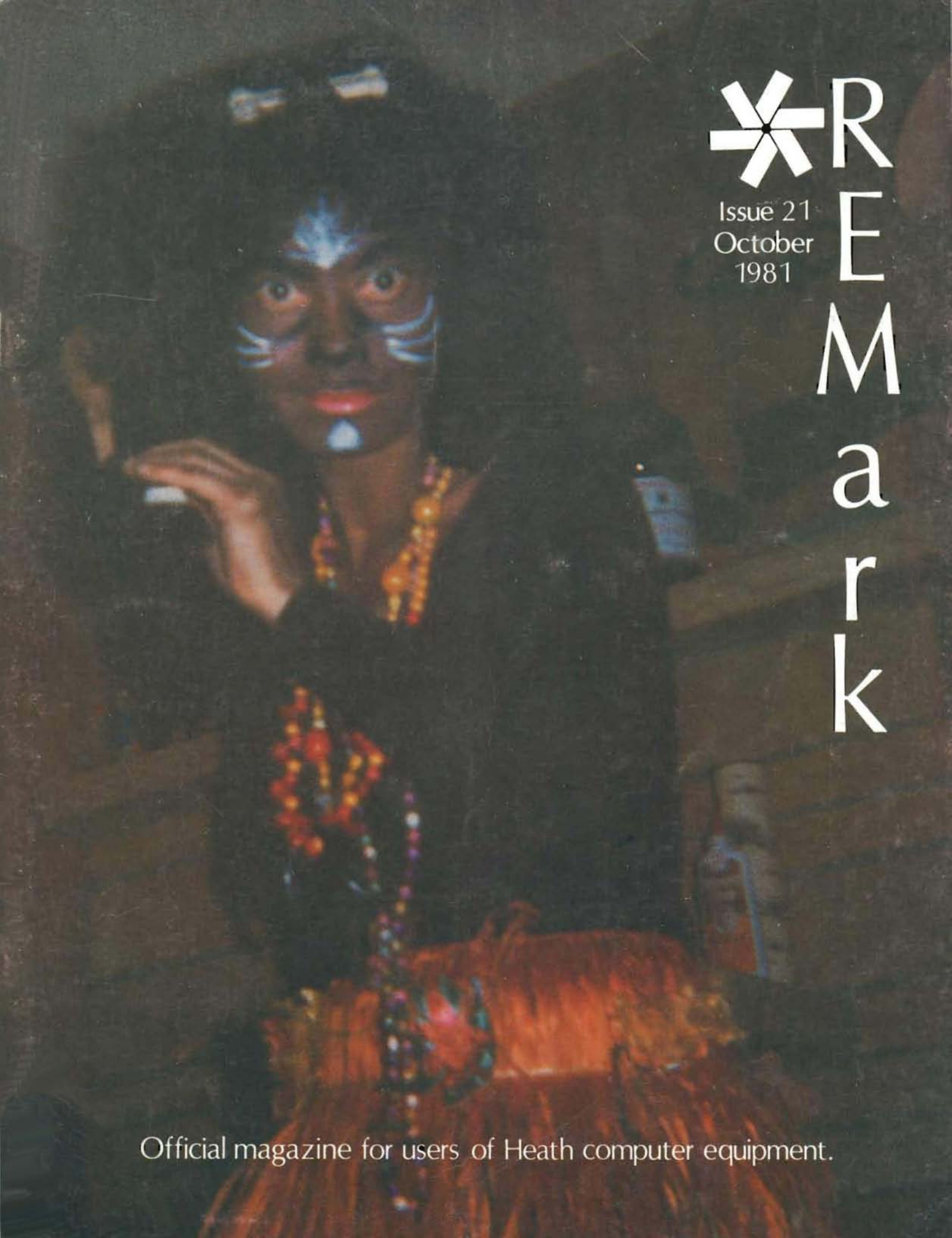


REMark

Issue 21
October
1981

Official magazine for users of Heath computer equipment.

on the cover . . . .

THE TOE!

Photo by: K. Andersen

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

Anyone Can Build a Computer

Several years ago, I had the pleasure of meeting Nino D'Agostino. He's a regular Heathkit customer. He has built small kits and large. The largest was the GR-2001 Color Television, so it wasn't a surprise when he bought the H-8 computer. The surprise to most people is that Nino is totally blind! He has a little trouble reading the color code on resistors, so he does need a little help, but I imagine some day we'll solve that problem too.

Using the computer is no problem to him. He uses a cassette looking device, with which he inserts his fingers in a slot. A lens is attached on a short cable to the machine. When Nino places the lens to the C.R.T., and holds it on a letter, small pins in the slot vibrate under his fingers. It doesn't come out in Braile. What the lens sees, is the shape the pins vibrate. So Nino had to learn how to read print! Can you imagine not knowing what your name looks like?

Nino has an H-8, H-19, H-17, two drives, and the CAT modem. Now he has purchased a Votrax and is currently learning how to use it. Now when he fires up his computer it talks to him! It has opened new doors for Nino, and he's having a ball.

Thanks to Larry Bollman of Heath's Factory Computer Service Department for this information about Nino. Nino is obviously an individual with a lot of heart. GOOD LUCK NINO! From all Huggies and the HUG staff.

Yo! Ho! Ho! and a Bottle of Rum!

HUG has received many inquiries from the various local HUG groups concerning the duplication of the HUG Library for addition to the software that they can offer to other members of the local group. Jim Blake once discussed this situation in a previous issue of REMark. The answer, at that time, remained unclear as to the course of action open to the clubs throughout the country. Several of the "locals" have asked that I look into this situation for a better definition for their membership. Here is what I found:

The software offered by the Heath Users' Group REMAINS THE PROPERTY OF THE AUTHOR! HUG has been given permission by the respective authors to release this software to the National HUG membership ONLY. This means that HUG software cannot be duplicated for any reason without written permission of the original author of the particular program. The release given to the Heath Users' Group and the information supplied in the individual membership package indicated that the original author maintained all rights to his work.

As the software battle rages, more and more of the individuals creating user programs are becoming increasingly "picky" about the work they wish to release to the world. Fortunately, our membership has contributed a wealth of programming knowledge for mere mention of their efforts in REMark knowing that they are supporting the different areas of responsibility of the National HUG. Their programs are being sold at a nominal charge to support the continued development of REMark, the HUG BB, and HUG itself. For those of you that have contributed, we can express the deep appreciation of all HUG members that have benefitted from your efforts.

HUG can not give authorization to duplicate the HUG Library. Further, duplication of the membership library or the software that has been developed by loyal HUGs all over the world may be considered SOFTWARE -------

Heath/Zenith on Foreign Shores

Did you know that there has been pixel graphics on the '89' for one and a half years? Did you know that there are some '89' systems with 160 meg of on line hard disk storage? Did you know that there are many systems with 10 X 10 meg of on line hard disk storage? Did you know

that there have been 380K floppy disks for the '89' for 6 months. Did you know that there are character sets for the '89' and '19' that cover all the major languages in the world, like French, German, Danish, Norwegian, Swedish, Italian, Spanish, Hebrew and even Arabic ????

Well now you know!

Did you know that there is a three port parallel Centronics Compatible I/O card for the '89'? Did you know that there is an eight channel, 12 bit A/D card with software for the '89'? Did you know that there is an IEEE interface for the '89'? Did you know that there is a mod kit for the '19' and the '89' serial card to convert from RS-232 to 20ma active and/or passive? Did you know that there is a complete EPROM/ROM burning package for the '89' computer? Did you know that your Volkswagen was checked at the end of the production line to see if all was well-By a Z89??? (No snide remarks now!!!) Did you know that Autoscribe is in all the above languages?

Did you know that there is even more!??

Well now that I've caught your attention I guess it would be a good idea to introduce myself and then give you all a little explanation of just what I mean by all of the above. I'm new to the Heath Factory but not to Heath, (I've been with Schlumberger and then Heath for a total of almost eight years) just back from four years living and working for Heath International in Europe. I was talking to Bob Ellerton (he's the Editor of this rag) a few days ago when he hit me up to write an article about "Heath on foreign shores". I'd been thinking about an international section in HUG for quite some time but I let him think it was his idea anyway. "Gee Whizz Bob what a great idea !!!"

So here I am, trying to think of how to start, for we have so much to talk about. I guess the first thing would be to let you all know that there really is a Heathkit and Zenith Data Systems beyond the friendly shores of Miami beach, Boston, and Malibu. Heath has been operating in Europe for over 18 years in many countries and in some of the larger countries there are even Heath owned stores, some very similiar to the ones you visit all over the United States. Currently we have Heath/Zenith operations in France, Belgium, Germany, Holland, Sweden, England, and Canada. At the same time we also have distributors in Switzerland, Austria, New Zealand, Australia, Norway, Denmark, Italy, Egypt, South America, the far east, and to tell the truth I'm running out of breath trying to name them all. In any case we are just about everywhere, EVEN CHINA.

Because the computer market is very different outside the U.S.A. (it's hard to justify games when everything costs twice to two and a half times what you pay for the same thing) the Heath/Zenith computer products are really put to work. As an example, with a few changes to the software and a HP bar code reader going via a serial I/O there are now shoe stores all over France that have a cash register that looks like (you guessed it!) a WH89 computer. In Northern Ireland the H8 system has had some ROM's put in the place of memory and now runs and operates complete factories in the area of process control. The Ford motor company (English Ford) has the complete factory and assembly line being controlled by two H8's, in tandom. If you go to a dentist in Sweden and the doctor wants to have a complete medical history of your mouth he will get it from his handy 89/47 combo. If you are Dutch and under the age of 16 you will be learning beginning program theory with an 89/87 system. And, if your country's leader goes by the name of Margaret Thatcher and you are a University student learning beginning to advanced programming; you will know the '89's keyboard and syntax in your sleep. In Germany, the newspaper and magazine you are reading may very well have had the type set by a Z89 and I already told you of the Volkswagen. And, if you are a bullfight enthusiast in Spain your ticket was printed with a (no not a TRSXX) '89' system.

To make a long story short (I didn't get much space this time), we as a family of Heath/Zenith computer freaks, have some family ties just about everywhere in the world. Over the coming issues of REMark I will tell you much more about what is being done in other countries in the areas of work and play, along with interviews with the people that make it happen. So until then, "tally ho old sport".

Best Regards
John R. Beran

Correction to "Losing Weight"

I made a slight boo-boo in my article "Losing Weight with HDOS 2.0" in REMark #19. In the program listing REDUCE.ASM, you should remove the line LXI SP,42200A, which is line no. 44. This line resets the stack before the name of the file to be reduced is fetched, and if a tic count occurs before the name is completely copied, it will be messed up. In a program this small, there is no need to reset the stack at all, so just removing the line will fix it.

PS:

Making Neighbors out of HDOS and CP/M

How often have you found yourself either trying to DBUG a program or simply run a program written under both HDOS and CP/M only to find the wrong operating system installed in your computer? Or, maybe your dream is to have a program that can be run on either operating system on the same disk for comparison! Well, the following text explains how you can build a "friendly" disk containing a directory for both CP/M and HDOS. With a little playing around you will discover some other very useful reasons for making a CO-RESIDENT CP/M and HDOS DISK!

To understand a little about the two disk structures, let's first take a look at what must happen on each disk. When you are "building" the HDOS disk, you must first run INIT which establishes the following on your fresh disk:

DIRECT.SYS, RGT.SYS, and GRT.SYS

Using the utility DUMP from HUG, take a look at the disk after the INIT procedure. You will find that the BOOT information has been written on track 0. Further, HDOS creates the RGT.SYS at track 1 sector 0, the GRT.SYS at track 14 sector 8, and the DIRECT.SYS at track 13 sector 0. The remaining portions of the disk are filled with the ASCII characters "GL" as an alternating pattern to complete the formatting under HDOS. Now, all of this may seem like wasted print but, you will find it to our advantage.

Fortunately, CP/M doesn't use any of the tracks or sectors described above for its' directory. The formatted CP/M is "filled" with E5(HEX) in a continuous pattern with the directory on five-inch disks appearing at track 3. AH HA! Now we know that HDOS does not disturb the CP/M directory track (3), at least during INIT. So, let's see what we can do with this information.

