 on any standard 5-inch HDOS disk. If you run DUMP and look at track 13, sector 2 of the disk you are working with, you will see something like the printout below.

```
                  Disk SY0:          Track 13          Sector 2

        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  ----------------
0000:  48 44 4F 53 00 00 00 00 53 59 53 00 00 03 F0 00  HDOS....SYS.....
0010:  06 15 01 63 15 C3 16 48 44 4F 53 4F 56 4C 30 53  ...c...HDOSOVL0S
0020:  59 53 00 00 03 F0 00 16 22 02 63 15 63 15 48 44  YS......".c.c.HD
0030:  4F 53 4F 56 4C 31 53 59 53 00 00 03 F0 00 23 28  OSOVL1SYS.....#(
0040:  01 63 15 63 15 53 59 53 43 4D 44 00 00 53 59 53  .c.c.SYSCMD..SYS
0050:  00 00 03 E0 00 2C 31 02 63 15 63 15 50 49 50 00  .....,1.c.c.PIP.
0060:  00 00 00 00 41 42 53 00 00 03 E0 00 34 3C 02 C3  ....ABS.....4<..
0070:  16 C3 16 53 59 00 00 00 00 00 00 44 56 44 00 00  ...SY......DVD..
0080:  03 C0 00 40 4F 02 63 15 C3 16 44 4B 00 00 00 00  ...@O.c...DK....
0090:  00 00 44 56 44 00 00 03 C0 00 50 57 01 63 15 63  ..DVD.....PW.c.c
00A0:  15 45 52 52 4F 52 4D 53 47 53 59 53 00 00 03 A0  .ERRORMSGSYS....
00B0:  00 58 5D 01 63 15 63 15 53 45 54 00 00 00 00 00  .X].c.c.SET.....
00C0:  41 42 53 00 00 03 A0 00 60 65 02 63 15 63 15 46  ABS.....`e.c.c.F
00D0:  4C 41 47 53 00 00 00 41 42 53 00 00 03 A0 00 68  LAGS...ABS.....h
00E0:  69 02 63 15 63 15 4F 4E 45 43 4F 50 59 00 41 42  i.c.c.ONECOPY.AB
00F0:  53 00 00 03 A0 00 6C 75 02 63 15 63 15 45 44 49  S.....lu.c.c.EDI

Modify this sector (Y/N) <N> ?
```

Note that each file name takes up 8 bytes regardless of the name length, with zeros filling in the unused bytes. Similarly, the extension always uses 3 bytes. Following the extension bytes are 12 bytes containing various information about the file. The fourth byte in this field is the flag byte. Use DUMP to patch the flag bytes of HDOS.SYS, HDOSOVL1.SYS, and PIP.ABS to 00. In the example I have shown, you would patch addresses 0E, 3C, and 6A, but the addresses may be different on your disk. Do not patch the flag byte for HDOSOVL0.SYS. Now exit dump and REDUCE those three files. Then run DUMP again and patch the flag bytes of HDOS.SYS and HDOSOVL1.SYS to F0, and the flag byte of PIP.ABS to E0. You could use FLAGS to restore the flags to PIP, but you must use DUMP to restore them to the HDOS files, because there is an extra flag, C for "Contiguous", that FLAGS cannot set. To view this flag in the directory, type CAT/JGL. After you reduce HDOS.SYS and HDOSOVL1.SYS on a disk, you cannot boot it, but can use it as the source disk for a SYSGEN, and the resulting disk can be booted, and will still have the reduced size files.

PS:

```
REDUCE.ASM -- System file size reducer                      HEATH ASM #104.06.00
by P. Swayne  3-Jun-81                                      04-Jun-81  Page    1

                              00003  * THIS PROGRAM REDUCES THE SIZE OF HDOS SYSTEM
                              00004  * FILES BY REMOVING THE FILE HISTORY SECTOR
                              00005  * AT THE END OF THE FILE.
                              00006
                              00007  * EQUATES
                              00008
   000.000                    00009  .EXIT    EQU     0
   000.001                    00010  .SCIN    EQU     1
   000.004                    00011  .READ    EQU     4
   000.005                    00012  .WRITE   EQU     5
   000.010                    00013  .LOADO   EQU     10Q
   000.041                    00014  .CTLC    EQU     41Q
   000.042                    00015  .OPENR   EQU     42Q
   000.043                    00016  .OPENW   EQU     43Q
   000.046                    00017  .CLOSE   EQU     46Q
   000.050                    00018  .DELETE  EQU     50Q
   000.052                    00019  .SETTOP  EQU     52Q
   000.057                    00020  .ERROR   EQU     57Q
   031.136                    00021  $TYPTX   EQU     31136A
```

```
042.200                      ORG     42200A
042.200              START   EQU     *
042.200  041 266 043         LXI     H,EXIT
042.203  076 003             MVI     A,3
042.205  377 041             SCALL   .CTLC           SET CONTROL-C EXIT
042.207  257                 XRA     A
042.210  377 010             SCALL   .LOADO          LOAD FIRST OVERLAY
042.212  332 271 043         JC      ERROR
042.215  076 001             MVI     A,1
042.217  377 010             SCALL   .LOADO          LOAD SECOND OVERLAY
042.221  332 271 043         JC      ERROR
042.224  041 377 377         LXI     H,-1
042.227  377 052             SCALL   .SETTOP         FIND TOP OF USER RAM
042.231  021 366 377         LXI     D,-10           SUBTRACT 10
042.234  031                 DAD     D
042.235  042 377 043         SHLD    MAXMEM          STORE MEMORY END
042.240  377 052             SCALL   .SETTOP         AND SET IT
042.242  332 266 043         JC      EXIT
042.245  041 000 000         LXI     H,0
042.250  071                 DAD     SP              LOCATE STACK POINTER
042.251  061 200 042         LXI     SP,42200A       RESET STACK
042.254  175                 MOV     A,L
042.255  376 200             CPI     80H             FILE NAME ENTERED?
042.257  302 331 042         JNZ     GETFILE         YES, CONTINUE
042.262  315 136 031         CALL    $TYPTX
042.265  012 105 162         DB      12Q,'Error -- No file name entered.',7,212Q
042.326  303 266 043         JMP     EXIT
042.331  021 001 044 GETFILE LXI     D,FNAME         STORE FILE NAME HERE
042.334  006 021             MVI     B,17            17 CHARACTERS MAX (INC. EOL)
042.336  176         GETFLO  MOV     A,M             GET A CHARACTER
042.337  043                 INX     H               MOVE TO NEXT CHARACTER
042.340  376 040             CPI     40Q             SPACE?
042.342  312 336 042         JZ      GETFLO          IGNORE SPACES
042.345  267                 ORA     A               END OF NAME?
042.346  022                 STAX    D               STORE CHARACTER
042.347  312 023 043         JZ      GOTFILE         IF END, GO ON
042.352  023                 INX     D               MOVE TO NEXT LOCATION
042.353  005                 DCR     B               DECREMENT CHAR COUNTER
042.354  302 336 042         JNZ     GETFLO          GET REST OF FILE NAME
042.357  315 136 031         CALL    $TYPTX
042.362  012 105 162         DB      12Q,'Error -- File name too big.',7,12Q+200Q
043.020  303 266 043         JMP     EXIT
043.023  021 025 044 GOTFILE LXI     D,DEFALT
043.026  041 001 044         LXI     H,FNAME
043.031  076 001             MVI     A,1
043.033  377 042             SCALL   .OPENR          OPEN THE FILE FOR READ
043.035  332 271 043         JC      ERROR
043.040  052 377 043         LHLD    MAXMEM          GET MEMORY END
043.043  021 345 333         LXI     D,-LEND
043.046  031                 DAD     D               FIND SPACE AVAILABLE
043.047  104                 MOV     B,H
043.050  016 000             MVI     C,0             BC = SPACE AVAILABLE
043.052  021 033 044         LXI     D,LEND          PUT FILE HERE
043.055  076 001             MVI     A,1
043.057  377 004             SCALL   .READ           READ IN THE FILE
043.061  332 126 043         JC      CHECKRD         CARRY MUST BE SET
043.064  076 001             MVI     A,1
043.066  377 046             SCALL   .CLOSE
043.070  315 136 031         CALL    $TYPTX
043.073  012 105 162         DB      12Q,'Error -- File too big',7,212Q
043.123  303 266 043         JMP     EXIT
043.126  376 001     CHECKRD CPI     1               END OF FILE?
043.130  302 271 043         JNZ     ERROR           IF NOT, BAD READ
043.133  076 001             MVI     A,1
043.135  377 046             SCALL   .CLOSE          CLOSE THE FILE
043.137  072 033 044         LDA     LEND            GET FILE TYPE
043.142  376 377             CPI     0FFH            IS IT MACHINE CODE?
043.144  302 324 043         JNZ     BADFILE         IF NOT, EXIT
043.147  072 034 044         LDA     LEND+1          GET CODE TYPE
043.152  267                 ORA     A               ABSOLUTE BINARY FILE?
```

```
043.153  312 200 043           JZ      ABS                 .ABS FILE
043.156  376 001               CPI     1                   POSITION INDEPENDENT CODE?
043.160  312 166 043           JZ      PIC                 "PIC" FILE
043.163  303 324 043           JMP     BADFILE             WRONG KIND OF FILE
043.166  052 035 044   PIC     LHLD    LEND+2              POINT TO SIZE
043.171  021 006 000           LXI     D,6                 6 BYTES FOR FILE INFO
043.174  031                   DAD     D                   ADD 6 TO SIZE
043.175  303 207 043           JMP     WRITE               WRITE THE FILE
043.200  052 037 044   ABS     LHLD    LEND+4              POINT TO SIZE
043.203  021 010 000           LXI     D,8                 8 BYTES FOR FILE INFO
043.206  031                   DAD     D                   ADD 8 TO SIZE
043.207  174           WRITE   MOV     A,H                 GET HIGH SIZE VALUE
043.210  074                   INR     A                   ADD 1
043.211  107                   MOV     B,A
043.212  016 000               MVI     C,0                 BC = SIZE MULTIPLE
043.214  305                   PUSH    B                   SAVE SIZE
043.215  021 025 044           LXI     D,DEFALT
043.220  041 001 044           LXI     H,FNAME
043.223  377 050               SCALL   .DELETE             DELETE OLD FILE
043.225  301                   POP     B
043.226  332 271 043           JC      ERROR               CAN'T DELETE
043.231  305                   PUSH    B
043.232  021 025 044           LXI     D,DEFALT
043.235  041 001 044           LXI     H,FNAME
043.240  076 001               MVI     A,1
043.242  377 043               SCALL   .OPENW              OPEN FOR WRITE
043.244  301                   POP     B                   RESTORE SIZE
043.245  332 271 043           JC      ERROR
043.250  021 033 044           LXI     D,LEND              POINT TO THE FILE
043.253  076 001               MVI     A,1
043.255  377 005               SCALL   .WRITE              WRITE THE FILE
043.257  332 271 043           JC      ERROR
043.262  076 001               MVI     A,1
043.264  377 046               SCALL   .CLOSE
043.266  257           EXIT    XRA     A
043.267  377 000               SCALL   .EXIT               RETURN TO HDOS

043.271  365           ERROR   PUSH    PSW                 SAVE ERROR CODE
043.272  315 136 031           CALL    $TYPTX
043.275  012 106 151           DB      12Q,'File error --',40Q+200Q
043.314  361                   POP     PSW                 GET ERROR CODE
043.315  046 007               MVI     H,7                 BELL
043.317  377 057               SCALL   .ERROR              REPORT ERROR
043.321  303 266 043           JMP     EXIT                RETURN TO HDOS

043.324  315 136 031   BADFILE CALL    $TYPTX
043.327  012 105 162           DB      12Q,'Error -- File is not machine code.',7,212Q
043.374  303 266 043           JMP     EXIT

043.377                MAXMEM  DS      2
044.001                FNAME   DS      20
044.025  123 131 060   DEFALT  DB      'SY0ABS'

044.033                LEND    EQU     *                   STORAGE AREA STARTS HERE

044.033  000                   END     START
```

