REMark

Issue 4 • 1978

Official magazine for users of Heath computer equipment.

HUG MATERIALS AVAILABLE TO HUG MEMBERS

HERE IS A COMPLETE LIST OF MATERIALS AVAILABLE TO MEMBERS TO DATE.

HUG BINDER	885-4	$ 4.00
HUG TEE S	885-1100	$ 4.50
SHIRTS M	885-1101	$ 4.50
L	885-1102	$ 4.50
SOFTWARE TAPE I	885-1009	$ 7.00
SOFTWARE VOLUME I	885-1008	$ 9.00
ADVENTURE (H8) (disk)	885-1010	$10.00
HDOS PROGRAMMING GUIDE	885-1018	$ 5.00
HDOS DEVICE DRIVER	885-1019	$10.00
ID-4001 INTERFACING* (cassette)	885-1017	$ 5.00
SOFTWARE (disk)	885-1016	$10.00

*Available after December 15.

NOTE: Always place your orders on the green order form and include payment plus shipping and handling.

on the stack

>CAT

"REMark" is a HUG membership magazine published quarterly. A subscription cannot be purchased separately without membership. The following membership rates apply.

	U.S. Domestic	Canada & Mexico	Internat'l
Initial	$14	$16	$24
Renewal	$11	$13	$18

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

Send payment to: Heath User's Group, Hilltop Road, St. Joseph, MI 49085. Back issues that are available cost $2.50 postpaid to U.S. destinations. Request for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heathkit Electronic Centers or technical consultants.

HUG Manager and Editor .... Jim Blake
Graphics .... Ron Hungerford

REMark

. . . . .A HAPPY MARRIAGE

MACHINE LANGUAGE SUBROUTINES AND BASIC

By: Chris Kern
Apt. V-839
201 I St., S.W.
Washington, D.C. 20024

When I entered Dartmouth in the middle 1960's, shortly after BASIC was introduced, it was an article of faith among those of us who learned the new programming language that it was possible to make the computer do anything in BASIC that you could make it do with any other language — including the arcane "machine language" that the people in the computer center were fond of making obscure references to.

There may have been some truth to that — there may even be today — but anyone who has used a BASIC interpreter in a microcomputer knows it has its limitations. The most obvious one is execution speed. BASIC spends most of its time translating the user's program: looking up variable references, deciphering arithmetic operators and the like. It's easy to use, of course, but where high speed execution is necessary — for controlling other machines, for example, or where unusually complex computations are involved — machine code is often necessary, too.

Fortunately, it is possible to include machine language subroutines in Heath BASIC programs. (The programs described in this article are all written for Extended BASIC, but the principles involved apply to the shorter version as well.) The USR(function in Heath BASIC is designed expressly for this purpose. Actually, USR(is a call to a machine language program that has been entered in high memory above the workspace for the BASIC interpreter.

To use this function, BASIC must be configured with reserved memory space at high memory. Instructions for implementing the USR(function, along with a simple example, can be found in the Heath software manual at page 5-111.

The USR(function was intended to permit the inclusion of a user-defined mathematical function that is not supplied by BASIC — in other words, another function which would be available to any BASIC program. But it can serve other purposes as well. It might be used to trigger a machine language program to sound the H8's audio oscillator, for example. And by using BASIC's POKE command, it is possible to write a machine language subroutine into memory during the execution of a BASIC application program. This means machine language instructions can be made part of the application program instead of part of the interpreter.

AN EXAMPLE

Listing 1 is the source code for BEEP, a machine language program tc sound the H8's audio oscillator. It was written with the Heath Text Editor. (The BASIC addresses referenced in the listing are taken from the software manual's entry point table for Heath Extended BASIC. The program starts at the beginning of reserved high memory, which in my system is at offset-octal address 154.000.) First, the program decodes the USR(function argument, which it gets from BASIC's floating point accumulator. The value of the argument must lie between 0 and 255 decimal (377 octal). This value becomes a key to the length of the beep. The rest of the program is similar to the HORN routine in the H8 ROM panel monitor. There is a listing of this routine in the Heath Software Manual.

To use this program as a subroutine in a BASIC program, it is necessary to assemble the source code into machine code. Then the octal machine instructions created by the Heath assembler need to be converted to decimal numbers, which can be read by the BASIC interpreter. Finally, these decimal numbers must be POKEd into reserved memory in sequence. It is possible to do this with separate POKE commands for each instruction, but it's easier to write the machine language subroutine during execution of the BASIC program with READ and DATA statements:

```
FOR A = first address TO last address
READ D
POKE A,D
NEXT
DATA machine instructions, machine
        instruction, etc.
```

Unfortunately, this is rather cumbersome if the machine language subroutine is a long one. For those, who like me, have no printer, it is necessary to (1) copy down each of the machine instructions generated by the assembler, (2) convert the octal code to decimal code using a conversion table or an decimal computer program, (3) write each of the decimal instructions into a DATA statement in the BASIC program — all without making a mistake.

See listing 1, on Page 6.

DECOCT

A better way is to use the computer to generate the decimal instructions that will be executed by BASIC sub-routines. Listing 2 is a BASIC program, DECOCT, to do this. First, it's necessary to complete the assembly process described above by generating a binary object code tape of the BEEP program. DECOCT reads this tape, converts each binary machine instruction into decimal, and enters the resulting decimal program into a BASIC datafile. (This last step is done after DECOCT has finished running by using a PUT command to dump the datafile onto tape.)

Here is a summary of the steps to this point:

(1) Write a program, using the text editor, for a machine language subroutine to be entered in reserved high memory.

(2) Assemble the program, using the assembler, and produce a binary object tape.

(3) Use the BASIC program DECOCT to read the binary tape, convert it to decimal and set up a BASIC datafile containing the decimal program along with the starting and ending addresses for it.

(4) Dump the datafile onto tape using BASIC's PUT command.

(Note, by the way, that DECOCT uses just the technique I have been describing — a call to a machine language subroutine — when it calls the panel monitor's tape handling routine to read the binary tape.)

See *listing on 2, Page 6.*

MORSCII

The BASIC program that uses BEEP to sound the H8's horn won't store the machine instructions in DATA statements. Instead, it will take them out of the datafile generated by DECOCT. As a result, this program must not be executed by using the RUN command. RUN wipes out all variable values; it would zero the S (starting address), E (ending address) and D (data bytes) variables contained in the datafile. This program will have to be executed by the commands: GOTO (first line of program): CONTINUE. And it will have to make provision for clearing any variables that must be set to zero when the program begins.

Listing 3, MORSCII, is a BASIC program which accepts characters from a terminal and outputs them as MORSE code on the H8's oscillator. It uses the datafile generated by DECOCT from the BEEP program. To create the machine language file for MORSCII, use BASIC's GET command to append the datafile to the MORSCII program. After this has been done, the FDUMP command can be used to dump MORSCII and the subroutine datafile onto tape together. FLOAD is subsequently used to load the MORSCII datafile combination. And, as noted, CONTINUE is the command used to execute the program instead of RUN.

Here is a summary of the final stages in the sequence:

(5) Write an application program that uses the machine language subroutine.

(6) Append the datafile containing the subroutine with BASIC's GET command.

(7) Execute the program using BASIC's CONTINUE command.

(8) Save the combination on tape using FDUMP; load the combination in the future with FLOAD.

It is still possible, of course, to write a BASIC program with a machine language subroutine that is contained in DATA statements, and for very short subroutines this may be desirable. DECOCT helps out by printing a list of all the decimal machine instructions in sequence (the format is designed to fit within the 12-line limitation of the Heath video terminal).

But where long subroutines are involved, the automated procedure will be considerably easier.

The listing for MORSCII is too long to be printed in this magazine, but is available upon request.

ANOTHER APPLICATION

The USR(function is not the only way to include a machine language subroutine in a BASIC program, although it will often be the most convenient method. Where the numbers transferred between the main program and the subroutine are to be operated on in floating point format, or where there are repeated calls to the machine language subroutine in the BASIC program, the USR(function simplifies program writing.

But it is also possible to transfer data by POKEing the number or numbers to be operated on by the machine language subroutine directly into memory, and there will be times when it is not convenient to transfer control from the main program to the subroutine with a USR(statement. One such occasion is where the machine language subroutine is to be activated in response to a CPU interrupt, for example when a device being controlled by the computer is ready for more instructions. Or a clock interrupt could be used to permit the execution of a machine language subroutine at regularly scheduled (in this case, 2 millisecond) intervals.

For example, a machine language subroutine to calculate the time of day could be activated at the beginning of a BASIC program. It would continue to operate each time a clock interrupt took place (every 2 msec) while the remainder of the BASIC program was executing. At any point when the time of day was needed, the BASIC program could retrieve it from memory locations serviced by the subroutine. That means the main program could calculate how long a given operation took, or it could use the time-of-day values to decide when to begin a particular operation of its own. Best of all, the operation of the time-of-day clock would be essentially invisible to the user, whose program would operate pretty much as it would if the machine language subroutine wasn't there. (Obviously, if the clock interrupt subroutine was long enough, its drain on CPU processing time would begin to show.)

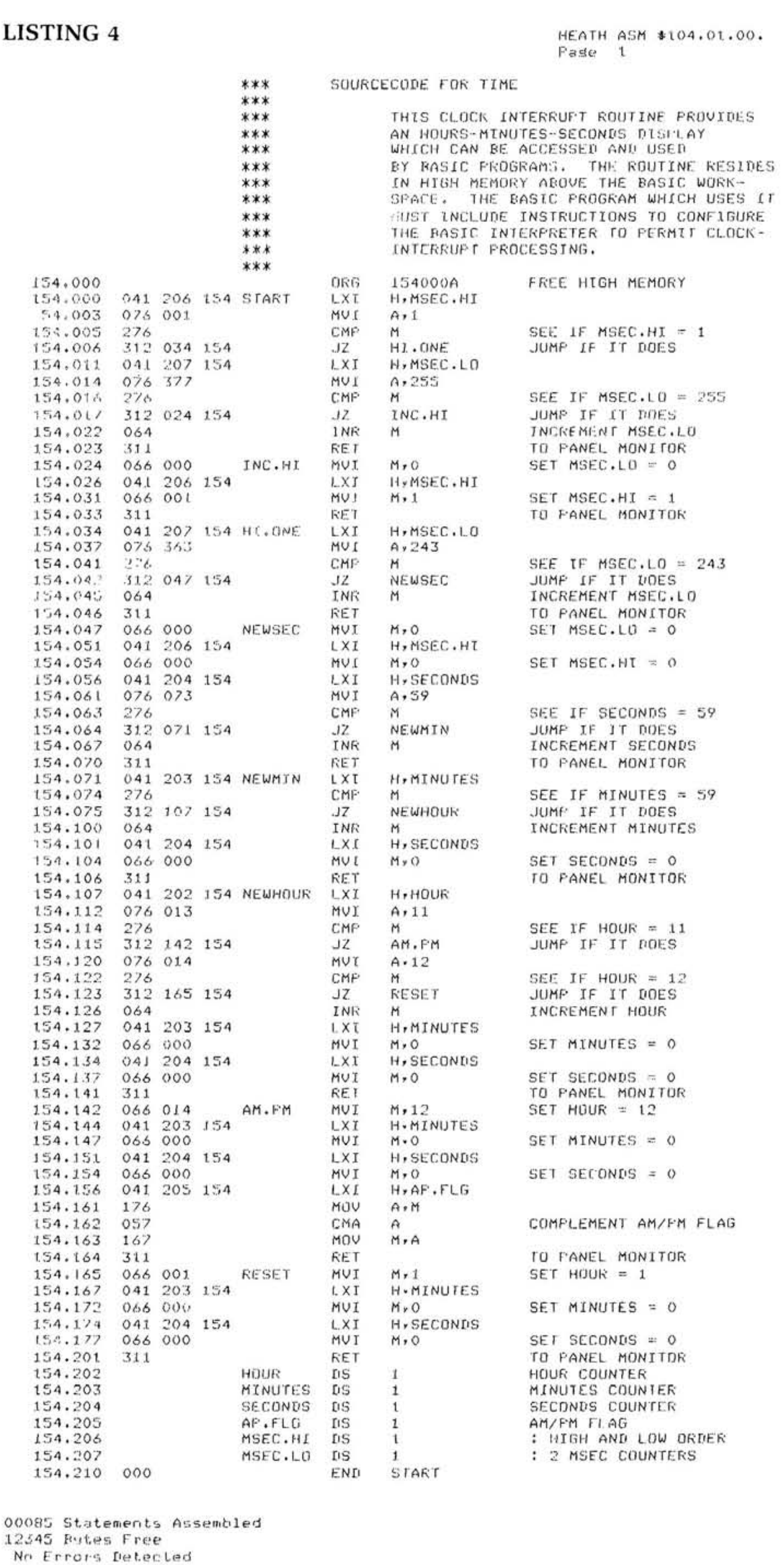
Listing 4 is the source code for such a machine language sub-routine. As in the BEEP program above, it starts at the beginning of reserved high memory, which is 154.000 offset-octal in my system. The subroutine counts clock interrupt cycles until one second has elapsed. Then it increments a seconds counter at a designated memory location. If according to standard timekeeping rules it is time to do so, it also increments a minutes counter, an hours counter and flips an A.M./P.M. flag. The real time of day must be set initially by the BASIC program unless only elapsed time is wanted.

See listing 4 on next page.

BASICLOCK

The TIME subroutine can be assembled, converted to decimal and placed in a BASIC datafile in the same manner as the BEEP subroutine was for MORSCII. The BASIC program that uses the TIME su! routine can access the time of day by taking the appropriate values for the hour, minute, second and A.M./P.M. flag out of

LISTING 4



```
                                                      HEATH ASM $104.01.00.
                                                      Page  1

                          ***         SOURCECODE FOR TIME
                          ***
                          ***              THIS CLOCK INTERRUPT ROUTINE PROVIDES
                          ***              AN HOURS-MINUTES-SECONDS DISPLAY
                          ***              WHICH CAN BE ACCESSED AND USED
                          ***              BY BASIC PROGRAMS.  THE ROUTINE RESIDES
                          ***              IN HIGH MEMORY ABOVE THE BASIC WORK-
                          ***              SPACE.  THE BASIC PROGRAM WHICH USES IT
                          ***              MUST INCLUDE INSTRUCTIONS TO CONFIGURE
                          ***              THE BASIC INTERPRETER TO PERMIT CLOCK-
                          ***              INTERRUPT PROCESSING.
                          ***
154.000                           ORG    154000A       FREE HIGH MEMORY
154.000  041 206 154 START        LXI    H,MSEC.HI
154.003  076 001                  MVI    A,1
154.005  276                      CMP    M             SEE IF MSEC.HI = 1
154.006  312 034 154              JZ     HI.ONE        JUMP IF IT DOES
154.011  041 207 154              LXI    H,MSEC.LO
154.014  076 377                  MVI    A,255
154.016  276                      CMP    M             SEE IF MSEC.LO = 255
154.017  312 024 154              JZ     INC.HI        JUMP IF IT DOES
154.022  064                      INR    M             INCREMENT MSEC.LO
154.023  311                      RET                  TO PANEL MONITOR
154.024  066 000     INC.HI       MVI    M,0           SET MSEC.LO = 0
154.026  041 206 154              LXI    H,MSEC.HI
154.031  066 001                  MVI    M,1           SET MSEC.HI = 1
154.033  311                      RET                  TO PANEL MONITOR
154.034  041 207 154 HI.ONE       LXI    H,MSEC.LO
154.037  076 363                  MVI    A,243
154.041  276                      CMP    M             SEE IF MSEC.LO = 243
154.042  312 047 154              JZ     NEWSEC        JUMP IF IT DOES
154.045  064                      INR    M             INCREMENT MSEC.LO
154.046  311                      RET                  TO PANEL MONITOR
154.047  066 000     NEWSEC       MVI    M,0           SET MSEC.LO = 0
154.051  041 206 154              LXI    H,MSEC.HI
154.054  066 000                  MVI    M,0           SET MSEC.HI = 0
154.056  041 204 154              LXI    H,SECONDS
154.061  076 073                  MVI    A,59
154.063  276                      CMP    M             SEE IF SECONDS = 59
154.064  312 071 154              JZ     NEWMIN        JUMP IF IT DOES
154.067  064                      INR    M             INCREMENT SECONDS
154.070  311                      RET                  TO PANEL MONITOR
154.071  041 203 154 NEWMIN       LXI    H,MINUTES
154.074  276                      CMP    M             SEE IF MINUTES = 59
154.075  312 107 154              JZ     NEWHOUR       JUMP IF IT DOES
154.100  064                      INR    M             INCREMENT MINUTES
154.101  041 204 154              LXI    H,SECONDS
154.104  066 000                  MVI    M,0           SET SECONDS = 0
154.106  311                      RET                  TO PANEL MONITOR
154.107  041 202 154 NEWHOUR      LXI    H,HOUR
154.112  076 013                  MVI    A,11
154.114  276                      CMP    M             SEE IF HOUR = 11
154.115  312 142 154              JZ     AM.PM         JUMP IF IT DOES
154.120  076 014                  MVI    A.12
154.122  276                      CMP    M             SEE IF HOUR = 12
154.123  312 165 154              JZ     RESET         JUMP IF IT DOES
154.126  064                      INR    M             INCREMENT HOUR
154.127  041 203 154              LXI    H,MINUTES
154.132  066 000                  MVI    M,0           SET MINUTES = 0
154.134  041 204 154              LXI    H,SECONDS
154.137  066 000                  MVI    M,0           SET SECONDS = 0
154.141  311                      RET                  TO PANEL MONITOR
154.142  066 014     AM.PM        MVI    M,12          SET HOUR = 12
154.144  041 203 154              LXI    H.MINUTES
154.147  066 000                  MVI    M.0           SET MINUTES = 0
154.151  041 204 154              LXI    H,SECONDS
154.154  066 000                  MVI    M,0           SET SECONDS = 0
154.156  041 205 154              LXI    H,AP.FLG
154.161  176                      MOV    A,M
154.162  057                      CMA    A             COMPLEMENT AM/PM FLAG
154.163  167                      MOV    M,A
154.164  311                      RET                  TO PANEL MONITOR
154.165  066 001     RESET        MVI    M,1           SET HOUR = 1
154.167  041 203 154              LXI    H.MINUTES
154.172  066 000                  MVI    M,0           SET MINUTES = 0
154.174  041 204 154              LXI    H,SECONDS
154.177  066 000                  MVI    M,0           SET SECONDS = 0
154.201  311                      RET                  TO PANEL MONITOR
154.202              HOUR         DS     1             HOUR COUNTER
154.203              MINUTES      DS     1             MINUTES COUNTER
154.204              SECONDS      DS     1             SECONDS COUNTER
154.205              AP.FLG       DS     1             AM/PM FLAG
154.206              MSEC.HI      DS     1             : HIGH AND LOW ORDER
154.207              MSEC.LO      DS     1             : 2 MSEC COUNTERS
154.210  000                      END    START

00085 Statements Assembled
12345 Bytes Free
 No Errors Detected
```

memory with a PEEK statement. Although the time-of-day subroutine uses octal notation internally (or, more precisely, binary), the BASIC interpreter will convert the numbers to the appropriate decimal values when it accesses them.