We know that an INITed HDOS disk is filled with "GL" where there is no information that is used by DIRECT.SYS, GRT.SYS, and RGT.SYS. Therefore, the directory track (track 3) for CP/M is now full of these "GL's". Further, we now know that CP/M must locate its' directory at track 3 which must contain "E5's". If we were to somehow write a bunch of E5's to track 3, then by theory, CP/M would then be able to establish a directory here even if the disk was set up using HDOS. If we carry this procedure a litte further, we can establish an area (protected) that CP/M can use and that HDOS will ignore. How do we do this? Easy! First we INIT a fresh disk then perform the MBASIC program described in FIG. A.

```
10 CLEAR 300
20 PRINT "THIS PROGRAM FORMATS HDOS DISKS FOR CO-RESIDENT CP/M USE.":PRINT
30 LINE INPUT "ENTER DRIVE TO BE FORMATTED: ";DR$: PRINT
40 IF RIGHT$(DR$,1)<>":" THEN DR$=DR$+":"
50 OPEN "O",1,DR$+"CPM.SPC": REM ---- CPM.SPC IS THE AREA SET ASIDE FOR CP/M
60 FOR I=1 TO 114: REM ---- SECTOR COUNTER
70 PRINT #1, STRING$(255,&HE5): REM ---- FILL SECTOR WITH 255 HEX E5's
80 NEXT I: REM ---- GET THE NEXT SECTOR
90 CLOSE
100 PRINT "FUNCTION COMPLETED.":PRINT
110 PRINT "ANYMORE DISKS TO BE FORMATTED? <N> ";:A$=INPUT$(1):PRINT A$:PRINT
120 IF A$="Y" THEN PRINT: GOTO 30
130 SYSTEM
```

FIG. A

Let's take a close look at the program and see what will happen. Lines 10, 20, 30, and 40 are designed to be "user friendly" and require little explanation other than line 30 which selects the drive our INITed disk is installed in. Line 50 begins the real "meat" of the program. Line 50 establishes an HDOS file we are calling CPM.SPC (CP/M SPACE). Next (line 60), we provide a FOR NEXT loop of 1 to 114 times as a sector counter (more on this later). Line 70 is the important one. In line 70, we write to the file called CPM.SPC the hex value E5 into each of 255 bytes available in each sector. Then, in line 80 we repeat this write procedure 114 times with the NEXT I statement.

WHY 114 SECTORS YOU ASK?! Well, our new file called CPM.SPC can only occupy an area up to track 13 of the disk. Track 13 is the area were HDOS will normally place its' DIRECTORY. Therefore, we must use caution to preserve this track so as to retain the ability to have both the HDOS DIRECTORY at track 13 and make room for the CP/M directory which CP/M places at track 3. The 114 sector "CPM.SPC" file will then be large enough to accomodate CP/M files that DO NOT exceed 20K bytes of memory.

OK! Now we have created the file called CPM.SPC and we would like to protect this area from HDOS. Easy! Place the "W" flag on CPM.SPC so that it becomes a portion of the system. This little procedure is done with FLAGS.ABS and will reserve this area for CP/M even if you choose to call CPM.SPC by a different name under CP/M. We can now PIP our HDOS or CP/M files (CP/M not to exceed 20K) to the disk and the disk is usable by both operating systems.

OTHER ADVANTAGES ....

One of the other benefits of "building" a CP/M disk using the HDOS INIT and our little format routine is the ability to perform a MEDIA TEST. HDOS performs a media test using the TEST.ABS routine. This is a definite plus over CP/M as it does not. Now we have the ability to test media and create a CP/M disk. Remember, you CAN use the entire disk for CP/M if you desire by simply destroying the directory track for HDOS. CP/M will not care about the information on track 13 as only its' directory track (track 3) requires the E5's.

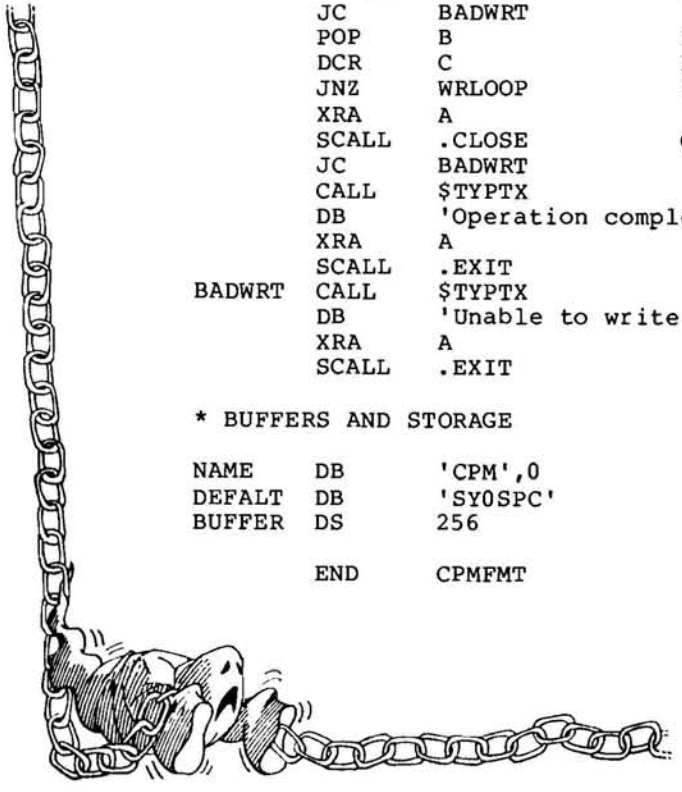
Before we terminate this little session, FIG. B is an Assembly Language version of our BASIC program provided by PS:. (He always likes to do it the hard way.) Anyway, I hope you will find this information is as useful as I did.

EOF BE:

FIG. B

```
        TITLE   'CPMFMT -- CPM DISK FORMATTER'
        STL     'BY PATRICK SWAYNE  31-JUL-81'
* THIS PROGRAM WRITES A 114 SECTOR FILE CONSISTING
* OF 0E5H's.  IF THE FILE IS WRITTEN ON A NEWLY
* INITIALIZED DISK, IT CAN BE WRITTEN ON BY CP/M,
* AND CAN BE USED AS A CP/M-HDOS CO-RESIDENT DISK
* IF THE AMOUNT OF CP/M DATA IS KEPT TO 20K OR
* LESS, OR IT CAN BE USED AS A REGULAR CP/M DISK.

* EXTERNALS

.EXIT   EQU     0
.SCOUT  EQU     2
.WRITE  EQU     5
.OPENW  EQU     43Q
.CLOSE  EQU     46Q
$TYPTX  EQU     195EH

        ORG     42200A

CPMFMT  LXI     H,0
        DAD     SP              FIND STACK POINTER
        MOV     A,L
        CPI     200Q            DRIVE ENTERED?
        JNZ     GOTDRV          YES, GET IT
        CALL    $TYPTX
        DB      'No drive specification entered.',8AH
        XRA     A
        SCALL   .EXIT
```

```
GOTDRV  LXI     D,DEFALT                STORE DRIVE HERE
        MVI     C,3             MAX CHARACTERS IN DRIVE NAME
GLOOP   MOV     A,M
        CPI     ' '             SKIP OVER SPACES
        INX     H
        JZ      GLOOP
        STAX    D               STORE CHARACTER
        INX     D
        DCR     C               GOT ALL OF NAME?
        JNZ     GLOOP           IF NOT, GET MORE

* FILL BUFFER WITH 0E5H'S

        XRA     A               A = COUNT
        LXI     H,BUFFER        POINT TO BUFFER
        MVI     B,0E5H          B HOLDS DATUM
FILL    MOV     M,B             FILL THE BUFFER
        INX     H
        DCR     A
        JNZ     FILL

* WRITE THE FILE

        LXI     H,NAME          GET FILE NAME
        LXI     D,DEFALT
        XRA     A               USE CHANNEL 0
        SCALL   .OPENW
        JNC     WRITE
        CALL    $TYPTX
        DB      'Unable to open file.',8AH
        XRA     A
        SCALL   .EXIT
WRITE   MVI     C,114           WRITE 114 SECTORS
WRLOOP  PUSH    B               SAVE COUNT
        LXI     B,256           WRITE ONE SECTOR
        LXI     D,BUFFER        DATA IS HERE
        XRA     A
        SCALL   .WRITE          WRITE THE SECTOR
        JC      BADWRT
        POP     B               RESTORE COUNT
        DCR     C               FINSIHED?
        JNZ     WRLOOP          IF NOT, CONTINUE
        XRA     A
        SCALL   .CLOSE          CLOSE THE FILE
        JC      BADWRT
        CALL    $TYPTX
        DB      'Operation completed.',8AH
        XRA     A
        SCALL   .EXIT
BADWRT  CALL    $TYPTX
        DB      'Unable to write file.',8AH
        XRA     A
        SCALL   .EXIT

* BUFFERS AND STORAGE

NAME    DB      'CPM',0
DEFALT  DB      'SY0SPC'
BUFFER  DS      256

        END     CPMFMT
```



MBASIC to Machine Code Link Revisited

(HDOS AND CP/M)

In REMark Issue #12 we published an article called "MBASIC to Machine Code Link", in which we presented a way to link from Microsoft BASIC (under HDOS) to a machine code program (.ABS file). However, since that article made no attempt to explain what was going on, and since there is a possible "bug" in the routine presented then, I will do it all over again. Also, since that article came out, we have discovered how to do the same thing in CP/M, which I will present here.

First, let me explain what we mean by "link" in this article. I am not referring to the process of linking two or more relocatable machine code files together to make one runnable program, but rather the process of causing a program to load and execute another program, sort of like CHAIN or RUN in BASIC, except that the second program is in machine code.

PART ONE -- HDOS

In HDOS, there is a built-in system call (.LINK) that can accomplish this. In assembly language, it is used like this:

```
        LXI     H,NAME  POINT TO FILE NAME
        SCALL   .LINK   LINK TO THE FILE
        JMP     ERROR   COULD NOT LINK
NAME    DB      'SY0:FNAME.EXT',0
```

The file name descriptor, "SY0:FNAME.EXT", can contain any legal device name, file name, and extension. You can use .LINK to go from an MBASIC program to a machine code program if you make a USR function that points the HL register at an appropriate file name descriptor, and calls the HDOS .LINK function. This is what the routine in REMark #12 does. The possible "bug" mentioned above is that the routine does not alter the stack pointer, and since .LINK does not change it either, the new program is loaded with the stack at where ever MBASIC had it, which could cause problems. Below is a new routine for linking from MBASIC to machine code programs which not only sets the stack for you, but is more "elegant" than the old routine. It was developed by Greg Chandler, a Heath engineer.

```
2000 '  THESE LINES CAN BE ADDED TO AN MBASIC
2010 '  PROGRAM TO CAUSE IT TO LINK TO A
2020 '  MACHINE CODE PROGRAM.
2030 '
2040 FOR I=1 TO 24:READ B9:'            READ MACHINE CODE DATA
2050 U9$=U9$+CHR$(B9):NEXT I:'          MAKE STRING FROM MACHINE CODE
2060 Z9=VARPTR(U9$):'                   LOCATE STRING PARAMETERS
2070 X9=PEEK(Z9+1):'                    GET STRING ADDRESS LOW BYTE
2080 IF PEEK(Z9+2)>127 THEN@
     X9=((127 AND PEEK(Z9+2))*256+X9) OR &H8000@
     ELSE X9=PEEK(Z9+2)*256+X9:'        CALCULATE USR FUNCTION ADDRESS
2090 DEF USR0=X9:'                      SET USR ADDRESS
2100 '  LINK TO THE MACHINE CODE PROGRAM
2110 PRINT USR0("SY0:SEABATTL.ABS"+CHR$(0));" CANNOT BE EXECUTED"
2120 '
2130 '  MACHINE CODE DATA STARTS HERE
2140 '
2160 DATA &O376,&O003,&O300,&O353,&O043,&O176,&O043,&O146
2170 DATA &O157,&O353,&O041,&O000,&O000,&O071,&O061,&O200
2180 DATA &O042,&O345,&O353,&O377,&O040,&O341,&O371,&O311
```

I have added comments to the routine to make it easier to follow. The DATA statements contain the machine code that will be executed by the USR function. The values of these DATA statements are put into a string (U9$), and VARPTR is used to get the address of that string, which is used as the address for USR0. The only strange part of this routine is line 2080. This is to get around one of the idiosyncrasies of MBASIC. In some cases, if a number will be used as a 16-bit value (even if it is not an integer type, %), MBASIC will not allow the number to be

larger than 32767. In line 2080, we want to multiply the high byte of the string address by 256 and add the low byte to produce the complete address. If the result of the multiplication is greater than 32767, something is flagged internally that causes MBASIC to refuse to assign the value to USR0 (line 2090). But if we AND off the high bit before the multiplication and OR it back afterwards, MBASIC thinks everything is OK.