EOF

PS from PS: One thing that I did not mention is that when a system file has its File History Record (also called the Patch History Table) intact, PATCH has the ability to change it even if it is write protected, but when you remove the History sector, you must also remove the write protect flag (if any) in order to patch the file with PATCH. If you have modified PATCH as shown in the article "Whither STAT" in REMark #17, you can still patch write protected files with a History sector even though you do not have to enter a Patch ID.

New HUG Software

885-1209 DND Game for CP/M $ 20.00

This is a CP/M version of HUG's popular Dungeons and Dragons game. This game is virtually identical to the HDOS version (885-1093) described in REMark #16. As with the HDOS version, the object of the game is to find the Lord Master of the 50-level Heathkit Dungeon. There is a never-ending supply of Monsters you must fight and other obstacles to overcome as you make your way deeper and deeper into the huge dungeon. One item we forgot to mention in the REMark #16 description is that this is a real-time game. This means that you have only a short time in which to decide what to do. If you wait too long, a monster could appear, or you could get teleported (ZAP!) to another part of the dungeon.

This program requires 64k of memory, an H19 or H89, Heath CP/M, MBASIC 5.2, and two drives. Even though this is a BASIC program, it plays very fast. A Dungeons and Dragons master player told us that it is the best computer implementation of the game that he has seen.

885-1098 H8 Color Graphics .ABS/.ASM $20.00

This is a collection of color graphic software for the HA-8-3 Color Graphic Board for the H8. The following programs are included:

MUSICK -- This program paints a colorful kalaidoscope on your color monitor while playing music through the HA-8-3's Sound Generator. It plays the same song files as the HA-8-2 Music Board except that only 3 channels are available. One sample song is included.

COMPOSE -- This program is part of the software for the Music board. It allows you to enter songs from conventional sheet music and compile them into files required by the MUSICK program.

GLOBE -- This program draws a rotating line drawing of a globe.

BLKJCK -- A standard computer blackjack game with the cards drawn in color.

DOODLE -- A program for the kids that lets them draw simple pictures using the arrow and function keys on an H19.

AFLAG -- This program paints an American flag on the color monitor.

The source listings for all programs is included, as well as all necessary .ACM files. Requires at least 32k, HDOS, and an H8 with the HA-8-3.

885-1099 H8 Color in Tiny Pascal $20.00

These programs show one of the best uses for HUG's Tiny Pascal (885-1086), that is, to write graphic software. This disk includes the following:

STRING -- This program draws interesting line patterns on the screen that resemble "string art". It includes some procedures that may be useful in other programs, such as one to draw a line from any one point on the screen to any other point. NOTE: This program requires the 9918A color processor.

TEST -- This program provides an easy way of determining the values being loaded onto the HA-8-3 board by joysticks and pushbuttons. These values are displayed on the H19 terminal and are loaded into the various graphic and sound chips. The author, Fred Pospeschil, presents in his documentation a proposed standard for joysticks so that software can be easily traded.

JIM -- This is a game (named after Fred Pospeschil's son) in which you must shoot down a Darth Vader like ship as it moves across the screen. There are sound effects, scoring, and a wide range of user selectable speeds. Two versions are provided. One requires a joystick to move and fire the "Phasor cannon", while the other allows you to do it from a terminal.

These programs require at 32k of memory (48k if you want to re-compile them), HDOS, and an H8 with the HA-8-3. Some require an H19. You will need Tiny Pascal if you want to compile the programs.

NOTE: HUG is presenting this software mainly to help you write other color graphic software. The games have little asthetic value when compared to something like DND, but should help you understand how the color board works.