Listing 5, BASICLOCK, is a program which demonstrates the operation of the time-of-day subroutine. The program has two distinct parts. The second part, which formats the time-of-day and displays it on a video terminal, will normally be replaced by a user program that needs the time of day. The first part, which sets up the subroutine, requires some explaining.

Writing the machine language subroutine into reserved high memory is done in the same way as in the MORSCII program. But before a clock interrupt subroutine can be executed in Heath BASIC, two additional operations must be performed. The first is to place the starting address of the subroutine into the UIVEC clock interrupt jump provided by the H8 panel monitor. This jump vector is described in the panel monitor listing provided in the Heath software manual at pages 1-26 and 1-60. Then the BASIC interpreter must be configured to accept clock interrupts. With Heath BASIC — the 8K version — this is done by setting the bit at memory address 040.010 as described in the software manual. But with Extended BASIC — as I learned after an hour or so with a disassembler — a more complicated procedure must be followed. The memory locations which must be changed, and the values to be POKEd into them — are described in the BASICLOCK listing.

The two techniques described here can be combined to produce BASIC programs with much more sophisticated machine language subroutines. Device control is probably the most important application since many machines will need instructions from the computer faster than a BASIC program can normally supply them. Machine language subroutines can also speed up the computation of complex mathematical expressions. And they can reduce — or eliminate — the waiting periods between moves in complicated computer games.

See listing 5, on Page 8.

EOF

LISTING 1

```
                                                   HEATH ASM #   4.01.00.
                                                   Page  1

                           *        SOURCECODE FOR BEEP
                           *
                           *        SOUNDS H8 AUDIO OSCILLATOR
                           *        ENTRY: (A) = DURATION IN MILLISECONDS/2
                           *
040.067                    M2       EQU   40067A        USR( ARG MANTISSA (2)
040.070                    M1       EQU   40070A        USR( ARG MANTISSA (1)
040.071                    EX       EQU   40071A        USR( ARG EXPONENT
000.200                    CB.SPK   EQU   200A          SPEAKER ENABLE
040.011                    CTLFLG   EQU   40011A        FRONT PANEL CONTROL BITS
040.033                    TICCNT   EQU   40033A        CLOCK TIC COUNTER
154.000                             ORG   154000A
154.000  041 071 040 BEGIN          LXI   H,EX
154.003  076 210     ROTM1          MVI   A,210Q        : DECODE BASIC'S FLOATING
154.005  276                        CMP   M             : POINT ACCUMULATOR
154.006  312 024 154                JE    ADDUP         - - - - - - - - - - - - -
154.011  064                        INR   M             : FOR A DESCRIPTION OF
154.012  072 070 040                LDA   M1            : THIS ACCUMULATOR, SEE
154.015  017                        RRC                 : SOFTWARE MANUAL,
154.016  062 070 040                STA   M1            : P. 5-101.
154.021  303 003 154                JMP   ROTM1
154.024  072 070 040 ADDUP          LDA   M1
154.027  007                        RLC
154.030  062 070 040                STA   M1
154.033  072 067 040                LDA   M2
154.036  007                        RLC
154.037  041 070 040                LXI   H,M1
154.042  206                        ADD   M
154.043  127                        MOV   D,A           (D)=BEEP DURATION
154.044  076 200                    MVI   A,CB.SPK
154.046  041 011 040                LXI   H,CTLFLG
154.051  256                        XRA   M             VALUE OF CTLFLG W/BEEP
154.052  136                        MOV   E,M           (E)=OLD CTLFLG VALUE
154.053  167                        MOV   M,A           START BEEP
154.054  056 033                    MVI   L,#TICCNT
154.056  172                        MOV   A,D           (A)=BEEP DURATION
154.057  206                        ADD   M
154.060  276         COUNTER        CMP   M             SEE IF TIME TO STOP
154.061  302 060 154                JNE   COUNTER       IF NOT, GO BACK
154.064  056 011                    MVI   L,#CTLFLG
154.066  163                        MOV   M,E           STOP BEEP
154.067  311                        RET
154.070  000                        END   BEGIN

00043 Statements Assembled
12395 Bytes Free
 No Errors Detected
```

LISTING 2

```
1000 REM DECOCT
1010 CNTRL 0,9000
1020 PRINT
1030 INPUT "APPROXIMATE NUMBER OF BYTES IN PROGRAM TO BE ENTERED: ";B
1040 DIM A(B),D(B)
1050 LINE INPUT "CHOOSE INPUT METHOD ('KEYBOARD' OR 'BINARY TAPE'): ";I$
1060 IF ASC(I$)=66 OR ASC(I$)=75 THEN 1090
1070 PRINT "YOU MUST TYPE 'KEYBOARD' OR 'BINARY TAPE.'"
1080 GOTO 1050
1090 IF ASC(I$) <> 75 THEN 1110
1100 GOTO 5000
1110 IF ASC(I$) <> 66 THEN 3000
1120 GOTO 6000
3000 PRINT
3010 CLEAR N3:CLEAR N4:CLEAR N5
3020 LINE INPUT "WANT PROGRAM DISPLAYED (YES OR NO)? ";Q3$
3030 IF Q3$ < "Y" THEN 4000
```

Listing 2 (cont'd)

```
3040 LINE INPUT "WANT ADDRESSES DISPLAYED IN DECIMAL OR OCTAL? ";Q4$
3050 LINE INPUT "WANT DATA DISPLAYED IN DECIMAL OR OCTAL? ";Q5$
3060 PRINT "TYPE A 'RETURN' TO SCROLL DISPLAY.  TYPE A 'CONTROL-B'"
3070 PRINT "TO CORRECT AN ERROR:"
3080 PRINT
3090 PRINT TAB(15);"ADDRESS";TAB(25);"DATA"
3100 FOR J=0 TO I-1
3110 N=A(J)
3115 PRINT TAB(15);
3120 IF Q4$ < "O" THEN 3180
3130 GOSUB 8000
3140 IF INT(LOG(N)/LOG(10)) = 5 THEN 3180
3150 FOR L=1 TO 5-INT(LOG(N)/LOG(10))
3160 PRINT "0";
3170 NEXT
3180 PRINT MID$(STR$(N),2);TAB(25);
3190 N=D(J)
3200 IF Q5$ < "O" THEN 3300
3210 CLEAR N3:CLEAR N4:CLEAR N5
3220 GOSUB 8500
3230 O=2
3240 IF N=0 THEN 3270
3250 O=2-INT(LOG(N)/LOG(10))
3260 IF O=0 THEN 3300
3270 FOR L=1 TO O
3280 PRINT "0";
3290 NEXT
3300 PRINT MID$(STR$(N),2);
3310 IF K=7 THEN 3330
3320 IF (K-7)/12 <> INT((K-7)/12) THEN 3340
3330 PAUSE
3340 K=K+1
3350 PRINT
3360 NEXT J
4000 PRINT
4010 LINE INPUT "WANT TO PUT DECIMAL DATA ON MAG TAPE (YES OR NO)? ";Q6$
4020 IF Q6$ < "Y" THEN 4270
4030 CLEAR J:CLEAR K:CLEAR L:CLEAR O:CLEAR X
4040 CLEAR N:CLEAR N1:CLEAR N2:CLEAR N3:CLEAR N4:CLEAR N5
4050 Q0$="":Q1$="":Q2$="":Q3$="":Q4$="":Q5$="":Q6$=""
4060 Q7$="":Q8$="":Q9$="":I$="":N$="":D$=""
4070 CLEAR Q0$:CLEAR Q1$:CLEAR Q2$:CLEAR Q3$:CLEAR Q4$:CLEAR Q5$:CLEAR Q6$
4080 CLEAR Q7$:CLEAR Q8$:CLEAR Q9$:CLEAR I$:CLEAR N$:CLEAR D$
4090 S=A(0)
4100 E=A(I-1)
4110 CLEAR A(
4120 FOR I=E-S+1 TO B
4130 D(I)=0
4140 NEXT
4150 CLEAR B:CLEAR I
4160 PRINT
4170 PRINT "DECIMAL DATA FORMAT:"
4180 PRINT
4190 PRINT "STARTING ADDRESS = S";SPC(10);
4200 PRINT "ENDING ADDRESS = E"
4210 PRINT "PROGRAM BYTES = D("
4220 PRINT "    -- FIRST BYTE = D(0)"
4230 PRINT "    -- LAST BYTE = D(E-S)"
4240 PRINT
4250 PRINT "WHEN PROGRAM ENDS, USE 'PUT' COMMAND TO RECORD"
4260 PRINT "DECIMAL DATA ON MAG TAPE."
4270 END
5000 REM KEYBOARD ENTRY SUBROUTINE
5010 LINE INPUT "WANT TO ENTER STARTING ADDRESS IN DECIMAL OR OCTAL? ";Q0$
5020 LINE INPUT "STARTING ADDRESS? ";Q1$
5030 IF Q0$ < "O" THEN 5100
5040 FOR I=1 TO LEN(Q1$)
5050 IF MID$(Q1$,I,1) = CHR$(32) THEN 5070
5060 N$ = N$ + MID$(Q1$,I,1)
5070 NEXT
5080 Q1$ = N$
5090 CLEAR I
```

Listing 2 (cont'd)

```
5100 N=VAL(Q1$)
5110 IF Q0$ < "O" THEN 5130
5120 GOSUB 7000
5130 A(I)=N
5140 PRINT
5150 LINE INPUT "WANT TO ENTER DATA IN DECIMAL OR OCTAL? ";Q2$
5160 PRINT
5170 PRINT "ENTER DATA ONE BYTE AT A TIME.  ENTER 'FIX' TO"
5180 PRINT "CORRECT AN ERROR.  ENTER 'END' WHEN FINISHED."
5190 PRINT
5200 PRINT "ADDRESS";TAB(10);"DATA"
5205 PRINT
5210 IF Q0$ < "O" THEN 5270
5220 GOSUB 8000
5230 IF INT(LOG(N)/LOG(10)) = 5 THEN 5270
5240 FOR J=1 TO 5-INT(LOG(N)/LOG(10))
5250 PRINT "0";
5260 NEXT
5270 PRINT MID$(STR$(N),2);TAB(10);
5280 LINE INPUT "";D$
5290 IF D$ >= "F" THEN 9000
5300 IF D$ < CHR$(58) THEN 5310
5305 GOTO 3000
5310 N=VAL(D$)
5320 IF Q2$ < "O" THEN 5370
5330 IF VAL(MID$(D$,1,1)) > 3 THEN 5900
5340 IF VAL(MID$(D$,2,1)) > 7 OR VAL(MID$(D$,3,1)) > 7 THEN 5900
5350 CLEAR N3:CLEAR N4:CLEAR N5
5360 GOSUB 7500
5370 D(I)=N
5380 A(I+1) = A(I)+1
5390 I=I+1
5400 N=A(I)
5410 GOTO 5210
5900 PRINT "DATA NOT IN OCTAL FORMAT.  RE-ENTER BYTE."
5910 PRINT TAB(10);
5920 GOTO 5280
6000 REM BINARY TAPE SUBROUTINE
6010 REM
6020 REM WRITE CALL TO PAM 'RMEM' ROUTINE INTO MEMORY
6030 POKE 28668,205
6040 POKE 28669,177
6050 POKE 28670,1
6060 POKE 28671,201
6070 REM CHANGE ADDRESS OF 'USRFN' TO 28668(D)
6080 POKE 17990,252
6090 POKE 17991,111
6100 PRINT "PLACE TAPE IN MACHINE.  TYPE A 'RETURN' WHEN READY."
6110 PAUSE
6120 CNTRL 2,2
6130 X=USR(0)
6140 CNTRL 2,0
6150 PRINT
6155 PRINT CHR$(7);"BINARY TAPE OKAY."
6160 FOR I=0 TO B
6170 IF PEEK(I+27648)=199 THEN 6210
6180 A(I)=I+27648
6190 D(I)=PEEK(I+27648)
6200 NEXT
6210 REM RETURN ADDRESS OF 'USRFN' TO 27648(D)
6220 POKE 17990,0
6230 POKE 17991,108
6240 GOTO 3000
7000 REM OCTAL-TO-DECIMAL SUBROUTINE
7010 IF N <= 377377 THEN 7050
7020 PRINT "NUMBER EXCEEDS 16 BITS."
7030 GOTO 9000
7040 REM HIGH 3 DIGITS
7050 N5 = INT(N/100000)
7060 N = N - N5*100000
7070 N4 = INT(N/10000)
7080 N = N - N4*10000
```

Listing 2 (cont'd)

```
7090 N3 = INT(N/1000)
7100 N = N - N3*1000
7430 REM LOW 3 DIGITS
7500 IF N <= 377 THEN 7540
7510 PRINT "NUMBER (OR OFFSET-OCTAL SEGMENT) EXCEEDS 8 BITS."
7520 GOTO 9000
7540 N2 = INT(N/100)
7550 N = N - N2*100
7560 N1 = INT(N/10)
7570 N = N - N1*10
7580 N = N5*16384 + N4*2048 + N3*256 + N2*64 + N1*8 + N
7590 RETURN
8000 REM DECIMAL-TO-OCTAL SUBROUTINE
8010 IF N <= 65535 THEN 8050
8020 PRINT "NUMBER EXCEEDS 16 BITS."
8030 GOTO 9000
8040 REM 256-65535 DECIMAL
8050 N5 = INT(N/16384)
8060 N = N - N5*16384
8070 N4 = INT(N/2048)
8080 N = N - N4*2048
8090 N3 = INT(N/256)
8100 N = N - N3*256
8500 IF N <= 255 THEN 8540
8510 PRINT "NUMBER EXCEEDS 8 BITS."
8520 GOTO 9000
8530 REM 0-255 DECIMAL
8540 N2 = INT(N/64)
8550 N = N - N2*64
8560 N1 = INT(N/8)
8570 N = N - N1*8
8580 N = N5*100000 + N4*10000 + N3*1000 + N2*100 + N1*10 + N
8590 RETURN
9000 REM ERROR SUBROUTINE
9010 PRINT "NOTE ADDRESS OF ERROR. TYPE 'RETURN' TO CONTINUE."
9020 PAUSE
9030 PRINT "WANT TO RE-ENTER DATA FROM THIS POINT (TYPE 'RE-ENTER') OR"
9040 PRINT "BEGIN NEW PROGRAM RUN (TYPE 'NEW RUN')? ";
9050 LINE INPUT "";Q7$
9060 IF Q7$ < "R" THEN 9999
9070 CLEAR K:CLEAR L:CLEAR N$
9080 PRINT
9090 PRINT "CHOOSE DECIMAL OR OCTAL FOR SPECIFYING FIRST ADDRESS"
9100 PRINT "TO BE CORRECTED: ";
9110 LINE INPUT "";Q8$
9120 LINE INPUT "ENTER FIRST ADDRESS TO BE CORRECTED: ";Q9$
9130 IF Q8$ < "O" THEN 9190
9140 FOR J=1 TO LEN(Q9$)
9150 IF MID$(Q9$,J,1) = CHR$(32) THEN 9170
9160 N$ = N$ + MID$(Q9$,J,1)
9170 NEXT
9180 Q9$ = N$
9190 N=VAL(Q9$)
9200 IF Q8$ < "O" THEN 9220
9210 GOSUB 7000
9220 FOR J=0 TO I
9230 IF N=A(J) THEN 9270
9240 NEXT
9250 PRINT "ADDRESS NOT RECORDED.  RUN PROGRAM AGAIN."
9260 END
9270 LINE INPUT "WANT TO ENTER NEW DATA IN DECIMAL OR OCTAL? ";Q2$
9280 PRINT
9290 PRINT "RE-ENTER ALL DATA BEGINNING WITH THE SPECIFIED ADDRESS."
9300 PRINT "ENTER 'END' AS DATA WHEN DONE."
9310 I=J
9320 IF Q4$ = "" THEN 9340
9330 Q0$ = Q4$
9340 PRINT
9350 PRINT "ADDRESS";TAB(10);"DATA"
9360 GOTO 5210
9999 END
```