In line 2110, the file name descriptor to be used by .LINK is a string argument of USR0. MBASIC enters the USR code with the DE registers pointing to three bytes containing the character count and address of the string. Below is an assembly listing of the USR code that shows how this information is used.

```
                        00001   *       THIS ROUTINE IS A USR PROGRAM THAT CAN BE
                        00002   *       INCORPORATED INTO AN HDOS MBASIC PROGRAM TO
                        00003   *       CAUSE A LINK TO A MACHINE LANGUAGE PROGRAM.
                        00004   *       BY G. CHANDLER, HEATH CO.
                        00005
000.040                 00006   .LINK   EQU     40Q
042.200  376 003       00007   MLINK   CPI     3               IS DATA TYPE STRING?
042.202  300           00008           RNZ                     RETURN IF NOT
042.203  353           00009           XCHG                    PUT STRING PARAM ADDR IN HL
042.204  043           00010           INX     H               SKIP OVER STRING LENGTH
042.205  176           00011           MOV     A,M             GET STRING ADDRESS LOW
042.206  043           00012           INX     H
042.207  146           00013           MOV     H,M             GET STRING ADDRESS HIGH
042.210  157           00014           MOV     L,A             HL = STRING ADDRESS
042.211  353           00015           XCHG                    SAVE ADDRESS IN DE
042.212  041 000 000   00016           LXI     H,0
042.215  071           00017           DAD     SP              LOCATE CURRENT STACK
042.216  061 200 042   00018           LXI     SP,42200A       SET STACK TO DEFAULT LOCATION
042.221  345           00019           PUSH    H               SAVE OLD STACK
042.222  353           00020           XCHG                    HL = STRING ADDRESS
042.223  377 040       00021           SCALL   .LINK           TRY TO LINK TO NEW PROGRAM
042.225  341           00022           POP     H               LINK FAILED, GET OLD STACK
042.226  371           00023           SPHL                    SET IT
042.227  311           00024           RET                     AND RETURN TO MBASIC
042.230  000           00025           END     MLINK
```

Note that the program first checks to see if the A register contains the number 3. This is to make sure that the argument to USR0 was a string. Then the address of the string is extracted and saved in DE. Next, the stack pointer is located and its value is saved, and the stack is reset to the default HDOS value, 42200A. The old stack value is pushed onto the new stack, which means that the new program is actually entered with the stack at 042176, but that is still a safe location. The address of the file name is returned to HL, and .LINK is called. If the link is successful, the rest of the code is not used, but if it is not successful the old stack is retrieved and reset and the program returns to MBASIC. NOTE: If you use B H BASIC and want to link to a machine code program, you don't have to bother with the above. To link to SEABATTL.ABS, as in the MBASIC example, use UNFREEZE "SY0:SEABATTL.ABS".

PART TWO -- CP/M

In CP/M there is no .LINK system call for loading and executing one program from another one, but there is a way to simulate it. You can insert a command line into the CCP's (Console Command Processor) command buffer, and then jump directly to the CCP. This method was shown to us by HUG member Marvin Fichter. In assembly, it looks like this:

```
                ;THIS ROUTINE CAN BE USED TO LOAD AND EXECUTE A
                ;MACHINE LANGUAGE PROGRAM FROM ANOTHER ONE.

                LINK:
0000 3A0200             LDA     2               ;GET BIOS PAGE
0003 D616               SUI     16H             ;FIND CCP PAGE
0005 67                 MOV     H,A
0006 2E00               MVI     L,0             ;HL = CCP START
0008 E5                 PUSH    H               ;SAVE IT
0009 EB                 XCHG
000A 1E07               MVI     E,7             ;DE = COMMAND BUFFER
```

```
000C 212500             LXI     H,COMMAND       ;POINT TO COMMAND
000F 0E13               MVI     C,(COMEND-COMMAND) AND 0FFH
0011 7E         LOOP:   MOV     A,M             ;GET A CHARACTER
0012 12                 STAX    D               ;STORE IT
0013 23                 INX     H
0014 13                 INX     D               ;INCREMENT POINTERS
0015 0D                 DCR     C
0016 C21100             JNZ     LOOP            ;LOOP UNTIL FINISHED
0019 E1                 POP     H               ;GET CCP ADDR
001A E5                 PUSH    H               ;SAVE AGAIN
001B 2E88               MVI     L,88H           ;HL = COMMAND POINTER
001D 3608               MVI     M,8             ;SET IT
001F E1                 POP     H               ;GET CCP ADDR
0020 3A0400             LDA     4               ;GET CURRENT DISK
0023 4F                 MOV     C,A             ;PUT IT IN C
0024 E9                 PCHL                    ;JUMP TO CCP

0025 11         COMMAND:DB      (COMEND-COMMAND-2) AND 0FFH
0026 4D42415349         DB      'MBASIC STARTREK',0
0036 =          COMEND: EQU     $

0036                    END
```

The first thing this program does is locate the CCP page address by subtracting 16H from the BIOS (Basic I/O System) page address. This makes this method possibly version dependent, because the size of the CCP may be different for different versions of CP/M. The value 16H may be valid only for Heath/Zenith CP/M version 2.2.02. If you are using Magnolia or DG CP/M, consult your documentation for the size of the CCP and its distance from the BIOS. After the program gets the address of the CCP, it adds 7 to it to get the address of the command buffer. Into this buffer it places the character count of the command, the command line itself, and a trailing zero. Then the program adds 88H to the CCP address to locate the command pointer. It inserts an 8 here indicating that the first command character starts 8 bytes after the start of the CCP. Then the program gets a number representing the current default system disk from location 4 in low memory and puts it in the C register. If we did not do this, CP/M would assume that the default system disk is the hardware system disk. Finally, control is transferred to the CCP, which processes the command line that was inserted and executes the new program.

A machine program can be run from MBASIC in CP/M using the same method. Below is a routine that will do it.

```
2000 '  THESE LINES CAN BE PLACED IN A CP/M MBASIC
2010 '  PROGRAM TO CAUSE IT TO LINK TO A MACHINE
2020 '  LANGUAGE PROGRAM
2030 '
2040 CCP=PEEK(2)-&H16:'                 GET CCP PAGE ADDRESS
2050 BUF=CCP*256+7:'                    GET ADDRESS OF COMMAND BUFFER
2060 COM$="STAT *.*"+CHR$(0):'          THIS IS WHAT WE'RE LINKING TO
2070 POKE BUF,LEN(COM$)-1:'             PUT COMMAND LENGTH INTO BUFFER
2080 FOR I=1 TO LEN(COM$)
2090 POKE BUF+I,ASC(MID$(COM$,I,1)):'   PUT COMMAND INTO BUFFER
2100 NEXT I
2110 PTR=CCP*256+&H88:'                 GET ADDRESS OF COMMAND POINTER
2120 POKE PTR,8:'                       SET IT
2130 IF CCP>127 THEN
     CCP=((127 AND CCP)*256) OR &H8000
     ELSE CCP=CCP*256:'                 GET CCP ADDRESS
2140 DEF USR0=CCP:'                     SET USR0 TO CCP ADDRESS
2150 CCP=USR0(0):'                      EXIT TO CCP
```

Note that the MBASIC routine does not set up the C register with the default system drive as with the assembly program. This means that you must leave the hardware system drive as the default system drive. You can still link to programs on other drives by specifying them in the command line. To run STAT from drive B: in our example, use B:STAT in line 2060. The MBASIC routine follows the assembly routine closely, and does not need more explanation than the comments included. We used the same trick in line 2130 that was used in the HDOS version, in case the CCP address is above 32767. If the attempt to link fails, control is returned to CP/M instead of to the MBASIC program, as with the HDOS version.

In order to use the linking routine in CP/M, you must protect the CCP from being overwritten by MBASIC. This can be done by setting the top address used by MBASIC to 1600H bytes below the BIOS (in the case of Heath CP/M) with the /M switch at start up. You can calculate the correct address by using DDT to Display the data at location 2. Then use the H command to subtract 16 (hex) from that value, and use the result as the high byte of an even page address. If you have 64k, that address will be D400H, and if you have 48k, it will be 9400H (Heath CP/M only). For example, if you have a 64k system, you would load MBASIC with

```
MBASIC /M:&HD400
```

This will set up MBASIC with the CCP preserved.

PS:

More Changes to HDOS 2.0 Bootup

In REMark Issue #16, I presented some patches for HDOS 2.0 to eliminate the need to type RETURN after <BOOT> and after the date prompt at boot-up. In this article, I will present alternate ways of doing the same thing. If you already patched your disks using the old methods, and decide you like these ways better, you should first use the "old values" in REMark #16 to restore your disks to the original configuration.

1. Eliminate CR after <BOOT>

If you boot up a standard HDOS 2.0 disk and do nothing when the <BOOT> prompt appears, it will eventually continue the boot process because there is a time delay that waits about a minute for you to respond, then continues without a response. This patch will shorten that delay to one or two seconds, eliminating the need to hit RETURN, but still giving you the opportunity to type I for Ignore or C for Check (you have to be fast, though). Use DUMP (from 885-1062) to make the patch.

TRACK 0 SECTOR 2

LOCATION	OLD VALUE	NEW VALUE
2C	3C	02 or 03

2. Eliminate CR after date

The patch given in REMark #16 caused HDOS to print whatever date was stored in memory or on the disk (if no date was in memory) at boot-up, but not wait for confirmation or a new date. This patch eliminates the date prompt altogether at boot-up if there is a date in memory, but prompts for confirmation (with RETURN) or a new date if there is not. This means that the first time you boot up for the day after turning on your computer, you will be prompted for a date, but every time you boot up afterwards you will not.

TRACK 2 SECTOR 0

LOCATION	OLD VALUE	NEW VALUE
2F	CA	C8
30	4C	00
31	31	00

You may want to use this date patch for your working disks, and the other one for your game disks, since having a curren. date is not important on them.

PS:

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. NOTE: HUG Tiny Pascal has recently been updated to a newer version. Before you make this patch, be sure that the "Old Data" in your version matches that which is given here.

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:

Binary Search Routine for MBASIC

William N. Campbell, M.D.
885 Smithbridge Road
Glen Mills, PA 19342

A "Binary Search" is the RAPID acquisition of a desired record from a file starting with a "key word", and using a familiar algorithm. The algorithm used in a "Binary Search" is similar to the method we frequently use to "look up a phone number" or to "find a desired word in a dictionary". The "key word" in a mailing list file might well be a person's last name.

There are 2 prerequisites. The records in the random file to be "searched" must be "ordered" (alphabetized, in a file of names, for example). And, the records must be "packed" (there must be NO blank or empty records in the random file). I have previously presented programs to create and maintain such an ordered, packed random disk file (such as a mailing list) in REMark, issue 10.

The accompanying 2 programs (BIN64.BAS and BIN85.BAS) written in Heath's MBASIC, will find ANY desired record in a random, ordered, packed disk file containing 1000 names and addresses in 7 seconds or less. The only difference in the 2 programs is that BIN64.BAS is "fielded" as 4*64 (4 records in each 256 byte sector) and contains "Remarks" to explain the logic of the program, whereas BIN85.BAS is fielded as 3*85 (3 records in each 256 byte sector) and contains no Remarks. BIN64.BAS can be used "as is" to search a random, ordered, packed disk file created and maintained as per the random file article in REM, issue 10.

As written, the routine will extract and present for your inspection ALL the records which "match" (are identical with) the desired "search word". For example, if you have 5 records in a name and address file, with identical last names (the last names must be in the first field of each record), then the program will find ALL 5 records and display them for you.