HUG Products List

CASSETTE SOFTWARE (H8 and H88)

MISCELLANEOUS COLLECTIONS

Part Number	Description	Selling Price
885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00
885-1009	Tape I Cassette	$ 7.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation and Program Listings	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation and Program Listings	$ 12.00
885-1026	Tape III Cassette	$ 9.00
885-1036	Tape IV Cassette	$ 9.00
885-1037	Volume IV Documentation and Program Listings	$ 12.00
885-1057	Tape V Cassette	$ 9.00
885-1058	Volume V Documentation and Program Listings	$ 12.00

UTILITIES

885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00

PROGRAMMING LANGUAGES

885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT on Cassette H8 Only	$ 11.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1085	PILOT Documentation	$ 9.00

AMATEUR RADIO

885-1027	Morse8 Cassette H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00

HDOS SOFTWARE (H8 with H17 or H89)

MISCELLANEOUS COLLECTIONS

885-1024	Disk I H8/H89	$ 18.00
885-1032	Disk V H8/H89	$ 18.00
885-1044	Disk VI H8/H89	$ 18.00
885-1064	Disk IX H8/H89	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00

GAMES

885-1010	Adventure Disk H8/H89	$ 10.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV Music H8 Only	$ 23.00
885-1067	Disk XI Graphic Games .ABS and B H BASIC	$ 18.00
885-1068	Disk XII MBASIC Graphic Games	$ 18.00
885-1088	MBASIC Graphic Games	$ 20.00
885-1093	DND Game for HDOS MBASIC and H89 or H8/H17/H19	$ 20.00
885-1096	MBASIC Action Games MBASIC and H89 or H8/H17/H19	$ 20.00

UTILITIES

885-1019	Device Drivers (HDOS 1.6)	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1050	M.C.S. Modem for H8/H89	$ 18.00
885-1060	Disk VII H8/H89 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.	$ 18.00
885-1061	TMI Cassette to Disk H8 only	$ 18.00
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM	$ 25.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1075	HDOS Support Package H8/H89	$ 60.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1079	HDOS Page Editor	$ 25.00
885-1080	EDITX H8/H19/H89	$ 20.00
885-1082	Programs for Printers H8/H89	$ 20.00
885-1083	Disk XVI RECOVER, etc.	$ 20.00
885-1092	RDT Debugging Tool H8/H89 Disk	$ 30.00
885-1095	HUG SY: Device Driver HDOS 2.0	$ 30.00
885-1098	H8/HA-8-3 Color .ABS/.ASM	$ 20.00
885-1099	H8/HA-8-3 Color in Tiny Pascal	$ 20.00

PROGRAMMING LANGUAGES

885-1038	WISE on Disk H8/H89	$ 18.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1059	FOCAL-8 on Disk H8/H89	$ 25.00
885-1078	HDOS Z80 Assembler	$ 25.00
885-1085	PILOT Documentation	$ 9.00
885-1086	Tiny Pascal Disk	$ 20.00
885-1094	HUG Fig-Forth H8/H89 2 Disks	$ 40.00

BUSINESS, FINANCE AND EDUCATION

885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1070	Disk XIV Home Finance H8/H89	$ 18.00
885-1071	SmBusPkg III 3 Disks H8/H19/H89	$ 75.00
885-1091	Grade and Score Keeping	$ 30.00
885-1097	Educational Quiz Disk MBASIC and H89 or H8/H17/H19	$ 20.00

AMATEUR RADIO

885-1023	RTTY Disk H8 Only	$ 22.00
885-1052	Morse8 Disk H8 Only	$ 18.00

H11 SOFTWARE

885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00
885-1033	HT-11 Disk I	$ 19.00

CP/M SOFTWARE (version 1.43 -- ORG 4200H)

885-1201	CP/M (TM) Volumes H1 and H2	$ 21.00
885-1202	CP/M Volumes 4 and 21-C	$ 21.00
885-1203	CP/M Volumes 21-A and B	$ 21.00
885-1204	CP/M Volumes 26/27-A and B	$ 21.00
885-1205	CP/M Volumes 26/27-C and D	$ 21.00

Vectored to page 26

Using the HDOS Type-Ahead Buffer in BASIC

By: Don M. Deck
P.O. Box 989
Lone Pine, CA 93545

The article "ANOTHER POINTER TO THE TYPE-AHEAD BUFFER" by Jay H. GOLD in Issue 13 of REMark lists offset values for the HDOS Type-Ahead Buffer which are based on a location known as HIGHDAT. The offsets are apparently version independent, at least for HDOS 1.6 and 2.0. The address of HIGHDAT is contained in S.DLINK at 040.346 (split-octal) or 8422 (decimal).

L.D. Barron, 1441-B N. Red Gum, Anahiem, CA 92806 developed a package of programs labeled "LOAD-N-GO" designed to automate game and similiar disks to make them "user-proof". Among the programs is a DOCOM.BAS program running in Microsoft or Benton Harbor BASIC that is designed to allow any BASIC program to input commands into the HDOS Type-Ahead Buffer. The original program was based on HDOS 1.6 and the offsets were based on the address of high memory as found in S.HIMEM.

The following is an update of the DOCOM.BAS program which is Version independent , operating on both HDOS 1.6 and 2.0. It is based on the Jay Gold offsets from HIGHDAT. The program is also independent of memory size.

```
10 REM     ****************************************************
20 REM     *                                                  *
30 REM     *                   DOCOM.BAS                      *
40 REM     *              MBASIC OR B.H.BASIC                 *
50 REM     *                 HDOS 1.6 OR 2.0                  *
60 REM     *   This routine allows the user to POKE commands  *
70 REM     *   into the HDOS type-ahead buffer from a BASIC   *
80 REM     *   program.  The effect is the  same as if the    *
90 REM     *   commands had been typed from the console.      *
100 REM    *                                                  *
110 REM    ****************************************************
120 REM    *                                                  *
130 REM    *   ENTRY: C$='COMMAND LIST'.  The string should   *
140 REM    *   be in the  same form  as for submit-files and  *
150 REM    *   docoms (i.e. C$="com1;com2;..comN).  Note use  *
160 REM    *   of ';'  in  place of C/R, and  that the final  *
170 REM    *   ';' will be provided by the routine. Commands  *
180 REM    *   may  be concatenated   with  each  other, and  *
190 REM    *   with CHR$(n) for inserting control characters  *
200 REM    *   , quotes, etc., prior to entrance.  The total  *
210 REM    *   length  of  the  string    cannot  exceed  99  *
220 REM    *   characters.                                    *
230 REM    *                                                  *
240 REM    ****************************************************
250 REM    *                                                  *
260 REM    *   EXIT:   No  exit  unless  an  error  occurs -  *
270 REM    *   remember, the commands will be executed as if  *
280 REM    *   they had been typed from the console.          *
290 REM    *                                                  *
300 REM    ****************************************************
310 REM    *                                                  *
320 REM    *   USES: C$,T0,T1,T2,T3,T4,T5,T6,T7,T8,T9,V,I as  *
330 REM    *   local variables.                               *
340 REM    *                                                  *
350 REM    ****************************************************
```

```
360 :
370 REM         ******************** DEFINITIONS ********************
380 REM C$= COMMAND STRING
390 REM T0= ADDRESS (DECIMAL) OF S.DLINK
400 REM T1= ADDRESS (DECIMAL) OF LINE COUNTER
410 REM T2= ADDRESS (DECIMAL) OF POINTER TO ADDRESS OF NEXT ENTRY= TAIL POINTER
420 REM T3= ADDRESS (DECIMAL) OF POINTER TO ADDRESS OF FIRST ENTRY= HEAD POINTER
430 REM T4= ADDRESS (DECIMAL) OF POINTER TO ADDRESS OF BUFFER START
440 REM T5= HIGH BYTE (DECIMAL) OF ADDRESS TO BE STORED
450 REM T6= LOW BYTE (DECIMAL) OF ADDRESS TO BE STORED
460 REM T7= LINE COUNT
470 REM T8= ADDRESS (DECIMAL) OF START OF BUFFER (ALSO FIRST ENTRY IN BUFFER)
480 REM T9= ADDRESS (DECIMAL) OF NEXT ENTRY AFTER LAST ENTRY IN BUFFER
490 REM V= BYTE (DECIMAL) FROM COMMAND STRING
500 REM I= LOOP VARIABLE
510 REM         ****************************************************
520 :
530 REM This is a 'dummy' command line for MBASIC.  Routine begins at line 1000
540 C$="SYSTEM;CAT;BYE"
550 :
1000 REM START OF ROUTINE - Add the final C/R to the string.
1010 C$=C$+";":T7=0
1020 :
1030 REM Add HDOS pointer offsets to S.DLINK (Reference REMark #13, page 24)
1040 T0=PEEK(8422)+(256*PEEK(8423)):T1=T0+2:T2=T0+6:T3=T0+8:T4=T0+10
1050 :
1060 REM Determine the starting address of the type-ahead buffer.
1070 T8=PEEK(T4)+(256*PEEK(T4+1))
1080 :
1090 REM Move the command string into the HDOS type-ahead buffer.
1100 FOR I=1 TO LEN(C$)
1110 V=ASC(MID$(C$,I,1)):IF V=59 THEN V=10:T7=T7+1
1120 POKE T8+I-1,V:NEXT I
1130 :
1140 REM Put buffer start address into head-pointer location.
1150 T5=INT(T8/256):T6=T8-(T5*256):POKE T3,T6:POKE T3+1,T5
1160 :
1170 REM Put buffer start address + LEN(C$) into the tail-pointer location.
1180 T9=T8+LEN(C$):T5=INT(T9/256):T6=T9-(T5*256):POKE T2,T6:POKE T2+1,T5
1190 :
1200 REM Put number of lines into the line-counter location.
1210 POKE T1,T7:END
```