LISTING 5

```
00100 REM BASICLOCK
00110 REM WRITE MACHINE LANGUAGE SUBROUTINE FROM DATAFILE INTO MEMORY
00120 FOR A=S TO E
00130 POKE A,D(A-S)
00140 NEXT A
00150 REM WRITE ADDRESS OF SUBROUTINE INTO CLOCK INTERRUPT VECTOR (040.040)
00160 POKE 8224,0
00170 POKE 8225,108
00180 REM CONFIGURE X-BASIC INTERPRETER TO ACCEPT CLOCK INTERRUPT PROCESSING
00190 POKE 8908,193
00200 POKE 8909,131
00210 POKE 8910,129
00220 POKE 13817,129
00230 REM CLOCK SHOULD BE RUNNING AT THIS POINT
00300 REM SET TIME
00310 REM
00320 REM THE NEXT FOUR MEMORY LOCATIONS -- CORRESPONDING TO THE HOUR,
00330 REM MINUTES AND SECONDS COUNTERS, AND THE AM/PM FLAG -- WILL
00340 REM VARY ACCORDING TO THE ADDRESS AT WHICH HIGH MEMORY IS RESERVED
00350 REM FOR MACHINE LANGUAGE SUBROUTINES.
00360 REM
00370 INPUT "SET EXACT SECOND (PRESS 'RETURN' AT MARK): ";T
00380 POKE 27780,T
00390 INPUT "SET MINUTE: ";T
00400 POKE 27779,T
00410 Z=T+5:IF Z > 59 THEN Z=Z-60
00420 INPUT "SET HOUR: ";T
00430 POKE 27778,T
00440 LINE INPUT "AM OR PM? ";Q$
00450 IF Q$ < "P" THEN POKE 27781,0:GOTO 470
00460 POKE 27781,255
00470 REM DISPLAY TIME
00480 FOR I=1 TO 11
00490 PRINT
00500 NEXT I
00510 H=PEEK(27778)
00520 M=PEEK(27779)
00530 C=PEEK(27780)
00540 F=PEEK(27781)
00550 PRINT TAB(20);
00560 IF H < 10 THEN PRINT " ";
00570 PRINT MID$(STR$(H),2);TAB(22);":";
00580 IF M < 10 THEN PRINT "0";
00590 PRINT MID$(STR$(M),2);TAB(24);":";
00600 IF C < 10 THEN PRINT "0";
00610 PRINT MID$(STR$(C),2);TAB(27);
00620 IF F < 255 THEN PRINT "A.M.";:GOTO 640
00630 PRINT "P.M.";
00640 FOR I=1 TO 15
00650 PRINT CHR$(8);
00660 NEXT I
00670 PAUSE 200
00680 IF M <> Z THEN 510
00690 REM
00700 REM IT IS NOT NECESSARY TO RECONFIGURE THE BASIC INTERPRETER WITH
00710 REM THE FOLLOWING VALUES AT THE END OF THE PROGRAM RUN.  IF THESE
00720 REM BYTES ARE LEFT AS THEY WERE SET IN THE FIRST PART OF THIS
00730 REM PROGRAM, THE TIME-OF-DAY CLOCK WILL CONTINUE TO RUN AFTER
00740 REM BASICLOCK HAS ENDED.  HOWEVER THE BASIC INTERPRETER M U S T
00750 REM BE RECONFIGURED WITH THE VALUES GIVEN IN THE NEXT SIX INSTRUCTIONS
00760 REM BEFORE A LOAD OR DUMP TO MAG TAPE IS EXECUTED.
00770 REM
00780 POKE 8908,192
00790 POKE 8909,130
00800 POKE 8910,128
00810 POKE 13817,128
00820 POKE 8224,237
00830 POKE 8225,67
00840 END
```

SPACE WARS PATCH FOR DISK SYSTEM

Change the following lines in Space Wars before attempting to run on floppy disk system.

```
3250 PRINT "? ";:X=0:A$="":L6=L7:IF K3<1THEN L6=300
3260 X=X+2.4:IF X>L6THEN A=-1:PRINT :GOTO 3300
3270 ~~REM~~ T8=CIN(0):IF T8<0 GOTO 3260
3280 A$=A$+CHR$(T8):IF T8<>10GOTO 3260
3285 ~~A$=LEN(A$)-1)~~ A$=LEFT$(A$,LEN(A$)-1)
3290 FOR I=1TO LEN(A$):A=ASC(MID$(A$,I,1))
3292 IF A=32THEN A$=MID$(A$,I+1):GOTO 3290
3294 NEXT I:A=VAL(A$)
3300 X=X/1000:GOSUB 2520:RETURN
```

SEE ISSUE #5
P. 15

WELCOME TO ADVENTURE!

Artificial Intelligence for your HDOS — see Back Page for details on how to get your copy. This is a short sample run by an amatuer we both know — :JB:

Somewhere nearby is colossal cave, where others have found fortunes in treasure and gold, though it is rumored that some who enter are never seen again. Magic is said to work in the cave. I will be your eyes and hands. Direct me with commands of 1 or 2 words. I should warn you that I look at only the first four letters of each word, so you'll have to enter "northeast" as "NE" to distinguish it from "north". (Should you get stuck, type "HELP" for some general hints. For information on how to end your adventure, etc., type "INFO".)

This program was originally developed by Willie Crowther. Most of the features of the current program were added by Don Woods (Don @ SU-AI). This microprocessor version was done by J.G. Letwi of Heath Co.

You are standing at the end of a road before a small brick building. Around you is a forest. A small stream flows out of the building and down a gully.
W

You have walked up a hill, still in the forest. The road slopes back down the other side of the hill. There is a building in the distance.
N

You are in open forest, with a deep valley to one side.
E

You are in a valley in the forest beside a stream tumbling along a rocky bed.
DNSTREAM

At your feet all the water of the stream splashes into a 2-inch slit in the rock. Downstream the streambed is bare rock.
W

You're at slit in streambed.
NE

There is no way to go that direction.

```
At your feet all the water of the stream splashes into a 2-inch slit in the rock. Downstream the streambed is bare rock.
DNSTREAM

You are in a 20-foot depression floored with bare dirt. Set into the dirt is a strong steel grate mounted in concrete. A dry streambed leads into the depression.

The grate is locked.
Unlock grate

You have no keys!
Get keys

I don't see any keys.
S

You're in forest.
S

You are in open forest, with a deep valley to one side.
E

You are in a valley in the forest beside a stream tumbling along a rocky bed.

Enter building

You are inside a building, a well house for a large spring.

There are some keys on the ground here.

There is a shiny brass lamp nearby.

There is food here.

There is a bottle of water here.
Get food

OK
Get water

OK
Get keys

OK
Get lamp

OK
S

You're at slit in streambed.
DNSTREAM

You're outside grate.

The grate is locked.
Unlock grate

The grate is open.
enter

You are in a small chamber beneath a 3 × 3 steel grate to the surface. A low crawl over cobbles leads inward to the west.

The grate is open.
W

You are crawling over cobbles in a low passage. There is a dim light at the east end of the passage.

There is a small wicker cage discarded nearby.
Get cage

OK
E

You're below the grate.
The grate is open.
W

You're in cobble crawl.
W

It is now pitch dark. If you proceed you will likely fall into a pit.
Light lamp

Your lamp is now on.

You are in a debris room filled with stuff washed in from the surface. A low wide passage with cobbles becomes plugged with mud and debris here, but an awkward canyon leads upward and west. A note on the wall says "MAGIC WORD XYZZY".

A three foot black rod with a rusty star on an end lies nearby.
Get rod

OK

You are in an awkward sloping east/west canyon.
W

You are in a splendid chamber thirty feet high. The walls are frozen rivers of orange stone. An awkward canyon and a good passage exit from east and west sides of the chamber.

A cheerful little bird is sitting here singing.
Get bird

The bird was unafraid when you entered, but as you approach it becomes disturbed and you cannot catch it.
E

You are in an awkward sloping east/west canyon.
W

You're in bird chamber.

A cheerful little bird is sitting here singing.
W

At your feet is a small pit breathing traces of white mist. An east passage ends here except for a small crack leading on.

Rough stone steps lead down the pit.
D

You are at one end of a vast hall stretching forward out of sight to the west. There are openings to either side. Nearby, a wide stone staircase leads downward. The hall is filled with wisps of white mist swaying to and fro almost as if alive. A cold wind blows up the staircase. There is a passage at the top of a dome behind you.

Rough stone steps lead up the dome.
D

You are in the hall of the mountain king, with passages off in all directions.

A hugh green fierce snake bars the way!

Throw rod

OK
Snake

What do you want to do with the snake?
Kill

Attacking the snake both doesn't work and is very dangerous.
Get rod

A little dwarf just walked around a corner, saw you, threw a little axe at you which missed, cursed, and ran away.

You're in hall of mists.
```

You should be on Page 26.

READING RTTY ON THE ET3400

By: Louis C. Graue
624 Campbell Hill Rd.
Bowling Green, ohio 43402

ET-3400 owners without teletype machines can read RTTY with the program presented here. The message flows across the seven segment readouts ticker tape fashion, that is, as each new letter is added, the message moves one space to the left.

PREPARATION

Besides loading the program, there are two other steps necessary. One is to get the mark and space tones, translated to logic high and low, into the processor. This is most simply accomplished by using a tone demodulator IC (NE567, available at Radio Shack for less than $2) as shown in Figure 1. Hook it to your speaker or phone jack terminals (also connect ground) and tune in the RTTY signal so that the LED flashes on as each space tone is produced. If you have tuned properly and there is no noise on the signal, then the words should start flowing across the readouts. With QRM or QRN, a meaningless batch of characters will flow across the readouts.

To improve reception under noisy conditions, you need a better terminal unit. I constructed one similar to the one in reference 1 and this makes it possible to get nearly perfect copy under poor conditions.

The second step is to get used to the character set necessary to display all the symbols on the seven segment readouts. First look over the character set as described in the program, then find a transmission being sent by a hunt and peck typist. The words will then stand still long enough for you to get a good look at them. After a while you will easily recognize the words and can then start to copy faster transmissions.

After a few slow sessions, I was able to read the W1AW bulletin presented at 60 words per minute.

THE PROGRAM

Similar programs have been published for other processors (reference 2 and 3) and provided ideas for this one. The microprocessor is doing the same thing as a mechanical decoder. First it waits for the start bit, then when it has detected this bit, it delays a set time until the middle of the first data bit and samples each succeeding data bit near the center of the pulse. The resulting 5 bit code is stored in a memory byte.

The flow chart in Figure 2 explains what happens next. Notice that a blank is always displayed when shifting from letters to figures or vice versa. For example, station call letters are displayed as "W 1 AW" rather than "W1AW". A space appears after the last letter before a period, question mark, or comma. This could be corrected by adding more steps to the program, but since it does not harm the readibility, if you are expecting it, I chose to stick to the shorter program.

Once you get used to this method of displaying RTTY, you will have a lot of fun watching it on your ET-3400.

EOF

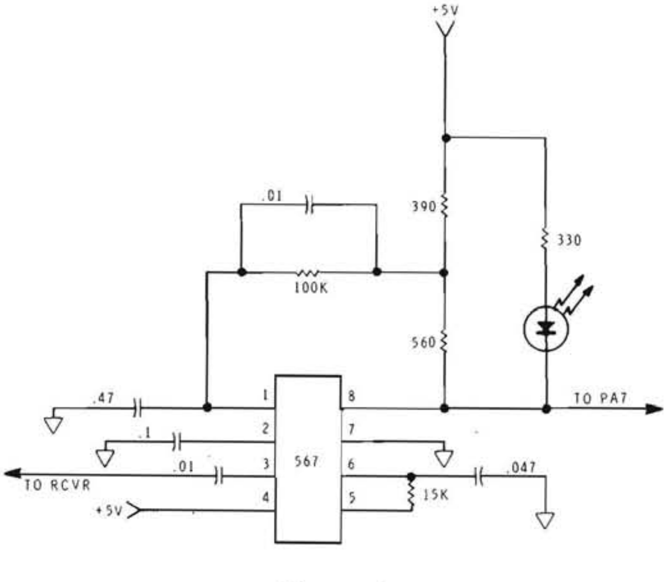


Figure 1

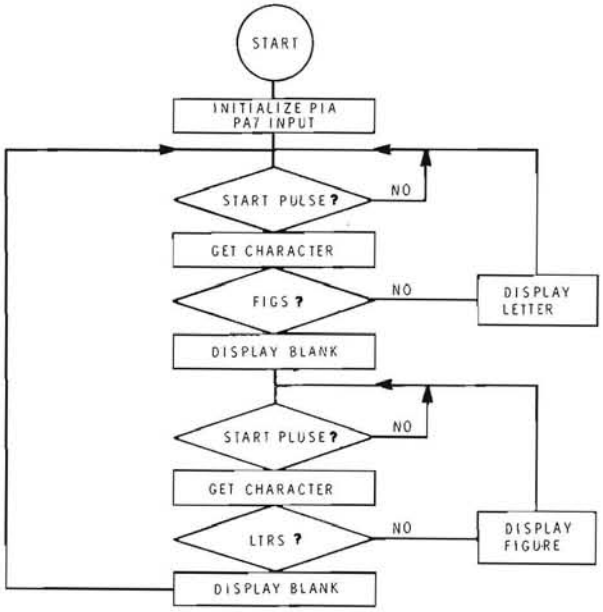


Figure 2

ROUTINE CALLING PROGRAM DATE 9/1/78

PROGRAM RTTY READER PAGE ______ OF ______

by K8TT CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	CP CODE	OPERAND	COMMENTS
0000	CE 7F 04		LDX	#$7F04	SET UP PIA
0003	FF 80 00		STX		PA7 is INPUT
0006	C6 01		LDA	B,#$01	SET UP HIGH PART
0008	D7 C2		STA	B	OF ADDRESS FOR LOOKUP TABLE
000A	8D 24	BEGIN	BSR	GETCH	CONSTRUCT BAUDOT CHARCTER
000C	96 C1		LDA	A	FETCH BAUDOT CHARACTER
000E	97 C3		STA	A	FORM POINTER TO
0010	DE C2		LDX		CONVERSION TABLE
0012	81 1B		CMP	A,#$1B	IS THIS BAUDOT "FIGS"?
0014	26 02		BNE	"LTRS"	IF NOT, GO TO "LTRS"
0016	20 06		BRA	"FIGS"	OTHERWISE, GO TO "FIGS"
0018	A6 00	"LTRS"	LDA	A,X,$00	FETCH SEGMENT CODE FROM TABLE
001A	8D 48		BSR	DISPLAY	DISPLAY THE CHARACTER
001C	20 EC		BRA	BEGIN	GO GET NEXT BAUDOT CHARACTER
001E	A6 20	"FIGS"	LDA	A,X,$ 20	FETCH SEGMENT CODE FROM TABLE
0020	8D 42		BSR	DISPLAY	DISPLAY IT
0022	8D 0C		BSR	GETCH	GO GET NEXT BAUDOT CHARACTER
0024	96 C1		LDA	A	FETCH BAUDOT CHARACTER
0026	97 C3		STA	A	FORM POINTER TO
0028	DE C2		LDX		CONVERSION TABLE
002A	81 1F		CMP	A,#$1F	IS THIS BAUDOT "LTRS"?
002C	26 F0		BNE	"FIGS"	IF NOT, GO TO "FIGS"
002E	20 E8		BRA	"LTRS"	OTHERWISE, GO TO "LTRS"

ROUTINE GETCH DATE 9/1/78

PROGRAM RTTY READER PAGE ______ OF ______

BY K8TT CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	CP CCDE	OPERAND	COMMENTS
0030	C6 05	GETCH	LDA	B,#$05	SET UP 5 BIT COUNTER
0032	B6 80 00	STPLS	LDA	A	TEST PA7 FOR START PULSE
0035	26 FB		BNE	STPLS	IF NOT, TRY AGAIN
0037	8D 1D		BSR	DELAY	OTHERWISE, DELAY ONE PULSE WIDTH
0039	8D 22		BSR	DHALF	SAMPLE AT MIDDLE OF PULSE
003B	B6 80 00	NEXT	LDA	A	LOAD ACC WITH PULSE INFO
003E	74 00 C1		LSR		SHIFT CHAR STORAGE RIGHT ONE BIT
0041	9A C1		ORA		"OR" ACC WITH CHAR STORAGE
0043	97 C1		STA	A	STORE ACC IN CHAR STORAGE
0045	8D 0F		BSR	DELAY	DELAY ONE PULSE WIDTH
0047	5A		DEC	B	DECREMENT BIT COUNT BY ONE
0048	26 F1		BNE	NEXT	IF CHARACTER NOT COMPLETE, GET NEXT BIT
004A	8D 11		BSR	DHALF	OTHERWISE, DELAY ½ PULSE WIDTH
004C	74 00 C1		LSR		SHIFT CHAR STORAGE RIGHT
004F	74 00 C1		LSR		THREE
0052	74 00 C1		LSR		TIMES
0055	39		RTS		RETURN TO CALLING PROGRAM

ROUTINE DELAY & DHALF
PROGRAM RTTY READER
BY K8TT

DATE 9/1/78
PAGE ______ OF ______
CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	OP CODE	OPERAND	COMMENTS
0056	CE 04 18	DELAY	LDX	#$0418	WAIT
0059	09	WAIT	DEX		ONE
005A	26 FD		BNE	WAIT	PULSE WIDTH
005C	39		RTS		RETURN TO CALLING PROGRAM
005D	CE 02 0C	DHALF	LDX	#$020C	WAIT
0060	09	WAIT	DEX		ONE HALF
0061	26 FD		BNE	WAIT	PULSE WIDTH
0063			RTS		RETURN TO CALLING PROGRAM
0064	97 94	DISPLAY	STA	A	PUT CHAR AT END OF STRING STORAGE
0066	CE 00 8E		LDX	#$008E	SHIFT CONTENTS OF
0069	A6 01	SHIFT	LDA	A,X,$01	STRING
006B	A7 00		STA	A,X,$00	STORAGE
006D	08		INX		BACK
006E	8C 00 94		CPX	#$0094	ONE
0071	26 F6		BNE	SHIFT	STEP
0073	CE 00 86		LDX	#$0086	TRANSFER
0076	A6 08	MOVE	LDA	A,X,$08	STORED
0078	A7 00		STA	A,X,$00	STRING
007A	08		INX		TO
007B	8C 00 8C		CPX	#$008C	OUTPUT
007E	26 F6		BNE	MOVE	STRING
0080	BD FC BC		JSR	REDIS	MOVE FIRST CHAR TO "H" DISPLAY
0083	BD FE 52		JSR	OUTSTR	DISPLAY THE STRING
0086					0086 TO 008B RESERVED FOR OUTPUT STRING
008C	80				DECIMAL POINT TO END STRING
008D	39				RETURN TO CALLING PROGRAM
008E					008E to 0094 RESERVED FOR STRING STORAGE