The programs also print out the absolute position of the record (the record "number") for your inspection.

```
 10 REM   BIN64.BAS   BINARY SEARCH ROUTINE - FINDS RECORD(S) USING LAST NAME
 20 REM    Note file must be alphabetized in ascending order and packed!!
 30 REM    Takes 7 seconds or less to find any record by last name in 1000 recs
 40 REM    MUST be random file, fielded as 64*4, last name first in records
 50 REM    Will find ALL records with same last name.
 60 REM    Assumed records created as per random file article in REM, issue 10
 70 REM
 80 CLEAR (1000):DEFINT A-Z:REM  clear string space & declare num var as integers
 90 PRINT "Enter complete name of random, ordered file to be searched"
100 LINE INPUT "(Example - SY1:TEST.DAT).....? ";P$
110 OPEN "R",1,P$
120 PRINT:PRINT "Type desired last name to be searched for using ALL CAPS";
130 PRINT ", then hit return.":REM  assuming last name all CAPS in file.
140 PRINT "If done type 0 (zero) and return"
150 LINE INPUT "Enter now, then hit Return.....? ";X$
160 IF X$="0" THEN 480
170 IF LEN(X$)=0 THEN 120:REM  If hit ONLY Return by mistake, length is zero.
180 X$=X$+"\":REM  tack on delimiter - make sure last name ended for search
190 REM Next 5 lines get total # of records in file and put in H
200 FOR I=0 TO 3:FIELD 1,(I*64) AS D$,64 AS R$(I):NEXT I
210 S=LOF(1):REM  S contains # sectors in file. Next 3 lines find last record.
220 GET #1,S
230 FOR H=1 TO 3:IF ASC(LEFT$(R$(H),1))=0 THEN 240 ELSE NEXT H:S=S+1:H=0
240 H=H+((S-1)*4):REM  H now contains highest numbered record in file.
250 L=1:REM  L = lowest and H = highest numbered records
260 N=INT((L+H)/2):REM  divide # of records by 2
```

```
270 IF N=0 THEN 360
280 GOSUB 460:REM  GOSUB examines record # N and puts rec contents in Y$
290 PRINT "Upper =";H;"  Lower =";L;"  Looking at rec #";N:REM  educational line
300 REM  next line does matching and makes sure we are comparing last names only
310 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 370:REM if <>0 found matching name!
320 IF L>=H THEN 360:REM   if L>=H then record not found
330 IF Y$>X$ THEN 350:REM Note if Y$<X$ we drop to 340, if Y$>X$ then 350
340 L=N+1:GOTO 260:REM   reset lower limit, then go back and divide by 2.
350 H=N-1:GOTO 260:REM  reset upper limit, then go back and divide by 2.
360 PRINT:PRINT MID$(X$,1,LEN(X$)-1);" is not in file":GOTO 120
370 REM  now we go back, record by record looking for identical last names
380 N=N-1
390 IF N=0 THEN 420
400 GOSUB 460
410 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<> 0 THEN 380:REM if still match goto 380,
420 N=N+1:REM           else end search and print record(s).
430 GOSUB 460
440 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 450 ELSE GOTO 120
450 PRINT:PRINT "Record #";N;"is ";Y$:GOTO 420
460 S1%=INT((N-1)/4)+1:S2%=N-4*(S1%-1):REM setup, field, get rec, then Return
470 FIELD #1,((S2%-1)*64) AS D$,64 AS R$:GET #1,S1%:Y$=R$:RETURN
480 CLOSE:PRINT:PRINT "Bye-bye!":END
```

```
10 REM    BIN85.BAS       binary search routine - finds record using last name
20 REM    note that file must be ordered and packed!!!
30 REM   fielded as 85*3  Takes 7 seconds to find any record in 1000 records
40 REM   Will find ALL records with same last name.
50 REM
60 CLEAR (1000):DEFINT A-Z
70 PRINT:PRINT "Type desired last name in the ordered random file. USE ALL CAPS"
80 PRINT "If done type 0 (zero) and return"
90 LINE INPUT "Enter now..... ";X$
100 IF X$="0" THEN 400
110 IF LEN(X$)=0 THEN 70
120 X$=X$+"\"
130 P$="SY1:FMLST.DAT"
140 OPEN "R",1,P$
150 FOR I=0 TO 2:FIELD 1,(I*85) AS D$,85 AS R$(I):NEXT I
160 S=LOF(1)
170 GET #1,S
180 FOR H=1 TO 2:IF ASC(LEFT$(R$(H),1))=0 THEN 190 ELSE NEXT H:S=S+1:H=0
190 H=H+((S-1)*3)
200 L=1
210 N=INT((L+H)/2)
220 IF N=0 THEN 290
230 GOSUB 380
240 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 300
250 IF L>=H THEN 290
260 IF Y$>X$ THEN 280
270 L=N+1:GOTO 210
280 H=N-1:GOTO 210
290 PRINT:PRINT MID$(X$,1,LEN(X$)-1);" is not in file":CLOSE:GOTO 70
300 N=N-1
310 IF N=0 THEN 340
320 GOSUB 380
330 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<> 0 THEN 300
340 N=N+1
350 GOSUB 380
360 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 370 ELSE CLOSE:GOTO 70
370 PRINT:PRINT "Record #";N;"is ";Y$:GOTO 340
380 S1%=INT((N-1)/3)+1:S2%=N-3*(S1%-1)
390 FIELD #1,((S2%-1)*85) AS D$,85 AS R$:GET #1,S1%:Y$=R$:RETURN
400 PRINT:PRINT "Bye-bye!":END
```

EDITOR'S NOTE: Doc just called to say that if any of you would like a copy of his article presented in Issue 10 of REMark, please send him two disks along with return postage. His disks will have the article and corrections from Issue 11 of REMark. Send your disks to: Doc Campbell; 885 Smithbridge Road; Glen Mills, PA 19342.

CP/M??

As members of HUG, virtually all of you are using HDOS and are probably getting very proficient with its procedures. To many of us (HDOS users), there is no other operating system that we wish to use because HDOS is such an excellent user oriented operating system. However, as we look through computer magazines and available software we realize that there must be something to that ever popular CP/M(tm) operating system.

Control Program for Microcomputers (CP/M), one of the first sophisticated microcomputer operating systems, can easily be a part of HDOS users environment without sacrificing or giving up the operating system that we have grown so fond of. It is not the intent of this article to start any kind of battle or "discussion" of the pros and cons of one system verses the other. CP/M is the most widely used operating system in the microcomputer world and many of us should become more aware of it, because sooner or later all Heath users will most likely encounter this popular system.

Because CP/M is a major part of the microcomputer world, we are going to begin a series of articles on how to use CP/M. The articles will continue, based upon the response of you, our users. Our intent is to familiarize anyone, who is interested, in learning and understanding more about the CP/M operating system.

At this point, it is assumed that you are aware that an operating system is the "interface" or "link" between the computer and the user. An operating system must be able to handle input and output of data from the CPU to any and all peripheral devices. It must also provide file management, to make that job independent of the programmers responsibilities. Loading and execution of user programs is the last of the interfacing capabilities that an operating system must be able to handle.

CP/M was written to accomplish these functions for 8080, 8085, and Z80 based microcomputers. Gary Kildall, the author of CP/M, with the aid of John Torode, completing the disk controller, created the first CP/M disk operating system. Thus began the era of CP/M. In 1976, Mr. Kildall started Digital Research and since that time Digital Research has offered more advanced versions of CP/M and added related software products to aid the user of the CP/M operating system. The greatest advantage of the CP/M operating system is not the sophistication of the system or the availablity of software, but rather the compatiblity between different manufacturer's microcomputers. Thus the popularity of CP/M.

For those of you who are just beginning in microcomputers, it will be good for us to stop and explain the different levels of computers and how they interrelate. You can find these facts in most any computer book, but we will go through them here so as not to be guilty of leaving someone behind.

The most important step is to show the software hierarchy of microcomputers. The two systems, CP/M and HDOS, are shown together, NOT as a pro/con comparison, but to familiarize us that the two systems relate. See CHART ONE.

The intent of CHART ONE is to show the different levels of computer programs and to imply how each level affects the user of a respective level. The user based program, the HIGHEST LEVEL, requires the least amount of knowledge to run the programs. Many of these jobs are "turnkey" operational, which allows anyone, be it secretary or game enthusiast, to run the programs with little or no knowledge of computers. The higher level languages of interpreters and compilers, allow a programmer to be concerned with meeting the needs of a "client" or a specific program, rather than needing to know what the CPU and operating system are doing with the "data" that they enter.

The transient program area, the area that includes user programs that directly support or relate to the operating system in general and the microprocesser in particular, is the area in which any serious programmer will become familiar. With microcomputers, most (if not all) of you use many of these transient programs everyday to do your "normal" operations.

Because of the design expertise necessary to create an operating system, this level is the lowest level that most of us will ever become familiar, and then only

slightly . We may do patches to or create device drivers for the operating system, but actually know very little about the operating system. The designing of the processors, the LOWEST LEVEL, and writing of the interpreters and compilers are left to a few select individuals.

HIGHEST LEVEL User based programs	1) Small Business Package	2) Education Package
	3) Data Management Systems	4) Mailing Lists, etc.
Interpreters and Compilers	1) BASIC	2) FORTRAN
	3) PASCAL	4) COBOL
Transient programs of system	1) PIP	2) ED 3) DDT
	4) SYSGEN	5) INIT or FORMAT 6) ASM
The Operating System	1) Heath Disk Operating System (HDOS)	
	2) Control Program for Microcomputers (CP/M)	
LOWEST LEVEL	1) The CPU (ROM)	

CHART ONE

We may learn about and "touch" the lower levels, however, virtually all the work we do is an output to the highest level. Even though the programs are written to help users of that level, doesn't that actually include all of us? Each level has its purpose and the users to support and aid in contributing to the other levels. Isn't that what it is all about?

At this point, the general overview of the operating system has been given. If you have specific questions about the history of CP/M or need more information on the general use of microcomputers, it is suggested that you research any of a number of 8080, 8085 or Z80 microprocessor books.

With that behind us, we will begin with our multi-part guide to using CP/M. We will assume very little, so as to explain to the very beginner;

PART I: How to get started with CP/M?

The very basics of getting started with CP/M, is not as easy to the beginner as it may appear unless the beginner has talked to someone who is familiar with the procedure for setting up the operating system. If you have purchased an H89/Z89 or H8 system, the chances are that you are unable to run CP/M on your computer. There is a hardware modification that must be made to your machine to allow CP/M to be recognized as the operating system. An explanation why CP/M will not run on a "regular" Heath computer will be given in another PART. What is important to know, is that you must purchase an "ORG 0 (ZERO)" kit to execute CP/M, unless your machine has this "Extended Configuration" on it already.

The first step for the beginner is to research his system to find if he does have the "Extended Configuration Option" on his computer. There is a difference between the H89 and the H8, so we will look at each one separately.

The H8 owners have two options to purchasing the "Extended Configuration". You

TO PAGE 28

New Brew Review

885-1212 CP/M Utilities I $ 20.00

This disk is a collection of programs for the Heath/Zenith CP/M user. It contains the following:

DISASM -- This is an intelligent 2-pass 8080/Z80 disassembler. It makes labels at all jumps and calls within the program, and can optionally add "comments" based on the ASCII value of the data being disassembled. Its output can go to your console or printer, or to a disk file. It requires a HEX file for input, which you can make with the UNLOAD program described below. You can also convert an HDOS program to hex using IHEX from 885-1089, transfer the result to CP/M with HTOC (below), and disassemble it.

UNLOAD -- This program is the opposite of the CP/M LOAD program. It converts a COM file to an Intel-type HEX file. You can use UNLOAD to prepare files for DISASM, or for making ASCII files from COM files for transmission over a modem. The CP/M LOAD program can be used to convert the hex files back to COM files.

HTOC -- This is an improved version of the H8COPY program (for copying files from HDOS to CP/M) that was released on disk no. 885-1207. It has been modified to read 8-inch (single side single density) HDOS disks as well as 5-inch HDOS disks. You can specify any drive B through E for the HDOS disk, and any drive A through E for the CP/M disk, so you can copy files from 5-inch HDOS to 8-inch CP/M and vice-versa. Note: the source for this program is not included on this disk due to lack of space, but 885-1207 has been updated, and has the HTOC source.

MPLINK -- This is an H19/H89 version of a popular public domain modem program. It features automatic log-on, file save and transmit, and optional XON recognition. It makes use of the H19/H89 function keys and the 25th line.

HSORT -- This is a CP/M version of the popular HUG SORTER program from 885-1044. It reads in an ASCII file, alphabetically sorts it by lines, and writes the result to an output file.

ONECOPY -- This is a single drive copy utility for CP/M. Although Heath CP/M provides for single drive copying, you can only do it if your system is configured for one drive. With ONECOPY, you can copy with one drive even though your system has several drives. It allows you to copy files larger than memory by prompting you to swap disks.

ERRORS -- This program reports the number of soft errors on your 5-inch disks since the last cold boot. It helps you monitor the condition of your drives.

All of the above programs require CP/M version 2.0 or higher, and at least 32k of memory. All programs include source except DISASM and HTOC.

SOFTWARE UPDATE

885-1078 HDOS Z80 ASSEMBLER $ 25.00

The HUG Z80 Assembler has been updated to a completely new version. It is now fully compatible with the HDOS 2.0 Assembler, including cross reference capability and PIC code handling. The disk includes two versions: one with octal output and one with hex output. This assembler uses extended Intel mnemonics, which means that all 8080 instructions use the same mnemonics as the Heath assembler, and all Z80 instructions are like 8080 mnemonics. This allows you to assemble existing programs with this assembler without modification. The documentation included cross references this assembler's mnemonics to Zilog mnemonics. This program requires HDOS and at leas 32k of memory.

HUG Products List

Part Number	Description	Selling Price
	CASSETTE SOFTWARE (H8 and H88)	
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
885-1039	WISE on Cassette H8 Only		$ 9.00
885-1057	Tape V	Cassette	$ 9.00
885-1058	Volume V Documentation and Program Listings		$ 12.00

HDOS SOFTWARE (H8/H17 or H89 -- 5-inch only)

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 (H19/H89)		$ 18.00
885-1068	Graphic Games (H19/H89)	*	$ 18.00
885-1088	Graphic Games (H19/H89)	*	$ 20.00
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	*	$ 20.00
885-1096	Action Games (H19/H89)	*	$ 20.00
885-1103	Sea Battle Game (H19/H89)		$ 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	Fixed Point Package H8/H89		$ 18.00
885-1075	HDOS Support Package H8/H89		$ 60.00
885-1077	TXTCON/BASCON H8/H89		$ 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
885-1089	MACRO, CTOH, and misc Utilities		$ 20.00
885-1092	RDT Debugging Tool H8/H89		$ 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 H8/H89		$ 19.00
885-1059	FOCAL-8 H8/H89		$ 25.00
885-1078	HDOS Z80 Assembler		$ 25.00
885-1085	PILOT Documentation		$ 9.00
885-1086	Tiny Pascal H8/H89		$ 20.00
885-1094	HUG Fig-Forth H8/H89 2 Disks		$ 40.00

BUSINESS, FINANCE AND EDUCATION

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

AMATEUR RADIO

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

* Means MBASIC is required

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 (5-inch only)

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

The above CP/M products are 2 disks each.

885-1207	TERM and H8COPY		$ 20.00
885-1208	HUG Fig-Forth H8/H89 2 Disks		$ 40.00
885-1209	Dungeons and Dragons Game MBASIC and H89 or H8/H19		$ 20.00
885-1210	HUG Editor		$ 20.00
885-1211	Sea Battle Game for CP/M		$ 20.00
885-1212	CP/M Utilities I		$ 20.00

% Means CP/M 1.43 only (ORG-4200)
%% Means CP/M 1.43 or 2.2 (Heath)
Other CP/M disks are for 2.2

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.