EDITORS NOTE: The "dummy" argument listed for LINE 540 must be changed to operate under B.H. BASIC to read as follows: C$="BYE;Y;CAT;BYE"
This change demonstrates the same expected results under B.H. BASIC as are obtained under the original program written in MBASIC. As suggested by the documentation, you can change LINE 540 to perform any task that you choose (keep this in mind as you construct the program). The example argument will perform the following sequence:

1. EXIT MBASIC (or B.H. BASIC if LINE 540 is modified)
2. CATalog THE DISK IN SY0:
3. EXIT HDOS TO PREPARE FOR RE-BOOT

EOF

TINY PASCAL PATCH

A bug in Tiny Pascal (HUG part no. 885-1086) has been brought to our attention. This bug, which is in the TRANSLAT.ABS program, causes a crash if you try to compile a program on an H8 with the Extended Configuration Option. Below is a patch that will correct the problem. You should make the patch whether you have the Extended Configuration Option or not. Because there are other versions of Tiny Pascal floating around besides the HUG version, be sure to check that the "Old Data" is as indicated before making the patch.

File TRANSLAT.ABS

Address	Old Data	New Data
64126	042	315
64127	333	320
64130	054	066
66320	patch	042
66321	area	333
66322		054
66323		021
66324		200
66325		042
66326		031
66327		311

PS:

HUGBB Via MicroNET

Many of you (if not all) are having trouble with the HUGBB . . . i.e. duplicate msg #'s, retrieving msg's and who knows what else . . . I have been trying to determine what or who is at fault or the cause of the problems . . . I finally talked to Richard Taylor and he explained many aspects of the HUGBB system that most certainly helped me understand what could be happening . . . He is checking on some of the points he mentioned to me . . . If you want an absolute date when the HUGBB will be "fixed" . . . well, I don't know . . . Give us about a week . . . As you will read in REMark (issue #14), I have been "put" in charge of the HUGBB and I intend to keep (or make) this a "good" thing for you the user . . . Right now I need your patience while I try to coordinate what the HUGBB should be doing to be running properly . . . Wednesday, I spent the whole day monitoring every single message and I found 77 msgs marked (X) . . .HELP. . . SYSOP <TLJ>

Those of you who were members on the Bulletin Board may remember that message which I left to "ALL". That message was dated the 23rd of January and was headed "HUGBB's PROBLEMS". Wow, have things changed since then. WE HAVE FINALLY GONE TO THE NEW HUG SIG!!!!!! Russel Renshaw has completed (for the most part) the new Bulletin Board system which will be used by most of the Special Interest Groups (SIG).

The change was developing just after REMark 18 had been sent to the printers. Now issue 19 is shipped for August so we have been on the new SIG for sometime now. Virtually everyone is pleased with the new Bulletin Board. We all can praise Russel Renshaw for an excellent job of writing the software for the SIG's.

Just because I say that we have moved, this does not mean literally. To get to the HUG SIG, you enter "R HUG" at the MicroNET prompt, just as before. The move was done internally, which means MicroNET moved us to the new software.

I will not do much explaining of how the new HUG SIG works . . It is probably more advantageous if I publish the main menu and the Information option of the system. This will give the new and prospective HUG SIG members a taste of the BB.

```
Command:
 B  - read SYSOP bulletins
 D  - delete message
 E  - exit from HUG
 H  - help (types this list)
 I  - Instructions for using HUG
 L  - leave a message
 M  - HUG membership information
 OP - user changeable options
 R  - retrieve message
      (R, RF, RI, RM, RN, RR, RS)
 S  - scan message headers
 SM - markable scan
 U  - view user login records
 V  - user interests
 X  - program database

Command: I


SIG Instructions

Retrieve Commands:

RF # - Retrieves messages in ascending serial order from number #
RI # - Retrieves an individual message number #
RM   - Retrieves marked messages (marked in SM, see below)
RN   - Retrieves messages left since last on
RR # - Retrieves messages in reverse serial order from number #
RS   - Retrieves messages by searching.
```

NOTE:
To see the main menu, enter at the "Command:" a <CR>. The menu shown is that which members see. As a non-member you will see a short version and be able to only <L>eave a message to me (*SYSOP). For non-members of the HUG SIG, be sure to read the <M>embership option before <L>eaving a message.

Please note that the menu is in alpha-order and the options are not intended to be done from top to bottom. The most difficult but yet useful option is "OP". You may wish to study it very carefully because it can save you time and money by choosing the suboptions that are most beneficial to you.

```
Options while Retrieving Messages (after Option prompt):
C or <CR>   - Continues with next message
D           - Deletes a message sent to you or by you
NS          - No stop between messages unless one is sent to you or by you
RE          - Reply to this message.  Prompts are automatic.  Retrieval
     continues after reply message is left.
T or Crtl-P - Interrupts retrieval and returns you to command level.


Scan Commands:
S #  - Scans message headers in ascending serial order from number #
SM   - Scans message headers and allows you to mark them for RM retrieval.


Leave Command:
L - Leave a message.  Prompts are self-explanatory.


Setting User Changeable Options (OP):
ST  - Stops output between retrieved messages
NS  - No stop between retrieved messages (ST or NS status marked with *)
LL  - Sets terminal line length at number of characters
NL  - Sets number of lines displayable on your terminal
PG  - Sets up output as paged.  Pauses between pages set by NL. Note that paged
     output is normally meaningful only if the STop option is in effect also.
NPG - No paging.  No pauses between output pages. (PG or NPG status marked with
     *).
BR  - (Brief) Suppress typing of possible Commands, Subcomands, or Options.
     This may also be set by the Brief option in DEFALT.
NB  - Not brief.
TWM - At login this option will default to printing any messages that have
        left for your name or User ID, since you were last on.
MWM - This will mark any messages that have been sent to you for Marked
        Retrieval, since you were last on.


Multiple Commands:
Multiple commands can be separated by semicolons.
Example: OP;LL 32;T;RR 9900
     This means do OP command, set line length at 32 characters, return to top
     of main menu, and reverse retrieve from #9900.


Output Formatting:

     At the beginning of an input line:
. (period) - Forces output to begin a new line
.>#        - Moves left margin # of spaces to right
.<         - Moves beginning of line to column 1, regardless of the .|# value.
.>0        - Clears margin set.


Control Keys:
Ctrl-P  - Aborts output and returns user to the command level
Ctrl-S  - Freezes output
Ctrl-Q  - Restarts output after Ctrl-S
Ctrl-O  - Allows user to skip message text while retrieving
Ctrl-C  - Aborts output and allows instant exit.

Other Features:
B - Allows member to read bulletins from Sysop
M - Allows user to read membership information
T - Prints main command list and updates user's record of when last on
U - Allows member to read or search database of user interests
V - Views login record of those accessing the BB
X - Allows member to enter the program database.


Command: E
                                                      *SYSOP  <TLJ>
```

Column Indicators for Your Editor

by Dean K. Gibson
UltiMeth Corporation
24025 Fernlake Dr.
Harbor City, CA 90710

The program listed below can perform two functions that will make editing easier. One function is to number the columns on your H89/H19 screen using the 25th line. Tab stops are indicated on this line by vertical bars ( ). The other function is to load the overlays, which allows the HDOS EDIT program to run much faster when doing disk I/O. You can remove the code that loads overlays if you wish by changing the value of the label LOADOVL. If you use the ASNI mode for terminal function control, you can change the value of the label ANSI to assemble the program for that mode. The ANSI mode is a bit "safer" than the Heath mode because it is less likely that a random string of characters sent to the console would "trigger" one of the terminal functions in the ANSI mode.