ROUTINE LOOK UP TABLE "LTRS"
PROGRAM RTTY READER
BY K8TT

DATE 9/1/78
PAGE ______ OF ______
CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	OP CODE	OPERAND	COMMENTS
0100	00				SPACE = NULL
0101	4F				E = E
0102	00				SPACE = LINE FEED
0103	77				A = A
0104	00				SPACE = SPACE
0105	5B				S = S
0106	06				I = I
0107	3E				U = U
0108	00				SPACE = CARRIGE RETURN
0109	3D				D = D
010A	05				R = R
010B	3C				J = J
010C	15				N = N
010D	47				F = F
010E	4E				C = C
010F	07				K = K

ROUTINE LOOK UP TABLE "LTRS"   DATE 9/1/78

PROGRAM RTTY READER   PAGE ____ OF ____

BY K8TT   CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	OP CODE	OPERAND	COMMENTS
0110	0F				= T
0111	49				= Z
0112	0E				= L
0113	3F				= W
0114	17				= H
0115	3B				= Y
0116	67				= P
0117	63				= Q
0118	1D				= O
0119	1F				= B
011A	7B				= G
011B	00				SPACE =FIGURES
011C	76				= M
011D	36				= X
011E	2B				= V
011F	00				SPACE = LETTERS
0120	00				SPACE = NULL
0121	79				= 3
0122	00				SPACE = LINE FEED
0123	01				= - (HYPHEN)
0124	00				SPACE = SPACE
0125	00				SPACE = BELL
0126	7F				= 8
0127	70				= 7
0128	00				SPACE = CARRIGE RETURN
0129	36				= $
012A	33				= 4
012B	20				= ' (APOSTROPHE)
012C	10				= , (COMMA)
012D	28				= ! (EXCLMATION POINT)
012E	09				= : (COLON)
012F	4E				= ((OPEN PARENTHESIS)
0130	5B				= 5
0131	22				= " (QUOTE)
0132	78				=) (CLOSE PARENTHESIS)
0133	6D				= 2
0134	00				SPACE = #
0135	5F				= 6
0136	7E				= Ø
0137	30				= 1
0138	73				= 9
0139	65				= ?
013A	31				= &
013B	00				SPACE = FIGURES
013C	0C				= . (PERIOD)
013D	25				= /
013E	71				= ; (SEMI-COLON)
013F	00				SPACE = LETTERS

REFERENCES

1. Build a Drift-Free T.U., J.C. Cain, VE7DBK, 73 Magazine, September 1977, p114.
2. RTTY with the KIM, 73 Magazine, September 1977, p110, Wilfred J. Gregson II, K4GCM.
3. Try your KIM-1 on RTTY, Jim Overstreet, WA5DXP, 73 Magazine, October 1977, p 88.

The BASIC IDEA

By Sam Cox, etal

It will be hard for you at HUG to regularly publish entire programs that will be of general interest to members. If REMark were to be printed on a montly basis, you could devote a single issue to one or two topics. On the current quarterly schedule, you will certainly disenchant many readers with such a plan. Moreover, most members want to develop their own versions of games and application programs. It seems to me that the proper place for software exchange is through the PROGRAM LIBRARY.

On the other hand, it is entirely appropriate for HUG to publish application independent functions, subroutines, etc. to illustrate and promote sound programming practices. Toward this end, I propose that REMark initiate an interactive forum for communicating program ideas between members.

Possibly entitled THE BASIC IDEA, the forum will consist primarily of BASIC and ASSEMBLY LANGUAGE short subjects of general interest. For the most part, the contents will be user developed /tested subroutines and single-line functions. Other useful subjects would be discussions of disciplined programming, documentation, and readability.

Entries should be short and of such a general nature so as to be readily adaptable to a specific end-use. Minimum documentation should include:

1) STATEMENT OF PURPOSE
2) VARIABLE DEFINITION
3) CALLING SEQUENCE
    - Entry Conditions
    - Variables Affected
    - Exit Conditions

THE BASIC IDEA will be an aid to programmers developing modular application packages and to others learning new and effective ways of using the language. We would all benefit through such communication.

To get the ball rolling, I submit the enclosed BASIC subroutines and single-line functions for your consideration.

The ability to define single-line functions in B.H. BASIC is a very powerful feature of the language. Functions can save memory space and improve program readability. A familiar set of personal functions can greatly reduce program development time. In some cases, the whole purpose of a program or subroutine can be changed be redefining the functions involved. Some single-line functions I have used are:

MODULUS
COMPLEMENT
BIT
LOGARITHMIC CONVERSION
FACTORIAL
TRIM$
JUSTIFY$

Functions that are too involved to fit on a single-line may be implemented as subroutines in B.H. BASIC. Of course, dedicated program variables must be used to pass arguments to subroutine 'functions'. Three useful routines that I have developed are:

MONEY FORMATTER
DECIMAL/OCTAL CONVERSION
DECIMAL/SPLIT-OCTAL CONVERSION

MODULUS FUNCTION

Common Format:	R = A MOD B
Definition:	R is the remainder after dividing B into A
Variables:	R = Remainder (Integer: $B>R\geqq 0$)
	A = Argument (Integer: $A\geqq 0$)
	B = Base (Integer: $B>0$)

A general Modulus Function for Benton Harbor BASIC is:

```
DEF FN M(A,B) = A - INT(A/B)*B
```

If your application requires Modulus to a specific base only, you can save typing by defining a more specific function.

For example, for R = A MOD 2,

```
DEF FN M2(A) = A - INT(A/2)*2
```

COMPLEMENT FUNCTION

Common Format:	R = CMP(A)
Definition:	CMP(ARG) returns the logical complement of the binary argument.
Variables:	R = Result (Integer: 0 or 1)
	A = Argument (Integer: 0 or 1)

In Benton Harbor BASIC, CMP(A) becomes

```
DEF FN C(A) = (A + 1) - INT(A + 1)/2)*2
```

or, alternatively:

```
DEF FN C(A) = FN M2(A + 1)
```

The second definition is in terms of the previoulsy defined A MOD 2 function and is completely equivalent to the first.

BIT FUNCTION:

Common Format:	R = BIT(A,B)
Definition:	R is the logical state of bit #B in the binary representation of the decimal argument A Rightmost bit (LSB) is bit #0.
Variables:	R = Result (Integer: 0 or 1)
	A = Argument (Integer: $0\leqq A$)
	B = BIT # (Integer: $0\leqq B$)

Using the previously defined A MOD 2 function, 'BIT' is easily implemented in Benton Harbor BASIC.

```
DEF FN B(A,B) = FN M2(INT(A/2↑B))
```

For example, let A = 98, (A_2 = 01100010), then;

```
FN B (A,0) = 0        FN B (A,4) = 0
FN B (A,1) = 1        FN B (A,5) = 1
FN B (A,2) = 0        FN B (A,6) = 1
FN B (A,3) = 0        FN B (A,7) = 0
```

How does 'BIT' work?

Recall that dividing a binary number by 2 is equivalent to right-shifting that number by one place. Then, dividing by 2^B will right-shift the number B places.

A MOD 2 returns the binary value (1 or 0) of the LSB of A.

Thus, INT(A/2↑B) MOD 2 returns the binary value of bit #B of argument A.

LOGARITHMIC CONVERSIONS

Purpose: To generate logarithms of any base from the built in natural log function

Definition: $LOG_B(N) = LOG_e(N)/LOG_e(B)$

Variables: B = Base (Integer: B>1)
N = Argument (Real: N>0)

n Benton Harbor BASIC, a general log conversion function is:

```
DEF FN L(N,B) = LOG(N)/LOG(B)
```

Of course, you may wish to define more specific functions for conversion to specific bases. For conversion to common logarithms, (base 10), use:

```
DEF FN L0(N) = LOG(N)/LOG(10)
```

For base 2 logarithms, a preferable function is:

```
DEF FN L2(N) = LOG(N)/.693148
```

where .693148 is used in place of LOG(2) to ensure integer results for arguments such as 1024 and 16384.

FACTORIAL FUNCTION

Common Format: R = FACT (N)

Purpose: A fast approximation to the factorial function.

Accurate to within 0.0003%.

In Benton Harbor BASIC, this somewhat complicated function is:

```
LET P2 = 2*3.141592654:  REM — This is 2 PI
DEF FN F(N) =c6z1/(12*N)))
```

This expression does have one limitation; it doesn't work for N = 0. The first noticable error occurs for N = 9.

TRIM$ FUNCTION

Purpose: To return the string equivalent of numeric arguments without leading and trailing blanks.

Definition One: DEF FN T1$(X$) = MID$(X$,2, LEN(X$)-2)
The argument, X$, is the string equivalent of a numeric value obtained from the library function, STR$(. For example, X$ =STR$(X).

Definition Two: DEF T2$(X) = MID$(STR$(X),2, LEN(STR$(X))-2)
The argument, X, is any numeric expression.

JUSTIFY$ FUNCTION

Purpose: To right-justify an input string within a character field of fixed or selectable width.

A general JUSTIFY$ function for Benton Harbor BASIC is:

```
DEF FN J$(X$,X) = LEFT$(" -spaces- ",
X-LEN(X$))+X$
```

where: X = width of character field
X$ = string to be justified. (LEN(X$) X)

For justifying to columns of fixed width, you can be more specific. If your columns need to be 10 characters wide, use:

```
DEF FN J$(X$) = LEFT$(" -10 spaces- ",10
-LEN(X$)) + X$
```

In any case, be sure to use enough blanks in the LEFT$(function to accomodate the worst-case. (The number of spaces should be at least equal to X_{max})

MONEY FORMATTER SUBROUTINE

Purpose: To provide reasonable formatting for monetary values similar to that obtainable with "PRINT USING" statements in others BASIC's.

Variables: A = ENTRY VALUE (unchanged by routine)
A$ = EXIT STRING
L = Length of A$ (no external significance)
L1 = Temp. variable used in placing commas (no external significance)

Function Used: JUSTIFY$ FUNCTION (see attached description)

Calling Sequence: A = (money value)
GOSUB 'FORMATTER'
PRINT A$

Listing:

```
100 REM — MONEY FORMATTER
105:
110 REM — TWO DIGITS FOR CENTS
```

. . . MORE IDEAS

```
115:  A$ = STR$(INT(100*A))
120:  L = LEN(A$)-2
125:  A$ = MID$(A$,2,L-2) + "." + MID$(A$,L,2)
130 REM  – INSERT COMMAS AS NECESSARY
135:  L1 = 2
140:  L1 = L1 + 4
145:  L = LEN(A$)
150:  IF L>L1 THEN A$ = LEFT$(A$,L-L1) +"," RIGHT$(A$,L1):
      GOTO 140
155 REM  – DOLLAR SIGN (OPTIONAL)
160:  A$ = "$" + A$
165 REM  – RIGHT JUSTIFICATION (OPTIONAL)
170:  A$ = FN J$(A$,10)
175 REM  – SUBROUTINE EXIT
180:  RETURN
```

DECIMAL TO OCTAL CONVERSION SUBROUTINE

Purpose: To convert decimal numbers to octal equivalent

Variables: A = ENTRY VALUE (destroyed by routine)
A1 = Temp. variable (no external significance)
A$ = EXIT STRING (no leading or trailing blanks)

Functions Used: MODULUS (see attached description)
TRIM$ (see attached description)

Calling Sequence: A = (decimal integer
GOSUB "DEC/OCT"
PRINT A$

Limitation: Restricted to positive integers

Listing:

```
100 REM  – DECIMAL TO OCTAL CONVERSION
105:
110 A$ = ""
115 A1 = FN M(A,8)
120 A$ = FN T2$(A1) + A$
125 A = INT(A/8)
130 IF A>0 THEN 115
135 RETURN
```

DECIMAL TO SPLIT — OCTAL SUBROUTINE

Purpose: Convert positive decimal integers to split-octal notation.

Variables: B = ENTRY VALUE Destroyed by routine)
B$ = EXIT STRING
A = Entry to DEC/OCTAL conversion routine

Functions and Subroutines Used:

MODULUS FUNCTION (see attached description)
ZERO-FILL FUNCTION
DEF FN Z$(X$) = LEFT$("000",3-LEN(X$))
+ X$ (to insert leading zeros)
DECIMAL – TO – OCTAL CONVERSION SUBROUTINE
(starts at line # 100: see description elsewhere)

Calling Sequence: B = (decimal value)
GOSUB "DEC / SPLIT-OCT"
PRINT B$

Listing:

```
200 REM  – DECIMAL TO SPLIT  – OCTAL CONVERSION
205:
210 REM  – LOW ORDER BYTE
215: A = FN M(B,256)
220: GOSUB 100
225: B$ = FN Z$(A$)
220: GOSUB 100
225: B$ = FN Z$(A$)
230 REM  – HIGH ORDER BYTE
235: A = INT(B/256)
240: GOSUB 100
245 B$ = FN Z$(A$) + "." + B$
250 RETURN
```

DECIMAL TO BASE-B CONVERSION

Purpose: Convert positive decimal integers to an equivalent integer in another number system.

Limitations: Decimal arguments restricted to positive integers.
Allowed bases: 2-36

Variables:

```
ENTRY D = decimal argument (unchanged by routine)
      B = base
EXIT B$
result (base-B string equivalent of decimal argument)
TEMP B9, D9 (no external significance)
```

Required Externals:

```
DEF FN M(X,Y) = Y-INT(X/Y)*Y: REM – (X MOD Y)
FUNCTION N$ = "0123456789ABCDEFGHI JKLMNOP-
QRSTUVWXYZ"
```

Calling Sequence:

```
B = (Base desired)
D = (decimal number)
GOSUB 600
PRINT B$;" = "; D
```

. . . MORE IDEAS

Listing:

```
600 REM  – DECIMAL / BASE-B CONVERSION
605 D9 = D : B$ = ""
610 B9 c6v FN M(D9,B)
615 B$ = MID$(N$,B9+1,1)+B$
620 D9 = INT(D9/B)
625 IF D9<0 THEN 610
630 RETURN
```

PRINT PAGE$

A common way of clearing the screen of a CRT terminal is to execute enough PRINT statements to scroll the text off the top. Here is an alternative method that is both faster and memory conservative. Simply define a string variable (say C$ for CLEAR$) that consists of a number of linefeed characters (LF = 10_{10} = 12_8). For the H9 terminal, 12 linefeeds will clear the display, so:

```
LET C$ = CHR$(10) + CHR$(10)
LET C$ = C$ + C$ + C$ + C$ + C$ + C$
```

Then, PRINT C$ will clear the H9 screen. At 9600 baud, this really goes fast! Similarly, PRINT LEFT$(C$,5) will rapidly space 5 blank lines.

You may wish to include a carriage return character (CR = 13_{10} = 15_8) ahead of C$. Then simply define:

```
C$ = CHR$(13) + C$
```

This method of clearing the screen has another important advantage. When you change the terminal or printing port of your H8 system, you can easily change all those CLEAR SCREEN commands to something more suitable for your new device.

RANDOMIZE:

An effective RANDOMIZE statement for Benton Harbor BASIC is:

```
PAUSE RND(-PEEK(8219))
```

where 8219_{10} is the address of the low-order byte of TICCNT. This sure beats having the user enter a seed value every time.

CENTER$ FUNCTION:

Purpose: A convenient means of centering titles for reports, etc.

Definition:

```
LET B$ = "-10 spaces-"
LET B$ = B$ + B$ + B$ + B$
DEF FN C$(X$) = LEFT$(B$,40-INT(LEN(X$)/2))+X$
```

Typical Use:

```
T$ = "TITLE OF REPORT"
PRINT FN C$(T$)
```

PROGRAM READABILITY

Here are some ideas for producing readable BASIC language programs.

A) Let blank program lines separate logically distinct sections of code. This can be done by typing a single colon (:) after a new line number.

B) Title each program block with a 3-5 word title descriptive of its function. Within a module, indent remark statements 3-4 spaces to emphasize submodule features. Indented remarks really stand out.

C) Label each user defined function with an easily pronouncible title.

D) With the absence of a RENumber command in Benton Harbor BASIC, it makes sense to organize programs around blocks of line numbers.

Example: Subroutine A – lines 100 to 199
Subroutine B – lines 200 to 299
etc.

POSITION$ SUBROUTINE

Purpose: Determine the presence and location of one string within another.