Interface Your ET/ETA-3400 to the SS-50 Bus

By: George H. Kelm
P.O. Box 160
Yap, Caroline Is., TT
Guam, IS 96943

This article describes how to interface a Heathkit Microprocessor Trainer ET-3400 and a Heathkit Memory and Input/Output Accessory ETA-3400 to a SS-50 buss. It provides the ET/ETA-3400 owner with suggestions on hardware and software requirements necessary to expand the units into a more useful and flexable system.

Almost everyone who has the ET/ETA-3400 MUST have, at one time or another, thought about expanding it into a bigger more useful system. The following article will describe how I interfaced my units to the SS-50 Buss.

The entire project, because of the money and the time that's involved, took over a year and consisted of the following parts:

1. Readdressing the Trainer's RAM ICs 14-17
2. Modifications to the ET/ETA-3400
3. Construction of a wirewrap interface card
4. Software rewriting

The below items are needed to complete the project:

1. Heathkit ET-3400 Microprocessor Trainer
2. Heathkit ETA-3400 Memory I/O add-on
3. SS-50 Motherboard and Power Supply
4. Memory card(s), etc.
5. LOTS of time, patience and some money!

READDRESSING THE TRAINER RAM ICs 14-17

This part of the mod is really not required, but it's an easy way to start, and you gain .5K of RAM to be used for scratchpad and stack. My thanks to James Greger for his help with this and the RE line modification.

Those of you who have both the trainer and the add-on know that when you purchase the add-on, Heath tells you to pull ICs 14-17, and not to reinsert them when you are using the ETA-3400 as this would mean that the trainer is addressing two RAMs in the 0000-01FF(HEX) area. To change the RAM addressing, CUT THE TRACE THAT CONNECTS IC-3 PIN 13 AND IC-2 PIN 1, THEN RUN A JUMPER FROM IC-3 PIN 13 TO ONE OF THE IC-2 PINS AS SHOWN IN FIG. 1. I used A000-A1FF(HEX) as this is the address Southwest uses. You should note that if you ever would like to run the trainer by itself, you will have to install a SPDT switch so you can readdress these RAMs back to 0000-01FF(HEX).

BY THE WAY, if you don't know how to tell pin 1 of an IC from a capacitor, you SHOULD NOT TRY THESE MODIFICATIONS or at least, have assistance from someone who does!

THE ET-3400 "RE" LINE

There seems to be a lot of misunderstanding about the uses of the RE line. This line controls a set of bi(two)-directional buffers which allow the CPU to either READ from RAM or another address (port) or, by changing the direction of the buffer, to WRITE to RAM or another address. When the RE line is low, the buffers are in the READ direction, when the RE line is high, the buffers are in the WRITE direction.

The RE line is required by the ETA-3400 add-on, and the line is brought out so the ETA-3400 can control the line and turn the buffers in the direction the ETA-3400 needs for proper operation. The problem here is if you try to add additional memory cards and you tie into the RE line at the Trainer's 40 pin connector, you will have two or more RAM's all trying to fight for control of the RE line. If however, we move the diode to each memory card, then each card will be able to use the RE line correctly.

This is also easy to modify. REMOVE THE DIODEAND THE WIRE THAT HEATHKIT HAS YOU INSTALL IN THE TRAINER, AND REPLACE IT WITH A WIRE BETWEEN THE SAME PINS. THIS IS FROM THE RE CONNECTOR TO PINS 6 AND 35 ON THE 40 PIN CONNECTOR. SECOND, OPEN THE ETA-3400 AND CUT THE TRACE RUNNING FROM THE 1C-107 PIN 1 TO PINS 6 AND 35 ON THE 40 PIN CONNECTOR. THEN INSTALL THE DIODE YOU REMOVED FROM THE TRAINER OVER THE CUT ON THE TRACE. THE DIODE

SHOULD BE INSTALLED WITH THE BANDED END TOWARD IC-107. WHILE YOU HAVE THE ETA-3400 OPEN, CUT THE TRACE BETWEEN PINS 15 AND 16 OF THE 40 PIN CONNECTOR. THIS LAST CUT WILL FREE PIN 26 FOR THE NEXT STEP.

VMA LINE REROUTING

In the ET/ETA-3400 system, the VMA line is ANDed with the 02 line by IC-5 and run to the ETA-3400 as the VMA,02. The SS-50 BUSS requires a separate VMA line. TO DO THIS, CUT THE TRACE IN THE TRAINER BETWEEN PINS 15 AND 16 ON THE 40 PIN CONNECTOR. This leaves the 02 line on pin 15. LAST, RUN A WIRE FROM PIN 26 ON THE 40 PIN CONNECTOR TO THE VMA OUTPUT CONNECTOR ON THE UNDERSIDE OF THE TRAINER.

That completes the mods to the trainer and add-on. Now our 40 pin connectors have pinouts as shown on FIG. 3. The original pinouts are shown in FIG. 2.

SYSTEM CHECKOUT

Well, if you've stuck with this so far, you'll want to be sure that you didn't harm anything in any of the steps so far. Locate those 2112's that were supplied with the ET-3400 and the course. Insert them in the IC sockets on the face of the trainer. Next reconnect the 40 pin ribbon cable between the ET-3400 and the ETA-3400, then close everything up after final inspection for loose wires, or any other problems. Power-up the system, and use your ET-3400 memory exam/change keys or the terminal to look at addresses A000-A1FF(HEX). If you've done everything OK, then you should see good memory at these locations.

If you have a set of memory tests, run the tests on the addresses A000-A1FF(HEX). If you have not yet purchased a set of memory tests, use your "SLIDE" control and SLIDE out of ROM, the memory test at 1A34-1A8E(HEX) in the ETA-3400. I'd suggest you relocate it starting at 0134, by punching in SLIDE 1A34,0134,FF (CR). Next, use your memory exam and change the following:

New Address	From	To
0181	CE 1000	CE 0100
0168	CE 00DF	CE A1FF
016B	8C FFFF	8C A000

Before starting the memory tests, use your memory exam/change and set all of the A000-A1FF(HEX) addresses to 00. Lastly, jump to the memory test which now starts at 0134(HEX) by typing in G 0134 (CR). The altered memory test program will now check your new memory.

WIREWRAP CARD CONSTRUCTION

Cut a piece of perf-board to 5" X 9" (SWTPCO standard for SS-50 buss cards) and use 5 minute Epoxy to mount 5 each of the 10 pin MOLEX female connectors on one side (the 9 inch side) of the card. Purchase or fabricate two 40 pin connectors with the same pin spacings as the ones you see on the trainer/add-on. I used two 40 pin wirewrap IC sockets to make a connector by carefully cutting each of the sockets down the middle, and gluing what was the outside of the connectors together. Mount the connectors on the top edge of the board with Epoxy, and finally, mount a 14 pin IC socket in the middle of the board.

Once the Epoxy is hard, you should use a fine felt-tipped pen to label one of the 40 pin connectors as "Trainer" and the other as "Add-on". Transfer, to the card, all of the pin numbers and uses shown in FIG 4 and list 1.

Lastly, the big job, is to wirewrap the board following the connections listed in FIG 4. I'd suggest you use one color of wire for the data lines, another color for the address lines, and so on. This makes it much easier later if you have to correct mistakes.

CHECKOUT (AGAIN)

Once you are sure you have the wirewrap board wired correctly, insert a 7404 in the IC socket on the board. Now disconnect the 40 pin cable connecting the trainer to the add-on. Connect one end of the 40 pin cable to the connector on the board marked "Trainer", and the other end to the trainer itself. Connect a second 40 pin cable to the connector on the board marked "Add-on", and the free end to the add-on itself. Now TRIPLE-CHECK that you have the cable plugs correct, that is, pin 1 on the trainer should connect to VMA,02 on the card, etc. It's easy to get these connectors backwards as there isn't any index pin to prevent you from plugging it either way!!

Finally, power-up the system WITHOUT THE CARD CONNECTED TO THE SS-50 BUSS. You should find that the trainer and the add-on will operate as before you started with the exception that you will now have the addition of the Trainer RAMs at A000-A1FF(HEX). If the units don't operate, you have a possible wiring error on the wirewrap card. DO NOT GO ANY FURTHER until you correct the problem!!!!

GETTING THE SS-50 BUSS GOING

I used Thomas Instrumentation memory cards, which are the same pinouts as the Southwest System shown on list 1. However, before you connect the

ET/ETA-3400 to your SS-50 buss, it would be a good idea to go back and recheck your pinouts and signal requirements against those listed. My system only required the inverting of the VMA line and the 02, but yours may be different, and in that case, there are several unused inverters on the 7404 for you to use.

SS-50 BUSS LINES

Listing #1 gives you a brief description of the pinouts, names, and definitions of the SS-50 buss used by Southwest and most companies using the SS-50 buss. Once you have rechecked your memory card requirements against it, you can plug your wirewrap card into the SS-50 buss. Make sure to plug the INDEX pin to prevent insertion of the card into the buss incorrectly.

With your wirewrap card plugged into the SS-50 buss, but without any other cards plugged into the buss, power-up the ET/ETA-3400 and check to be sure it operates normally. IF NOT, check the SS-50 buss and correct the problem.

Once you pass this test, set your memory card addressing switches or jumpers to any address between 2400 and 8000(HEX), and plug the memory card into the buss. Before you do so, check to make sure there is a plug in the correct hole of the card marked INDEX. Power-up the SS-50 buss, but not the ET/ETA-3400 and check the voltage across the +12V, -12V, and +8V lines to ground (GND). These voltages can vary by + or - 20% and still be acceptable. Next, check the voltages on your memory cards. These should be + or - 5% for the card to work correctly. If you find any high or low voltages, be sure to correct them before going to the next step. If you have a friend who has an SS-50 buss computer, the ideal set would be for you to ask his help, and if possible, have him test your memory boards in his computer.

THE "RE" LINE (AGAIN)

Almost done! Consult the data that came with your memory card, and locate the data buffers on the schematic. On most cards there will be two buffers, one for D0-D3, and the other for D4-D7 just like the ET-3400. If your buffers have two enable lines, one low to read and another to write, you're in luck. Simply connect a diode similiar to the one now relocated in the ETA-3400 (a GD510) to the pin that goes low for a READ. THE BANDED END SHOULD BE NEAREST THE IC. Connect the end of the diode to the UD2 pin which connects to the RE line. If your memory cards use one line like the ET-3400, (high to WRITE and low for READ) you may have to invert the signal using one of the spare 7404's on the wirewrap card.

If all else fails, do as I did, and connect the diode to first one pin then the other on the memory card buffer until you find the one that works! On the Thomas 24K RAM card, the correct pin is IC-105 pin 8.

IT'S UP!

Once you have the card operating, run memory tests, and/or use the program in the first of this article. You'll have to change the address to match those on your card.

SOFTWARE

As it comes from Heathkit, the ETA-3400 is set to use 0000-23FF(HEX). Any memory you add will have to be higher than this. This is not too limiting until you start getting more than 4-8K of memory. Most of the good programs written for this much memory assumes that you have free memory from 0000- and up.

If you would like to free the memory from 0000-0800(HEX), you will have to seriously consider rewriting the ETA-3400 monitor ROM, and then burn it into EPROM which can be set to E000(HEX) and up. To work on the program, use the SLIDE and move the Monitor program to RAM. It will be up to you to change all three BYTE instructions to the new address for your EPROM. Most of the data that needs to be changed is easy to spot, just use your "I" command, and go thru the program. One "hidden" piece of data that is not a three BYTE instruction is at 1482(HEX) C6 14. This is the MSB of the return address, and should be changed to the MSB of your EPROM's address. The address table must also be rewritten to the new address. Lastly, you'll have to work to readdress the PIA, which is now at 1000(HEX).

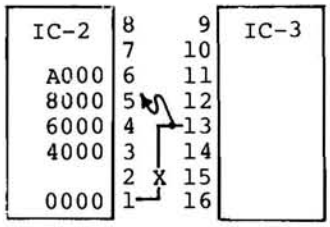
One word of caution. Even after you reburn your monitor to EPROM and locate it above free memory, the ETA-3400 makes heavy use of the 00 page, 0000-00FF(HEX). This will require that you rewrite some of the commercially available programs because of addressing conflicts.

THE END

In conclusion, I should say thanks again to James Greger for all of his assistance. It has been fun and educational, and helped keep me out of the bars on the weekends! If you like to work with hardware, I hope you will try these ideas. You can write me for assistance, but please enclose an 18 cent SASE with your letter. GOOD LUCK!

FIGURE #1

READDRESSING THE TRAINER
RAM IC's 14-17



Cut the trace at X and jumper from IC-3 pin 13 to a pin on IC-2 which gives you the address you need for your system.