This program links to EDIT when it finishes setting up the column indicators, so giving the command PRED (if PRED.ABS is on your system disk) will set up the indicators and overlays, and start EDIT. If you use another editor, insert its name at the label EDITOR. If you use an editor that requires a filename in the HDOS command line such as HUG's ED, enter the filename after PRED (PRED FNAME.EXT). To erase the column indicators, enter PRED ?.

```
PRED.ASM -- A PRE-PROCESSOR FOR EDITORS                HEATH ASM #104.06.00
                                                       11-Jun-81  Page    1

                        * THIS PROGRAM SETS UP COLUMN NUMBERS ON THE
                        * 25TH LINE, THEN LINKS TO AN EDITOR.  IT ALSO
                        * CAN LOAD OVERLAYS, IF YOU WISH.  IT CAN BE
                        * ASSEMBLED FOR HEATH OR ANSI MODES.
                        * ENTER "PRED ?" TO ERASE COLUMN INDICATORS.

                        * BY DEAN K. GIBSON     MODIFIED BY P. SWAYNE

                        * OPTIONS

000.001                 ANSI    SET     1               ASSEMBLE FOR HEATH MODE
                        *                               SET TO 0 FOR ANSI MODE
000.000                 LOADOVL SET     0               LOAD OVERLAYS
                        *                               SET TO 1 TO OMIT LOAD

                        * EXTERNALS

000.000                 .EXIT   EQU     0
000.003                 .PRINT  EQU     3
000.010                 .LOADO  EQU     10Q
000.040                 .LINK   EQU     40Q

                        * PROGRAM

                                LON     I               LIST OPTIONAL CODE
104.200                         ORG     42200Q
104.200  041 000 000    PRED    LXI     H,0
104.203  071                    DAD     SP              GET SP VALUE
104.204  175                    MOV     A,L
104.205  376 200                CPI     200Q            ANY ARGUMENT ENTERED?
104.207  312 236 104            JZ      PRED1           IF NOT, GO ON
104.212  176            GETARG  MOV     A,M             GET A CHARACTER
104.213  376 040                CPI     ' '
104.215  043                    INX     H               INCREMENT POINTER
104.216  312 212 104            JZ      GETARG          IGNORE SPACES
104.221  376 077                CPI     '?'             WANT TO ERASE?
104.223  302 236 104            JNZ     PRED1           IF NOT, WRITE INDICATORS
104.226  041 036 105            LXI     H,ERASE
104.231  377 003                SCALL   .PRINT          ERASE INDICATORS
```

```
104.233  257                  XRA    A
104.234  377 000              SCALL  .EXIT              RETURN TO HDOS
104.236  041 277 104  PRED1   LXI    H,COLUMNS          PRINT COLUMN NUMBERS
104.241  377 003              SCALL  .PRINT
000.000                       IF     LOADOVL
104.243  257                  XRA    A                  LOAD OVERLAY 0
104.244  377 010              SCALL  .LOADO
104.246  076 001              MVI    A,1                LOAD OVERLAY 1
104.250  377 010              SCALL  .LOADO
                              ENDIF
104.252  041 262 104          LXI    H,EDITOR           LINK TO YOUR EDITOR
104.255  377 040              SCALL  .LINK
104.257  257                  XRA    A
104.260  377 000              SCALL  .EXIT              ERROR EXIT
104.262  123 131 060  EDITOR  DB     'SY0:EDIT.ABS',0
104.277  012          COLUMNS DB     12Q
000.001                       IF     ANSI
                              DB     33Q,'[>1h',33Q,'[s',33Q,'[25;1f'
                              ELSE
104.300  033 170 061          DB     33Q,'x1',33Q,'j',33Q,'Y',31+25,31+1
                              ENDIF
104.311  012 061 056          DB     12Q,'1.......¦10.....¦..20...¦....30'
104.351  056 174 056          DB     '.¦......40.......¦50.....¦..60...¦....'
105.017  067 060 056          DB     '70.¦.......',12Q
000.001                       IF     ANSI
                              DB     33Q,'[u',200Q
                              ELSE
105.033  033 153 200          DB     33Q,'k',200Q
                              ENDIF
105.036  012 105 162  ERASE   DB     12Q,'Erasing column indicators'
000.001                       IF     ANSI
                              DB     12Q,33Q,'[>1l',212Q
                              ELSE
105.070  012 033 171          DB     12Q,33Q,'y1',212Q
                              ENDIF
105.075  000                  END    PRED
```

EOF

New Monitors for the H89

by Dean K. Gibson
UltiMeth Corporation
24025 Fernlake Dr.
Harbor City, CA 90710

MRTHEX and MTROCT are replacement monitor ROM's for the Heathkit H89 computer that provide increased HDOS processing speed and additional debugging capabilities. The following features are provided:

1. The monitor clock interrupt processing routine has been rewritten, giving an effective 16% CPU speed increase to HDOS programs.

2. The clock value in memory (TICCNT) has been extended to a four byte counter.

4. Additional monitor debugging commands have been provided to display and alter the registers, to read and write to I/O ports, and to single-step programs, and to restart HDOS.

5. Full compatibility with HDOS and CP/M (ORG 0) has been maintained. Each ROM includes all of the common Heath H89 ROM entry points (except for cassette tape processing).

6. This new monitor ROM is available in either octal (MTROCT) or hexadecimal (MTRHEX) versions (specify which).

Each ROM is available for $50 (CA addresses $53), which includes documentation and shipping. Source code is NOT provided. For more information, write to Dean Gibson at the address above.

EOF

BUGGIN' HUG

Dear HUG,

Persons converting BASIC sources to B.H. BASIC or Microsoft BASIC, be advised that Logic Statements evaluate "True" statements differently. Most BASICs evaluate "False" statements as 0 (zero). "True" statements are evaluated as:

B.H. BASIC	TRUE= 65535
MBASIC	TRUE= -1 (MINUS)
SOME OTHERS	TRUE= +1 (POSITIVE)

A statement such as:

```
X=Q+100*(Q>99)
```

should be recorded if it is contained in the program you are attempting to convert.

Larry Lankston
3475 St. Catherine Street
Florissant, Missouri 63033

Dear HUG,

People interested in expanding the ET-3400/ETA-3400 system might like to know that an intra-connector is available that will allow a "T" connection to be made to the 40 pin output connector. The connector is made by AP Products, part number AP 922576-40-R Model IC-40, and is available from Priority Electronics; 16723 Roscoe Blvd.; Sepulveda, CA 91343 for $9.00. A three foot, forty wire extension that fits into the intra-connector is also available, part number AP 924005-36-R, for $12.99.

Clive Oakes
225 Lisgar Street Apt. 1001
Ottawa, Ontario K2P OC6

Dear HUG,

The following modification to the H-9 Video Terminal will be of interest to those who use their H-9 on a time-share system. The modification forces any incoming lower case characters to appear as upper case, thus eliminating the "garble" you get when the H-9 tries to display lower case. Actually, the information isn't garble at all. You can construct a very logical conversion table to decipher lower case transmissions.

The modification does not effect normal operation (i.e. H-9/H-8 operation) at all. The only problem that might occur is if you attempt to alter the H-9 so it actually displays lower case (remember, this mod only changes lower case to upper case, it does not display lower). If you ever do this modification, be sure to restore your H-9 if you ever require service.

FORCED UPPER CASE:

On the Character Generator Board:

1. Break the foil of pin #22 of the Character Generator Chip (IC205) going to the "feedthrough" hole underneath the center of the chip.
2. Connect a jumper from pin #11 of IC203 to pin #22 on the Character Generator Chip (IC205).