Variables:
ENTRY S1$ = substring to find
S2$ = string to look in
EXIT P = numerical position of first occurrence of S1$ in S2$ (P>0)
-or-
= 0 (S1$ not found in S2$)
-or-
= -1(LEN(S1$)>LEN(S2$) or LEN(S1$) =0)
TEMP L1, L2, P1 (no external significance)

Calling Sequence (Typical):

```
S1$ = "SUBSTRING"
S2$ = "HOST STRING"
GOSUB 1000
IF P = -1 THEN PRINT "LENGTH ERROR"
IF P = 0 THEN PRINT S1$;" NOT IN ";S2$
IF P>0 THEN PRINT S1$;" STARTS AT CHARACTER";P
```

Listing:

```
1000 REM  – POSITION$
1005 L1=LEN(S1$) : P1=LEN(S2$) + 1 –L1
1010 P=1 : IF P1<1 OR L1<1 THEN P=-1 : RETURN
1015 IF MID$(S2$,P,L1)=S1$ THEN RETURN
1020 P=P+1 : IF P<P1 GOTO 1015
1025 P=0 : RETURN
```

Continued on Page 24.

when you know how. :JB:

TED 8 + HASL 8 = NO HASSLE

OK. . . Class, today we are going to learn how to write a simple assembly language program using TED 8 and HASL 8. Here it is, step-by-step. Fire up your machine and pay attention to teacher.

Load TED 8 . . . Hit the 'GO' button on the front panel of your H8 and hit 'RETURN'. (This by-passes the opening dialogue, but we don't need to get into that. It is explained in the manual and we only got a couple pages to do this in.)

Type 'T'. The word TAB will be completed for you. This allows you to set tabs. This program was written with the tabs set at ten spaces. Now type, 10,20,30,40,50. This will provide you with five fields of ten spaces each.

Now type 'I'. The word INSERT will be completed for you. Now we are ready to insert text in TED 8. What text? We are going to write a program that will:

- Print characters on the terminal
- Get characters off the keyboard
- Store each character in consecutive memory locations
- Retrieve each character
- Test for end of line
- Print each line 10 times on the terminal

Sound complicated? Duck soup! Let't take one thing at at time. What's the name of the program? 'WRITE MY NAME TEN TIMES'. OK. HASL 8 has some psuedo-ops, one of which is 'TITLE' and it goes in the operand field. Who's waving his hand in the back of the class? What's an operand field and what difference does it make? Geez, you want to know everything? Ok . . here is a strict format that must be followed. (one of the reasons we set tabs) When writing your program there are four key ingredients.

```
LABEL  OPCODE  OPERAND  COMMENTS & DOCUMEN-
                        TATION
```

They will be explained as we go along. For now; however, give your program a title like so.

```
TITLE  'WRITE MY NAME TEN TIMES'
```

Good! Now where is this program to reside in memory? Pick a spot. We are going to start this one at 040100 split octal. Here's how we tell HASL 8 about it.

```
ORG  040100A  THE 'A' MEANS SPLIT OCTAL,(Q MEANS
              OCTAL AND HASL 8 DEFAULTS TO DECI-
              MAL)
```

Now what? We got a title and the starting location in memory. We have to do some necessary housekeeping, but we don't have space to discuss that here and it requires an explanation of some of the hardware. This routine is identified as 'INIT' in the source listing.

Some more housekeeping chores include telling HASL 8 about such things as where is your terminal . . . What's a carriage return, etc. and we do this with 'EQUATE' statements.. Why? Well, let's say you wrote a big long program for a system that had its terminal at PORT YYY and you wanted to run the program on a system with its terminal at PORT XXX. Can you imagine how long it would take you to go through the entire program and change all the YYY's to XXX? If we identify the 'TERM' in an equate statement, then all we have to do is change one statement, reassemble and the job is done! So here's stuff we have to equate.

```
TERM      EQU       372Q      OUR TERMINAL IS AT
                              PORT 372 OCTAL
CR        QU        015Q      CARRIAGE RETURN IS
                              15Q. FROM NOW ON WE
                              DON'T HAVE TO RE-
                              MEMBER THAT. . WHEN
                              YOU WANT A CAR-
                              RIAGE RETURN. . . JUST
                              TYPE CR
LF        EQU       012Q      THIS IS A LINE FEED
                              ANYTHING ELSE YOU
                              THINK YOU MIGHT
                              WANT SHOULD BE
                              IDENTIFIED HERE. . FOR
                              INSTANCE A ROUTINE
                              OUT OF PAM 8. THE
                              HORN. WANT TO BEEP
                              THE HORN? OK. .
$HORN     EQU       002136A   DONE! FROM NOW ON,
                              ALL YOU HAVE TO DO
                              IS 'CALL $HORN' AND
                              THE PROGRAM WILL
                              'KNOW' WHAT A
                              $HORN IS AND WHERE
                              TO FIND IT.
```

All right, let's begin. Type. .

```
BEGIN     LXI       SP,STACK  DEFINE THE STACK
                              AREA. EASY ENOUGH.
```

```
*See that '*'? If you make any comments like this or want to leave
*a space, preceed the line with an '*'! The HL register is a neat
*'POINTER'. It can be used to point to locations in memory, like
our message. So. . .
*
*LET'S DEFINE THE 'FIELDS' AGAIN.

*LABEL    OPCODE OPERAND    COMMENTS         OR
                            DOCUMENTATION
*
          LXI       H,$MESAG  MAKE THE (HL) RE-
                              GISTER PAIR POINT TO
                              OUR MESSAGE
$MESAG    DB        CR,LF,LF, 'WHAT IS YOUR
                              NAME?' ,LF,LF,CR,0
```

*
*Now what? We want to get one character at a time out of
*$MESAG and print it on the terminal. But first, we must realize
*that the computer can spit out characters at a blinding speed
*and no terminal can print that fast so we have to waste time
*between each character before sending out another. Here's
*how we do that. . Another hardware lesson. When the USART
*on the H8-5 is busy sending a character out, bit 0 will be low.
*Stated another way, bit 0 high means I'm ready, send me
*another character. So we can test bit zero between each charac-
*ter to insure that none are lost. So let's write a subroutine for
*this purpose to be 'CALLED' anytime we want it. This avoids
*'RE-INVENTING THE WHEEL' each time and reduces typing.
*Remember that all input and output of the 8080 is through the
*accumulator. How do we input the status of the USART.
*No problem. Don't type this routine now . . put it down near the
*end.
*

```
$TEST      IN       TERM1      THE STATUS PORT IS
                               372Q+1 or 373Q
           RAR                 ROLL BIT 0 INTO THE
                               CARRY FLAG AND
                               TEST FOR READY
           JNC      $TEST      JUMP IF CARRY FLAG
                               NOT SET. OR WASTE
*                              TIME IN A LOOP UNTIL
                               THE TERMINAL CAN
*                              COPE WITH ANOTHER
                               CHARACTER. WHEN
*                              HE CAN, RETURN.
           RET                 GO BACK TO CALLING
                               PROGRAM.
*
```

*OK. We got a message stuck down in memory someplace and
*the terminal is ready to accept a character. . Let's send him
*one. Looking through my OPCODE tool box, we find one that
*says 'MVI A,M' or move immediate the byte that the (HL)
*register is pointing to, to the accumulator. Do it
*

```
$GETCHR    CALL     $TEST      WAIT AROUND UNTIL
                               READY
           MVI      A,M        PLOP GOOD OL'
                               CHARACTER IN A
           OUT      TERM       AND SEND HIM TO
                               TERMINAL. NOW
                               WASN'T THAT EASY?
           CPI      0          SEE IF IT IS THE LAST
                               CHARACTER AND IF IT
                               IS
           JZ       $RCHAR     BAIL OUT OF THIS
                               ROUTINE AND GOTO
*                              ROUTINE CALLED
                               '$RCHAR'.
           INX      H          IF NOT, MAKE THE
                               (HL) POINT TO THE
                               NEXT CHARACTER
           JMP      $GETCHR    AND GO WAIT. KEEP
                               GOING. THIS IS FUN.
```

*Next project? Get characters from the key board. Same prob-
*lem we had as before. . We can't type as fast as the computer
*can gobble key strokes off the key board, so we have him wait
*around for the action. Let's write another subroutine that will
*wait on us. . Call is '$WAIT'. Don't type this routine in yet
*either.
*

```
$WAIT      IN       TERM1      LOOK AT THE STATUS
                               WORD AGAIN
           ANI      002Q       BIT 1 IS THE GUY WE
                               HAVE TO CHECK ON
                               THIS TIME
           JZ       $WAIT      IF IT IS SET, THAT
                               MEANS DATA IS AV-
                               AILABLE SO
           RET                 GO GOBBLE HIM UP.
*
*     HERE'S THE GOBBLE ROUTINE
$RCHAR     LXI      H,$INBUF   MAKE (HL) POINT TO
                               $INBUF WHICH WE
*                              WILL DEFINE LATER
                               AS THE PLACE WHERE
*                              WE WILL STORE OUR
                               NAME
           CALL     $WAIT      WAIT FOR DATA AV-
                               AILABLE
           IN       TERM       GOBBLE CHARACTER
                               AND
           ANI      177Q       LOOK HIM OVER.
                               DON'T NEED PARITY
                               BIT SO STRIP IT.
           MOV      M,A        STICK HIM IN MEM-
                               ORY
           INX      H          MAKE HL POINT TO
                               NEXT EMPTY MEM-
                               ORY LOCATION
*
```

*Wait a minute. . Question from the class. . . Some clown wants
*to know how we are going to tell if the guy is done typing his
*name on the keyboard. Glad you asked. Well, he will probably
*type a carriage return when he's done. Right? Ok. We can use
*the CPI OPCODE to check on it. If it's a CR, then what?? Jump
*on zero (it matches) to next task.
*

```
           CPI      CR         IS IT CARRIAGE RE-
                               TURN?
           JZ       PRINT      MUST BE. . GO PRINT
                               MY NAME TEN TIMES.
*
```

*Whoa! Another problem. Ya gotta tell these computers every
*thing. I don't see any means by which the character we type on
*the keyboard would be displayed on the screen! Bummer!
*Fix it.
*

```
*          OUT      TERM       ECHO CHARAC.
                               HAPPY NOW?
*
```

*OK we got a character in the accumulator, we've sent it back
*to the terminal (although it's still safe and sound in the
*accumulator and memory location N) and we've tested to see
*if it's the end of the line. Next. It isn't end of line, so go
*gobble another character.
*

```
           JNZ      $RCHAR     DOESN'T MATCH. . GO
                               GOBBLE UP ANOTHER
                               ONE
*
```

```
*Next project. . We have our name safely tucked away in mem-
*ory. Now we said we wanted to see it printed ten times. Right?
*Let's do it.
*
*Ten times, huh? Need a counter. . Got lots of registers we
*haven't used. So let's just reach in and blindly pick out.
*.'B'.
*
         MVI     B,10        MOVE IMMEDIATE 10
                             TO B
PRINT    LXI     H,$INBUF    MAKE THE (HL) POINT
                             TO THE BEGINNING
*                            OF OUR NAME
PRT      CALL    $TEST       SEE? WE NEED THIS
                             ROUTINE AGAIN, BUT
*                            DON'T HAVE TO
                             WRITE IT AGAIN. WAIT
*                            TILL TERMINAL IS
                             READY
         MOV     A,M         GET A CHARACTER
                             OUT OF MEMORY, PUT
                             IN
*                            ACCUMULATOR
         OUT     TERM        SEND TO TERMINAL
         INX     H           INCREMENT H AND
                             MAKE HIM POINT TO
                             NEXT
*                            LETTER
         CPI     CR          SEE IF WE ARE AT THE
                             END OF LINE
         JNZ     PRT         NOPE. . NOT DONE. GO
                             GET NEXT CHARAC-
                             TER
         DCR     B           YEP IT WAS! COUNT
                             ONE LINE. DECRE-
                             MENT B
         JNZ     PRINT       B HASN'T COUNT
                             DONE TO ZERO YET,
                             SO
*                            KEEP GOING
         HLT                 OTHERWISE HALT.
                             QUIT. ALL DONE!
*
*One more thing HALT doesn't make it for HASL 8. . Must tell
*him 'END' so. .
*
         END     BEGIN       THE OPERAND BEGIN
                             MAKES THE PC SET UP
*                            TO 040100 SO YOUR
                             PROGRAM IS ALL
*                            READY TO GO WHEN
                             IT IS LOADED.
```

Now for the loose ends

Type in the INIT routine from the source listing.

Remember, we loaded the HL register with a label which defines out $MESAG and $INBUF. So we have to save space for these. Here's how.

```
$INBUF   DS      80          OK. 80 BYTES ARE RE-
                             SERVED FOR YOU TO
                             TYPE IN YOUR NAME.
```

```
DS                           DEFINE STORAGE
                             AREA
```

Same goes for $MESAG. Only in this case, we tell HASL 8 what to put in those locations.

```
$MESAG   DB      LF,CR,      'TYPE YOUR
                             NAME!',LF,CR
```

The same with the stack area. You'll notice we reserved 100 bytes for the stack. Don't need that much and there's a better way, but that'll do for this discussion.

Ok. Now that we have all of this typed, let's see if it will assemble.

As you probably have noticed, we didn't cover everything, including the use of the editing commands. Perhaps we can do that another time. If you are still in the insert mode, type control 'C' and put your program on tape. Here's how.

```
--NEWOUT/TYPE MY NAME TEN TIMES/

--$SAVE
```

'Make sure you have the tape in record!' This writes your program to tape but leaves it in the computer too, just in case the tape messes up.

When the tape stops and you get the prompt type;

```
--STOP OUTPUT
 SURE?Y
```

this puts and end of file marker on the tape

Load HASL 8
Hit 'GO'
And here's what happens

Hit CR if you want the listing to appear on your terminal, otherwise enter the decimal equivalent of your line printer.

Yes we want a binary image.

Yes we want HASL 8 to dump a binary image on tape so have a blank tape in your 'write' deck.

After HASL 8 has dumped the object code (binary image) on your 'write' deck, load your object code by using the front panel 'LOAD' button and see if it will execute.

During the listing, any errors detected will be flagged so you can identify them. To correct, you will have to load TED 8. . . And retrieve the source code for editing.

When you get the prompt type *NEWIN!!*

Don't need to put the title in unless you want to.

- - FILL

Fill buffer with your program.

