REMark

Issue 13 • December 1980

Official magazine for users of Heath computer equipment.

on the cover . . . .

Yes, Virginia, there is ORG-0 CP/M for Heath 8-bit computers.

CP/M® is a registered trademark of Digital Research Corp.
Illustration by Ray Massa.

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
Software Developer .....Jon Falkner

REMark

"REMark #2 Is Due in 10 Days!"

It was almost three years ago. I'm standing in my office surrounded by hundreds of cassette based programs and some paper tape stuff too, all of which need to be prepared for distribution "yesterday"! Plus, REMark #2 consists of 28 empty pages. But this job is going to be a "skate", to borrow a line from the kids. How big a deal can it be to publish four magazines a YEAR? It's gotta be fun to play with software all day and dub it to another cassette. There can't be many phone calls or much correspondence, can there? They said that if you accept this new job as HUG manager, REMark 2 must be ready for production in ten days! O-o-o-o-o-K. REMark got out the door and all those cassettes were turned into one. By the way, that cassette (885-1008) has 82 programs on it (some tacky) and is still one of the best selling HUG products for reasons I don't understand. We had 2300 members. All mad. But, in a few days, Sue came on the scene and bailed us out of a mountain of paper work and membership details.

In a few weeks Gerry joined us and I pulled the same trick on him. "We need to produce another cassette. QUICK! But nothing to it. Takes no time at all. Fun. You'll love it," I told him. He swallowed it! He took a cut in salary to join this madness! So Gerry, Sue and I plug along for a year and pretty much get ahead of you (or so I thought) and then Gerry starts hinting that his wife is questioning to whom is he married, since he is spending nights and weekends on HUG business. Funny thing. I was hearing the same tune a home. So... enter a guy that parted company with the Louisville Heathkit Electronic center. "Good guy... shouldn't let him get away." They said. "I can have him here for an interview tonight." I go... "Are you crazy??? I'm leaving in the morning for China and don't even have a tooth brush packed." (They don't have tooth paste in China. Therefore, don't need tooth brushes.) So anyway, I reluctantly meet Jon Falkner at the hotel at nine that night and he comes up with all the right answers. So I phone from O'Hare airport my decision. And he has been cranking out new disks, contributing to the magazine, answering your goofy questions on the phone and your correspondence all the while leading me into a life of false security.

Meanwhile, the results of that survey we took a year ago are bugging HUG and I start looking around for someone to take over the magazine and turn it into a monthly happening. Concurrently, I'm looking for a software guru for SOFTSUFF. As a result of the SOFTSTUFF guru search, Pat Swayne shows up in my office one morning and reminds me he is the guy that wrote FOCAL for Heath Cassette system AND, as I remember, it was not only well written, but very well documented. AH HAH! Two for the price of one.five! Well, HUG's gain is SOFTSTUFF's loss. You're looking at Pat's effort. And, meanwhile he has put FOCAL and his Tiny BASIC on disk.

Then, one day, Sue starts getting grouchy. (She does that alot.) But I don't understand. She's only secretary to me and all of the HUG staff, handles the 300 or more HUG membership applications per month, processes too many address changes a month, plus trys to keep up with me in foreign places.

Meet Nancy. The first thing she did was help Sue organize an expensive dinner party to show me the door! Swell kid. She is hereinafter wholly responsible for any mistakes on your membership status, typos in all correspondence, (even if she didn't type it) and all busy signals when you call.

Meet Bob Ellerton. Your new HUG manager. He's well over six feet tall and surpasses that in dedication and character. He's been involved in the computer product line from the day he and I first tried to get the proto H8 to say "your H8 is up and running". But it increased our vocabulary.

So, Bob, here's what you have to deal with.... A hellava great staff... and over 10,000 HUG members that have made me tick. I hope that you will have the luxury of staring out the window and dreaming about the future... and above all, making the members' investment in HEATH hardware rewarding!

Incidentally Nancy, I've never taken the time. Can I be a HUG member?

JB:

Home Control with the ET3400

By Michael C. Frieders
9318 Clanbrook Ct.
Fairfax, Va. 22031

INTRODUCTION

Ever since I bought my H8 computer my Heathkit ET-3400 microcomputer has been sitting in the closet, neglected. I have always thought that I would like to have the ET-3400 employed in some control application that I would not want to have the H8 system tied up with. One such control application is home control with the BSR Home Control System. The ET-3400 is ideally suited to this task, and a number of articles describing computer interfaces to the BSR system have appeared recently. One article, by Mark Garetz, was published in the Nov. 1979 Creative Computing, and is the interface used here. Another article by Steve Ciarca appeared in the January, 1980 Byte, and is primarily aimed at the S-100 bus. Part I of this article will cover the hardware aspects of the ET-3400/BSR system. Part II will feature software for the system.

PART I: HARDWARE

The BSR system consists of a command console, a cordless controller, lamp modules (up to 300 watts), appliance modules (up to 15 amps), and a wall switch module. It transmits commands over the AC lines in the house, and therefore is very convenient. The BSR system is available in two models, one with and one without the ultrasonic remote control ability. Most retail stores sell only the non-ultrasonic version because it is less expensive. The ultrasonic version, which is necessary for this application, is available nationwide from Sears as follows:

Ultrasonic Command Console	#34H5456	$44.99
Cordless Controller	#34H5457	23.99
Lamp Module	#34H5458	15.99
Appliance Module	#34H5459	15.99
Wall Switch Module	#34H5461	19.99

I was able to buy the command console and cordless controller from Sears on sale for a total of $53.19. These prices are from the 1980 Fall/Winter Catalogue, page 1028. Discount catalog showrooms in the Washington, D.C. area offer the various BSR modules for under $12.

I will not attempt to duplicate the Garetz article here, so I will not go into great detail about any of the components or the theory behind the design. Basically, the circuit interfaces to the cordless controller, and simulates the pressing of the keys by sending one byte hex codes through a parallel port. The remote cordless controller was chosen for the interface because it operates at -9V and eliminates dealing with the hot ground, 120V circuit in the command console.

The interface parts fall into two categories; those needed to build the parallel port, and those needed for the BSR interface:

PARALLEL PORT	BSR INTERFACE
(1) 6820 PIA	(3) CD4051
(1) 74LS27	(6) 2.7K res.
(1) 7400	(1) 470 res.
(1) 74LS30	(1) .1 cap.
(2) 1K res.	(1) 1N757 zener diode
hook-up wire	(1) 13 wire ribbon cable

The parts for the parallel port are included in the ET-3400 course. The BSR interface parts can be purchased from James for $4.29, or from electronics supply stores for about $6.

The interface circuit is presented in two parts in figure 1. On the left is shown the parallel port, which can be built on the ET-3400 breadboard. This circuit, with software, sets up side B of the PIA as a parallel output port at address 8002. The B side is chosen because of its extra drive capabilities. Only the lower six bits

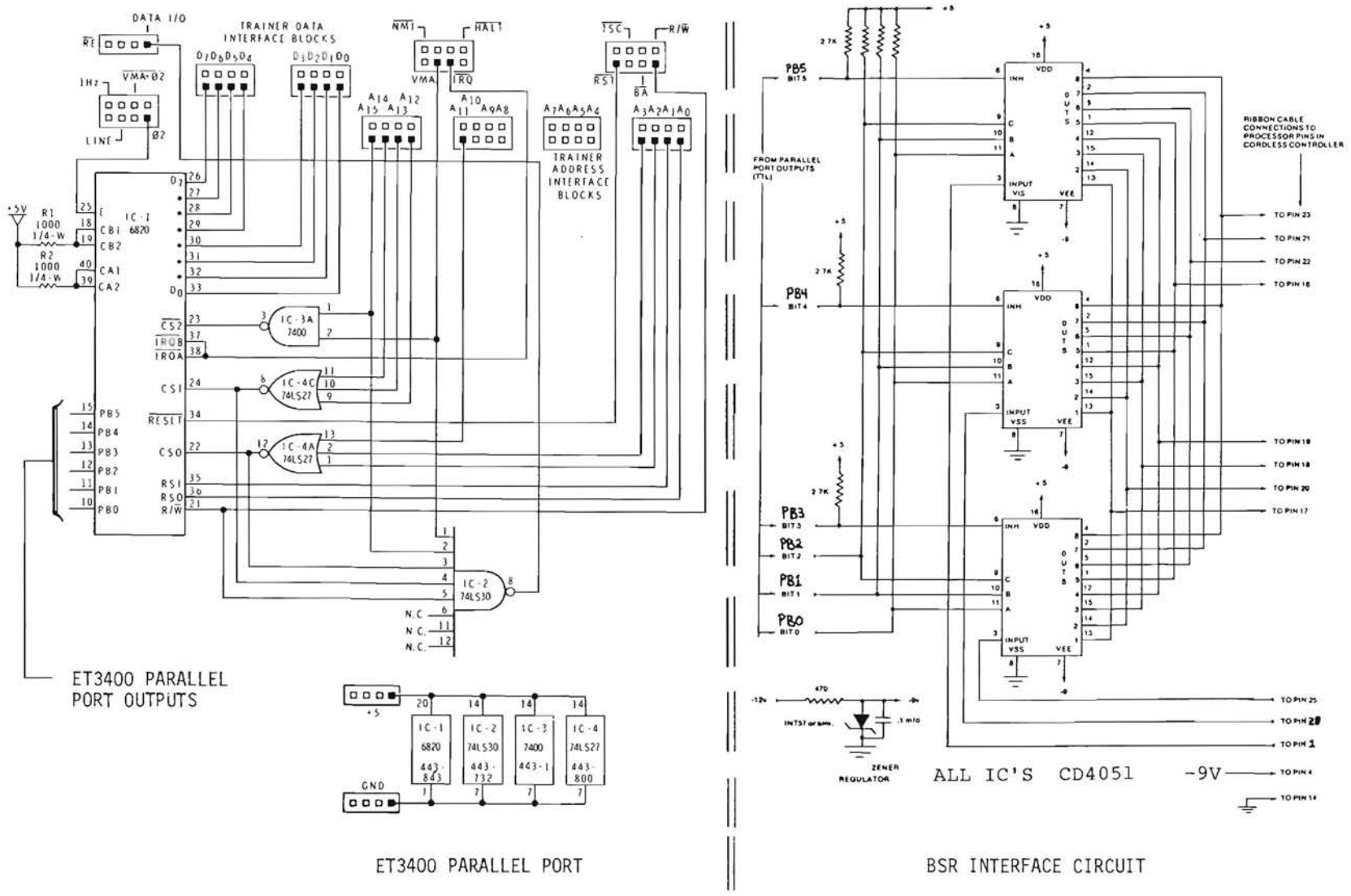


Figure 1. INTERFACE CIRCUIT BETWEEN THE HEATHKIT ET3400 MINICOMPUTER AND THE BSR HOME CONTROL SYSTEM

of the output are used here. The right side of figure 1. shows the BSR interface circuit, including a zener regulator to provide a -9V source from the ET-3400 -12V supply. Although the -9V source is connected to the cordless controller, the controller still requires a battery to provide sufficient current to drive the ultrasonic transducer. The BSR interface consists of three CD4051 1-of-8 output analog multiplexers to simulate the keyboard matrix in the BSR controller. Each multiplexer takes an input signal (column in the matrix) and switches it to one of eight outputs (rows in the matrix), therefore simulating the pressing of a key on the controller. The hex byte sent thru the parallel port selects the proper multiplexer and output combination. It should be noted that figure 1. gives corrected BSR pin numbers for the CD4051 inputs. Those given in the Garetz article are wrong!