Jon Giorgini
1321 Cherokee Avenue
West St. Paul, MN 55118

Dear HUG,

For owners of the ETA-3400 who, like myself, have found space available for USR programs too restrictive (programs seem to run into trouble if the lower address for Tiny BASIC is raised above 125) here is a suggestion. Lower the upper limit for BASIC and place the USR program above that point. For up to 1024 bytes of machine code, the procedure is as follows:

Enter the MON with DO-1400 and type G 1C00, followed by "BYE". Then type M22, and change 0F to 0B. After entering the machine code starting as 0C00, be sure to re-enter BASIC with "B". If you wish to live dangerously, ignore the above and just put the machine language program in a high address! As long as Tiny BASIC program does not enter this space, all will go well!

Lee Aamodt
Rt. 5, Box 251
Santa Fe, NM 87501

EGG NOG —

Remember way back in Issues 13 and 14 of REMark when we discussed the effects of temperature and humidity on our precious disks (see "The Magic Egg" and "Disk Care - Or Else")? Well, here we go again!

As probably most users' realize, the life of the ole' 40 track drive is limited. Heath has announced plans to release double-sided, double-track, and double-bit density drives in the near future (this fall we hope). And, HUG just released the fine SY: provided by Dean Gibson. This means we are looking at a new set of problems when, not only caring for our disks, but purchasing them as well! YUP! you gotta buy special disks or risk future problems with the data stored on the disk.

From what information I have been able to obtain, it appears that there are two types of mylar used to "build" the disk and both have unique properties as far as humidity is concerned. Humidity again is the bad guy who can cause the "egg" effect to occur. Additionally, one of the two types of mylar is allowed to expand and contract more with changes in long term weather. This means that the disk acts somewhat like a sponge. During periods of sustained humidity, as in summer months, the disk will expand and maintain this shape. However, in winter months, when the humidity is considerably less, this particular type of mylar "dries out" and the disk is capable of shrinking. The amount of this change is, of course, determined by the environment that the disk is stored in or operated in.

To fully understand the effects of the expansion, let us quickly review what a new drive will see when it looks at a disk. The first new term that will be of major concern will be "double-track density". This means that there will be twice the number of tracks (we referred to them as grooves in Issue 14 of REMark) on a single side of the disk. Secondly, the term "double-sided" means that we will have two READ/WRITE heads, one for each side of the disk. And last, the term "double-bit density" means we will be capable of storing twice the data in one single track of the disk. There you have it! Some simple math will tell you that a single drive with all of these new features (and a new controller board for double-bit density) will replace EIGHT of the drives you are probably using now!

OK!...Where does the disk come into play? Well, the one specification that becomes critical is the double-track density. With double-track density, the head size must decrease thus increasing the accuracy required of the track placement on the disk itself. If we were to allow the expansion described, the smaller head-gap would simply not tolerate the situation and errors would abound!

One thing we can do to protect our software from the effects of humidity is to be sure the disks we purchase are designed for double-track density (usually specified for 96 TPI or Tracks Per Inch). This type of disk uses a better grade of mylar with respect to the effects of humidity. Although unnecessary, it may be a good idea to purchase this type of disk even if you intend to stay with the lower capacity drives for a while especially for those disks that you want do retain and do not use too often. The second reason for better disks is a slight change in the oxide coating which will ensure improved performance when and if you go to the double-bit density controller. As suggested earlier, double-bit density enables us to write twice the information on a single track. Therefore, the coating (magnetic oxide) on the disk must be closely controlled for quality to ensure accurate and reliable data transfer.

As a brief summary, disks that are specified for 96 TPI will, for the most part, be made of better stuff. Improved mylar and quality coatings will decrease the number of errors and lost data that we may have if using our older disks that were designed to operate with low capacity drives.

That's it for now! I hope things don't change too quickly or I'll have to figure out what else we can do with an EGG!

BE:

Local HUG News

ELECTRONIC CENTERS SUPPORT RBBS....

The Downers Grove Heath Electronics Center has recently activated their RBBS message only board. Access can be obtained by calling (312) 852-1305. Typing several CARRIAGE RETURNS will put you on the board. The operating hours are listed below:

7PM-8AM MON, TUES, WEDS, & FRI
9PM-8AM THURS
6PM SAT TO 8AM MON

The Heath Electronics Center in Louisville also supports an RBBS that can be accessed via MCI for lower rates should you care to try this one. This board is accessed by calling (502) 245-7811. CARRIAGE RETURN is the terminal identifier for the log-on. Their board hours are listed here:

6PM-9AM TUES, WEDS, & THURS
10PM-9AM FRI
6PM ON SAT THROUGH MON

NOHUG (New Orleans HUG) recently announced that their board will be up soon, but to date we have no further information other than it will be operated from an H-89 using a Corvus hard disk.

Broward Users' Group (BUG) is meeting on the third Thursday of the month at 7:00PM. For further information on the club and its activities, you are invited to attend the regular meetings at 275 SW 27th Avenue; Fort Lauderdale, FL 33312.

OMAHUG currently mails bulletins to approximately 250 individuals. They have scheduled meetings for the third Sunday of each month. The actual meetings begin at 7:30 but the BS begins at 6:30 and all are welcome. The location of the meeting is the Heath Electronics Center located at 9207 Maple Street; Omaha, NE. For further information, just drop a note ATTENTION: OMAHUG in care of the Center.

CINHUG begins its meeting at 6:30 on the second Tuesday of each month. Their highly active group is taking steps to publish the clubs first newsletter, I/O PORT, via the SOURCE. Articles and Ads must be received by the last Saturday of the month to ensure they will be included. The meetings are conducted at the Heath Electronics Center; 10133 Springfield Pike; Woodlawn, OH 45215. For further information, contact Roger Svoboda at the Center by calling (513) 771-8850.

CONHUG (Connecticut HUG) is a newly formed group that meets on the first Wednesday of each month at the Heath Electronics Center in Avon, CT. For additional information contact Tom Carbone by phone at (203) 658-0819. Tom is also very active on MNET and his number is 70300,245 in case you would like to take the modem route.

A local Heath group has recently formed in Abilene, Texas. If you would care to obtain details, contact: AUG; P.O. Box 1651; Abilene, TX 79604; or call (915) 676-1027.

PittsburgHUG meets at the Pittsburgh Heath Electronics Center on the third Tuesday of the month at 7:00 PM. For further information, contact John C. Schultz by calling (412) 793-7681.

The Frazer Heath Users' Group (FUG) meets on the first Saturday of the month at 4:00 PM. Meetings are usually conducted at the Heath Electronics Center in Frazer, PA. FUG indicates that some meetings are held at the Paxon Hollow Country Club, and that members or individuals that are interested should contact the Center by calling (215) 647-5555 for meeting information. For additional information, the official club address is 1641 Princess Anne Drive; Lancaster, PA 17601. The phone number is (717) 397-3146. Anyone interested in forming an auxiliary group which would meet in Lancaster, PA, is requested to contact either Nial Crawford at (717) 299-3836 or Dave Hendrie at (717) 397-3146.

A 35 member group know as the Wright-Patterson HUG has formed in the Dayton, Ohio area. Meetings of W-P HUG are held at 4:00 PM on the first Thursday of each month. These meetings are held at Wright-Patterson AFB. For additional information, phone Jim Moore at (513) 236-4915, or write: W-P HUG; 4110 Spruce Pine Ct.; Dayton, Ohio 45424. Jim is the President Pro Temp of W-P HUG.

Christian L. Wilson is interested in contacting individuals in any branch of the Military Service who currently own an H-89 for the purpose of exchanging ideas and information. He mentioned that he is very interested in software for the auto-answer type modems that would operate on the H-89. You may contact Christian at the following address: Christian L. Wilson, ETC; USS Juneau LPD-10; FPO San Fran, CA 96669.

EOF

Vectored from page 17

885-1206	CP/M Games Disk	$ 21.00

CP/M SOFTWARE (version 2.2 -- ORG 0)

885-1207	TERM and H8COPY	$ 20.00
885-1208	HUG Fig-Forth H8/H89 2 Disks	$ 40.00
885-1209	DND Game for CP/M MBASIC and H89 or H8/H17/H19	$ 20.00

MISCELLANEOUS

885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-4	HUG Binder	$ 5.75

CP/M is a registered trademark of Digital Research Corp.