Proceed with your editing.

```
                      *           THE USART ON THE H8-5 MUST BE PROPERLY INITIALIZED AS OUTLINED
                      *           IN THAT MANUAL. WE CAN NOT EXPLAIN EVERY LITTLE DETAIL IN THIS
                      *           LIMITED SPACE, BUT, HOPEFULLY WE CAN SHOW YOU HOW TO WRITE A SHORT
                      *           PROGRAM AND HOW TO USE TED 8 AND HASL 8. HERE WE GO. ANY LINE,
                      *           SUCH AS THIS ONE, THAT IS PRECEEDED BY AN '*' IS IGNORED BY HASL 8.
                      *           THEREFORE, WE CAN USE THIS METHOD TO DOCUMENT WHAT WE ARE DOING.
                      *           THIS IS STRONGLY RECOMMENDED, SINCE, IF YOU'RE LIKE ME, YOU WONT
                      *           HAVE THE SLIGHTEST IDEA WHY YOU WROTE A PROGRAM THE WAY YOU DID
                      *           A FEW WEEKS FROM NOW.
                      *
                      *
                      *
                      *           HERE IS THE NECCESSARY FORMAT FOR WRITING AN ASSEMBLY LANGUAGE PROGRAM.
                      *
                      *
                      * LABEL   OPCODE    OPERAND    COMMENTS (MY TABS ARE  SET FOR TEN SPACES EACH)
                      *
                      *
040.100                         ORG       040100A    'A' MEANS SPLIT OCTAL..SEE MANUAL.
                      *
                      *         ORG TELLS HASL 8 TO PUT OUR PROGRAM TOGETHER STARTING AT 040100
                      *         YOU MAY 'ORG' YOUR PROGRAM ANYWHERE YOU HAVE MEMORY.
                      *         IN OTHER WORDS, WHEN YOU HAVE THIS PROGRAM WRITTEN AND WANT IT TO
                      *         RESIDE IN HIGHER MEMORY, YOU COULD CHANGE THE VALUE GIVEN TO 'ORG'
                      *         AND REASSEMBLE (USING HASL-8). SIMPLE.
                      *
                      *         NOW WE MUST DEFINE SOME THINGS THAT THE DUMB COMPUTER DOESN'T
                      *         KNOW ABOUT. FOR INSTANCE. ON WHICH PORT IS YOUR TERMINAL? WE
                      *         TELL HIM ABOUT SUCH THINGS WITH  EQUATE STATEMENTS
                      *
                      *
000.372               TERM      EQU       372Q       CONSOLE DATA OUT IS AT PORT 372Q
000.015               CR        EQU       15Q        THAT'S CARRIAGE RETURN AND
000.012               LF        EQU       12Q        THIS MEANS LINE FEED
002.136               $HORN     EQU       002136A    MIGHT  WANNA BLOW THE HORN FOR SOME REASON
                      *                              SEE PAM 8 LISTING.
                      *
040.100   061 204 041 BEGIN     LXI       SP,STACK   DEFINE STACK AREA
                      *         SINCE WE WILL BE 'CALL'ING SUBROUTINES, WE MUST RESERVE AN AREA
                      *         FOR THE CPU TO STORE 'RET'URN ADDRESSES.
040.103   315 235 040           CALL      INIT       GET USART ALL SQUARED AWAY
                      *
                      *          $TYPLN IS CALLED TO TYPE ONE LINE OF TEXT ON THE
                      *           CONSOLE DEVICE
                      *
                      * $TYPLN IS A LABEL WHICH IS SYMBOLICALLY UNIQUE.. THERE CAN ONLY BE
                      *         ONE ROUTINE CALLED $TYPLN... $MESAG WILL BE DEFINED LATER BY HASL-8
                      *         SINCE WE DON'T KNOW HOW LONG OUR PROGRAM IS GOING TO BE AND THEREFORE
                      *         CANNOT GIVE THE (HL) REGISTER PAIR AN ABSOLUTE ADDRESS FOR $MESAG.
                      *
                      *
040.106   041 355 040 $TYPLN    LXI       H,$MESAG   SET HL TO BEGINNING OF MESSAGE
                      *
                      *
040.111   315 226 040 $GETCHR   CALL      $TEST
040.114   176                   MOV       A,M        FLOP CHARACTER IN THE ACCULUMATOR
040.115   346 177               ANI       177Q       DON'T NEED PARITY BIT..STRIP IT
040.117   376 000               CP1       0          SEE IF END OF MESSAGE
040.121   312 133 040           JZ        $RCHAR
040.124   315 262 040           CALL      $CDOUT     GO PRINT IT
040.127   043                   INX       H          BUMP POINTER
040.130   303 111 040           JMP       $GETCHR    GO DO ANOTHER
                      *
                      *
                      *
                      *
                      *
                      *
040.133   041 205 041 $RCHAR    LXI       H,$INBUF   SET (HL) TO START OF BUFFER WHERE
                      *                              WE WILL STORE THE ENTIRE LINE WHICH
                      *                              IS TYPED ON THE KEY BOARD.

040.136   315 216 040 $RCHAR1   CALL      $WAIT      DO WE HAVE A CHARACTER YET?
                      *
040.141   315 265 040           CALL      $CDIN      NOW WE DO
040.144   346 177               ANI       177Q       STRIP IT
040.146   167                   MOV       M,A        STORE IT AWAY
040.147   043                   INX       H          BUMP THE POINTER
040.150   376 015               CPI       CR         SEE IF END OF MESSAGE
040.152   315 262 040           CALL      $CDOUT     ECHO THE CHARACTER
040.155   302 136 040           JNZ       $RCHAR1    GO GET ANOTHER CHARACTER
040.160   066 012               MVI       M,LF       GIVE HIM A LINE FEED
                      *
                      *
                      *         $PRINT - PRINT MY NAME TEN TIMES
                      *
                      *
                      *
```

```
040.162  006 012                 MVI     B,10       SET UP COUNTER
040.164  041 205 041 $WAIT.      LXI     H,$INBUF   RETURN THE POINTER TO BEGINNING OF MESSAGE
040.167  315 226 040 PRT         CALL    $TEST
040.172  176                     MOV     A,M        GET CHARACTER FROM BUFFER
040.173  315 262 040             CALL    $CDOUT     AND PRINT IT
040.176  043                     INX     H          BUMP POINTER
040.177  376 012                 CPI     LF         SEE IF END OF LINE
040.201  302 167 040.            JNZ     PRT
040.204  005                     DCR     B          YEP.. END OF LINE, COUNT IT
040.205  302 164 040             JNZ     $WAIT.
040.210  041 276 040             LXI     H,$MESAG2 LET'S DO IT AGAIN!
040.213  303 111 040             JMP     $GETCHR
                     *
                     *
                     *
                     *   $WAIT - $WAIT IS CALLED TO WAIT ON A KEYSTROKE ON THE TERMINAL
                     *
                     *
                     *
040.216  333 373     $WAIT       IN      TERM+1
040.220  346 002                 ANI     002Q       ISOLATE READY BIT
040.222  312 216 040             JZ      $WAIT      FOOL AROUND
040.225  311                     RET
                     *
                     *
                     *
                     *
                     *
                     * $TEST -$TEST IS CALLED TO TEST THE TERMINAL BUSY BIT
                     *
                     *
                     *
040.226  333 373     $TEST       IN      TERM+1
040.230  037                     RAR                STICK BIT 0 IN THE CARRY FLAG SO WE CAN TEST IT
040.231  322 226 040             JNC     $TEST      CARRY BIT NOT SET.. KEEP TESTING
040.234  311                     RET                OK, HE'S READY
                     *
                     *
                     *
                     *
                     * INIT      INIT IS CALLED TO FIX UP USART
040.235  076 201     INIT        MVI     A,201Q     MAKE SURE THIS GUY'S READY!
040.237  315 273 040             CALL    $CSOUT
040.242  076 100                 MVI     A,100Q     INTENAL RESET
040.244  315 273 040             CALL    $CSOUT
040.247  076 116                 MVI     A,116Q     TELL HIM ABOUT SUCH THINGS AS 8 BIT CHARACTERS
040.251  315 273 040             CALL    $CSOUT
040.254  076 005                 MVI     A,005Q     ENABLE RCVR AND TXMTR
040.256  315 273 040             CALL    $CSOUT
040.261  311                     RET
040.262  323 372     $CDOUT      OUT     TERM
040.264  311                     RET
040.265  333 372     $CDIN       IN      TERM
040.267  311                     RET
040.270  333 373     $CSIN       IN      TERM+1
040.272  311                     RET
040.273  323 373     $CSOUT      OUT     TERM+1
040.275  311                     RET
040.276  015 012 012 $MESAG2     DB      CR,LF,LF,' CONGRATULATIONS!..TYPE SOMETHING ELSE!  ',LF,CR,0
040.355  015 012 012 $MESAG      DB      CR,LF,LF,LF,'OK... I''M WAITING FOR YOU TO TYPE YOUR NAME!',LF,CR,0
041.040                          DS      100        RESERVE ROOM FOR THE STACK
041.204              STACK       DS      001        STACK STARTS HERE
041.205              $INBUF      DS      80
041.325  000                     END     BEGIN

00166 Statements Assembled
20194 Bytes Free
 No Errors Detected
```

EOF

Heres a one liner that will return the date in HDOS BASIC.

```
10 D$ ="": FOR I=8383 TO 8391:D$ =D$+CHR$(PEEK(I)):NEXT :PRINT D$
```

BITS AND NIBBLES

MULTITASK SOFTWARE WEATHER STATION/H8 SYSTEM

Brousing through the latest Heath catalog, I see the new super nifty (ID-4001) computerized weather station advertised. We won't elaborate it's features here, but . . . you can hook it up to your H8 (H8-2) and have the computer constantly monitor and print out the weather information in several formats even while you are using the H8 for other things! Neat!? Currently the software is available only through HUG and is cassette based. We plan on having it for the H8 disk system very soon and we will let you know as soon as it is ready . . . order the cassette software and documentation on the green order form as P/N 885-1017 . . . include a check or money order for $5.50. Note: this software is written around the latest cassette distribution tape released this month. It will not work with any of the earlier versions since the console driver routines were changed to accommodate the new 4 port serial board which is being announced. *:JB:*

NO MORE BACK ISSUES OF REMARK

Sorry, our stock of issues #1 and #2 is depleted. A limited quantity of #3 still exists, however.

MORE SOFTWARE

If you thought HUG software tape I and Volume I is super, wait till you get a peek at Volume II. The quality and use is greatly improved and you will receive an admended catalog soon. Keep up the great work. Now that we're in off the beaches (at least up here), we expect some really super programs coming in. Also, with the availability of the expansion accessory for the ET-3400 and the floppy system for the H11, we hope to start seeing some work from these users. As a matter of fact, we will have a super contest next issue just for the H11 and ET-3400 users. Neat prizes too!

MEETING

May I take a few minutes of your time to have a quick 'one-on-one' meeting? Thank you . . . The subject of our meeting is to discuss procedures for submitting software.

The first software catalog and the documentation of the software in Volume I and Tape I could be a lot better. We promise Volume II and Tape II to be much better . . . with your help.

If memory allows, document your program as much as possible so that anyone can look at it and determine what you had in mind when you wrote it. This includes identifying subroutines, possibly operating instructions that appear when the program is first run and is self deleting. It is also nice if you identify variables. Example; REM A$ = ADDRESS . . . N$ = NAME ETC. anything you can do to explain your program which will save us phone calls and possibly yourself phone calls and correspondence.

Also, please limit BASIC program lines to 65 characters. This will improve the appearance of your program when published in a future volume. If your documentation is type-written, we will most likely reproduce it directly.

Now, on cassettes . . . unless you specifically request it, we will not return your cassette. However, if you ask, we will return your cassette as soon as the master is made for distribution. If your program is not included in the library we will return all materials.

Authors of programs accepted for inclusion in a future release of HUG Software will automatically receive a free copy of the next release as their renumeration. In addition, the authors' membership will be automatically extended one year upon the prevailing expiration date.

On disk based software: yes, we are accepting software on disk. And we recognize that they are expensive. Therefore, as soon as your program is accepted, (usually the same day it is received), we will return your diskette.

End of meeting . . . Thanks for your time and cooperation.

HDOS PROGRAMMERS GUIDE

For the HDOS user that considers himself a sophisticated programmer, we have a very informal paper prepared by the author of HDOS that will help you communicate with the disk in assembly language programs. This document is not written in a tutorial manner, and it uses sophisticated terms without explan ation. Heath technical support will be minimal. You're on your own. Order HUG P/N 885-1018. It costs $5. Use the green order form please.

Also, many of you HDOS users have asked for more device drivers . . . we have them. The driver comes on diskette in source with instructions on how to adapt it to your particular needs. Order HUG P/N 885-1019. Cost is $10.

CONTEST #4

One super prize for the 'best' program submitted before Jan. 15, 1979! Santa presents an ID-4001 Computerized Weather Station to the person that submits the winning program regardless of which Heath machine it is written for. . See the latest Heath catalog for a description of the Weather Station. . . Please mark all materials with 'Contest #4'. Entries must be postmarked by Jan. 15, 1979. Submit tape or disc if applicable. Same rules as in the past apply. Only one prize will be given. Decision of the judges final.

. . . MORE IDEAS

BASE-B TO DECIMAL CONVERSION

Purpose: Convert a positive integer of arbitrary base to a decimal integer.

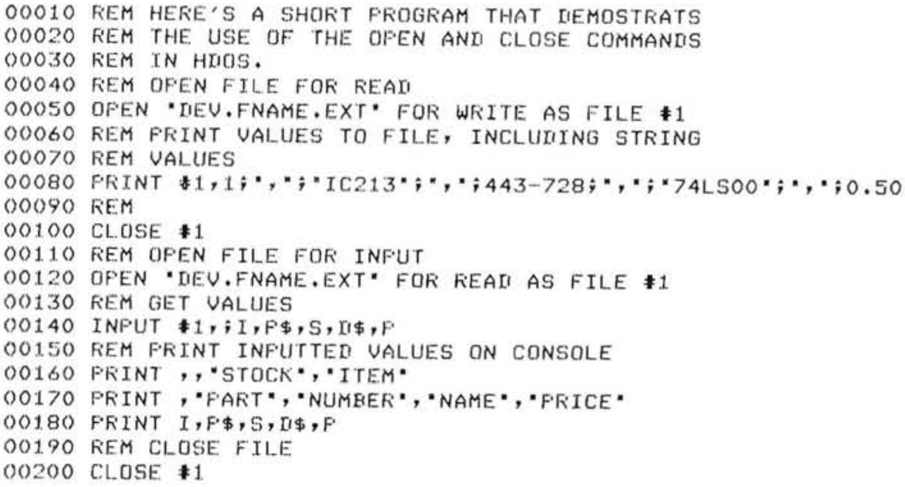
Limitations: Allowed Bases: 2-36
Restricted to positive integer arguments

Variables:
ENTRY B = base of entry integer
B$ = Entry integer
– each digit must be in the set 0-9, A-Z
– no leading/trailing blanks
EXIT D = result
-or-
= -1 (if B$ in error for base B)
TEMP B9, P9, P (no external significance)

Calling Sequence:
B$ = (entry integer)
B = (base of B$)
GOSUB 900
PRINT D;" = ";B$

Listing:

```
900 REM – BASE-B / DECIMAL CONVERSION
905 D = 0 : P9 = LEN(B$) : IF P9<1 THEN D =−1: RETURN
910 FOR P = 1 to P9
915 B9 = ASC(MID$)(B$,P,1))−48
920 IF B9<0 or (B9>9 and B9<17) THEN D =−1 : RETURN
925 IF B9>19 THEN B9 = B9−7
930 IF B9>B THEN D =−1 : RETURN
935 D =B*D + B9
940 NEXT P
945RETURN
```



```
00010 REM HERE'S A SHORT PROGRAM THAT DEMOSTRATS
00020 REM THE USE OF THE OPEN AND CLOSE COMMANDS
00030 REM IN HDOS.
00040 REM OPEN FILE FOR READ
00050 OPEN "DEV.FNAME.EXT" FOR WRITE AS FILE #1
00060 REM PRINT VALUES TO FILE, INCLUDING STRING
00070 REM VALUES
00080 PRINT #1,1;",";"IC213";",";443-728;",";"74LS00";",";0.50
00090 REM
00100 CLOSE #1
00110 REM OPEN FILE FOR INPUT
00120 OPEN "DEV.FNAME.EXT" FOR READ AS FILE #1
00130 REM GET VALUES
00140 INPUT #1,;I,P$,S,D$,P
00150 REM PRINT INPUTTED VALUES ON CONSOLE
00160 PRINT ,,"STOCK","ITEM"
00170 PRINT ,"PART","NUMBER","NAME","PRICE"
00180 PRINT I,P$,S,D$,P
00190 REM CLOSE FILE
00200 CLOSE #1
```

EOF

H11 BASIC PATCHES

By: Bob Meister
59 Glade St. Apt. C-4
West Haven CT 06516

Dear Hug,

I am quite surprised at the lack of articles or programs for the H11 computer. As an owner of one, there have been a few problems which I have overcome and believe that other owners might also want to fix. I don't know if it is because I am a super programmer, or I am at the bottom of the experience list, but surely somebody out there has noticed some of these 'BUGS'.

I program D.E.C. PDP-8 computers for a living, so maybe this puts me out of the beginner class. Most of my work is in machine and assembly language, but occasionally, BASIC and FOCAL programs find their way into the computer. All of the 'BUGS' I attempted to fix were in the Heath version of BASIC 11. Out of necessity, and curiousity, I designed a disassembler in machine language to assist in the repair of the BASIC 'BUGS'. Here is a list of the problems followed by an assembly listing of their corrections.

PROCEED AT YOUR OWN RISK. :JB:

1. If, for some reason, you hit the 'BREAK' key on the terminal, or lower the 'HALT' switch on the CPU, and want to re-start BASIC, the only way the manual says to do it is by typing '0G'. This starts the program by jumping to the address at locations 000000 and 000002. This is known as a 'COLD START' since the entire text area is cleared and the BASIC intrepreter does an initial start. This is all well and good if you don't care what was in the machine anyway. I noticed that my friends' versions of BASIC for a 6800 CPU all had 'WARM START' locations which re-started BASIC without destroying the program area.

1. Solution: Change the address jumped to at locations 000000 and 000002 to a 'WARM START' location. If you really want a 'COLD START', you can always type 'SCR' to BASIC, clearing the user area. This location is the same place that program execution resumes after encountering a 'STOP' command.

2. If, for some reason known only to the ancient Greek Gods, the program should 'TRAP' through location 000004, there is no way of knowing just what happened, nor how to recover from it. Before problem 1 was solved, I found out that there were two definite conditions which would cause a 'TRAP' to location 000004: 'SAVE' to devise #2, but not having an interface respond to address 177514 and 177516 (line printer address), and doing some kinds of string manipulations.

2. Solution: I found in the reference manual of BASIC, and in my disassembly of BASIC, an error message, '% NEM'. This message indicates non-existent memory. Perfect, since this condition is usually the one that causes a 'TRAP' to location 000004. All that was necessary was to change the vector at this 'TRAP' to point to the '% NEM' error routine. Now, if for any reason, the BASIC program should get lost, it prints the error message and jumps back to the 'READY' mode through the 'WARM START' routine.

3. While not really a problem, I noticed that if there were many 'STOP' commands in BASIC, I had no idea which one caused the interpreter to stop executing commands. If there was only one 'STOP' instruction, then there was no problem. I noticed that any error message was followed by the line number where it occurred, if BASIC was 'RUNNING' a program.

3. Solution: Again, with the help of my disassembler, I found the routine which printed the 'STOP' message. This was moved to an empty area of memory, and with a little additional programming, voila . . . when the program encounters a 'STOP' command, it is followed by the line number that contained the 'STOP' instruction.

These additions have made BASIC a lot more fool-proof, at least the interpreter doesn't 'CRASH' if you look at it wrong. The changes were assembled using PAL-11S. After BASIC is loaded, and the initial dialog is answered, I stop the computer to insert the changes. BASIC erases memory (in an 8K system anyway) such that I must re-boot the absolute loader. No real problem. Once the absolute loader is in the computer, I can load in the patch tape. It is self starting at the 'COLD START' location, thereby clearing memory again and also resetting the stack pointers. There is no reason why I couldn't have used 'ODT' from the console and changed BASIC manually. One thing that can't be done is 'POKE' all of the changes. BASIC, when executing a 'PEEK' or 'POKE' command, changes a location (000004) so, if you try to access memory that isn't available, BASIC tells you about it. After completing the operation, that location is restored to its former contents. Try this operation . . . print peek (4) . . . and then examine location 000004 with ODT. The two will never be the same. This was also verified by a quick look at the disassembly of BASIC.

A problem with strings caused me to determine solution #2. Apparently, when 'RUN' is typed, BASIC scans most of the text and sets up variables in memory . . . except for strings. If you want a segment of a string, then that string must be stored in memory before trying to use it. For example:

```
20 PRINT SEG$(A$,2,4)
30 A$ = "ABCDEFG"
```

Will 'CRASH' by vectoring to non-existent memory. Apparently the string area of BASIC is in the highest area of memory, which, in an 8K system, is address 37776. Since the string hasn't been defined, the 'SEG' command tries to get characters after address 40000, which is a no-no. The proper, safe way is to define A$ before using it like this:

```
10 A$ = "ABCDEFG"
20 PRINT SEG$(A$,2,4)
```

Which prints . . . BCD

I have encountered no problems with the above fixes to BASIC. However, my system may not be representative of anyone else's. I am running an 8K system, teletype as console terminal, A high speed paper tape reader, and a 2400 baud CRT terminal acting as a line printer. If anyone has any problems or additional improvements to BASIC, I would definitely like to hear of them.