This project is exceedingly easy to complete, owing to the nature of the ET-3400. The circuits are wired using solderless breadboarding just as in the ET-3400 course. I built the parallel port on the ET-3400 breadboard. The extra breadboard supplied with the course was used for the BSR interface circuit. I mounted it with two-sided tape on the ET-3400 frame just below the permanent breadboard. The ribbon cable must be soldered to the IC in the BSR cordless controller. This is most easily done by applying solder to the IC pins before attaching the wires from the ribbon cable. Then place each wire on the correct pin and apply the iron directly to the wire to remelt the solder on the pin. This technique should be no problem for kit builders.

In a follow-up article in the Dec., 1979 Creative Computing Garetz presents an improved circuit that allows the two unused bits of the parallel port to select four house codes, therefore giving the computer control of up to 64 devices in the house. In Ciarca's article in Byte he describes a circuit similar to the one used here but using a 4-to-1 multiplexer to select the input signal to a single CD4051, therefore reducing the chip count to two.

After building the interface it can be tested with the following 6800 program:

```
0000    CE FF04   LDX #FF04
0003    FF 8002   STX #8002
0006    BD FCBC   JSR REDIS
0009    BD FE09   JSR IHB
000C    B7 8002   STA A
000F    20 F5     BRA F5
```

Using this small program hex codes can be entered from the ET-3400 keypad to operate up to 16 devices. A null code must be sent after each command to simulate taking your finger off of the BSR keys. The valid commands are:

COMMAND	HEX CODE	COMMAND	HEX CODE
NULL	FF	12	ED
1	D8	13	DE
2	E8	14	EE
3	D9	15	DF
4	E9	16	EF
5	DA	ON	F1
6	EA	ALL LIGHTS ON	F2
7	DB	BRIGHTEN	F3
8	EB	OFF	F5
9	DC	ALL OFF	F6
10	EC	DIM	F7
11	DD		

Of course it would be much more convenient if a program were written to take over the task of sending the codes at the right times during the day. Such a program will be presented in Part II. Further details on the hardware may be obtained from the articles cited above.

PART II: SOFTWARE

In this part of the article I will present a 6800 machine code program that will allow the ET-3400 to control up to 16 devices in the home according to a preset, recycling schedule input by the user. When the additional RAM is installed in the ET-3400, there is sufficient memory space available to allow over 50 five byte command strings to be processed in one day. By taking advantage of certain BSR

command formats many more devices can be controlled from the same memory space. This program, exactly as presented here, has been controlling lights and appliances in my own home for over three months without any problems

This program incorporates an extremely accurate 24 hour real-time clock that is interrupt driven by the 60 cycle line frequency available on the ET-3400. The program occupies the first 147 bytes of RAM plus one byte at address 00F7. The command stack begins at address 0100 in the second 256 bytes of RAM. The first three bytes of RAM are reserved for storing the current time; execution begins at address 0003. Hours, minutes, and seconds are displayed, left to right, on the ET-3400 LEDs as time ticks off. Each time that "minutes" is updated the program checks to see if any commands are scheduled to be sent at that hour and minute. If not, the clock continues, otherwise the program branches to a subroutine to process the command string scheduled for that time.

The command stack beginning at 0100 is made up of individual command strings entered in time sequence. Each string begins with two bytes, the hour and minute, indicating when this command string is to be processed. Following these two bytes are an arbitrary number of commands to the BSR controller. The last byte of each string must be 00. The byte following the last command string must be set to 99 to indicate the end of the command stack. When the last string has been processed and 99 is encountered the program automatically resets to the top of the command stack to recycle the command strings for the next 24 hour period. User interruption is only required when changes in the commands or the time are desired.

Command strings must be entered in time sequence relative to the time that the clock is started. All times should be entered in 24 hour format. A typical command stack might look like this:

ADDRESS	COMMAND	COMMENT	ADDRESS	COMMAND	COMMENT
0100	06	6:15 AM	010B	E8	DEVICE #2
0101	15		010C	F1	ON
0102	D8	DEVICE #1	010D	00	END STRING
0103	F1	ON	010E	23	11:30 PM
0104	00	END STRING	010F	30	
0105	06	6:45 AM	0110	F5	#2 OFF
0106	45		0111	00	END STRING
0107	F5	#1 OFF	0112	99	END STACK
0108	00	END STRING			
0109	10	10:30 AM			
010A	30				

Using only 19 of 256 bytes available, this command stack controls two devices twice each day. In this particular application a coffee pot has been turned on at 6:15 AM and turned off at 6:45. Device 2, a lighted plant stand, has been turned on for 13 hours each day. If the commands had been entered and the clock started after 10:30 AM on day 1 then the user would have entered the last string shown here at the top of the stack to maintain the time sequence of commands relative to the clock start time.

The user of this program should be aware of the following important points:

The user should be familiar with the many possibilities and permutations of BSR command sequences which allow the control of literally hundreds of devices.

A wire must be connected between "LINE" and "IRQ" on the ET-3400 sockets.

The RTI instruction must be entered at address 00F7. This is hard-wired by the ROM and can not be changed!

A command string should not exceed 62 bytes all inclusive (59 commands) to assure proper time accounting. The program automatically updates the clock during command processing.

If you do not have the optional 256 bytes of RAM installed on your ET-3400 the command stack can use the RAM area from 0094 to 00F6. Change bytes 005B/C and 007C/D to 0094. Be careful not to write over 00F7.

To change the time use EXAM and CHAN to set the time in the first three bytes. Use DO 0003 to start the clock at the time you set. Use AUTO 0100 to

enter your command stack. To reset the command stack at the top at any time use EXAM and CHAN to set bytes 005B/C to 0100. Adjusting the time and restarting the program will have no effect on the command sequence, unless by adjusting the time you have passed over the next time in the stack. If so, the command string(s) passed over will not be processed again until the next day.

The cordless controller should be kept at least ten feet from the BSR command console to avoid overdriving the ultrasonic receiver when commands are transmitted.

In this article I have presented an inexpensive, versatile method for using the ET-3400/BSR combination in home control. The possibilities for daily home control, vacation security, etc. are obvious. The hardware and software presented in this article can be extended to any computer having a latched parallel output port. It is easily transported to the H8 computer.

The program presented below may be easily modified to display days, hours, and minutes, and process commands on day/hour schedules. This would provide flexibility in scheduling from day to day, a real plus for vacation security. The program is presented with numerous comments and should be self explanatory. It may be loaded and run as a 24 hour clock before any of the interface circuits are built.

Good Luck!

```
ADDRESS   OPCODE    MNEMONIC       COMMENTS
-------   -------   ----------     ------------------------------------------
 0000     00                       hours
 0001     00                       minutes
 0002     00                       seconds
 0003     CE FF04   LDX  #FF04     set-up the PIA -
 0006     FF 8002   STX  #8002       with the B side out
 0009     86 FF     LDAA #FF       load BSR null command
 000B     B7 8002   STAA #8002     send null comand
 000E     FE 005B   LDX  #005B     get address of next time
 0011     96 00     LDAA 00        load hour from clock
 0013     A1 00     CMPA,X 00      compare clock hour with next hour
 0015     26 08     BNE  08        no match-continue clock
 0017     96 01     LDAA 01        hours match, load minutes from clock
 0019     A1 01     CMPA,X 01      compare clock minute with next minute
 001B     26 02     BNE  02        no match-continue clock
 001D     8D 3E     BSR  3E        times match! send commands
 001F     CE FFFD   LDX  #FFFD
 0022     A6 03     LDAA,X 03        load and display
 0024     BD FE20   JSR  #FE20       hours, minutes,
 0027     08        INX              and seconds
 0028     26 F8     BNE  F8
 002A     BD FCBC   JSR  #FCBC     reset display counters
 002D     CE 003D   LDX  #003D
 0030     09        DEX
 0031     27 04     BEQ  04          one second delay routine
 0033     0E        CLI              ("LINE" must be connected to "IRQ")
 0034     3E        WAI
 0035     20 F9     BRA  F9
 0037     8B 01     ADDA #01       increment seconds
 0039     19        DAA            decimal adjust seconds
 003A     A7 02     STAA,X 02      store seconds
 003C     81 60     CMPA #60       60 seconds yet?
 003E     26 DF     BNE  DF        if not, go back for next second
 0040     8D 0F     BSR  0F        else clear seconds and update minutes
 0042     81 60     CMPA #60       60 minutes yet?
 0044     26 C8     BNE  C8        if not, go back and check next time
 0046     09        DEX            else clear minutes -
 0047     8D 08     BSR  08          and update hours
 0049     81 24     CMPA #24       24 hours yet?
 004B     26 02     BNE  02        if not, don't clear hours yet
 004D     6F 01     CLR,X 01       else clear hours
 004F     20 BD     BRA  BD        go back and check next time
```

```
ADDRESS   OPCODE    MNEMONIC      COMMENTS
-------   --------  ----------    ------------------------------------------
 0051     6F 02     CLR,X 02      clear seconds or minutes
 0053     A6 01     LDAA,X 01     load minutes or hours        load
 0055     8B 01     ADDA #01      increment minutes or hours   and
 0057     19        DAA           decimal adjust               update
 0058     A7 01     STAA,X 01     store minutes or hours       routine
 005A     39        RTS           return
 005B     0100                    address of next time; scratch area
 005D     A6 02     LDAA,X 02     load next command to be sent
 005F     27 10     BEQ  10       if end of string set-up for next time
 0061     8D 1F     BSR  1F       else goto send-and-delay routine
 0063     86 FF     LDAA #FF      load BSR null command
 0065     8D 1B     BSR  1B       goto send-and-delay routine
 0067     96 02     LDAA 02         update seconds to
 0069     8B 01     ADDA #01        compensate for
 006B     19        DAA             sending commands,
 006C     97 02     STAA 02         and delays
 006E     08        INX           set index reg. for next command
 006F     20 EC     BRA  EC       go back and repeat for next command
 0071     A6 03     LDAA,X 03     load hour of next time
 0073     81 99     CMPA #99      is this the end of -
 0075     27 04     BEQ  04         the command stack?
 0077     08        INX
 0078     08        INX             if not, set the index reg.
 0079     08        INX             for the next time
 007A     20 03     BRA  03
 007C     CE 0100   LDX  #0100    else reset to the top of command stack
 007F     DF 5B     STX  5B       store address of next time
 0081     39        RTS           return to clock
 0082     B7 8002   STAA #8002    send command to PIA
 0085     DF 5B     STX  5B       store index reg. temporarily   send-
 0087     CE 001F   LDX  #001F                                   and-
 008A     09        DEX                                          delay
 008B     27 04     BEQ  04         1/2 second delay             routine
 008D     0E        CLI
 008E     3E        WAI
 008F     20 F9     BRA  F9
 0091     DE 5B     LDX  5B       restore index reg.
 0093     39        RTS           return to command processor

 00F7     3B        RTI           return from interrupt

 0100     <BEGIN COMMAND STACK HERE>

                                                          EOF
```

The Magic Egg

Many years ago (who knows exactly when?) the drum was invented, followed some time thereafter by the stringed instrument. Early drums, of course, were made from simple logs. Man advanced his "technology" and with this additional "wisdom", he discovered that animal hide pulled tightly over the end of the log added tonal qualities that had never been achieved before. Further, he noted that cat innards could be wound tightly to produce a strong adjustable string-like cord which, when connected on a long flat stick with a box attached, produced a musical note. More strings were added to the stickbox, and the box was renamed the Guitar (something to do with "guts" ... I guess).