FIGURE #2

PIN ASSIGNMENTS BEFORE MODIFICATIONS
40 PIN CONNECTORS ON --

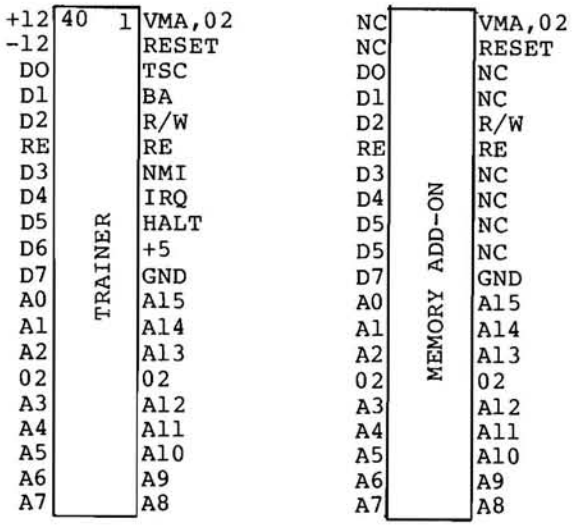


FIGURE #3
AFTER

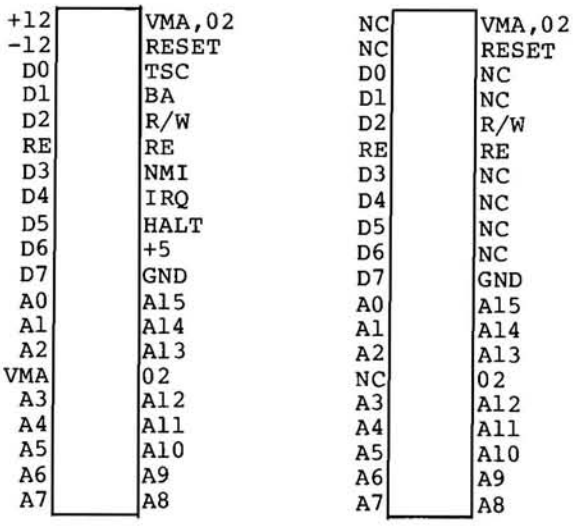


FIGURE #4

WIREWRAPP CARD PINOUTS/CONNECTIONS

```
SS-50 BUSS     TRAINER      ADD-ON      7400
PIN   LINE     LINE         LINE        PIN
 1    DO ----- DO -------- DO
 2    D1 ----- D1 -------- D1
 3    D2 ----- D2 -------- D2
 4    D3 ----- D3 -------- D3
 5    D4 ----- D4 -------- D4
 6    D5 ----- D5 -------- D5
 7    D6 ----- D6 -------- D6
 8    D7 ----- D7 -------- D7
 9    A15 ---- A15 ------- A15
10    A14 ---- A14 ------- A14
11    A13 ---- A13 ------- A13
12    A12 ---- A12 ------- A12
13    A11 ---- A11 ------- A11
14    A10 ---- A10 ------- A10
15    A9 ----- A9 -------- A9
16    A8 ----- A8 -------- A8
17    A7 ----- A7 -------- A7
18    A6 ----- A6 -------- A6
19    A5 ----- A5 -------- A5
20    A4 ----- A4 -------- A4
21    A3 ----- A3 -------- A3
22    A2 ----- A2 -------- A2
23    A1 ----- A1 -------- A1
24    A0 ----- A0 -------- A0
25    GND )---
26    GND )--- GND ------- GND ------ PIN-7
27    GND )---
28    +8V )            (X------X=CONNECTION)
29    +8V ) NC
30    +8V )            (NC=NO CONNECTION)
31    -12V NC
32    +12V NC
33    INDEX  (PLUGGED WITH A PIN)
34    M RESET NC
35    NMI ---- NMI
36    IRQ ---- IRQ
37    UD2 ---- RE -------- RE
38    UD1 NC
39    02 ---------------------------- PIN-2
40    VMA ---------------------------- PIN-4
41    R/W ---- R/W ------- R/W
42    RESET -- RESET ----- RESET
43    BA NC
44    01 NC
45    HALT --- HALT
46 TO 50  =  BAUD RATE LINES NC
                VMA ------------------ PIN-3
                02 -------- 02 ------- PIN-1
                +5V ------------------PIN-14
```

****NOTE: FIGURES #2 AND #3 ARE DIAGRAMS OF THE TRAINER AND ADD-ON CONNECTORS SHOWING THE SAME PINOUT CONFIGURATIONS FOR BOTH DETAILS WITH THE MODIFIED PINS INDICATED. FURTHER, THE PIN NUMBERS ARE SHOWN STARTING AT THE TOP LEFT WITH PIN 40 AND TERMINATING AT THE TOP RIGHT WITH PIN NUMBER 1.

LISTING #1

SS-50 PINOUT DESIGNATIONS

PIN NUMBER	SIGNAL	DESCRIPTION
1 TO 8	D0 TO D7	Data Buss Lines. Complement of the 6800 data lines (inverted).
9 TO 24	A0 TO A15	Address Lines. Same as ET-3400
25 TO 27	GND	Ground return for power.
28 TO 30	+8V	+8 VDC supply line.
31	-12V	-12 VDC supply line.
32	+12V	+12 VDC supply line.
33	INDEX	A plugged pin hole to prevent incorrect insertion of boards.
34	M RESET	Manual Reset. Active low. Input to a oneshot which inturn outputs pulse to reset CPU.
35	NMI	Nonmaskable Interrupt. Active low. Same as ET-3400.
36	IRQ	Interrupt Request. Active low. Same as ET-3400.
37 TO 38	UD1 & 2	User Defined lines. UD2 is used here for the RE line.
39	02	Clock 2 line. Inverted.
40	VMA	Valid Memory Address. Inverted. Same as ET-3400.
41	R/W	READ/WRITE line. High for a READ, low for a WRITE.
42	RESET	Reset line. This is the output of a oneshot (M RESET).
43	BA	Bus Available. Same as ET-3400.
44	01	Clock 1 line. Same as ET-3400.
45	HALT	Halt line. Active low. Same as ET-3400.
46 TO 50	110 TO 1200	Baud lines. Used for ACIA timing in the Southwest System.

NIFTY NEATS FROM HOYLE & HOYLE...

Janet Hoyle, of Hoyle & Hoyle Software, just gave us a call to fill this little gap with information on their latest program called PICTURE PERFECT which is a "receiving and collecting" utility for all you Ham types out there. They also offer A REMarkable Experience which is the next level in Adventure for the explorer in all of us. Plus, (get this!) Steer Kleer, an obstacle course game that will definitly test your manual dexterity. PICTURE PERFECT sells for $15.00 while their games go for $10.00 each. All of these pieces are valuable additions to your software library.

For additional information you can reach Janet Hoyle by writing:

Hoyle & Hoyle Software
716 South Elam Ave.
Greensboro, NC 27403

VARPTR in MBASIC

In issue 18 of REMark, there were two articles on MBASIC which touched on the user defined functions. We had several calls from individuals who wanted more of an explanation about them. Bob (and I) undertook to learn about the DIMensioned USeR DEFined function as it relates to something called a VARible PoinTeR (VARPTR) of MBASIC. I am writing what we have learned.

Our objective will be to write a program which will allow us to run the "BEEP" subroutine (REMark issues 15-18) using the VARPTR of MBASIC. First, we better start with a simple comparison of the two user subroutines in MBASIC from issue 18 of REMark. This should help us to get started. I will refer to the articles as "POKE" (page 10) and "REAL" (page 24) for simplicity.

Did you notice that both the "POKE" and "REAL" programs used assembly code in the user defined subroutines? "POKE" used OCTAL while "REAL" used HEX. As we know, the CPU is not dependent on any particular number base. So, what is the difference between these two user defined functions? Well, in "POKE", we DEFined the address and we placed our user program where we wanted it to be stored in memory. In "REAL", we DIMensioned an area and then MBASIC placed the user program where it wanted. Are there any advantages to either way of storing the user defined programs? Well, the answer is yes . . I think you will understand the advantages after completing this article.

By using as our example, the "Real-time function" program (from our "REAL" article), we will be able to see how MBASIC treats dimensioned user defined subroutines. We will study this program first and then we will be able to use the more difficult program, "BEEP" as a dimensioned user defined subroutine.

From the source code of "BEEP" in issue 17 of REMark (page 5), we were able to determine the OCTAL values to POKE into memory. But what is the source code to the HEX values in the "Real-time function" program? Here is the source code as you would see it if you were to write it in assembly code:

```
xxx xxx   006 000   MVI       M,0
xxx xxx   377 001   SCALL     .SCIN
xxx xxx   330       RC
xxx xxx   167       MOV       M,A
xxx xxx   311       RET
```

(The x's indicate "some" address in memory.)

Well, this program looks simple enough. How did we "disassemble" this source code from the HEX values of the "Real-time" subroutine? First, by taking a closer look at how MBASIC executes DIMENSIONED user defined subroutines, we will be able to understand the whole picture.

The normal steps for setting up a function call of an MBASIC program are to dimension, define and execute the call. By looking at the following lines, we can see that this is true:

```
1000 DEF USR0= VARPTR (U0(0))  REM: This line was omitted.
1005 X=USR0(0):

2010 DIM U0(3)
2020 U0(0)=&H36
2030 U0(1)=&H1FF
2040 U0(2)=&H77D8
2050 U0(3)=&HC9
2060 DEF USR0=VARPTR(U0(0))
2070 IF USR0(0)<>0 THEN 2070
```

(Note the new lines 1000 and 1005. These lines must appear this way. The reason why will be explained later.)

The next step is to "disassemble" the HEX numbers. To do this we will first convert the HEX values to OCTAL. This is done for the assembly language programmers who are more familiar with the OCTAL number base. From there we can determine from an 8080/Z80 Op Code Table what this program is to do. Here is the conversion:

&H36	=	&H0036	=	&000066	=	&000 &066	=	000 066
&H1FF	=	&H01FF	=	&001377	=	&001 &377	=	001 377
&H77D8	=		=	&167330	=	&167 &330	=	167 330
&HC9	=	&H00C9	=	&000311	=	&000 &311	=	000 311

Does the last column of OCTAL numbers look familiar? Glance back at the assembly program I gave you earlier. These octal numbers look strikingly similiar to those octal numbers. What is the difference? When DIMensioning subroutines, MBASIC loads two bytes per dimensioned variable, and thus we have two HEX (or OCTAL) numbers per variable. Ok, that's pretty easy!

This is where MBASIC tends to confuse matters . . . MBASIC reverses the order of the two bytes when storing them in memory. Huh? Let's compare the instructions and see if we can understand what MBASIC is doing.

* HEX VALUES *	* OCTAL EQUIVALENT *	* BYTES LOADED REVERSED *	* ASSEMBLY CODE *	* STORED IN MEMORY *
&H0036	000 066	066 000	066 000	xxxxxx 066
				xxxxxx 000
&H01FF	001 377	377 001	377 001	xxxxxx 377
				xxxxxx 001
&H77D8	167 330	330 167	330	xxxxxx 330
			167	xxxxxx 167
&H00C9	000 311	311 000	311	xxxxxx 311
			NOP	xxxxxx 000

What is shown are the variables in HEX, with the OCTAL equivalent followed by the OCTAL bytes shown reversed, as they will be loaded into memory. The next step was to show the bytes as they appear in the assembly code listing. The last column shows the addresses (x's) with their instructions or data as they would be stored in memory. What we have done is converted the HEX code from the "Real-time function" program to the OCTAL values shown in the order in which the bytes will be stored in memory.

When executed, what is each step of this user defined subroutine actually doing? The user defined program will set the Memory register to zero then check to see if a character has been entered. If no character has been entered CARRY is set and thus the Return on Carry. If a character has been entered then CARRY will not be set. The value in the A register will then be MOVed to the Memory register and the unconditional RETurn will be executed. A simple straight forward program!

Let's review the "Real-time" MBASIC program (issue 18, page 24). In line 2010 we DIMensioned four variables, U0(0) - U0(3). Lines 2020 -2050, we set the variables equal to the HEX values. Then in line 2060, two terms are there that we have not discussed.

The DEF USR0 is simply DEFining our USeR machine subroutine. The "0" is the number that distinguishes this subroutine from other subroutines. But what does the DEF USR0 actually do? It tells MBASIC at what address to begin executing the user defined subroutine. So far so good. What is this VARPTR?

The VARPTR(U0(0)) according to definition "returns the address of the variable" U0(0). What does that mean? Well, remember those "xxx xxx's"? We said when using the defined user function that MBASIC puts the subroutine any place it wants to. With the VARPTR we are able to access that address and then execute the subroutine! And that is exactly what happens.

The "xxx xxx's" are not important to us because the VARPTR finds and returns the address of the variable. Remember the "POKE" article and subsequent program? We determined where in memory we wanted to put the subroutine and then we called it from that specific address. We had to make sure that we did not POKE anywhere in memory that was occupied by something else. That in essence was what the entire article was about. Now by using the user defined function of MBASIC, we no longer have to worry about where we put the subroutine in memory because MBASIC does it all for us. Nice!

Well, that about raps it up for user defined functions using the VARPTR . . . or does it? Our example program from "REAL" used one and two byte instructions. What happens when we have three byte instructions? Do we just have to be sure the HEX values are in the proper order and use NOP's where necessary? Is that all there is to it? What if the three byte instruction is a jump?