By the way, there appear to be no flaws in either version of FOCAL-11, other than the fact that they are incredibly SLOW.

See listing on Page 31. :JB:

EOF

Dear Hug,

I received HUG 3 yesterday and was very pleased. Bill Nico's CAI program was great. Your HDOS PRIMER was especially exciting since I have recently ordered an H17.

The most useful item for me was the PUT/GET article by Bob Behar and Neal Rogers. I installed the GET command in a couple of my own programs and it worked fine. It should be noted that a typical calling sequence for these routines would be:

```
100 Z$ = "DATA FILE : GOSUB 6500 :
STOP
```

The STOP is necessary to return control to the command mode. It must be outside of the called routine because GETing apparently wipes out the subroutine stack.

Dennis Smith's REFLEC TEST is a good demo-program, but it was a little hard to read. I haven't tried Bob's MAIL LIST PROGRAM; I'll wait until I have hard copy.

Here are a couple of ideas for the future:

1. An article listing and detailing the use of the many useful memory locations and subroutines available in PAM-8 and CONSOLE-DRIVER.

2. A quick reference guide to Benton Harbor Software. It could be produced on a plastic page like the one included with the H-8 and sold as a HUG accessory.

Sam Cox
2412 Zenith Ct
Ft. Collins CO 80526

MORE ARTIFICIAL INTELLIGENCE. . .

OR IS IT?

```
There is a little axe here.
Get axe

You can't carry anything more. You'll have to drop something first.
Inv

What?
Invent

You are currently holding the following:
Set of keys
Brass lantern
Wicker cage
Black rod
Tasty food
Small bottle
Water in the bottle
Large gold nugget
Drop cage

OK
S

Get axe

I don't see any axe.
N

There is a threatening little dwarf in the room with you!

You're in hall of mists.

There is a small wicker cage discarded nearby.
There is a little axe here.
Get axe

OK
Throw axe

You attack a little dwarf, but he dodges out of the way.

You're on East bank of fissure.

Jump fissure

You didn't make it.

Oh dear, you seem to have gotten yourself killed. I might be able to help you out, but I've never really done this before. Do you want me to try to reincarnate you?
Y

All right. But don't blame me if something goes wrong. . .

--Poof!!--
You are engulfed in a cloud of orange smoke. Coughing and gasping, you emerge from the smoke and find . . .

You're inside the building.
Score

If you were to quit now, you would score 34 out of a possible 350.

Do you indeed wish to quit now?
Y

OK

You scored 34 out of a possible 350, using 157 turns.

You are obviously a rank amateur. Better luck next time.
```

EOF

!ERROR — SYNTAX

This info 'fell off' page 25 of issue #3 'Interfacing a Selecterm'

THE WIRE WHICH PRESENTLY IS ON OUR DB25P, PIN No.	AND CARRIES THE SIGNAL OF	SHOULD BE MOVED TO THE HEATH CONNECTOR PIN No.
1	D0	1
2	D1	2
3	D2	3
4	D3	4
5	D4	5
6	D5	6
7	D6	7
13	GROUND	9
10	READY	10
9	STROBE	11

We strongly suggest that you move one pin at a time to reduce the chance of error. This concludes the modification for the Heath H8 computer.

W. Louis Waggoner
Micro Computer Devices

!ERROR — SYNTAX

Page 29 of #3
Line 6290 should read:

```
6290 IF J>K then 6120
5050 IF I<100 then 5030
```

!ERROR — SYNTAX

Page 31 IC219-11

! ERROR — SYNTAX

Page 3-24 and 3-25 of the H8 Software Reference Manual.

Steps 3 and 5 are backwards. . .

Step 3 should read:

'Enter 116 377. . .'

and step 5 should read:

'Enter 005 377. . .'

--EDIT

Thought maybe some HUG members might be interested in the enclosed circuit which I have developed and installed in my H9 terminal. It allows me to do a screen erase under program control by sending a CNTRL E character to the terminal. When printing a lot of information, it is much easier to read by filling the screen, inserting a PAUSE, and then clearing the screen and start the new information at the top. Much easier to follow than scrolling.

The circuit is easy to build, costs under $5 if you have to buy all of the parts new (depends on your supply source), requires only 6 connections to the terminal, and requires no modifications of the existing boards. My circuit is assembled on a small piece of perf board and is installed under the righthand side of the keyboard, near the ERASE PAGE and PLOT keys.

I recently attended the opening of a new computer store in my area and had the chance to try other computers. Though I have many times sworn at Heath's command completion, I found myself missing it and making many more mistakes than I do on my own system! However, I did like the feel of the Hazeltine terminal keyboard a lot more than the H9. There's a big difference in price too. I guess the world is filled with compromises.

Reprinted from Kilobaud with their permission.

William C. Richter

You said you might be interested in how I hooked up my Centronics 101A printer.

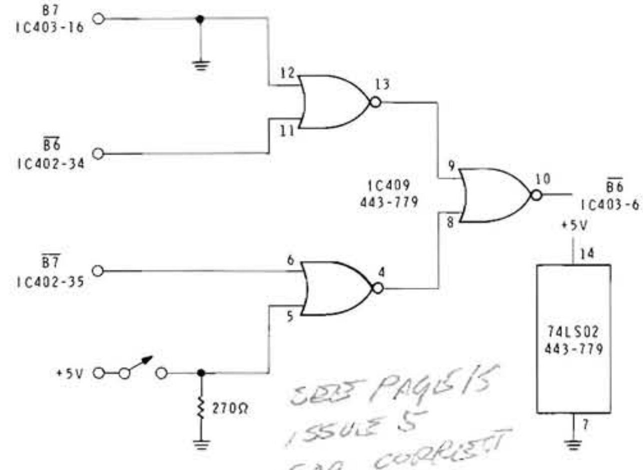
I hope this will help:

1. Wire the H8-2 parallel board per the instructions
2. Test all 3 channels shown on page 37.
3. Select a channel and move jumper from C2 to C3, remove A1 - A2 and B1 – B2 jumpers.
4. Cut trace between ICX05D-13 and ICX08A-3.
5. Add a jumper between ICX05C-10 and the low side of RX06 (1000 ohm) resistor.
6. Wire the 25 hole connector shell using the 9 feet of the 25 conductor cable as shown on pages 31 and 32.
7. Connect the Centronics 101A as follows:

	H8-2	101A
PIN# 1	WHT/BLK	DATA BIT 1
2	WHT/BRN	DATA BIT 2
3	WHT/RED	DATA BIT 3
4	WHT/ORG	DATA BIT 4
5	WHT/ YEL	DATA BIT 5
6	WHT/GRN	DATA BIT 6
7	WHT/BLU	DATA BIT 7
8	WHT/VIOL	DATA BIT 8
9	WHT/GRY	/-0
11	RED/YEL	DATA STROBE
16	WHT	ACKNOWLEDGE
17	GRY	+/-0

8. Tape back the 13 unused wires (make sure no wires touch).

Hank Derkinderen



SEE PAGE 15
ISSUE 5
FOR CORRECT
SCHEMATIC

If you are using a second H11-5 serial board to drive a remote printer device and are experiencing difficulty with the print #2 command typing the program in BASIC, try the following procedure:

1) Load BASIC
2) Set pad characters
3) Place RUN/HALT switch in HALT position
4) Type the following

```
200/xxxxxx LF
202/xxxxxx LF
204/xxxxxx LF   type data in from
                location 200
206/xxxxxx CR   type data in from
                location 202
```

5) Type P

You have now reentered BASIC.

This patch will make up for hardware differences between the H11-5 and DEC printer interface board.

Jim Moore
Factory Service

DOUBLE-SIDED DISK

Square 1 now makes a kit available that lets the user make "flippy" diskettes out of his "floppy" diskettes. Most diskette manufacturers coat and finish both sides of the diskettes, but package them in such a way that they are only usable on one side. With ordinary care, the user can modify the jacket of the diskette so the spare side can be used. Called the FLIPPY-DISK-KIT, it contains all the necessary tools to locate and accurately punch the extra holes in the jacket of the diskette. Instructions guide the user through the "anatomy" of a diskette explaining clearly the function of each hole and opening in the jacket, then the method of marking and punching the holes and testing the newly available side. Square 1 claims over 85 percent of the 5-1/4 inch diskettes can be successfully made usable on the "flip " side. The kit is designed to be used with any 5-1/4 inch hard sectored mini-diskette drive. Once the user buys the kit, he then gets the use of both sides of every diskette he buys, thus in effect, getting a 50 percent discount on his disk purchases. The kit contains instructions, double sided "flippy-plate", a unique pencil for making highly visible marks on the black diskette jacket, and a specially ground and polished hand punch for making the holes. Priced at $9.95 plus $1.00 shipping, the kit is available from: Square 1, 614 Eighteenth Avenue, Menlo Park, California 94025.

Perform 'MEDIA' check on flipside. :JB:

RTTY to H8?

*by: Robert Traub
Canada*

The word seems to be out that soon the U.S. Hams will have the use of ASCII code on RTTY. If this is true, then perhaps this article will be of interest to those who are on RTTY and also have the H8 H9 system.

At the present time, this system is being used by VE6OJ (me). It was built up in order to try and solve some of the interface problems that would be encountered. The circuits are used in operation with the DT-600 and XK-2 A.F.S.K. units, although they should apply to most if not all units.

The first problem was that of setting data into the H8. I will stop here a minute to explain that I did not want to bother with the computer at this stage, as the programs and syntax would be in the way. I used the H9 as a stand alone terminal, as this met all the interface requirements of the H8 and save direct results. Therefore, in order to get data into the H8(H9), I would require a RS-232 signal. As it works out the DT-600 uses both +12 and −12 volt supplies and the XK-2 uses a +5 volt supply. They are mounted in the same cabinet, so all the voltages required can be found in one cabinet. Figure 1 shows the schematic of a TTL to RS-232 interface card used for the conversion. The TTL level is taken from the 'DATA' output transistor on the DT-600 (Q1) called the Keyer. At this point, some may ask why I did not take the output from the slicer op-amp output; this was simply because it is a mark positive and RS-232 is a mark negative. The output from this transistor will be at logic level '0' when a mark signal is present; therefore it will have to be inverted; thus the inverter shown in the schematic. This is used to drive the 741 op-amp and produce the correct mark negative −10 volts and space positive +10 volts, which fully meet the RS-232 requirements. The output from this circuit is hooked to the input of the H8 (H9) and copy from off the air signals will result.

The second part of the interface is to transmit data to the AFSK unit from the H8(H9). Now as it works out, RS-232 is tailor-made to key the input transistor of the AFSK unit. Therefore, all that is required is to simply hook the output of the H8(H9) to the input of the keyer. The normal negative voltage from the RS-232 of the computer will hold the keyer on steady mark. Then as data is typed on the keyboard, the RS-232 will swing positive with the signal. This will cause the AFSK unit to follow suit and shift the tones to space as required by the data.

Now that you have that set up for ASCII, you will see that it is also an interface to the Baudot equipment. About all you need there is the software and a Baud rate clock, and the H8-H9 can be used with the standard Baudot equipment. This, however, does not interface the Baudot equipment for local copy or local loop.

This system has been used with great success and has resulted in better than expected reception of data. No modification was required to the DT-600 filters or the AFSK unit. The baud rate that this unit has been run at is 110 baud, and tests are now underway at 300 baud, just right for my LA-36 DECwriter. If you are using the H9 terminal only, all you need do to obtain the 110 baud is to have the baud rate key in the 'UP' position. For the H8 you would have to strap the serial I/O board for 110 baud.

EOF

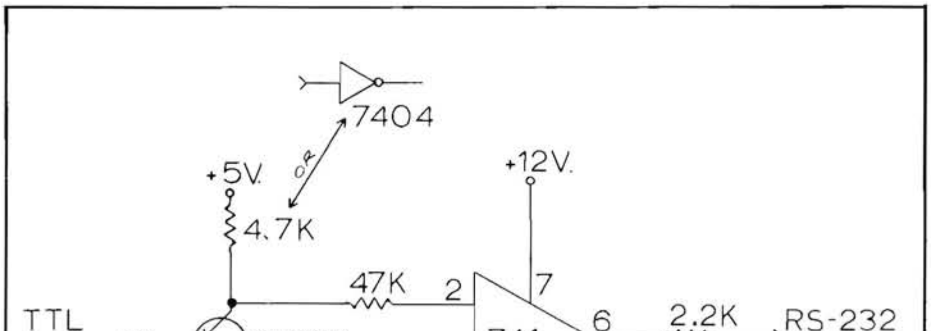


FIG. 1
TTL TO RS-232

MEETINGS and CLUB NOTICES

I would like to communicate with other HUG members in Germany. I am presently in the U.S. Army and would like to see if we could get a computer club started or is there already one in existance over here? Surely I cannot be the only individual in Germany with an H8 Heath computer!?

Robert E. Mimms
HHB 2/81st F.A.
APO New York NY 09322

South New Hampshire
Local HUG forming
Contact:
Perry Miller
1 Milhouse Rd A9
Milford N.H. 03053
603-673-8639

Beverly, MA
Contact:
Norman Hill
580 B. Cabot St.
Beverly MA 01915

Or contact the Heath Electronic Center in Peabody MA.

Baltimore – Annapolis Area
Contact:
Albert Richburg
PO Box 768
Severna Park MD 21146
765-3803 or 647-6471

Spokane, Seattle WA
Inland Empire Computer Club
Contact:
Charles Ballinger
E. 403 Dalke #2
Spokane WA 99207

Milwaukee WI
Contact:
Marvin N Lake Dr.
Milwaukee WI 53217

Redwood City, CA
BIG Club had their first meeting October 4
See Bob Bance at Heath Electronic Center
415-365-3157

Simsbury, CT

Our computer club is open to anyone interested in computers and our regular meetings are on the second Thursday of every month. Some of our meetings are held at various computer installations such as Connecticut General Life Insurance Company, June 8; Wethersfield Computer Center (Town Hall), July 13; Talcott Mountain Science Center, August 10; and other locations that have some impact on Electronic Data Processing.

For further information contact:
Harald Bender
The Computer Club
6 Maureen Drive
Simsbury CT. 06070.

Detroit, MI

Heathkit Computer owners in the Detroit area. Find out what other ET-3400, H8 and H11 owners are doing with their equipment.

Join the Detroit area Heath Users' Group and find out.

For information, call:
Jim Hauser
1-313-774-0098 Between 8am and 2pm

Honolulu, HI
HUGH – Heath Users' Group Hawaii
A club within the Aloha Computer Club
Gerry Cramm
2545A Lawrence Pl.
Kailua, HI 96734
phone 254-2319

Meeting each first Wednesday of the month at Lee Ward Community College in Honolulu at 1830 hours.

Tampa Bay, FL
Local club meets every 1st and 3rd Mondays
contact:
Heath Electronic Center
813-886-2541

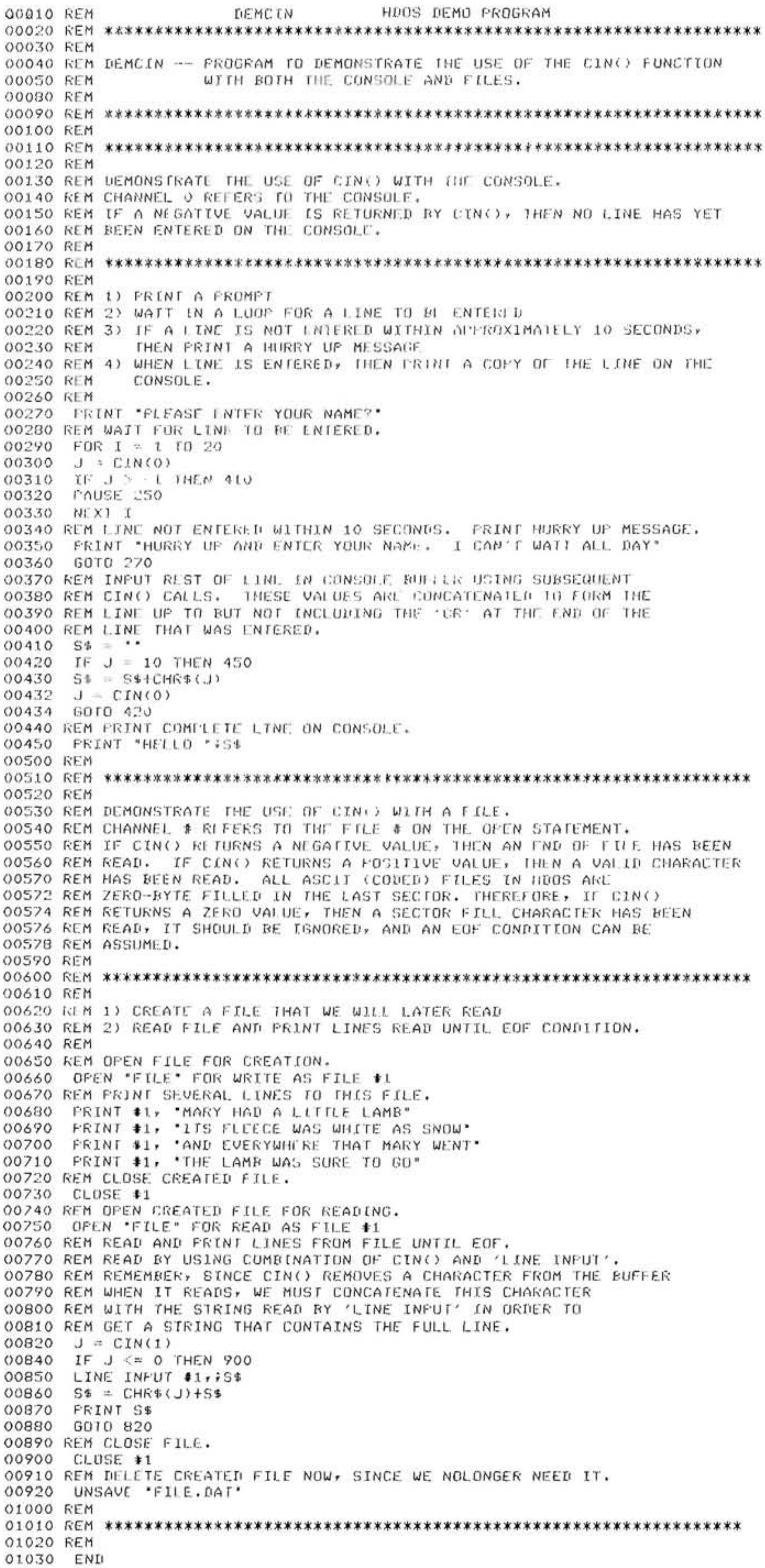
Tidewater, VA Area

If you are interested in joining a local H8 User's Group, please call me or send me your name, address and phone number. My idea is to have meetings about once a month. There are about fifty of us in the area. Let's start swapping ideas.