HDOS Scalls
The Straight Scoop

by John O. Corbett
Box 259
Avoca, NY 14809

The HDOS SCALLs are powerful tools in the hands of the machine language programmer. The ability to manipulate data and files is made easier through the use of SCALLs. In the following paragraphs, I will explain how to build a file using the .CHFLG SCALL, some quirks of the .CHFLG SCALL, analyze a short program using the .CHFLG SCALL, and finally build a FLAG.ACM file. First, what are some necessary elements in any program?

There are several important elements which must be included in every program. When you make a program, it is a good idea to use the .VERS SCALL. By using this SCALL, you will ensure that the program always contains the correct SCALLs for the HDOS operating system resident in RAM. After a file name has been inputted, the program should check to see if the file is actually in the directory. This can be accomplished by opening the file for a read. Next, we find that we must determine if the lock flag has been invoked. This check can be accomplished by trying to set a flag with a zero value in both the B and C registers. See pages 41 and 42 in the HDOS System Programmer's Guide (Models HOS-817-1 and HOS-847-1). If the flag cannot be set, an error code will be generated and the program control can then be directed to an error routine. However, if your HDOSOVL0.SYS file has been patched to unlock the lock flag (as with Jim Teixeira's SYSMOD -- see REMark #14 page 29), this check is unnecessary. The program should also include the normal error checking routine and XTEXT files or definitions used in assembling the file. Now that we have some of the essential program elements, what are some of the quirks of the .CHFLG SCALL?

One .CHFLG SCALL quirk, if the lock flag has not been invoked and you want to delete a flag, placing a 0 byte in the B and C registers then invoking the .CHFLG SCALL will not cause the flags to be deleted. Another quirk, if you load the same bit in both registers (B and C) as indicated in the Guide, you can only add this value to the flag values already in the directory and cannot change the flag from say an S flag to a W flag. For example, if the directory indicates that the W flag has been set on a particular file and you want to change the flag to an S flag, putting a 200Q in both registers will increase the directory value to 240Q indicating that the S and W flags have been set. In an effort to see how the Heath program FLAGS.ABS handled these two quirks, I disassembled the FLAGS.ABS program. I found that if I put a 377Q in the C register and the value of the flag(s) that I wanted in the B register, the previously identified flag(s) could be changed. So, how can I use this information to my advantage?

Let's see how we can put the above information to use and analyze a short program I wrote using the .CHFLG SCALL. From the beginning to the label FLG1A, the program is self-explanatory and should pose little difficulty. So, let's analyze it from FLG1A down to the label BADVER. Since the ending value must be in the B register, I used this register to store the total inputted value. First the program prompts for input, and checks each character entered. If an incorrect character is encountered, the B register is returned to the 0 state and the operator is asked for the correct flag(s). Otherwise, the value in the B register is OR'ed with the value MVI'ed to the A register then MOV'ed to the B register. When a NL (12Q) is encountered, program control is turned over to executing the .CHFLG SCALL, then the program jumps back to the file name inputting routine. Control-C is available if you want to leave the program. The last section to be analyzed is from the label BADVER to the label EXIT. The only way the program will ever get to BADVER is if the version of this program does not match the HDOS version. MVI'ing 50Q to the A register and then executing an .ERROR SCALL will cause "NOT CORRECT VERSION OF HDOS" to be printed on the system terminal. One word of caution when using the .ERROR SCALL. Whatever is loaded into the H register will be printed on the system terminal after printing the error message. This problem can be rectified by putting a LF (12Q) character in the H register before calling the .ERROR SCALL.

This program can be used inplace of the FLAGS program that comes with HDOS, and it takes less disk space. The default extension of .ABS takes into consideration the fact that most of the files you change the flags on have that extension. Now, let's fine tune the same program so it can be used as an .ACM file. First, remove the ORG and END statements, and any definitions that are defined in the main program. Most programs will have an input routine; therefore, you can omit the lines from the label FLG through the line MVI M,0. As previously stated, good programming dictates the use of the .VERS SCALL. Therefore, you can eliminate the lines from BEGIN through JC BADVER. However, the line MVI A,50Q should be included in the version routine before calling the .ERROR SCALL. The program will also probably include some kind of exit, so you can eleminate the next 3 lines. The next two lines allow the program to change flags on itself, so you can remove them if that is not your intention. Most programs will also have an error printing routine, so you can remove the lines from the label BADVER through SCALL .ERROR. In the final analysis, the lines from the label FLG1A through SCALL .CHFLG are necessary and are the main lines to be included in a FLAG.ACM file. JMP FLG should be changed to RET as the final change in this .ACM file. This is so that the routine can be called from the main program loop.

In conclusion, I have shown you how the .CHFLG SCALL can be used in an assembly language program, some quirks of the SCALL, and how to make a FLAG.ACM file.

Since this is the first in a series of articles explaining the use of a particular SCALL, your constructive criticism is appreciated and welcomed. Address your comments to:

John O. Corbett
Box 259
Avoca, New York 14809

```
FLAG.ASM -- .CHFLG SCALL DEMONSTRATION                 HEATH ASM #104.06.00
                                                       05-Jun-81  Page    1

                           *       PROGRAMED BY:   JOHN O. CORBETT
                           *                       BOX 259
                           *                       AVOCA, NY  14809
                           *       MODIFIED BY:    P. SWAYNE

000.000                    .EXIT    EQU     000Q
000.001                    .SCIN    EQU     001Q
000.002                    .SCOUT   EQU     002Q
000.006                    .CONSL   EQU     006Q
000.007                    .CLRCO   EQU     007Q
000.011                    .VERS    EQU     011Q
000.040                    VERS     EQU     20H                 VESION 2.0 IN BCD
000.041                    .CTLC    EQU     041Q
000.042                    .OPENR   EQU     042Q
000.046                    .CLOSE   EQU     046Q
000.055                    .CLEAR   EQU     055Q
000.057                    .ERROR   EQU     057Q
000.060                    .CHFLG   EQU     060Q
031.136                    $TYPTX   EQU     031136A
042.200                    USERFWA  EQU     042200A
                           *
042.200                             ORG     USERFWA
                           *
042.200                    BEGIN    EQU     *
042.200  377 011                    SCALL   .VERS
042.202  332 076 043                JC      BADVER
042.205  376 040                    CPI     VERS
042.207  332 076 043                JC      BADVER
042.212  076 003                    MVI     A,3
042.214  041 125 043                LXI     H,EXIT
042.217  377 041                    SCALL   .CTLC               SET CONTROL-C EXIT
042.221  076 377                    MVI     A,-1
042.223  377 055                    SCALL   .CLEAR              CLEAR CURRENT CHANNEL
```

```
                      *
                      *      INPUT FILE NAME
                      *
042.225                      FLG    EQU    *
042.225  315 136 031         CALL   $TYPTX
042.230  106 151 154         DB     'File Name =',240Q
042.244  041 130 043         LXI    H,FNAME
042.247  377 001      FLG1   SCALL  .SCIN
042.251  332 247 042         JC     *-2
042.254  167                 MOV    M,A
042.255  043                 INX    H
042.256  376 012             CPI    012Q
042.260  302 247 042         JNZ    FLG1
042.263  053                 DCX    H
042.264  066 000             MVI    M,0
                      *
                      *      ENSURE FILE IS IN DIRECTORY
                      *
042.266  076 001             MVI    A,1
042.270  021 153 043         LXI    D,DEFALT
042.273  041 130 043         LXI    H,FNAME
042.276  377 042             SCALL  .OPENR
042.300  332 100 043         JC     ERR               FILE IS NOT THERE
042.303  076 001             MVI    A,1
042.305  377 046             SCALL  .CLOSE
042.307  332 100 043         JC     ERR
                      *
                      *      ENSURE FILE IS NOT LOCKED
                      *
042.312  006 000             MVI    B,0
042.314  016 000             MVI    C,0
042.316  021 153 043         LXI    D,DEFALT
042.321  041 130 043         LXI    H,FNAME
042.324  377 060             SCALL  .CHFLG
042.326  332 100 043         JC     ERR
                      *
                      *      MAIN PROGRAM
                      *
042.331  006 000      FLG1A  MVI    B,0
042.333  315 136 031         CALL   $TYPTX
042.336  116 145 167         DB     'New Flag(s) =',240Q
042.354  377 001      FLG2   SCALL  .SCIN
042.356  332 354 042         JC     *-2
042.361  376 012             CPI    012Q
042.363  312 061 043         JZ     FLG8
042.366  346 137             ANI    137Q              CAPITALIZE ENTRY
042.370  376 123      FLG3   CPI    'S'
042.372  302 002 043         JNZ    FLG4
042.375  076 200             MVI    A,200Q
042.377  303 054 043         JMP    FLG7
043.002  376 114      FLG4   CPI    'L'
043.004  302 014 043         JNZ    FLG5
043.007  076 100             MVI    A,100Q
043.011  303 054 043         JMP    FLG7
043.014  376 127      FLG5   CPI    'W'
043.016  302 026 043         JNZ    FLG6
043.021  076 040             MVI    A,40Q
043.023  303 054 043         JMP    FLG7
043.026  315 136 031  FLG6   CALL   $TYPTX
043.031  007 111 154         DB     007Q,'Illegal Flag',212Q
043.047  377 007             SCALL  .CLRCO            CLEAR CONSOLE
043.051  303 331 042         JMP    FLG1A
043.054  260          FLG7   ORA    B                 COMBINE FLAGS
043.055  107                 MOV    B,A
043.056  303 354 042         JMP    FLG2
043.061  016 377      FLG8   MVI    C,377Q
043.063  021 153 043         LXI    D,DEFALT
043.066  041 130 043         LXI    H,FNAME
043.071  377 060             SCALL  .CHFLG
043.073  303 225 042         JMP    FLG
```

```
                    00105   *
                    00106   *       ERROR HANDLING
                    00107   *
043.076  076 050    00108   BADVER  MVI     A,50Q
                    00109
043.100  365        00110   ERR     PUSH    PSW
043.101  315 136 031 00111          CALL    $TYPTX
043.104  105 122 122 00112          DB      'ERROR - ',200Q
043.115  361        00113           POP     PSW
043.116  046 012    00114           MVI     H,12Q
043.120  377 057    00115           SCALL   .ERROR
043.122  303 225 042 00116          JMP     FLG
                    00117   *
                    00118   *       RETURN TO HDOS
                    00119   *
043.125  257        00120   EXIT    XRA     A
043.126  377 000    00121           SCALL   .EXIT
                    00122
043.130             00123   FNAME   DS      19
043.153  123 131 060 00124  DEFALT  DB      'SY0ABS'
                    00125
043.161  000        00126           END     BEGIN
```