In our "BEEP" program we use the Jump Not Zero (JNZ) command three times. Assembly language programmers know that the JMP and JNZ commands jump to an address NOT a name as is specified in the LABEL and OPERAND fields e.g. "JNZ LOOP". When we POKEd the "BEEP" program in issue 18, we POKEd the instructions into known locations. What if we don't know where the addresses are? What if they are ever changing? When using the dimensioned user defined subroutines, we do not know the addresses of any of the bytes of the subroutine because MBASIC takes care of that. Adding one character of code to an MBASIC program can change the user subroutine location by 30 OCTAL. Somehow, we need to do a relative and "variable" jump.

With the 8080 processor we are not able to do Relative Jumps, meaning jumping from ANY location backward or forward "X" number of locations from the source code. With MBASIC taking care of the locations, the addresses change with any variation in code or memory size. With these unknowns, what can we do?

From the definition of the VARPTR, we found that it "returns the address of the variable given as the argument". Could this be our answer? Each time MBASIC moves the subroutine around, the VARPTR will return that new address. So even though the addresses are varying, we can always determine the first address of the subroutine. That means we always have a "known" address, right? Can that be our "relative" address for doing jumps? Why sure! All we have to do is determine how many relative address locations do we need to jump to.

That really is all there is to it. It simply means that we need to determine how many addresses from the first variable, U0(0), is the jump to go. Let's look at a table showing the conversion from what we know we want in memory to the HEX values and its dimensioned variable.

First, I better briefly explain NOP's, as we may need to use this "operation" in one or all of the three byte instructions. A "No OPeration", NOP, is similiar to a NULL statement. It is neither an instruction or data and the CPU, simply bypasses it. NOTE: The NOP, which is 000 OCTAL or 00 HEX, cannot follow an instruction that requires data. The CPU will execute it as 000 OCTAL or 00 HEX, if it is looking for data, e.g. a jump looking for an address. This should not need any further explanation.

To execute the program properly, we want the program, once loaded in memory, to appear identical to the OCTAL codes of the source code on page 5 of REMark issue 17, with the exception of the addresses to jump to. The objective of the table is to show the addresses (x's) with the OCTAL codes (as we will need to "see" them in memory in order to execute "BEEP"), converted to the HEX values with the appropriate user defined variable.

* NUMBERED ADDRESS	* STORED IN MEMORY	* BYTES REVERSED	* HEX VALUES	* MBASIC VARIABLE
Relative	xxxxxx 006	005 006	05 06	U0(0)
(1)	xxxxxx 005			
(2)	xxxxxx 315	136 315	5E CD	U0(1)
(3)	xxxxxx 136			
(4)	xxxxxx 002	026 002	16 02	U0(2)
(5)	xxxxxx 026			
(6)	xxxxxx 377	036 377	1E FF	U0(3)
(7)	xxxxxx 036			
(8)	xxxxxx 377	035 377	1D FF	U0(4)
(9)	xxxxxx 035			
+ (10)	xxxxxx 000	302 000	C2 00	U0(5)
(11)	xxxxxx 302			
(12)	xxxxxx ???	??? ???	?? ??	U0(6)
(13)	xxxxxx ???			
(14)	xxxxxx 025	302 025	C2 15	U0(7)
(15)	xxxxxx 302			
(16)	xxxxxx ???	??? ???	?? ??	U0(8)
(17)	xxxxxx ???			

(18)	xxxxxx 005	302 005	C2 05	U0(9)
(19)	xxxxxx 302			
(21)	xxxxxx ???	??? ???	?? ??	U0(10)
(22)	xxxxxx ???			
(23)	xxxxxx 311	311	C9	U0(11)

(Note: The location of the "+" is the only address where a NOP is necessary.)

The 302 OCTAL is the JNZ instruction to an address we do not know ("??? ???"). As soon as we determine these addresses we have written our program. Look at the first instruction, the "xxxxxx 006". This is our "Relative" location. The x's are some address, we said that we can determine this address by using the VARPTR! The "DEF USR0=VARPTR(U0(0))" returns this very address!

Before we go on and confuse anyone, we better explain that the VARPTR returns a DECIMAL value as the address. This is going to make our job so simple, because now we do not have to worry about reversing the order of the bytes for MBASIC. By simply counting from the first "known" or "relative" address, to the address we want to jump to, we can add that decimal value to the U0(0) address. We will then have our jump completed.

The three JNZ's, LOOP, LOOP2 and LOOP3 (issue 17, page 5), jump to (2) 315 OCTAL, (7) 036 OCTAL and (9) 035 OCTAL, respectively. By adding the DECIMAL (2), (7) and (9) to the VARPTR(U0(0)), we know exactly where we want to jump no matter where MBASIC decides to put the subroutine.

Let's look at the program.

```
10 ' RUNNING "BEEP" AS A USER DEFINED FUNCTION USING THE VARPTR
20 CLEAR 5000: WIDTH 255: DEFINT A-Z:
30 PRINT:PRINT TAB(10) "Begin running BEEP":PRINT: PRINT:
40 DIM U0(12):
50 Z=VARPTR(U0(0)):
60 U0(0)=&H506 : U0(1)=&H5ECD: U0(2)=&H1602: U0(3)=&H1EFF:
70 U0(4)=&H1DFF: U0(5)=&HC200: U0(6)=Z+9    : U0(7)=&HC215:
80 U0(8)=Z+7    : U0(9)=&HC205: U0(10)=Z+2  : U0(11)=&HC9 :
90 DEF USR0=VARPTR(U0(0)):
100 PRINT USR(0):
110 PRINT VARPTR(U0(0)):
120 PRINT PEEK(VARPTR(U0(0))):
```

With all the background leading up to this program, there should be very little confusion about the HEX values and where they came from. In line 100, the PRINT USR(0) simply executes the user defined subroutine, U0. (For the time being, disregard the "0" value, which this statement will print on the screen.) The PRINT VARPTR(U0(0)), line 110, prints the value in DECIMAL of the first address of our subroutine, while in line 120 the value "6" will be printed. This DECIMAL "6" (006 OCTAL) is our first instruction as you will remember from our program and table shown above. The rest of the program should be self-explanatory, or you would not have read this far.

When you have entered and executed the program, I would suggest that you "look" at the memory addresses to verify what you have inputted. First, convert the DECIMAL address to SPLIT-OCTAL.

On the H89, enter the "SHIFT RESET" key combination as if to reboot, then instead of entering the "B" for boot, enter an "S". The computer will finish the word "Substitute". Enter the SPLIT-OCTAL value (that you converted from the DECIMAL address given from the program) and a <CR>. You will now see the SPLIT-OCTAL value and the 006 OCTAL value which is our first instruction. (HINT: In case you happen to make a slight error in converting to SPLIT-OCTAL, you may want to begin about 50 SPLIT-OCTAL addresses earlier.) Each time you hit the SPACE BAR, you will see the next sequential addresses and the OCTAL value that is stored in that location. These values should be identical to our table shown above. Note the jumps (302 OCTAL) and their addresses. Doesn't it just excite you to see these values right before your eyes?!! (For those of you with a 48k machine, if you enter the program verbatum, the address of the U0(0) will be 29996 DEC and 165 054 SPLIT-OCTAL. This may be of help to some of you.)

(For those of you who have the H8, simply do a soft-reset, the "0" and "MEM" keys hit simultanously. "ALTER" the memory to the converted SPLIT-OCTAL value and you can see the values on your front panel. Neato!!)

Actually, that gives you all the information you need to understand and use the VARPTR in MBASIC. As I look back on the article, I don't think it was all too difficult. I hope you learned something and also that you found it interesting.

Now, for those of you who like to experiment, I am going to throw something out to you that you may wish to investigate. MBASIC does some strange things sometimes, and to tell you the truth we are baffled and don't know what is going on with the following example.

The first important confusing part appears to relate to the rest of our question. The VARPTR of MBASIC is ever changing and jumping around. Note that I added (back in the earlier part of this article) the new line 1000 and 1005 in the "Real-time" subroutine and that line 90 and 100 are set up the same way in this "BEEP" program. The VARPTR is moving around and if PRINT USR0 is given without the DEF USR0, then the VARPTR will have moved and the subroutine will most likely not execute properly. If it has moved since the DEF USR0=VARPTR(U0(0)), then the PRINT USR0 begins executing wherever the VARPTR happens to be pointing.

Confusing? How about this? In the "POKE" program of issue 18 page 13, we used the "Q=USR1(BEEP)" to execute the subroutine. If you were to try "Q=USR0" instead of using the "PRINT USR0", you would find that it will not execute . . it just "kinda" sits there. The VARPTR using the "Q=USR0", returns an address five locations ahead of the U0(0) . . Huh? Bob and I looked at that several days and finally gave it up to the wind. Any ideas out there why changing the executing line will return a consistant, wrong address?

I guess to conclude this article, it is fair to say the VARPTR of MBASIC will do a nice job and has its productive place in programming. The "hidden" rules of using the VARPTR make it an interesting (to say the least) means of linking to user defined, machine code subroutines.

<TLJ>

Heath Related Products

From M.I.-8 -- Ted Benglen of Micro-Interface recently sent us a collection of Heath related hardware items which are quite interesting. You may find that some of Ted's hardware would be a useful addition to either your H-8 or H-89. Here is a list of the super offering from M.I.-8:

1. Modem Kit H-8/H-89
2. Digitalker H-8/H-89 (set vocab)
3. V8 Voice Synthesizer H-8/H-89
4. Parallel I/O and Clock H-8
5. BSR Home Control Interface H-8/H-89

Ted has complete information on each of these outstanding hardware additions to your computer system. You can contact Ted by writing to:

M.I.-8
822 East County Road 30
Fort Collins, CO 80525

SILVERMAN ASSOCIATES -- Bernard Silverman recently sent HUG information on two packages they have ready to go. I feel his letter contains some important details about the products.

baZic -- ... "baZic" is a BASIC interupter that will allow any Heath/Zenith H-89 or equal computer to run any NorthStar BASIC program under CP/M - and run faster and more efficiently. "baZic" was written using the full capability of the Z-80 processor and reports of up to 60% improvement over the NorthStar have been reported. ....

MONEY MAESTRO -- ... This product should appeal to the entire H-89 community since it makes big system features available to the home computer user. It handles an individuals personal banking - both at the office, for the small professional office, and at home for a personal home banking system. The system has built in Electronics Funds Transfer (EFT) capability so that when banks are ready to offer that service Money Maestro, with a simple disk exchange to put the user's bank number into the system, will allow a tie into EFT systems.

For additional information, contact:

Silverman Associates
4010 Opal Street
Oakland, CA 94609

FROM PAGE 15

will need to decide if you want to purchase a Z80 CPU Card, HA-8-6, which includes the Extended Configuration along with the upgraded microprocessor. If you are content with the 8080 microprocessor, your other option is to purchase the HA-8-8 Extended Configuration Option from the Heath catalog. The instructions of the kits need no explanation.

The H89 requires the H-88-7 Replacement ROM Kit to facilitate the ORG-0 operation of CP/M. The H89 already contains the Z80 microprocessor and therefore leaves only the one option to H89 owners.

The next step is to purchase the Digital Research's CP/M software package from the Heath catalog. For the 5 1/4" drive system, purchase the HOS-817-2 and the HOS-847-2 for 8" drive system.

When the hardware and software packages come, you will obviously need to do the hardware modifications as instructed, no comment is necessary from HUG. The software package, however, will need a brief explanation.

The first step is the signing of the "SOFTWARE LICENSE AGREEMENT". Be sure you read the entire agreement. The most confusing part of the agreement is, "each program license granted under this Agreement authorizes the customer to use the Licensed Program in any machine readable form on any single computer system (referred to as System). A separate license is required for each System on which the Licensed Program will be used." This means that your CP/M operating system can be run on only one computer. If you have more than one computer, you will need to purchase an individual CP/M operating system for each "system" you have. That is one reason why you must understand the entire agreement before signing it.

After reading, understanding, signing, and mailing the Agreement, you are then ready to begin to learn the CP/M operating system. It is at this point that this first part will be concluded. Part II will begin to explain the set up for CONFIGURing CP/M to your system so that you will finally be able to start to use the widely used CP/M operating system.

At this point, it may be wise to recommend some books to supplement the CP/M documentation. Heath is in the stages of preparing a CP/M Course (EC-1120) that includes tapes and manuals. The course will be available in January of 1982. I have viewed the first few chapters at this point and highly recommend it to the very beginner. The first chapter is probably familiar to most everyone and you will think that you have wasted your time, however, as you continue to listen and study, you will begin to feel more at ease. You will become familiar with the terms and definitions, and find a good basic understanding of CP/M.

I would suggest that you purchase another book or manual to complement the course from Heath. Howard W. Sams & Co., Inc. has published a number of beginner manuals including their "CP/M(tm) PRIMER" by Stephen M. Murtha and Mitchell Waite. I rely on this manual as a primary source of my studying of CP/M. Dale Lamm, from the HUG Bulletin Board, suggested the "CP/M Handbook" by Rodnay Zaks, published by SYBEX. One of these two sources supplemented with Heath's CP/M course will blend very well together and will give you an excellent start on learning the CP/M operating system.