Well ... while man was relaxing on his caboose with these two magnificent accomplishments, a lad named Murphy was born. It seems that Murphy (not your average child) invented a gaget that has and will continue to help man achieve higher levels of technology. Yup ... along came the "monkey" wrench! And, as you have probably guessed, Murphy threw the thing at the two strolling musicians one day. To man's great displeasure, the hide on the drum streched and his guitar went out of tune. Man's perfect discovery had a flaw. In fact, it seems that the two musicians were required to readjust their "hides" and "guts" continually.

The musicians started what has come to be known as R & D (research and

(vectored to page 17)

Build an EPROM Programmer

In the last issue of REMark, I promised to tell you how to build a board for the EPROMS you program with the programmer I presented in that issue. Fig. 1 is a schematic of the board. It can hold 16k of data in 8 2716 EPROMS occupying the last 16k of address space in the H8. That leaves you 40k of RAM space, or 48k with ORG 0. Switch SW1 allows you to disable the board in case you want to put RAM in that space. I am assuming that you will build the board on a wire wrap board, so I included all interfacing and address decoding.

Address decoding is done with a 74LS138 one-of-eight decoder. Since the inverted address lines are run straight to the decoder, the outputs are active in reverse order. That is, output Y7 selects the 2716 at the bottom of the 16k address space, Y6 selects the next one, etc. The Reset line from the H8 bus is connected to one of the decoder's enable inputs so that the board will be disabled when the computer is reset. (This is also done on Heath RAM boards.)

The 2716's are selected by their Chip Enable inputs instead of Output Enable so they will power down when they are not used. Each output from the decoder goes to one of the 2716's. All other connections to the 2716's are common, that is, they should be bridged from one to another, even though it is not shown on the schematic.

The outputs of the decoder, which are active low, are ORed together (the 74LS30 is used as a negative OR gate), and the result is ANDed with the Memory Read signal (MR) to produce a signal to enable the 74LS240 through which the data outputs from the 2716's must pass. The address lines that do not go to the decoder are connected through inverters to the 2716's.

I have been asked why people like myself who draw schematics for do-it-yourself projects don't number the pins on all of the chips, such as the 74LS00 and 74LS30

(vectored to page 18)

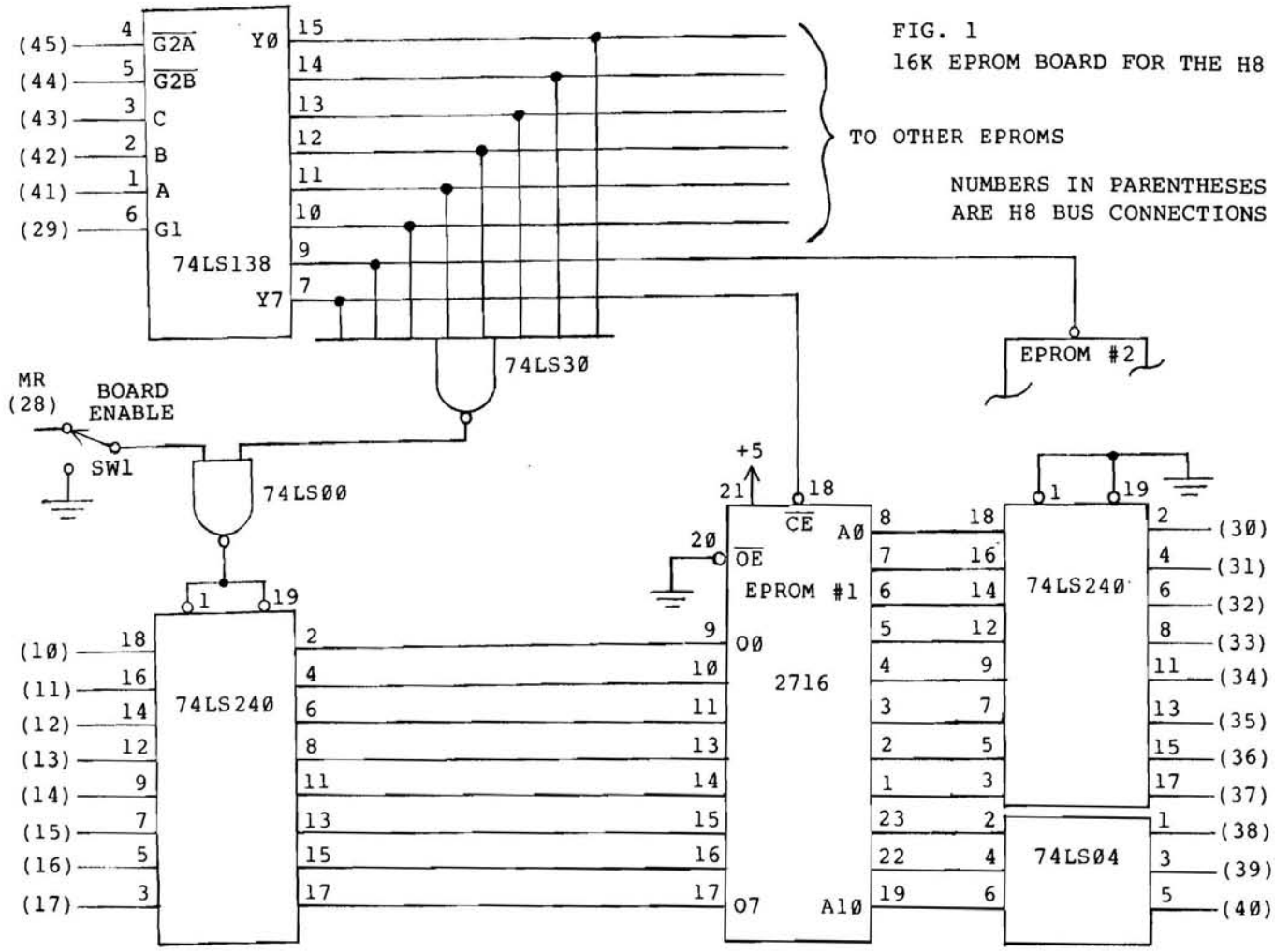


FIG. 1
16K EPROM BOARD FOR THE H8

H11A File Management

by Douglas H. Mc Neill, M.D.
Poynette Family Practice Center
PO Box 115 -- 330 N. Main St.
Poynette, WI 53955

One of the most frustating parts of a major programming effort is creating what appears to be a dynamite program only to find either strange or no output or the eruption of DCE (Device Channel Error) from the bowels of the operating system. When multiple files are used for I/O (up to 14 can be accessed with the H11A!), the potential for confusion rises logarithmically.

Also, the grand headache of remembering which channel was for what throughout the program is an unnecessary encumberance. The BASIC interpreter will support indirect I/O, for example:

```
INPUT #I:Q$
PRINT #I:Q$
```

With little overhead systematic file assignment is both possible and beneficial, for whereas the operating system will not tell you who's on first, a small program will. After an error has been detected by the system, you can give a command in the BASIC immediate mode that will tell you what's up. For the following program, that command would be GO TO 10000.

```
100 DIM(F$(6),F(6)
110 E9$='' FOR I=0 TO 6 F(I)=0 F$(I)='' NEXT I
120 REM ************************************************
130 REM *                                              *
140 REM *               FILE MANAGEMENT                *
150 REM *                                              *
160 REM *        ENTRY POINTS FOR FILE MANAGEMENT      *
170 REM *                FUNCTIONS:                    *
180 REM *                                              *
190 REM *        OPEN FOR INPUT  -        1040         *
200 REM *        OPEN FOR OUTPUT -        1160         *
210 REM *        CLOSE ONE FILE  -        1280         *
220 REM *        CLOSE ALL FILES -        1305         *
230 REM *        FILE STATUS REPT-        1410         *
240 REM *                                              *
250 REM *          FILE NAMES CARRIED IN Q$            *
260 REM *          VARIABLE I0 RESERVED FOR            *
270 REM *                FILE NUMBER                   *
280 REM *                                              *
290 REM ************************************************
1000 REM - Find the next available file
1010 FOR I0=1 TO 7 IF F(I0-1)=0 THEN 1030  NEXT I0
1020 E9$='All sequential files #1-7 in use' GOSUB 1410  STOP
1030 RETURN
1040 REM - Open a file Q$ for input
1050 GOSUB 1000
1060 ON I0 GO TO 1070 ,1080 ,1090 ,1100 ,1110 ,1120 ,1130
1070 OPEN Q$ FOR INPUT AS FILE #1 GO TO 1140
1080 OPEN Q$ FOR INPUT AS FILE #2 GO TO 1140
1090 OPEN Q$ FOR INPUT AS FILE #3 GO TO 1140
1100 OPEN Q$ FOR INPUT AS FILE #4 GO TO 1140
1110 OPEN Q$ FOR INPUT AS FILE #5 GO TO 1140
1120 OPEN Q$ FOR INPUT AS FILE #6 GO TO 1140
1130 OPEN Q$ FOR INPUT AS FILE #7
1140 F(I0-1)=1 F$(I0-1)=Q$
1150 RETURN
1160 REM - Open a file Q$ for output
1170 GOSUB 1000
1180 ON I0 GO TO 1190 ,1200 ,1210 ,1220 ,1230 ,1240 ,1250
1190 OPEN Q$ FOR OUTPUT AS FILE #1 GO TO 1260
1200 OPEN Q$ FOR OUTPUT AS FILE #2 GO TO 1260
1210 OPEN Q$ FOR OUTPUT AS FILE #3 GO TO 1260
```

```
1220 OPEN Q$ FOR OUTPUT AS FILE #4 GO TO 1260
1230 OPEN Q$ FOR OUTPUT AS FILE #5 GO TO 1260
1240 OPEN Q$ FOR OUTPUT AS FILE #6 GO TO 1260
1250 OPEN Q$ FOR OUTPUT AS FILE #7
1260 F(I0-1)=-1 F$(I0-1)=Q$
1270 RETURN
1280 REM - Close one file Q$, or all files (enter at line 1305)
1290 FOR I0=1 TO 7 IF F$(I0-1)=Q$ THEN 1310  NEXT I0
1300 E9$='File for close ('&Q$&') not located' GOSUB 1410  STOP
1305 FOR I0=1 TO 7 IF F(I0-1)<>0 GOSUB 1310  NEXT I0 GO TO 1400
1310 ON I0 GO TO 1320 ,1330 ,1340 ,1350 ,1360 ,1370 ,1380
1320 CLOSE #1 GO TO 1390
1330 CLOSE #2 GO TO 1390
1340 CLOSE #3 GO TO 1390
1350 CLOSE #4 GO TO 1390
1360 CLOSE #5 GO TO 1390
1370 CLOSE #6 GO TO 1390
1380 CLOSE #7
1390 F(I0-1)=0 F$(I0-1)=''
1400 RETURN
1410 REM - Report file status
1420 RPINT "File #    File Name     Status" PRINT
1430 F$="  #       'LLLLLLLLLL"
1440 FOR I1=1 TO 7 PRINT USING F$,I1,F$(I1-1);
1450 ON F(I1-1)+2 GO TO 1460 ,1470 ,1480
1460 PRINT TAB(25);'Output in progress' GO TO 1490
1470 PRINT TAB(25);'File available' GO TO 1490
1480 PRINT TAB(25);'Input in progress'
1490 NEXT I1
1500 IF E9$='' THEN 1540
1510 E8$='*************ERROR CONDITION*************'
1520 PRINT  PRINT E8$ PRINT '*';TAB(41-LEN(E9$))/2);E9$;TAB(40);'*'
1530 PRINT E8$
1540 RETURN
10000 GOSUB 1410  STOP
```

Simultaneous Equations

by George J. Sellers
1033 Bishop Walsh Road
Cumberland, MD 21502

For those of you into advanced algebra, here is a BASIC program you might find interresting. It solves simultaneous equations with up to 20 equations and 20 unknowns. It is based on what is called the Gauss-Jordan method with maximum pivot feature.