EOF

Russian Roulette for the ETA-3400

```
100 REM RUSSIAN ROULETTE GAME
110 REM BY THOMAS R. SCHUYLER, SEPTEMBER, 1980
120 REM FOR ET/ETA3400 TINY BASIC
130 PR" THIS IS A GAME OF RUSSIAN ROULETTE.  WE ARE"
140 PR" PLAYING WITH A SIX-SHOOTER REVOLVER.  FOR"
150 PR" YOUR TURN, YOU CAN EITHER STOP (TYPE 'S')"
160 PR" OR CONTINUE (TYPE 'C')."
170 S=0
180 C=1
190 N=0
200 Y=1
210 PR
220 PR
230 L=RND(6)+1
240 PR" I HAVE JUST LOADED A BULLET INTO ONE OF"
250 PR" THE CHAMBERS.  BEING A GENTLEMAN, I WILL"
260 PR" OF COURSE LET YOU HAVE THE FIRST TURN."
270 PR
280 PR" YOUR TURN, PLEASE";
290 INPUT A
300 IF A=1 GOTO 370
310 PR
320 PR" CHICKEN!!!  IS THERE ANYONE ELSE"
330 PR" (WITH GUTS) WHO WOULD LIKE TO PLAY";
340 INPUT A
350 IF A=1 GOTO 210
360 END
370 GOSUB 500
380 IF F=L GOTO 700
390 PR" LUCKY!  NOW IT'S MY TURN."
400 GOSUB 500
410 IF F<>L PR" W H E W ! ";
420 IF F<>L GOTO 280
430 PR
440 PR
450 PR
460 PR" BOY, DOES THAT SMART!!!"
470 PR" HOW ABOUT ANOTHER (IF YOU'LL EXCUSE THE PUN) ROUND";
480 GOTO 340
500 PR
510 F=RND(11)/2+1
```

```
520 PR"      - - - - W H I R R - - - - ";
530 I=20
540 I=I-1
550 IF I>0 GOTO 540
560 IF F=L GOTO 600
570 PR"CLICK! CHAMBER ";F;" WAS EMPTY."
580 PR
590 RETURN
600 PR"BANG!"
610 PR
620 RETURN
700 PR" I'M SORRY.  MY CONDOLENCES WILL BE SENT"
710 PR" TO YOUR FRIENDS.  YOUR INSURANCE WILL BE"
720 PR" PAID TO MY OWNER.  ANYONE ELSE CARE TO PLAY";
730 GOTO 340
```

Vectored from page 4

```
INCLP   PUSH    H
        LHLD    LC              POINT TO LINE COUNTER
        INR     M               INCREMENT IT
        POP     H
        RET

* STORAGE AREA

CHRCNT  DB      0
SAVCHAR DB      0
LC      DW      0
QTPT    DW      0
QHPT    DW      0
BSTPT   DW      0
BENPT   DW      0

* TABLE OF COMMANDS TO BE INSERTED INTO BUFFER

TABLE   DS      110
        DB      0

                                          EOF
```

Changing your address? Be sure and let us know since the software catalog and REMark are mailed bulk rate and it is not forwarded or returned.

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

IS THE INFORMATION ON THE REVERSE SIDE CORRECT? IF NOT FILL IN BELOW.

Name ____________________

Address ____________________

City-State ____________________

Zip ____________________

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

RENEWAL RATES		NEW MEMBERSHIP FEE IS:
US DOMESTIC	$15 ☐	$18 ☐
CANADA	$17 ☐ US FUNDS	$20 ☐
INTERNAT'L*	$22 ☐ US FUNDS	$28 ☐

* Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

Source Update

I have to apologize for some misleading information presented in REMark Issue 17. It appears that I did not have the entire picture when I developed the previous article. Here is some additional cost information. There is a minimum monthly charge for the use of STC (SOURCE TELE-COMPUTING). A $10.00 minimum is charged whether you use the system or not (the charge is not applied if you average a little over two hours per month). Further, an additional charge of 1.6 cents for 2K of storage area per day (at midnight) is tacked on to the monthly minimum charge. This means that if you were to store what would be equal to one five inch disk (HDOS) on the system for a month your bill could be as high as $25.00 without ever getting "ONLINE".

RUMORS...

We have heard through some reliable sources that MNET is looking into a monthly minimum charge. Let's get our group together for some lobbying on this topic by using "FEEDBACK". Let's get some group comments on the subject.

The purpose of supplying information on both systems is to keep you informed as to the direction of telecommunications. HUG does not favor either system as some people feel. Both MNET and SOURCE have their advantages and disadvantages,and it should be noted that we have users' on both, either due to locale or choice. HUG supports both systems as a "pure service" to you and to keep abreast of what we feel could very well be a future item similiar to the telephone today.

We at HUG wish to extend our thanks to all users' of the HUGBB's for their continued support and interest. If there is anything that you feel we can do to improve the BB services, please do not hesitate to let us know!

CAPT.B (BE:)

Vectored from page 8

rapid key strokes will produce unwanted characters to appear on the screen while in the menu mode. The program is intended to demonstrate handling techniques that could be used if you desire and is not intended as a finished debugged item. Also, the function keys used are "f1" for selecting the MENU or RUN modes (acts as a toggle) and the "GRAY KEY" to select END of program. The "SPACE BAR" is used to step through the various choices (1-10).

EOF

HUG BUG: In "Using the Epson MX-80" on page 3 of REMark #18, "JSR" should be changed to "JRS". The LPH24 driver should be used with the MX-80, and you should SET LP: LENGTH 13 to prevent the letter "B" from being typed when the driver is opened. The interface cable must be male at both ends.