In conclusion of this introduction to CP/M, I would like to say that by learning both the HDOS and CP/M operating systems, you will have more of an appreciation for both systems. They both possess their good and bad points, but understood together, you are able to accomplish your achievements by making a choice of which system will meet your goals the best.

<TLJ>

HUGBB Helps and Hints

One of the biggest causes of confusion to any interested person that would like to join the HUGBB via MNET or the SOURCE, is which software modem package to purchase. There are so many different packages available from HUG and SOFTSTUFF(tm) plus other sources, how does anyone know what the differences are? Let me just briefly discuss software modem packages and I hope that you will pickup enough information that you can decide for yourself which modem package will best fit your needs.

The purpose of a software modem package is to allow two computers to communicate

with each other via telephone lines. From that point, each package can branch and provide different options dependent on what type of computer is at the other end. Baud rates, filtering of some codes, transferring files to and from the other system, auto-log-on, and even now auto-dial are some of the different options that are available in modem packages. The best way to explain the different packages available from HUG is to give a brief abstract of each product.

P/N 885-1043 MODEM Heath to Heath: This package is the most primitive that is available and is intended for communications between Heath computers, ONLY. If you will only communicate between Heath systems this package will serve your purposes, but if you will eventually use it for communications between your system and MNET or the SOURCE, then you will not want this package. Selectable BAUD rates are 110, 150, and 300.

P/N 885-1050 M.C.S. Modem for H8/H89: This package was written for communications between a Heath computer and a time-share HOST such as MNET or the SOURCE. It does NOT communicate properly (without making modifications) between Heath computers. The source listing is included with the disk, however, it is a very difficult program to modify. The COPY mode of MCS is not a very efficient means of storing data in a buffer for saving to disk. Selectable BAUD rates include 300 and 1200. All in all, this package has all the necessary options to "talk" to MNET or the SOURCE but not another Heath system.

P/N 885-1089 Disk XVIII Misc H8/H89: This disk contains the software modem package HTERM. This package is the newest we have to offer and I highly recommend it. It has nine selectable BAUD rates from 110 to 9600. It allows the full use of the normal control characters and the ESCape functions of the H19/H89 CRT. It has full capabilities of transferring files to/from the other system. HTERM does not offer auto-logon. It can be used for any type of communication. I have used it to talk to MNET, the SOURCE, and my H11. Pat Swayne originally wrote it to communicate with the ET3400. A very versatile package.

P/N 885-1207 TERM and H8COPY: TERM is a CP/M modem package. It includes auto log-on to TYMNET or CompuServe. The BAUD rate may be changed within the source code. It has full transfer capabilities of files and is independent of the other system. An excellent modem package.

P/N 885-1212 CP/M Utilities I: This CP/M disk has the H19/H89 version of MPLINK as announced in this issue of REMark. (See page 16.) It features automatic log-on, normal file save and transmit, and optional XOFF and XON recognition. It makes use of the CRT function keys and the 25th line.

SOFTSTUFF(tm) CPS modem package; SF-9003 for HDOS, and SF-9103 for CP/M: This is probably the easiest package to learn of the ones we have discussed so far. It uses the 25th line to display the options that are available while in a particular function of CPS. The BAUD rate and other options are available through a series of menus. It includes auto-logon and a real-time clock display on the 25th line. CPS does not allow the full use of the control characters and the ESCape codes of the H19/H89 terminal to be processed from the HOST or other computer.

These modem packages available from HUG and SOFTSTUFF(tm) are not the only packages available in the software market. I have seen or heard of others that are very good and that offer the same features of these packages. (I understand there are even auto-dial packages, but have not used one as of yet.) Among some of the others I have used briefly or seen are versions of ITCOM, INTERCOM, and wow . . who knows what else.

Anyway there are many software modem packages available on the market. Because this "craze" of computer communications is in such an infant stage, don't be too critical about the many modem packages and that each one offers different features. As communications continue to become more sophisticated these will become primitive also. There is a whole new world beginning right before our eyes and it is getting bigger and bigger. Kinda' scarey!!

*SYSOP <TLJ>

Local HUG News

Roger Fraumann reports that the C-Quad (Champaign County Computer Club) is alive and well! Roger just took on the responsibility of representing the club for their activities. If you would like additional information about C-4 and this computer group, contact Roger at 412 Dorchester; Mahomet IL 61835.

SLHUG (St. Louis HUG) meets every month on the second Wednesday at the Heathkit Electronics Center located in Bridgeton MO. Meetings begin at 7:30 PM and visitors are always welcome to attend.

TO PAGE 31

Dear HUG,

I discovered that disabling CTRL-C and CTRL-A in MBASIC has great merit if you need to "protect" your software. This can be accomplished for HDOS MBASIC Version 4.82 as follows:

1. Locate sector 78 of MBASIC with a patch utility (HUG's DUMP or similiar).

2. Change the following bytes to "00":

Location		Old Value	
HEX	OCTAL	HEX	OCTAL
31	61	6B	153
32	62	65	145
38	70	71	161
39	71	65	145

Now when someone presses CTRL-C or CTRL-A, nothing at all happens. (HINT: develop your software before making this patch!)

Gary Hawthorne
Fair Lawn, NJ

Dear Bob,

I have recently completed a software modification to FBE Researches' Centronics 737.DVD. The modification allows tabs to be processed in the proportional spaced mode of operation. Now, in all three character fonts (10CPI,16,7CPI, PROPO), the .DVD aligns the tabs so they are in the same format as the H19-H8/H89 tabs are.

I am offering this modified version to owners of FBE's CT.DVD for three dollars (to cover shipping and handling) and their original FBE.DVD disk which will be returned with my CT.DVD and MCT.ASM files added to it.

Dan Morin
124 Plymouth Street
Manchester, NH 03102

Dear HUG,

It has come to my attention that the procedure for "Recovering A Deleted File" published in Issue 19 of REMark will not work under certain conditions. Referring to the article, step 2 of the procedure requires that the values at the 17th and 18th bytes be noted. If the value of the 17th byte is of a higher value than that located at the 18th byte, the procedure described will not result in a recovered file and could result in unpredictable disk performance if used.

This condition seems to exist when various GROUPS have been made available by previous deletes of small files and they were located in the GRT.SYS file at the lower addresses. As a result, HDOS, in its' effort to string together as many contiguous groups as possible, may start the file location record at a high address in the GRT.SYS file and actually fold around and end up at a lower address. In this case, the procedure described should not be used since the file is not strung together from low address to high address. I have experimented around with this condition and could not develop a guaranteed procedure unless the condition of the GRT.SYS file was known prior to the delete. This, of course, is unlikely since usually one does not anticipate an unintentional delete.

It appears that this condition happens rarely and usually on a disk that is nearly full or full to the delete. I apologize if this has caused any inconvenience to any of your readers and would welcome any solutions to the problem.

Donald Harton
2313 Covered Bridge Garth
Baltimore, MD 21234

Dear HUG,

Here is a little something I ran into, and would like to share. Under MBASIC 4.7 (also under 4.8) for HDOS a period (.) following the command EDIT, will designate the current line number for manipulation (see manual, page 4-6). Curiosity lead me to try this technique on some other commands. I found that the period (.) after the following commands works in the same way as it would when using EDIT:

AUTO	DELETE
RENUM	CONT
LIST	

A little experimentation will give you the hang of it quickly.

Joseph Gonzales
Trujillo Alto, PR 00760

FROM PAGE 29

The Al Lynch HUG meets on the first and third Wednesday of each month at the Tampa, Flordia Heathkit Electronics Center at 7:30 PM. For additional information, either contact the store or drop down on a Wednesday evening to see what is going on in the Flordia Center.

Paul Beland is looking for a users' group in his area or individuals that are interested in getting together in Cambridge. If you can help Paul, or if you just want to get to know another user, drop him a line at 453 Franklin Street; Cambridge, Mass. 02139

A little note from Jim Simpson ... Over a dozen Heathkit Hobbyist have formed a local Heath Users' Group in Charlotte, NC. For details on the monthly meetings, contact Jim Simpson (704) 374-6997 or Mike LaFleur (704) 364-9667 or write 2721 Picardy Place; Charlotte, NC 28209. Thanks ..... HUG Charlotte

CIHUG (Central Illinois Heath Users' Group) meets on the third Sunday of every other month. This new HUG has approximately 17 members that get together at 3:00 PM. For additional information, contact the Club President, Ronald Morgan at (309) 745-8313 in Washington IL. The clubs main center of population is the Peoria area.

James Piunti would like to contact other HUG members located near the RAF Chicksands, England. If you would like to get together with Jim, drop him a line at the following address: 6590 ESG Box 1001; APO New York, NY 09193

Bill Latham is interested in forming a Heath Users' Group in Guam. He feels that there may be enough interest by combining the Services that he is willing to give it a shot. Bill tentatively has called this group GHUG (Guam Heath Users' Group). If you would like to get together with Bill to discuss GHUG, contact him by writing:
William R. Latham
Capt, 1238-B Palau Loop
APO SF 96334

Alaska HUG---A new group is in the process of forming in Alaska. For further details contact Ben Sevier, P.O. Box 951, Eagle River, Alaska 99577 Phone (907)-694-9908.

Bulletin Boards .....

PNHUG (WA)	24 hrs.	(206)-246-4468
OCEAN (NJ)	24 hrs.	(201)-775-8705
FAIRLAWN (NJ)	AFTER HOURS	(201)-791-6983
PEABODY (MA)	AFTER HOURS	(617)-531-9332
SLHUG (MO)	24 hrs.	(314)-291-1854
MUG (KS)	24 hrs.	(913)-362-9583

NOTE: Please add the bulletin board list to any you may have already and watch for new boards as they are announced under the Local HUG News section in REMark.

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 _____

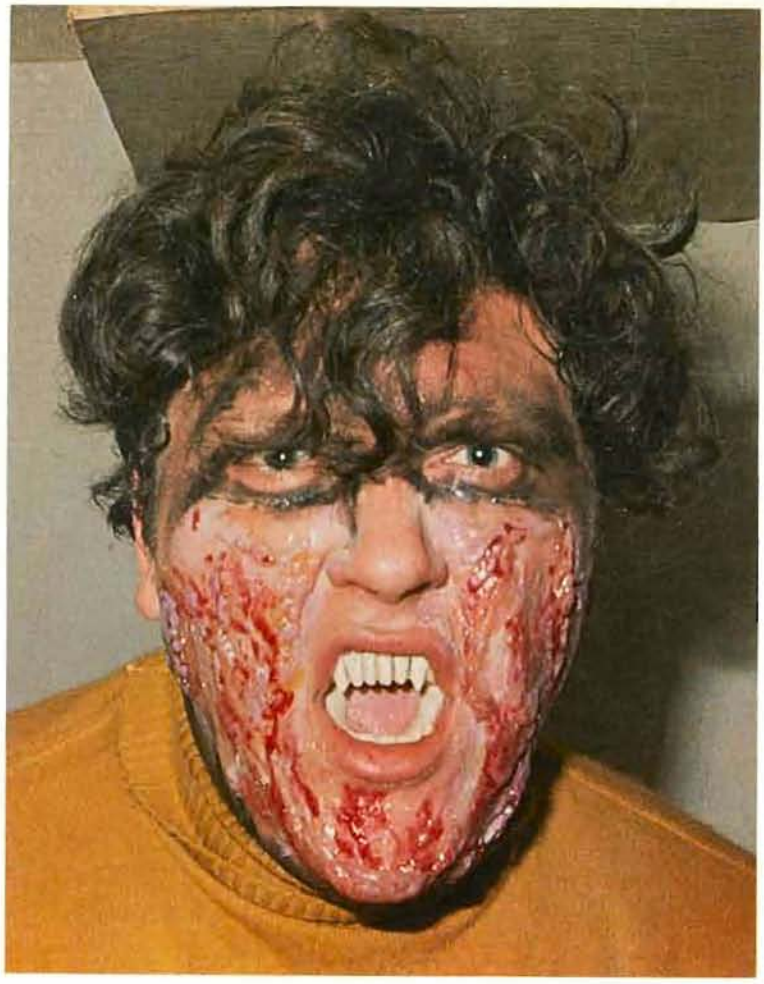
Zip _____

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.



BLACKMAX

Many of our HUG BB users have requested to be "officially" introduced to the infamous "BLACKMAX". Some of them (like Al Dallas of ANAHUG) are obsessed with trying to find out who this strange computer MYSTERY PERSON is. Well Al, and all you other curiousity seekers, here "HE" is (I guess?). Now, ask yourself one question....DO YOU REALLY WANT TO MEET THIS INDIVIDUAL? I might add, your HUG staff risked life and limb to get this picture and now our camera is a pile of jelly. Anyway, don't mention HDOS to this CP/M freak! That is, if you want to keep your computer in one piece! **NOTE: Hope this helps AL!

BE:

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable, please do not return.

885-2021