The input is organized with each of the coefficients of each equation being a row of matrix A, which is called the coefficient matrix. The right side of the equations are organized into column B, which is called the constant matrix. The solutions are also organized into a column (column X), and it is called the solution matrix. Thus A*X=B in matrix algebra.

The data are entered into the program by way of prompts for each column and row. The coefficient matrix can then be listed to verify the accuracy of the inputs and changed if necessary. Then the constants can be input, and after a short time the solution matrix is printed.

If matrix notation is not familiar to you, you should check an advanced algebra or other suitable text. This technique forms the basis of many important types of problems, for example, network theory, regression analysis, and economics. (See August 1980 BYTE for the most recent method of trying to solve large systems of linear equations, and Sept. 1980 Scientific American p. 207 concerning economics. For a FORTRAN version of this program, see FORTRAN IV Programming and Computing by James T. Golden, published by Prentice Hall, Englwood NJ 1965.)

Note: This program as written requires MBASIC to run because of multi-letter variable names.

```
10 PRINT "SIMULTANEOUS EQUATIONS":PRINT
20 REM
30 REM  GET COEFFICIENTS
40 REM
50 DIM A(20,21),X(20),LC(20),CK(20)
60 INPUT "ENTER NO. OF EQUATIONS";NM
70 IF NM>20 OR NM < 1 GOTO 60
80 PRINT:PRINT "INPUT COEFFICIENT MATRIX"
90 FOR J=1 TO NM:FOR I=1 TO NM
100 PRINT "COLUMN  ";J;"   ROW  ";I;
110 INPUT A(I,J):NEXT I:NEXT J:PRINT
120 INPUT "CHECK INPUT";AN$:IF LEFT$(AN$,1)="N" GOTO 210
130 FOR I=1 TO NM:PRINT "ROW ";I;"  ";
140 FOR J=1 TO NM:PRINT A(I,J);" ";
150 NEXT J:PRINT:NEXT I:PRINT
160 INPUT "CHANGE INPUT";AN$:IF LEFT$(AN$,1)="N" GOTO 210
170 INPUT "ROW, COLUNM, AND NEW VALUE";I,J,A(I,J):GOTO 160
180 REM
190 REM  GET CONSTANTS
200 REM
210 PRINT:PRINT "INPUT CONSTANT MATRIX"
220 FOR I=1 TO NM:PRINT "B(";I;")";
230 INPUT A(I,NM+1):NEXT I
240 REM
250 REM  SOLVE EQUATIONS
260 REM
270 FOR I=1 TO NM:CK(I)=0:NEXT I
280 NP=NM+1
290 FOR I=1 TO NM
300 IP=I+1
310 AX=0
320 FOR K=1 TO NM
330 IF (AX-ABS(A(K,I)))>0 GOTO 370
340 IF CK(K)>0 GOTO 370
350 LC(I)=K
360 AX=ABS(A(K,I))
370 NEXT K
380 IF ABS (AX)<=1E-06 GOTO 590
390 CK(L)=1
400 L=LC(I)
410 CK(L)=1
420 FOR J=1 TO NM
430 IF (L-J)=0 GOTO 470
440 F=-A(J,I)/A(L,I)
450 FOR K=IP TO NP
460 A(J,K)=A(J,K)+F*A(L,K):NEXT K
470 NEXT J
480 NEXT I
490 REM
500 REM  PRINT SOLUTIONS
510 REM
520 PRINT:PRINT "SOLUTION MATRIX:"
530 FOR I=1 TO NM
540 L=LC(I)
550 X(I)=A(L,NM+1)/A(L,I)
560 PRINT "X(";I;")";X(I)
570 NEXT I
580 END
590 PRINT "UNDERFLOW -- AX < 10E-6":END
```

SAMPLE RUN

```
SIMULTANEOUS EQUATIONS

ENTER NO. OF EQUATIONS? 5

INPUT COEFFICIENT MATRIX
COLUMN  1    ROW  1? 5
COLUMN  1    ROW  2? 3
COLUMN  1    ROW  3? 1
COLUMN  1    ROW  4? 2
COLUMN  1    ROW  5? 4
COLUMN  2    ROW  1? -1
COLUMN  2    ROW  2? 3
COLUMN  2    ROW  3? -2
COLUMN  2    ROW  4? 1
COLUMN  2    ROW  5? 6
COLUMN  3    ROW  1? 1
COLUMN  3    ROW  2? -2
COLUMN  3    ROW  3? 4
COLUMN  3    ROW  4? -1
COLUMN  3    ROW  5? -5
COLUMN  4    ROW  1? 1
COLUMN  4    ROW  2? -6
COLUMN  4    ROW  3? 2
COLUMN  4    ROW  4? -1
COLUMN  4    ROW  5? -3
COLUMN  5    ROW  1? 2
COLUMN  5    ROW  2? 3
COLUMN  5    ROW  3? -1
COLUMN  5    ROW  4? 2
COLUMN  5    ROW  5? 3

CHECK INPUT? YES
ROW 1    5  -1  1  1  2
ROW 2    3  3  -2  -6  3
ROW 3    1  -2  4  2  -1
ROW 4    2  1  -1  -1  2
ROW 5    4  6  -5  -3  3

CHANGE INPUT? NO

INPUT CONSTANT MATRIX
B(1 )? 4
B(2 )? -6
B(3 )? 4
B(4 )? -2
B(5 )? 1

SOLUTION MATRIX:
X(1 )2
X(2 ).333333
X(3 )-.5
X(4 ).833334
X(5 )-3
```

EOF

Local HUG News

A local HUG group has been formed for Heath users in the Memphis, Tennessee area. If you would like to join, contact Morris Proctor at (901) 523-6259 before 5 PM or call (901) 345-0035 after five. The club meets on the first Tuesday of each month.

PNHUG (Pacific Northwest Heath Users' Group) meets at the Seatle WA Heathkit store, 505 8th Avenue North, on the first Monday of each month that is not a holiday. Members generally arrive around 6 p.m. and the meeting starts formally at about 7 p.m. The meetings conclude by 9 p.m.

(vectored to page 14)

Tiny BASIC Tricks

SQUARE ROOTS

If you've been running Tiny BASIC on your ETA-3400 for very long, you are probably now used to rejecting programs from game books, etc., because they require something that Tiny BASIC can't do. But if you've thrown out a program because it needs square roots, go fish it out of the circular file, because now you can run it.

We're going to present a couple of routines to do the job. We will make both of them subroutines that start at line 1000, with A used as the input variable, and B as the output variable. To test them, you can use the following program.

```
10 PRINT "ENTER A NUMBER";
20 INPUT A
30 GOSUB 1000
40 PRINT "THE SQUARE ROOT IS ";B
50 PRINT
60 GOTO 10
```

One way to get square roots is to guess at it, then square your guess to see how close it is. Then you can work your way up or down until you get it right. Tiny BASIC can't guess, but it can check a number to see if it is right, and find the square root if we give it a place to start. A good place to start is 181, because that is approximately the square root of 32767, the highest number allowed in Tiny BASIC. The following routine uses this check and move down method to find square roots.

```
1000 B=182
1005 REM B=181+1 BECAUSE WE DECREMENT FIRST
1010 B=B-1
1020 IF B*B>A GOTO 1010
1030 RETURN
```

There are two things wrong with this approach. One is that it takes a long time, with the time increasing with smaller numbers. We can shorten the time required by checking the size of the number and changing the starting value of B accordingly, as in this example.

```
1000 B=182
1005 IF A<10000 THEN B=100
1010 B=B-1
1020 IF B*B>A GOTO 1010
1030 RETURN
```

This fix cuts nearly in half the time required to find the square root of 4 compared to the first example. You could add more entry checks to reduce the time further. But there is another objection to this approach. It always gives you the truncated integer of the square root, not a rounded off integer. For example, the square root of 8 is about 2.828, but this method gives you 2, not 3, which is closer.

There is another method for finding square roots which is quicker and more accurate (at the expense of a bit more code). This method makes use of an algorithm often included with electronic calculators back in the days when they only had 4 functions. The algorithm is

Yn+1=1/2(Yn+(X/Yn))

where X is the number we want the square root of, and Y is an approximation. We keep feeding the result of this equation back in until the error is acceptable. A routine using that algorithm appeared in Kilobaud MICROCOMPUTING Magazine, February 1980, page 172, by Tom Crawford, and is presented below with the publisher's permission.*

```
1000 REM    SQUARE ROOT ROUTINE.  RETURNS
1010 REM    ROOT OF A IN B.  USES C AND D
1020 IF A>1 GOTO 1050
1030 B=1
1040 RETURN
1050 B=A/182+2
1060 D=0
1070 C=(B+A/B)/2
1074 REM  CHECK IF C SHOULD BE ROUNDED UP
1075 IF (A-A/(2*B)*B*2)>B THEN C=C+1
1080 IF B=C THEN RETURN
1085 REM  CHECK FOR OSCILLATION OF RESULT
1090 IF (B-C)=D GOTO 1130
1100 D=C-B
1110 B=C
1120 GOTO 1070
1130 REM  PICK THE LARGER OF B OR C
1140 IF B>C THEN RETURN
1150 B=C
1160 RETURN
```

The first approximation is obtained in line 1050. This routine up remainders greater than .5, so the square root of 8 is given as 3 instead of 2 as with the first method. It also runs much faster, so if you have the room, you should use this method in your programs. Now, you can start coding that Lunar Lander game!

PS:

(vectored from page 13)

For St. Louis HUGgies, there is a local group (SLHUG) meeting at the Heathkit store at 3794 McKelvey Rd., Bridgeton MO 63044, phone (314) 291-1850. Meetings are held on the second Wednesday of the month at 7:30.

H88 Software from HUG

Yes, there is software available for the H88 computer from the Heath Users' Group. The real question is which software will work with the H88 and not just the H8 computer.

The software volumes one to five all have programs which will work with the H88. This also includes the tapes one to five.

The purchase of both the tape and the volume is not required, however the volumes do include, beside the listings, some additional information on some of the programs on the tapes. This information may be helpful in making some of the programs operate.

When loading the programs from the tapes, some of the programs, especially those on Tape I, will require that the command OLDLOAD be used instead of the LOAD command. (Use OLDLOAD on ALL programs written in Benton Harbor BASIC version 10.01.) Once a program has been loaded into memory it may be dumped to a new tape, using the DUMP command. The next time the program is loaded from the new tape the LOAD should be used.

The Volume I (885-1008) and Tape I (885-1009) have many programs that will work with the H88 computer. The programs that will NOT work are those for the H11 that are included in the volume and the machine code programs included on the tape. The machine code programs will not work because the H8 uses a different port number for the terminal than does the H88 computer.

The Morse code programs that are included in Volume I and on Tape I use the speaker on the H8 computer which is different than the speaker within the H88 computer. Remember all programs using the speaker or the front panel L.E.D.'s of the H8 will not operate properly on the H88 or H89 computers.

Even with the negative notes in the previous paragraphs about Volume I and Tape I there are over fifty programs that will work on the H88 computer.

The Volume I is still selling for the low price of $9.00 and the Tape I is selling for only $7.00. Even if both items are purchased that's fifty programs for only $16.00 plus shipping. Approximately $0.35 for each program, a bargain for any computer.

Volume II and Tape II also have over twenty programs which will work on the H88 computer. The Volume II sells for $12.00 (885-1013) and the Tape II sells for $9.00 (885-1012). Note there is a second tape for Volume II which contains only machine code programs and these will NOT work, on the H88 computer, without modifcation. The problem is with the port address of the H8 being different than the H88 computer. The second tape for Volume II sells for $9.00 (885-1014). This is source code intended for an experienced assembly language programmer. If this tape is purchased remember all port addressing must be changed for use with the H88 or H89 computers and the programs reassembled.

Volume III and Tape III are also very useful for the H88 computer. All the programs included in these two items should operate on the H88 except the Morse code program. The price for Volume III is $12.00 (885-1015) and the price of the Tape III is $9.00 (885-1026).

Volume IV and Tape IV have a collection of thirty programs, most of these will work with no modifications, on the H88 computer. The last four programs in this volume and on this tape will not work on the H88 computer. That still leaves twenty-six programs for the H88 computer. The price for Volume IV is $12.00 (885-1036) and the Tape IV is $9.00 (885-1037). Another fine bargain at $21.00 plus shipping for twenty-six programs (less than a dollar a piece).

NOTE: Tape III has Extended Benton Harbor BASIC Version 10.02 at the beginning of the tape. This version will NOT work on the H88 computer.

The last of the present volumes and tapes available through the Heath Users' Group is Volume V and Tape V. These two products are the first of the true H88 software products, that is they include several graphic programs for the H19 portion of the H88 computer. The Volume V sells for $12.00 (885-1057) and the Tape V sells for $9.00 (885-1058).

Future cassette programs are planned for both the H8 and H88, however HUG needs the support of the cassette user so that the less than a dollar a piece program will continue to be a part of the HUG library. HUG is looking for additional graphic programs for the cassette system. If you have some of these programs to submit, HUG would greatly appreciate hearing from you.

:GK:

HUG Products List

Part Number	Description		Selling Price
CASSETTE SOFTWARE			
MISCELLANEOUS COLLECTIONS			
885-1008	Volume I	Documentation	$ 9.00
885-1009	Tape I	Cassette	$ 7.00
885-1012	Tape II BASIC	Cassette	$ 9.00
885-1013	Volume II	Documentation	$ 12.00
885-1014	Tape II ASM	Cassette H8 Only	$ 9.00
885-1015	Volume III	Documentation	$ 12.00
885-1026	Tape III	Cassette	$ 9.00
885-1036	Tape IV	Cassette	$ 9.00
885-1037	Volume IV	Documentation	$ 12.00
885-1057	Tape V	Cassette	$ 9.00
885-1058	Volume V	Documentation	$ 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			
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
885-1060	Disk VII	H8/H89	$ 18.00
885-1062	Disk VIII	H8/H89 (2 Disks)	$ 25.00
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
885-1083	Disk XVI	Misc H8/H89	$ 20.00
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	H8/H19/H89 Games	$ 18.00
885-1068	Disk XII	MBASIC Graphic Games	$ 18.00
UTILITIES			
885-1019	Device Driver Disk H8/H89		$ 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
885-1061	TMI Load H8 Only		$ 18.00
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
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
BUSINESS AND FINANCE			
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
885-1054	SmBusPkg II 3 Disks H8/H19/H89		$ 60.00
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
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	$ 9.00
885-1033	HT-11 Disk I		$ 19.00
CP/M SOFTWARE (version 1.43)			
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
885-1206	CP/M Games Disk		$ 21.00
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.

(vectored from page 9)

developement). The guitar player was fairly successful. He replaced his "guts" with steel strings and minimized the effects of Murphy's wrench. The drummer, however, found that steel was an unacceptable substitute for his hides as noted by the ringing in his ears when he attempted to play the drum. The drummer, in his search for a replacement "hide" noticed that the tone of the drum varied higher and lower with temperature, humidity, and pressure. Therefore, he began his search for a substance which would give Murphy a hard way to go. He used paper, cloth, wood, glass, and a variety of other coverings on his drum. Unfortunately, all of these materials were useless for one reason or another ... he was lost.

One day while our drummer friend was attempting to play his instrument, a gentleman named Mylar introduced himself. He indicated that he had developed a substance which, although not perfect, could be used as a "hide" replacement. For reasons unknown, he called this stuff "Mylar", and it worked. The drummer was elated. Although weather effected the mylar, much greater extremes in temperature, humidity and pressure were required to cause changes in the new "hide".

As you can guess, Murphy was pretty upset. He didn't like musicians. In fact, he didn't like too much of anything as evidenced by his wrench showing up in so many locations. He had used one of his best wrenches on the drummer. It had taken hundreds of years to get the "monkey" out of the drum. Murphy vowed to get even with Mylar for his discovery.

Meanwhile, Mylar's discovery was being used widely in industry for all kinds of neat products. Somebody even coated Mylar's substance with magnetic oxide which allowed us to record music and voice to man's enjoyment. Technology was advancing rapidly. Man entered the age of the computer and discovered that magnetic mylar could store computer stuff just as if it were a voice or music.

"Where does this leave us?", you ask. Well, Mylar's round drum "hide" was coated with magnetic oxide so that Mr. Heath could use it to develope a computer. Mr. Heath obtained the round "hide" and a machine called a "drive" to put the "hide" in. Hundreds of recordings were made by a computer and sent to users of Mr. Heath's computers to use on their computer drives. Everybody was happy.

Remember Murphy? Murphy was not at all happy about people using Mylar's discovery in Mr. Heath's drive. So ... Murphy pulled out his biggest "monkey" and aimed it right at Mr. Heath's "floppy". BINGO! He was right on target! Murphy had fulfilled his vow. He had discovered that the mylar "hide", when subjected to heat, humidity, and pressure, would grow. And, to complicate matters further, when the "hide" was coated with the magnetic stuff, it would grow in the shape of an EGG!

Unknowingly, Mr. Heath's "Huggies" returned the drives as defective. At Mr. Heath's factory, the drives tested good. Everyone was puzzled about this new happening. Tests were developed, modified, rechecked, etc., until the drive experts said, "You must be encountering the magic egg! Your mylar hides are growing, causing incorrect response from Mr. Heath's computers." Thus, another

(vectored to page 30)

New HUG Products

885-1059 FOCAL-8 Disk H8/H89 $ 25.00

This is an HDOS version of the cassette FOCAL (FOrmula CALculator) interpreter (885-1045), with many inprovements and extensions. It includes disk I/O commands for storage and retrieval of programs and program variables and for chaining programs. Disk operations are done in large single blocks instead of many small ones, making them much faster than with B H BASIC or MBASIC. FOCAL-8 has an improved line editor for correcting mistakes and a trace mode for finding bugs. It allows multi-character variable names and can print numbers in either engineering or BASIC-like format. It has a printer control command that lets you send listings or program output to a line printer.

885-1086 Tiny Pascal Disk H8/H89 $ 20.00

This is the version of PASCAL that was originally presented in BYTE Magazine (Sept. - Nov. 1978) by Kin Man Chung and Herbert Yuen. It is an HDOS adaptation of the 8080 assembly language version by B. Gregory Louis, with HDOS file control commands added. It consists of compiler and a translator that translates P-code to 8080 machine code, and includes several sample Pascal programs. It is easy to learn and produces fast, efficient programs.

HUG BUG: In the KISS article in REMark issue #9, line 2060 of the program there should read:

```
2060 PRINT T$(I);: LINE INPUT":  ";A$(N,I)
```

Diablo Remote Control

by Lloyd E. Johnson
Leejon Engineering Co.
700 Highview Rd.
East Peoria IL 61611

Recently I discovered that the keyboard of the H89 can be used to type copy for printing by a printer with no more software than is provided by booting up with HDOS. This is very convenient for printing addresses on envelopes or for the typing of notes or memos for which a record need not be stored on disk or tape.

HDOS works much better for this than CP/M. HDOS is put into the mode for keying copy to its video screen by typing

```
PIP LP:=TT:<CR>
```

After entering this type-to-screen mode the only prompt is the cursor. What is typed to screen is not printed until CTRL-D is typed, then HDOS returns to the command mode. The first time the PIP LP:=TT: command is given the device driver LP.DVD is loaded. CTRL-D doesn't unload the driver, so a return to PIP LP:=TT: is almost instantaneous. Many screenfuls of copy can be typed before giving the CTRL-D order to print. The amount of RAM available in memory below the operating system apparently determines how much can be typed before printing is necessary.

In CP/M (version 2.2), the startup process is similar. The command required is

```
PIP LST:=CON:<CR>
```

However, in CP/M each letter is printed as it is typed. Any errors made appear both on the screen and on paper. The <CR> does not cause a line feed nor <LF> a carriage return, so you must use both to start a new line. Typing CTRL-Z will get you back into the command mode.

Since HDOS does not print until you hit CTRL-D, you can make corrections with DELETE or BACK SPACE, but you must make them before you hit RETURN. With CP/M, backspacing leads to type-overs. In HDOS, the cursor control keys not only move the cursor, but also embed signals which are sent to the printer. Everything that is keyed in is sent to the printer except characters that have been deleted.

The ability to send embeded signals turns out to be of more use than trouble. The Diablo printer (WH44) has certain functions that can be activated by special codes. I presume other printers, such as Qume and NEC, would respond to similar sets of codes. Following this article is a table of the functions of a Diablo printer that can be activated remotely.

Diablo sells an optional accessory called the XMEM2 printed circuit board that can be activated to provide the special functions marked with an asterisk in the table. This PCB, which contains IC's (ROMS) for word processing (and vector plotting) can be bought separately and plugged into a slot in Diablo printers. Special functions such as proportional spacing, justification of margins, automatic centering of a line, underlining, and bold and shadow printing are nice to have. They add style to the printed copy. It is also possible to generate these special functions with software, and in fact it is done with word processors.

Of these codes, ESC 9, for setting the left margin, will probably be used most often. Successive settings of the left margin to the right are cumulative, but it is possible to back space past the setting and set a new one. Note that the ESC E command will clear your H19/H89 screen.

Through effecient use of these remote functions, you could create your own RUNOFF-type program, with the printer doing most of the work.

(The table of Diablo functions is on the next page.)

(vectored from page 10)

in this design. There are two reasons. The first is that there is often more than one gate in the chip for small gates such as the 74LS00, and I am leaving it up to you to use the one(s) you want where you want them. The second is that for a gate like the 74LS30, all of the inputs are the same, and it does not matter which one goes where. There are data books available at electronic stores or from mail order houses that show the pinouts of all the TTL IC's available. You should not even attempt a project such as this one without one of those books even if all the pins are numbered, because the author (me) might have made a mistake. It is also common practice not to draw in the power and ground connections on the IC's (as was done on this schematic) so the drawing will be less cluttered. Here again, the data book will show you the connections.

PS:

CODES FOR REMOTE CONTROL OF DIABLO 1640 PRINTER FROM H89 KEYBOARD

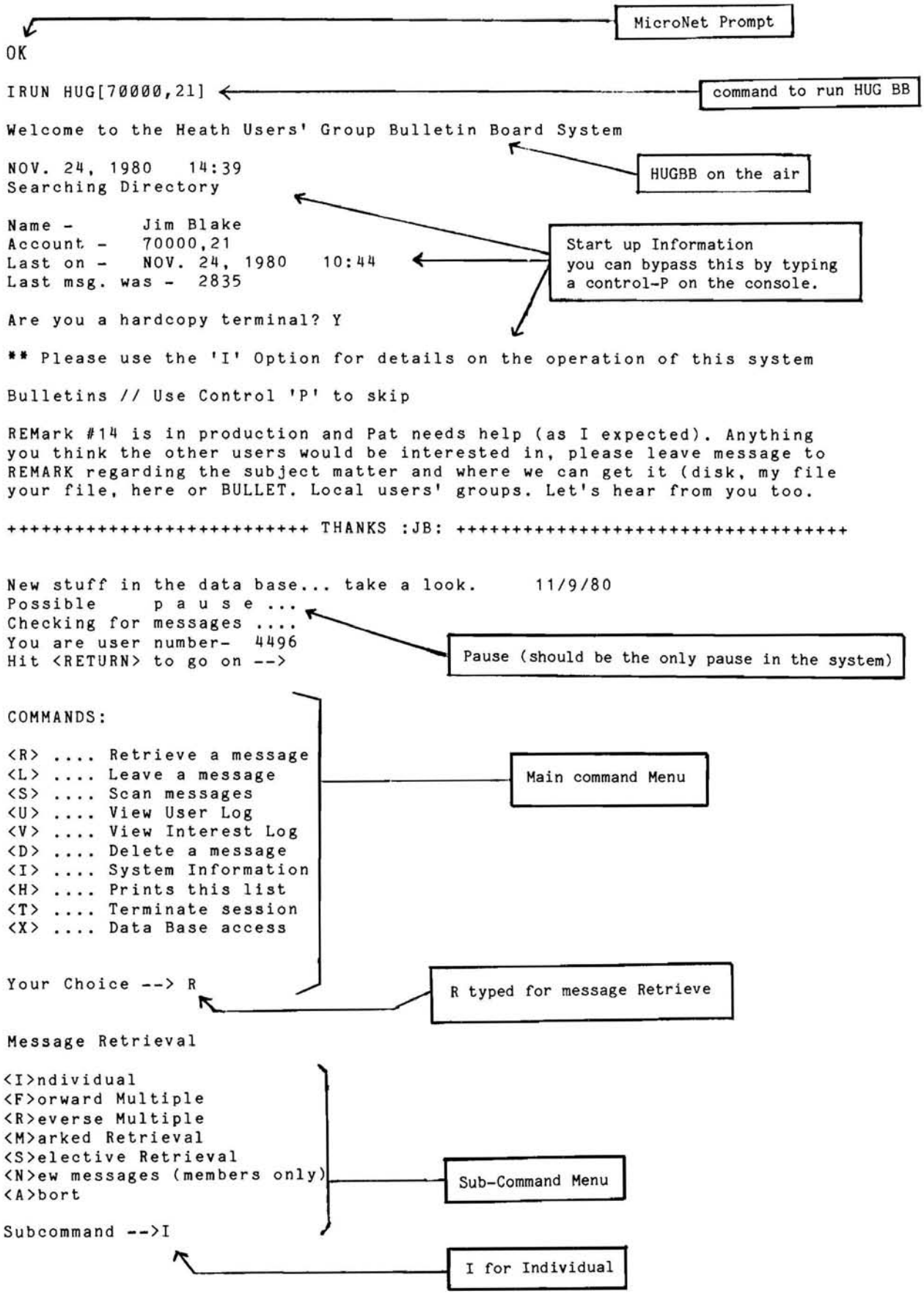
Diablo 1640 Printer Functions	Heath Keys	ASCII Chars	CODES Hex	Dec	Oct	Remarks
INITIAL SET-UP CODES MOST LIKELY TO BE USED:						
*Proportional Spacing ON	Blue	ESC P	1B,50	27,80	033,120	
*Proportional Spacing OFF	Red	ESC Q	1B,51	27,81	033,121	
Set HMI by Pitch Switch	f1	ESC S	1B,53	27,83	033,123	
*Auto Justification ON	DL	ESC M	1B,4D	27,77	033,115	off,ESC X
*Auto Center One Line	ESC =	ESC =	1B,3D	27,61	033,075	off,ESC X
*Bold Overprinting START	ESC O	ESC O	1B,4F	27,79	033,117	off,ESC &
*Shadow Printing START	f5	ESC W	1B,57	27,87	033,127	off,ESC &
*Underlining START	ESC E	ESC E	1B,45	27,69	033,105	clears CRT
Top Margin SET (as set)	f2	ESC T	1B,54	27,84	033,124	
Left Margin SET, at cursor	ESC 9	ESC 9	1B,39	27,57	033,071	+residual
INITIAL SET-UP CODES OCCASIONALLY USED:						
Clear All HMI and VMI Tabs	ESC 2	ESC 2	1B,32	27,50	033,062	
Graphic Mode ON	ESC 3	ESC 3	1B,33	27,51	033,063	clear w/CR
Graphic Mode OFF	ESC 4	ESC 4	1B,34	27,52	033,064	
Forward Print Mode ON	ESC 5	ESC 5	1B,35	27,53	033,065	default mode
Backward Print Mode ON	ESC 6	ESC 6	1B,36	27,54	033,066	clear w/CR
Print in Secondary Color	Up Crsr	ESC A	1B,41	27,65	033,101	
Clear Top and Bottom Margins	Rt Crsr	ESC C	1B,43	27,67	033,103	
*Add 20ms Settling Time	ESC %	ESC %	1B,25	27,37	033,045	off,ESC N
Set Lines/Page to (n)	ESC	CTRL-L (n)	1B,0C+	27,12+	033,014+	dflt(n=66)
CODES LIKELY TO BE USED DURING TYPING OF TEXT:						
Right Margin SET, at Crsr	ESC 0	ESC 0	1B,30	27,48	033,060	+residual
*Underscoring STOP	White	ESC R	1B,52	27,82	033,122	
*Bold and Shadow Print STOP	ESC &	ESC &	1B,26	27,38	033,046	
*(above)+Justify and Cntr. STOP	ESC X	ESC X	1B,58	27,88	033,130	
Lower Margin SET (as set)	IL	ESC L	1B,4C	27,76	033,114	
OTHER USEFUL CODES:						
Set HMI Tabs (at cursor)	ESC 1	ESC 1	1B,31	27,49	033,061	
Clear HMI Tabs (at cursor)	ESC 8	ESC 8	1B,38	27,56	033,070	
Negative Line Feed	ESC LF	ESC LF	1B,0A	27,10	033,012	
1/2 Space Line Feed	f3	ESC U	1B,55	27,85	033,125	
Negative 1/2 Space LF	Lf Crsr	ESC D	1B,44	27,68	033,104	
Vertical Tab SET (as set)	ESC -	ESC -	1B,2D	27,45	033,055	
Vertical Tabulation	CTRL-K	CTRL-K	0B	11	013	
Print Suppression ON	ESC 7	ESC 7	1B,37	27,55	033,067	clear w/CR
Character "£" or "¢"**	ESC Y	ESC Y	1B,59	27,89	033,131	
Character "¬" or "‾"**	ESC Z	ESC Z	1B,5A	27,90	033,132	
*Normal Settling Time	DC	ESC N	1B,4E	27,78	033,116	
Initiate Remote RESET	ESC	CTRL-P	1B,10	27,16	033,020	

*Functions available if printer is fitted with XMEM2 board for word processing.
**These characters will depend on your print wheel.

MicroNET Sample Run

The following is a sample run of MicroNET (and the HUG bulletin board) to let you know what it looks like and how it works. We have added comments (in boxes) to explain some of the inputs and outputs.

(sample run starts on the next page)


MicroNet Prompt
OK
IRUN HUG[70000,21]
command to run HUG BB
Welcome to the Heath Users' Group Bulletin Board System
HUGBB on the air
NOV. 24, 1980 14:39
Searching Directory
Name - Jim Blake
Account - 70000,21
Last on - NOV. 24, 1980 10:44
Last msg. was - 2835
Start up Information
you can bypass this by typing
a control-P on the console.
Are you a hardcopy terminal? Y
** Please use the 'I' Option for details on the operation of this system
Bulletins // Use Control 'P' to skip
REMark #14 is in production and Pat needs help (as I expected). Anything
you think the other users would be interested in, please leave message to
REMARK regarding the subject matter and where we can get it (disk, my file
your file, here or BULLET. Local users' groups. Let's hear from you too.
++++++++++++++++++++++++++ THANKS :JB: ++++++++++++++++++++++++++++++++
New stuff in the data base... take a look. 11/9/80
Possible p a u s e ...
Checking for messages ....
You are user number- 4496
Hit <RETURN> to go on -->
Pause (should be the only pause in the system)
COMMANDS:
<R> .... Retrieve a message
<L> .... Leave a message
<S> .... Scan messages
<U> .... View User Log
<V> .... View Interest Log
<D> .... Delete a message
<I> .... System Information
<H> .... Prints this list
<T> .... Terminate session
<X> .... Data Base access
Main command Menu
Your Choice --> R
R typed for message Retrieve
Message Retrieval
<I>ndividual
<F>orward Multiple
<R>everse Multiple
<M>arked Retrieval
<S>elective Retrieval
<N>ew messages (members only)
<A>bort
Sub-Command Menu
Subcommand -->I
I for Individual

```
Single Retrieval
The system contains messages  2547 to  2837

Enter Msg# you wish to see (C/R to end)-> 2837

Msg#-   2837
Date- NOV. 24, 1980   13:27
From- Jim Blake 70000,21
To- Howard Nurse
Subject- You Here?

You be here the 8th? :JB:

End of Msg-      2837
Delete this message (Y/N) -> N

Enter Msg# you wish to see (C/R to end)->

<R>etrieve   <L>eave   <S>can   <T>erminate  <D>elete
<U>ser Log   <H>elp    <I>nfo   <X>Data Base <V>Int Log

Your Choice --> I

  ****       W E L C O M E   T O   H U G B B  1.0  ****

(Use Control 'P' to end)

I  hope  you enjoy the new system and that you will take the
time  to  try  out  the many features. The only pause in the
system may occur after log on at 'Please hold on  ....'  and
the time may vary as to the system load.

        While this system is open to all, Heath H 11
        users will find items of special interest by
        typing IRUN MNET11.CMD[70110,403]

              * * * F E A T U R E S * * *

MARKED RETRIEVAL:

If  a message contains your name OR your account number, you
will be alerted as you log on and  the messages  will  be
placed in  an array for MARKED RETRIEVAL. You can also MARK
your own messages while doing a SCAN. To retrieve them,  use
the 'M' subcommand under RETRIEVAL.

Once  you pick up a message it is marked as retrieved and it
will not be announced on subsequent logons. You can also see
if messages have  been picked  up  by  looking  at  the
heading during a SCAN. To: Jim Blake (X) would signify that
I  have  picked it up. Please go through MARKED RET. once so
that your  old messages get zapped as  retrieved.  You  can
SKIP  OVER  the message text by using CONTROL 'O' which will
move you to the next message.

CONTROL 'P' is used as the ABORT KEY and will return you to
the short MENU immediately!

LOWER CASE  can  be  used  anywhere  since  all  the  SEARCH
routines  have been  written  to  match  any combination of
upper/lower case.

PLEASE USE THE 'T' COMMAND TO LEAVE THE PROGRAM.  Using  the

'T' COMMAND is to your advantage as it updates your personal
log  and allows the system to tell you what the last message
number was when you were last on.
```

Message Number

This is the Message

Short Command Menu

Running Information

This information tells about the Bulletin Board. This option can be run by the user any time a question comes up.

SELECTIVE RETRIEVAL will search any heading field, match your input and retrieve the message. Due to the fact that what you see on your screen has little to do with what has already been sent to buffers, it has been neccessary to include 'HIT <RETURN> ... ' between all multiple retrievals and scans. This allows you to MARK messages during a scan and it allows me to allow you to skip over the text of a message (trust me). As MNET has TYPE AHEAD, no time is lost since you can hit <RETURN> before the question appears (if you wish).

As always CONTROL 'S' freezes the output and CONTROL 'Q' restarts it.

** S Y S T E M I N F O R M A T I O N **

CBBS.CMD was written by Richard Taylor and is run by MNET-80, a TRS-80 Users Group of MICRONET. It was

subsequently adapted for use by the Heath Users Group by Richard Taylor with on going enhancement by the HUG staff. The system was designed in FORTRAN.XF4 which is MICRONET'S own extended FORTRAN.

As well as being a Computer Bulletin Board System, CBBS.CMD also gives access to the catalog of the 1000K data base which the group has put together. A sample catalog of files is available to non-members through the 'X' option on the board. To type out a file you must exit the program ('T') and type TYP FILENAME.EXT<70000,21>

If you are interested in becoming a member of the Heath, Users' Group, please leave a message to SYSOP or Jim Blake and he will forward the information to you. If you are a current member of HUG and want to become a member of this system and have access to whatever files are available, please so indicate leaving me your HUG ID # and as soon as you have been added to the system, you will know, since the system will 'remember' you when you sign on.

<R>etrieve <L>eave <S>can <T>erminate <D>elete
<U>ser Log <H>elp <I>nfo <X>Data Base <V>Int Log

Your Choice --> X

Mini-Menu Again

Data Base access
Checking Clearance

Jim Blake 70000,21 full clearance

Looking at Data Base

MEMBER.DIR List of Huggies on this system as of 11/11/80

Software available on the system 11/11/80

TIME.ASM Software clock for H8/H89

PICDEF.ACM
DVDDEF.ACM
CMDUTIL.ACM
PICDEF.ACM
DVDDEF.ACM Some requested common decks.

Micronet

Most cities have local numbers for Micronet. This cuts on Ma-Bells bill. The HUG Bulletin Board is on the Micronet system, so therefore the user must get on Micronet first. Micronet itself has many features, to include a large data base. The command IRUN HUG typed just as it appears plus a RETURN will put you on the HUG BB. HAVE FUN.

```
To access these files, return to the MicroNET
command level and type:

COPY FILENAME.EXT[70000,21] TO FILENAME.EXT

                    OR

          TYP FILENAME.EXT[70000,21]
:JB:

<R>etrieve   <L>eave   <S>can   <T>erminate  <D>elete
<U>ser Log   <H>elp    <I>nfo   <X>Data Base <V>Int Log

Your Choice --> T

Logging off:
Jim Blake 70000,21
Thank you for using HUG CBBS
Last message on system-     2837

OK

BYE
Off at 14:46 EST  24-Nov-80
Connect time =    0:50

dropped by host system
please log in:
```

This will tell you how to retrieve files from the Data Base.

Oh no the dreaded Menu again

Terminating

Back in Micronet

Leaving Micronet

Hang the Phone up and go to bed.

EOF

FORTRAN Corner

by James G. Jerling

In this edition of FORTRAN corner, we will divert from the discussion of FORTRAN itself and tell you how to run FORTRAN on a single drive system. The manual says that two or three drives are required, but with a few tricks you can get by with one, as long as your FORTRAN programs are not too big.

The first thing you need to do is to SYSGEN three disks. On each disk put a copy of PIP.ABS, and put ONECOPY.ABS on at least one of them. Put a copy of your favorite editor on the first disk. I recommend the HUG editor (P/N 885-1022) because it is easy to write FORTRAN programs with it. Also on the first disk you will need a copy of F80.ABS, which is the FORTRAN compiler. On the second disk put copies of L80.ABS (the linking loader) and FORLIB.REL (the FORTRAN library). The third disk will be used to store finished programs so they do not use up space on the first two.

When you have prepared your disks, boot up on the first one and use the editor to enter the source for your FORTRAN program. Compile it when it is finished. Watch for error messages while the compilation procedes and edit and re-compile if necessary. Now ONECOPY the compiled program (filename.REL) to the second disk. Boot up the second disk and run the linker by typing L80 filename/M followed by a RETURN. In a little while you will see a list of Library routines required by your program on the screen. This is for your information and is not needed for the rest of the linking process. When this finishes, type in filename/N and a RETURN (you are still in L80), and when that is done type in filename/E and a RETURN. L80 is now converting your .REL file into an absolute file and saving it on the disk. If everything runs satisfactorily L80 will return you to HDOS, and you can run your program by typing its file name.

If you need more disk space you can use STAND-ALONE as described in REMark issue #12. Just RESET when you change disks instead of re-booting. (After you run ONECOPY you will have to re-boot and re-set STAND-ALONE.) This requires a little more memory so you must have a minimum 48k system instead of the minimum 40k normally required. Good luck and have fun with FORTRAN. In the next issue we will get down to some basics and try to amplify some of the more obscure parts of the FORTRAN software manual, or answer some of your questions if we get some.

EOF

Another Pointer to the Type-Ahead Buffer

by Jay H. Gold
P. O. Box 2024
Des Moines IA 50310

I recently obtained the offset values for the type-ahead buffer in HDOS version 1.6 (see REMark issue #11). I wrote an assembly language routine using these numbers, and they worked fine until the system disk was reset during "stand-alone" operation. Resetting the system disk apparently does something to the input buffer and pointers.

The HDOS code documents a way to get around this. In low memory is a word labelled S.DLINK that is the address of an area in high memory called HIGHDAT. The various pointers to the buffer are located at fixed offsets from HIGHDAT, even when you reset the system disk. Below is a list of the offsets. I have included an assembly language program that puts into the type-ahead buffer anything at the label TABLE at the end of the program. A 12Q (newline code) must be inserted between each command, and the table must end with a zero byte. Just replace the commands at TABLE and re-assemble. You can name the program what you want to, and if it has an .ABS extension, just typing its name will cause the list of commands to be executed. This is also an easy way to make PROLOGUE.SYS files.

Type-Ahead Buffer Offsets from HIGHDAT

Address of HIGHDAT Area	040.346
Offset to Line Counter	2
Offset to Queue Tail Pointer	6
Offset to Queue Head Pointer	8
Offset to Pointer to Buffer Start	10
Offset to Pointer to Buffer End	12

The above offsets are in Decimal.

```
* THIS PROGRAM FILLS THE TYPE-AHEAD BUFFER
* WITH COMMANDS SUPPLIED AT THE LABEL "TABLE"
* AT THE END OF THE PROGRAM

* HDOS DEFINITIONS

.EXIT   EQU     0
USERFWA EQU     042200A
$HLIHL  EQU     030211A         LOADS HL WITH WORD STORED AT ADDRESS
*                               POINTED TO BY ENTERING VALUE OF HL
$CDEHL  EQU     030216A         COMPARES REG'S DE AND HL
$TYPTX  EQU     031136A         TYPES TEXT UNTIL HIGH BIT SET

* 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

        ORG     USERFWA

BEGIN2  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             GET BYTE IN TABLE
        INX     H               POINT TO NEXT BYTE IN TABLE
        ORA     A               CHECK IF LAST BYTE
        JZ      DONE            IF IT IS, WE'RE DONE
        STA     SAVCHAR         IF NOT, PUT BYTE IN SAVCHAR
        CALL    PUTIN           PUT IT IN THE BUFFER
        JMP     MOVEM           GET ANOTHER BYTE
DONE    XRA     A
        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
        LHLD    QHPT
        CALL    $HLIHL
        CALL    $CDEHL          IS BUFFER FULL?
        JZ      BUFUL           IF SO, REPORT IT
        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

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

* REPORT BUFFER FULL

BUFUL   CALL    $TYPTX
        DB      07,'Out of buffer space!',210Q
        JMP     DONE
```

```
* STORAGE AREA

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   DB      'CAT/S',12Q,'STAT',12Q
        DB      0
        END     BEGIN2
```

EOF

EDITOR'S NOTE: We have found that the HIGHDAT pointer to the type-ahead buffer is not only independent of system configuration, but it is also independent of the version of HDOS used, at least with 1.6 and 2.0. It is therefore possible to make the SUBMIT program on HUG disk no. 885-1060 version independent by replacing the file STUFF.ACM with one created as follows. First, copy the following files from 885-1060 to an empty diskette: SUBMIT.ASM, HDOS.ACM, ASCII.ACM, CONSL.ACM, and TTIO.ACM. Load SUBMIT.ASM into your editor (from the new diskette) and delete all the lines from the label PROG1 to the line SCALL .EXIT. Also, delete the two lines starting with the lable TABLE, and the line that starts HDOS.04. Change the line MVI M,CR to MVI M,NL.

Now, using your editor, enter Jay Gold's program, giving it the name STUFF.ACM, and make these changes. Do not enter the HDOS DEFINITIONS, the ORG line, or the END line. Change the lable BEGIN2 to PROG1. Replace the lines from LXI H,TABLE to RET with the following.

```
* PUT COMMANDS INTO TYPE-AHEAD BUFFER

        LXI     H,TABLE
MOVEM   MOV     A,M             GET BYTE IN TABLE
        INX     H               POINT TO NEXT BYTE IN TABLE
        STA     SAVCHAR         IF NOT, PUT BYTE IN SAVCHAR
        CALL    PUTIN           PUT IT IN THE BUFFER
        LDA     CHRCNT
        DCR     A
        STA     CHRCNT          UPDATE CHARACTER COUNT
        JNZ     MOVEM           GET ANOTHER BYTE
        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
```

Also, replace the lines at the end of the program with these lines.

```
        * 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
```

When you assmeble what you have created, you will have a SUBMIT program that runs under HDOS 2.0 as well as 1.6. Files created by it will also run under both versions. For example, if you make a file called M1 that mounts SY1:, it will run under HDOS 1.6 and 2.0. I have not tried this program out on 1.5, but it may run under it also. This approach can be used to fix other programs that use the type-ahead buffer.

PS:

CHASE — An Animated Graphics Game

by Tim Stanley
12801 Roma Avenue NE
Albuquerque NM 87123

Here is a game for your H88, H89, or H19 that the kids will have fun with. In this have, the computer plots a small circle and a square on the screen. The circle is moved randomly by the computer, and the square's motions are controlled by the keys on the number keypad. The object of the game is to catch the circle with the square. When you do, the terminal beeps and the computer tells you how many moves (directional changes) you took. Players can compete trying for as few moves as possible.

This program as listed requires MBASIC. To run it on B H BASIC (disk or cassette), change the INP function in line 410 to PIN. If you have an H8 with the H8-5 card, change the argument to INP from 232 to 250.

```
10 REM ***CHASE**** A TAG GAME BY TIM STANLEY FOR THE H88/H89
20 REM
30 E$=CHR$(27):C$=E$+"E":G$=E$+"F":G0$=E$+"G":Y$=E$+"Y"
40 PRINT C$:PRINT :PRINT :PRINT TAB(15);"***CHASE***"
50 PRINT "TRY TO CATCH THE CIRCLE WITH THE SQUARE"
60 PRINT "MOVE THE SQUARE BY PRESSING KEYS 1-9 ON THE NUMBER PAD"
70 PRINT "KEYS 2,4,6,8 MOVE THE SQUARE IN THE DIRECTION OF ARROW ON KEY"
80 PRINT "KEYS 1,3,7,9 MOVE THE SQUARE DIAGONALY. THE 5 KEY STOPS THE SQUARE"
90 PRINT "I WILL TELL YOU HOW MANY MOVES YOU TAKE AND BEST SCORE"
100 PRINT "(HINT: HORIZONTAL OR VERTICAL MOVES ARE ONE SPACE AT A TIME."
110 PRINT "DIAGONAL MOVES ARE ONE VERTICLE AND TWO HORIZONTAL.  THEREFOR"
120 PRINT "FASTEST HORIZONTAL PROGESS RESULTS FROM ALTERNATING UP AND DOWN"
130 PRINT "DIAGONAL MOTION.  BUT YOU MIGHT SKIP OVER THE CIRCLE)"
140 PRINT: PRINT: PRINT "PRESS THE RETURN KEY TO CONTINUE":LINE INPUT "";X9$
150 B=200
160 REM CLEAR SCREEN, TURN ON GRAPHICS MODE, TURN OFF CURSOR
170 PRINT C$;G$;E$+"x5";E$+"q"
```

```
180 REM PICK INITIAL COORDINATES FOR CIRCLE AND SQUARE
190 X0=30+INT(RND(1)*20):X1=30+INT(RND(1)*20):R=INT(RND(1)*8):Y0=8+R:Y1=16-R
200 REM DRAW LINE AROUND PLAYING AREA
210 PRINT TAB(2);"f";:FOR I=3 TO 78:PRINT "a";:NEXT I:PRINT "c"
220 FOR I=3 TO 22:PRINT TAB(2);"`";TAB(79);"`":NEXT I
230 PRINT TAB(2);"e";:FOR I=3 TO  78:PRINT "a";:NEXT I:PRINT "d"
240 FOR I=1TO 200
250 REM ERASE LAST CIRCLE
260 GOSUB 810
270 REM PICK NEW CIRCLE COORDINATES
280 REM MAKE SURE THEY STAY IN PLAYING AREA
290 X0=X0+INT(5*RND(1))-2
300 IF X0<3 THEN X0=3
310 IF X0>78 THEN X0=78
320 Y0=Y0+INT(3*RND(1))-1
330 IF Y0<3 THEN Y0=3
340 IF Y0>22 THEN Y0=22
350 GOSUB 800
360 REM TEST FOR CATCH
370 IF X0=X1 AND Y0=Y1 THEN GOTO 590
380 REM ERASE LAST SQUARE
390 GOSUB 830
400 REM GET DESIRED DIRECTION FOR SQUARE TO MOVE
410 C=INP(232)-48
420 REM IF INVALIDE DIRECTION SPECIFIED STOP SQUARE MOTION
430 IF C<1 OR C>9 THEN C=5
440 REM GET SQUARE COORDINATES BASED ON BUTTON PRESSED
450 ON C  GOSUB 700,710,720,730,740,750,760,770,780
460 REM MAKE SURE SQUARE DOSEN'T GO OUT OF PLAYING FIELD
470 IF X1<3 THEN X1=3
480 IF X1>78 THEN X1=78
490 IF Y1<3 THEN Y1=3
500 IF Y1>22 THEN Y1=22
510 REM DRAW NEW SQUARE
520 GOSUB 820
530 REM TEST FOR CATCH
540 IF X0=X1 AND Y0=Y1 THEN GOTO 590
550 NEXT I
560 PRINT E$
570 PRINT "YOU DIDN'T CATCH THE CIRCLE IN 200 MOVES"
580 GOTO 600
590 PRINT C$;CHR$(7);"YOU CAUGHT THE CIRCLE IN";I;"MOVES"
600 IF B>I THEN B=I
610 PRINT "THE BEST SCORE THIS GAME IS";B
620 PRINT: PRINT G0$;E$+"y5"
630 LINE INPUT " *PUSH RETURN TO CLEAR BUFFER! *";A$
640 LINE INPUT "DO YOU WANT TO PLAY AGAIN? ";A$
650 IF LEFT$(A$,1)="Y" OR  LEFT$(A$,1)="y" THEN GOTO 170
660 IF A$="" GOTO 170:REM    ALLOW CR TO RESTART GAME
670 PRINT "OK LETS PLAY AGAIN ANOTHER TIME"
680 STOP
690 REM SQUARE COORDINATE CHANGE ROUTINES
700 Y1=Y1+1:X1=X1-2:RETURN
710 Y1=Y1+1:RETURN
720 Y1=Y1+1:X1=X1+2:RETURN
730 X1=X1-1:RETURN
740 RETURN
750 X1=X1+1:RETURN
760 Y1=Y1-1:X1=X1-2:RETURN
770 Y1=Y1-1:RETURN
780 Y1=Y1-1:X1=X1+2:RETURN
790 REM ERASE AND DRAW ROUTINES
800 PRINT Y$+CHR$(31+Y0)+CHR$(31+X0);"^":RETURN
810 PRINT Y$+CHR$(31+Y0)+CHR$(31+X0);" ":RETURN
820 PRINT Y$+CHR$(31+Y1)+CHR$(31+X1);"p":RETURN
830 PRINT Y$+CHR$(31+Y1)+CHR$(31+X1);" ":RETURN
```

EOF

BUGGIN' HUG

Dear HUG,

I have a Heath H11A computer and the H19 terminal and H27 floppies. The H19 terminal has some interesting features which I am unable to use with the H11A. For example, in the off-line mode, the terminal can be used to edit text in a way that is simpler than the R-EDIT program in the HT=11 operating system. You could compose and edit a page of text on the H19, but how may this be transferred to floppy disk so this may be used, output to a printer, etc. The H19 Operation Manual on page 89 gives the following under "Additional Functions":

PXMT Transmit Page ESC p
Transmits lines 1 through 24. (The computer requires a special routine to use this feature.)

There are other special features of the H19 terminal which seem to work in the off-line mode and I have not figured out how to use them with the computer. Can I get some help from other HUG members?

Sincerely yours,

Frank Y. Speight
2011 Jackson Heights Dr.
Sebring FL 33870

EDITOR'S NOTE: The sequence Mr. Speight has given in his letter is the ANSI transmit page sequence. The Heath sequence is ESC #. As the book says, you need special software to use this feature. Basically, what you need is some kind of editor that would ignore and echo back all input from the terminal until you entered the Transmit Page sequence (or the software could send it), then accept everything as the page is transmitted, then go back to the ignore mode. Please note that this is not the only way to use the features of the H19 in an editor. A better way is to have the editor handle each feature separately itself by sending the proper sequences itself as needed instead of going off line.

PS:

Dear HUG,

Here is some information for users of Morse 8 (HUG part no. 885-1027). Having trouble with Morse 8 and your new H19 video terminal? One of the penalties of having a "smart" terminal is that some of the commands are processed by the terminal and never get to the computer.

The Morse 8 program uses Control-Shift-P, Control-I and Control-J as part of its command structure. If you look at page 19 of the H19 Operation Manual, you'll see that Control-I and Control-J are processed by the terminal. The following patch will replace Control-I with Control-K, and Control-J with Control-L.

Address	New Data
050 011	000
050 012	000
050 013	357
050 014	356

Control-Shift-P generates a different code on the H19 than on the H9. To get the required code, use Control-@. For those who want to use other control characters or make other changes, here is how Morse 8 processes control characters. An address pointer is loaded with 050 as the high byte and the ASCII value of the character as the low byte. This points to a table of functions. In the patches above, we have assigned the Lock function to address 050.013 (Control-K) and Freeze to 050.014.

With the changes above, Morse 8 will work on your H19

J. S. Gurske
2809 Dewey Court
Middleton WI 53562

Dear HUG,

If you are like me, you may have started your first system with an H8. Then you started to add this or that option to expand your home computer and before you knew it there was always something nicer coming out of BH. Then came the H14, great unit, but alas, they told you that an H8-4 was required, which you could not afford at the time. So you converted your existing H8-5 card and were off and running. Then came the new software with the LP: driver and all its lovely SET

options but the pocket book was already blank. To make a long story short, the following suggestions are offered and will work with either the H8-4 card or H8-5 card, and make life with the H14 more pleasant.

Since the verious SET options (for AT:, LP:, or whatever) are essentially fixed, you can only change print or vertical density with the HDOS SET command, right? Wrong! Sure, it isn't difficult to change things with SET, but it's a different story if you want to make changes while in BASIC or something.

First, I will tell you how to make changes while in HDOS (using single letter commands, if desired) without using SET. The prerequesite is to have the SUBMIT program written by R. Rudell (available on HUG part no. 885-1060 or 885-1075) which allows you to create an .ABS file of a set of instructions to be executed under HDOS. This, by the way, is great if you don't know how to write in anything other than BASIC (like me).

The first step is to use EDIT (HUG's ED will not work) to create files containing the printer commands that you will most likely need. For example, if you want your H14 to print 96 characters per line on the Narrow switch setting, you would do the following:

```
>EDIT
--INSERT
(ESC u)D
(CTRL-C)
--NEWOUT/P96./
--FLUSH
--BYE
>
```

The "ESC" and "u" are in parentheses because they do not echo to your terminal, but they will be contained in the file written to disk. After you make this file, use SUBMIT to create an .ABS file as follows:

```
>SUBMIT
Binary Output File: P96
[1] COPY AT:=P96
[2] (CTRL-D)
>
```

The file created is P96.ABS. I have used AT: in this example, but you could use LP:. Now, when you want 96 characters per line, give the command P96. If you do not have SUBMIT, you can do the same thing with the following:

```
>PIP AT:=TT:
(ESC u)D
(CTRL-D)
>
```

After using either of the above procedures, the H14 will print at the 96 character width in the narrow switch setting until you give another command or turn the power off. When you use the SUBMIT method, you can introduce several printer changes in one file. Another benefit of this method is that the original SET command settings are not disturbed.

You can also make printer changes in the middle of BASIC programs. In the following example, we want to change from 80 characters per line to 132, without using the print width switch.

```
10 E$=CHR$(27)
20 E1$=E$+CHR$(117)+" "
30 OPEN "AT:" FOR WRITE AS FILE #1
40 PRINT #1,E1$
50 REM Whatever follows
60 CLOSE #1
70 END
```

I have used the CHR$ function in line 20 to indicate that this method can be used on an H9. If you have an H19/H89, you change line 20 to read

```
20 E1$=E$+"u "
```

For MBASIC, you would have to change the OPEN statement. Needless to say, any number of print format changes can be made during a BASIC program without ever touching the H14.

Best regards,

G. L. Berendes, Jr.
37 President Kennedy Ave.
Blainville PQ Canada J7C 1T8

(vectored from page 17)

mystery was solved and Murphy was foiled again.

The moral of this story? Store and handle your floppies carefully, and you won't have egg on your hide!

BE:

EDITOR'S NOTE: In the next issue of REMark, Mr. Ellerton will present an article on how to test for the "magic egg", and present some specifics on caring for your floppies.

PS:

The Computer Group is expanding rapidly and we thought a recruiting ad in REMark would be an excellent way to reach a great number of qualified persons. Hope you don't mind us using the back page for this purpose. We won't make it a practice.

JB:

HEATH/ZENITH CAREER OPPORTUNITIES

Heath and the Zenith Computer Business Group, subsidiaries of Zenith Radio Corporation, are growing to meet the microcomputer systems demands of business and the home hobbyist.

If you would like to find out about having a hand in designing the hardware or software used by business or people like yourself, or help support its development, complete the reverse side of this form and mail it to us today!

(fold and tape or staple)

no postage necessary if mailed in the united states

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 363 BENTON HARBOR, MI

POSTAGE WILL BE PAID BY ADDRESSEE

HEATH COMPANY
ATTN: PERSONNEL DEPARTMENT
BENTON HARBOR, MI 49022

(Please Print or Type)

Name _____

Address _____

City _____ State _____

Zip _____

Telephone (_____) _____
Area code Home

_____
Business

Area of Interest

- ☐ Hardware Design: ☐ Digital ☐ Analog
- ☐ Software Design: ☐ Applications ☐ Systems
- ☐ Firmware Design
- ☐ Technical Writing: ☐ Hardware ☐ Software
- ☐ Customer Service

Educational Background: (Field of study, type of degree) _____

_____

Work History:

Current Title: _____

Nature of Work: _____

_____

_____

Previous Title(s): _____

Nature of Work: _____

_____

_____

Special Skills: _____

_____

_____

_____

☐ Call me immediately.

Heath/Zenith
An Equal Opportunity Employer