Jim Egerton
1049 Patrick Henry Way
Virginia Beach VA 23455
(804) 464-9487

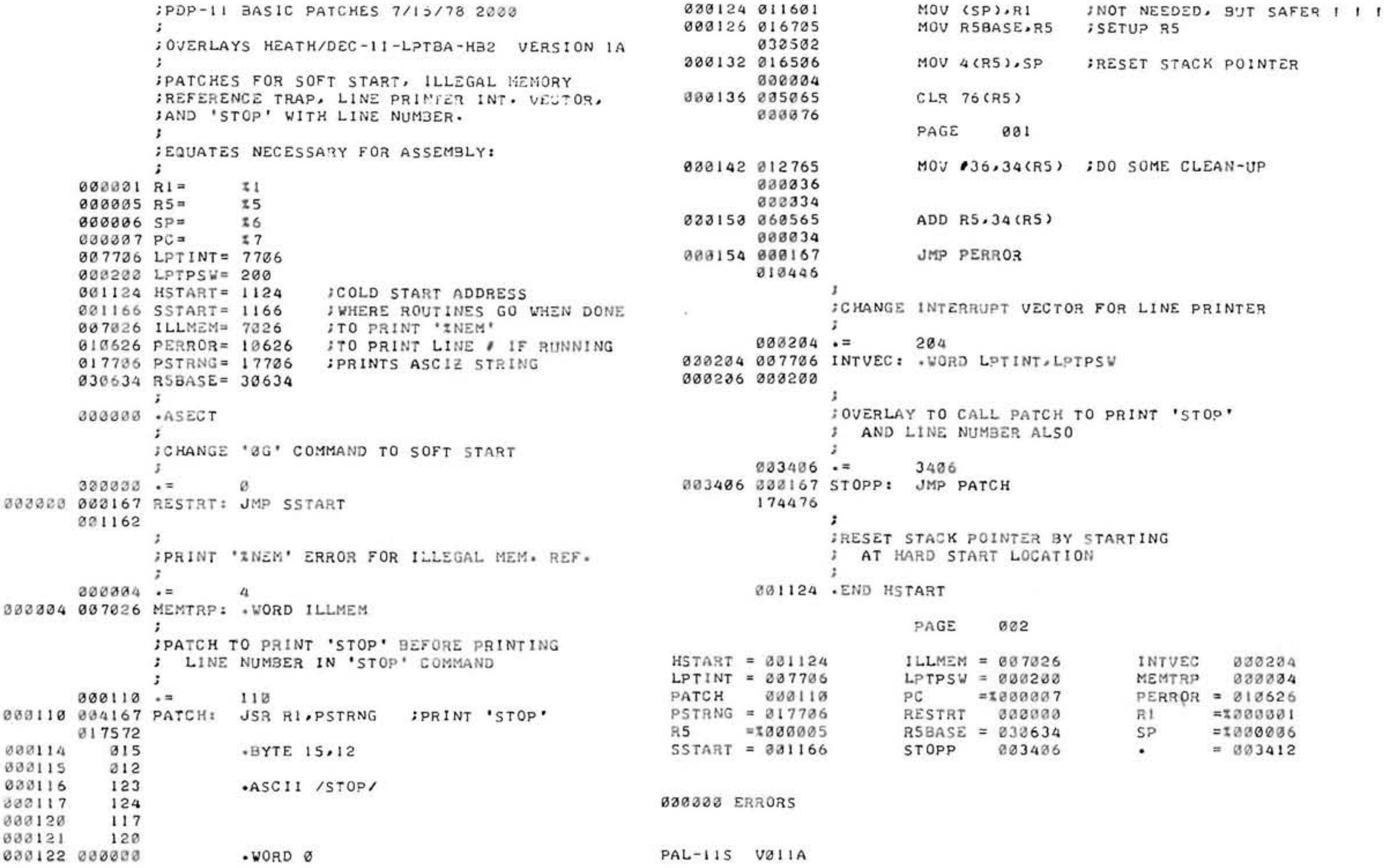
MEET HUM — H8 USERS — MONTREAL

Left to right: Kenneth Papineau, Secretary; Pierre Limoges, Vice-President; Bernard Tremblay, President; and George Girard, Manager, Heathkit Center in Montreal.



```
00010 REM                  DEMCIN          HDOS DEMO PROGRAM
00020 REM *******************************************************************
00030 REM
00040 REM DEMCIN -- PROGRAM TO DEMONSTRATE THE USE OF THE CIN() FUNCTION
00050 REM            WITH BOTH THE CONSOLE AND FILES.
00080 REM
00090 REM *******************************************************************
00100 REM
00110 REM *******************************************************************
00120 REM
00130 REM DEMONSTRATE THE USE OF CIN() WITH THE CONSOLE.
00140 REM CHANNEL 0 REFERS TO THE CONSOLE.
00150 REM IF A NEGATIVE VALUE IS RETURNED BY CIN(), THEN NO LINE HAS YET
00160 REM BEEN ENTERED ON THE CONSOLE.
00170 REM
00180 REM *******************************************************************
00190 REM
00200 REM 1) PRINT A PROMPT
00210 REM 2) WAIT IN A LOOP FOR A LINE TO BE ENTERED
00220 REM 3) IF A LINE IS NOT ENTERED WITHIN APPROXIMATELY 10 SECONDS,
00230 REM     THEN PRINT A HURRY UP MESSAGE
00240 REM 4) WHEN LINE IS ENTERED, THEN PRINT A COPY OF THE LINE ON THE
00250 REM     CONSOLE.
00260 REM
00270  PRINT "PLEASE ENTER YOUR NAME?"
00280 REM WAIT FOR LINE TO BE ENTERED.
00290  FOR I = 1 TO 20
00300  J = CIN(0)
00310  IF J > -1 THEN 410
00320  PAUSE 250
00330  NEXT I
00340 REM LINE NOT ENTERED WITHIN 10 SECONDS.  PRINT HURRY UP MESSAGE.
00350  PRINT "HURRY UP AND ENTER YOUR NAME.  I CAN'T WAIT ALL DAY"
00360  GOTO 270
00370 REM INPUT REST OF LINE IN CONSOLE BUFFER USING SUBSEQUENT
00380 REM CIN() CALLS.  THESE VALUES ARE CONCATENATED TO FORM THE
00390 REM LINE UP TO BUT NOT INCLUDING THE 'CR' AT THE END OF THE
00400 REM LINE THAT WAS ENTERED.
00410  S$ = ""
00420  IF J = 10 THEN 450
00430  S$ = S$+CHR$(J)
00432  J = CIN(0)
00434  GOTO 420
00440 REM PRINT COMPLETE LINE ON CONSOLE.
00450  PRINT "HELLO ";S$
00500 REM
00510 REM ******************************************************************
00520 REM
00530 REM DEMONSTRATE THE USE OF CIN() WITH A FILE.
00540 REM CHANNEL # REFERS TO THE FILE # ON THE OPEN STATEMENT.
00550 REM IF CIN() RETURNS A NEGATIVE VALUE, THEN AN END OF FILE HAS BEEN
00560 REM READ.  IF CIN() RETURNS A POSITIVE VALUE, THEN A VALID CHARACTER
00570 REM HAS BEEN READ.  ALL ASCII (CODED) FILES IN HDOS ARE
00572 REM ZERO-BYTE FILLED IN THE LAST SECTOR. THEREFORE, IF CIN()
00574 REM RETURNS A ZERO VALUE, THEN A SECTOR FILL CHARACTER HAS BEEN
00576 REM READ, IT SHOULD BE IGNORED, AND AN EOF CONDITION CAN BE
00578 REM ASSUMED.
00590 REM
00600 REM ******************************************************************
00610 REM
00620 REM 1) CREATE A FILE THAT WE WILL LATER READ
00630 REM 2) READ FILE AND PRINT LINES READ UNTIL EOF CONDITION.
00640 REM
00650 REM OPEN FILE FOR CREATION.
00660  OPEN "FILE" FOR WRITE AS FILE #1
00670 REM PRINT SEVERAL LINES TO THIS FILE.
00680  PRINT #1, "MARY HAD A LITTLE LAMB"
00690  PRINT #1, "ITS FLEECE WAS WHITE AS SNOW"
00700  PRINT #1, "AND EVERYWHERE THAT MARY WENT"
00710  PRINT #1, "THE LAMB WAS SURE TO GO"
00720 REM CLOSE CREATED FILE.
00730  CLOSE #1
00740 REM OPEN CREATED FILE FOR READING.
00750  OPEN "FILE" FOR READ AS FILE #1
00760 REM READ AND PRINT LINES FROM FILE UNTIL EOF.
00770 REM READ BY USING COMBINATION OF CIN() AND 'LINE INPUT'.
00780 REM REMEMBER, SINCE CIN() REMOVES A CHARACTER FROM THE BUFFER
00790 REM WHEN IT READS, WE MUST CONCATENATE THIS CHARACTER
00800 REM WITH THE STRING READ BY 'LINE INPUT' IN ORDER TO
00810 REM GET A STRING THAT CONTAINS THE FULL LINE.
00820  J = CIN(1)
00840  IF J <= 0 THEN 900
00850  LINE INPUT #1,;S$
00860  S$ = CHR$(J)+S$
00870  PRINT S$
00880  GOTO 820
00890 REM CLOSE FILE.
00900  CLOSE #1
00910 REM DELETE CREATED FILE NOW, SINCE WE NOLONGER NEED IT.
00920  UNSAVE "FILE.DAT"
01000 REM
01010 REM ******************************************************************
01020 REM
01030  END
```

Vectored from Page 25.



```
              ;PDP-11 BASIC PATCHES 7/15/78 2000
              ;
              ;OVERLAYS HEATH/DEC-11-LPTBA-HB2  VERSION 1A
              ;
              ;PATCHES FOR SOFT START, ILLEGAL MEMORY
              ;REFERENCE TRAP, LINE PRINTER INT. VECTOR,
              ;AND 'STOP' WITH LINE NUMBER.
              ;
              ;EQUATES NECESSARY FOR ASSEMBLY:
              ;
       ØØØØØ1 R1=     %1
       ØØØØØ5 R5=     %5
       ØØØØØ6 SP=     %6
       ØØØØØ7 PC=     %7
       ØØ77Ø6 LPTINT= 77Ø6
       ØØØ2ØØ LPTPSW= 2ØØ
       ØØ1124 HSTART= 1124     ;COLD START ADDRESS
       ØØ1166 SSTART= 1166     ;WHERE ROUTINES GO WHEN DONE
       ØØ7Ø26 ILLMEM= 7Ø26     ;TO PRINT '%NEM'
       Ø1Ø626 PERROR= 1Ø626    ;TO PRINT LINE # IF RUNNING
       Ø177Ø6 PSTRNG= 177Ø6    ;PRINTS ASCIZ STRING
       Ø3Ø634 R5BASE= 3Ø634
              ;
       ØØØØØØ .ASECT
              ;
              ;CHANGE 'ØG' COMMAND TO SOFT START
              ;
       ØØØØØØ .=      Ø
ØØØØØØ ØØØ167 RESTRT: JMP SSTART
       ØØ1162
              ;
              ;PRINT '%NEM' ERROR FOR ILLEGAL MEM. REF.
              ;
       ØØØØØ4 .=      4
ØØØØØ4 ØØ7Ø26 MEMTRP: .WORD ILLMEM
              ;
              ;PATCH TO PRINT 'STOP' BEFORE PRINTING
              ;  LINE NUMBER IN 'STOP' COMMAND
              ;
       ØØØ11Ø .=      11Ø
ØØØ11Ø ØØ4167 PATCH:  JSR R1,PSTRNG   ;PRINT 'STOP'
       Ø17572
ØØØ114    Ø15         .BYTE 15,12
ØØØ115    Ø12
ØØØ116    123         .ASCII /STOP/
ØØØ117    124
ØØØ12Ø    117
ØØØ121    12Ø
ØØØ122 ØØØØØØ         .WORD Ø
```

```
ØØØ124 Ø116Ø1         MOV (SP),R1      ;NOT NEEDED, BUT SAFER ! ! !
ØØØ126 Ø167Ø5         MOV R5BASE,R5    ;SETUP R5
       Ø325Ø2
ØØØ132 Ø165Ø6         MOV 4(R5),SP     ;RESET STACK POINTER
       ØØØØØ4
ØØØ136 ØØ5Ø65         CLR 76(R5)
       ØØØØ76
                      PAGE    ØØ1

ØØØ142 Ø12765         MOV #36,34(R5)  ;DO SOME CLEAN-UP
       ØØØØ36
       ØØØØ34
ØØØ15Ø Ø6Ø565         ADD R5,34(R5)
       ØØØØ34
ØØØ154 ØØØ167         JMP PERROR
       Ø1Ø446
              ;
              ;CHANGE INTERRUPT VECTOR FOR LINE PRINTER
              ;
       ØØØ2Ø4 .=      2Ø4
ØØØ2Ø4 ØØ77Ø6 INTVEC: .WORD LPTINT,LPTPSW
ØØØ2Ø6 ØØØ2ØØ
              ;
              ;OVERLAY TO CALL PATCH TO PRINT 'STOP'
              ;  AND LINE NUMBER ALSO
              ;
       ØØ34Ø6 .=      34Ø6
ØØ34Ø6 ØØØ167 STOPP:  JMP PATCH
       174476
              ;
              ;RESET STACK POINTER BY STARTING
              ;  AT HARD START LOCATION
              ;
       ØØ1124 .END HSTART

                      PAGE    ØØ2

 HSTART = ØØ1124       ILLMEM = ØØ7Ø26       INTVEC   ØØØ2Ø4
 LPTINT = ØØ77Ø6       LPTPSW = ØØØ2ØØ       MEMTRP   ØØØØØ4
 PATCH    ØØØ11Ø       PC     =%ØØØØØ7       PERROR = Ø1Ø626
 PSTRNG = Ø177Ø6       RESTRT   ØØØØØØ       R1     =%ØØØØØ1
 R5     =%ØØØØØ5       R5BASE = Ø3Ø634       SP     =%ØØØØØ6
 SSTART = ØØ1166       STOPP    ØØ34Ø6       .      = ØØ3412

ØØØØØØ ERRORS

PAL-11S  VØ11A
```

------------------------------ CUT ALONG THIS LINE ------------------------------

HUG MEMBERSHIP RENEWAL FORM

You can determine your expiration date by examining the last six digits of your ID number — example: 780202 indicates your membership began 02/02/78 and expires one year from then.

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 APPROPIATE BOX AND RETURN TO HUG

NEW MEMBERSHIP?
FEE IS:

RENEWAL RATES			
US DOMESTIC	$11 □		$14 □
CANADA	$13 □	US FUNDS	$16 □
INTERNAT'L*	$18 □	US FUNDS	$24 □

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

THE BACK PAGE —

CONTEST #2 WINNERS

Many excellent programs were submitted for the second software contest. A team of local amatuers personally reviewed each program before declaring the winners. The winners were: Bruce McNair (Howell NJ). His program calculates and lists antenna headings to all 'ARRL' countries, US States, Canadian provinces, Australian territories and Soviet Republics. Third place in the judges opinion.

Ed Willis (Charleston W VA) Ed's program, 'HANDY HAM PROGRAMS' will solve ohms law problems, design dipole, quad and beam antennas and calculate parallel resistances. Second place.

Roger Gascon (Montreal Est Que Canada) First place honors were captured by Roger's program which keeps records on your stations activity. it allows you to 'SEARCH', 'LIST', and 'ENTER' information regarding station contacts. The judges felt this program to be the most useful of those submitted. Congratulations to the winners! Many other fine programs were submitted and will appear in Volume II later this year.

CONTEST #3 WINNERS!

Happily, we can include the winners of contest #3 in this issue also. As you may remember, contest #3 was three contests in one . . . one for the ET 3400 users . . . one for the H11 users and yet another for the H8 users. Some of the software engineers and two tech consultants reviewed the entries and selected the following as winners.

ET-3400: Louis Graue of Bowling Green, Ohio for his program which appears on page 10. ET 3400 owners can read RTTY as the message flows across the LEDs in ticker tape fashion.

H11: Francis Roy of Hull, Que Canada for his program written in 4K FOCAL. A pretty tough chess game with some interesting features.

H8: Mark Ignatius of Lakewood, Ohio. Mark's program is co-resident with either 10.01 or 10.02 Extended B.H. BASIC and renumbers BASIC programs at the stroke of a key in any increment. The command 'renumber' features command completion and defaults to renumbering programs beginning at line 10 with increments of 10. The user can, however, specify different increments, starting at a different line number. Written in assembly language, it loads through the front panel of the H8 and automatically takes care of all housekeeping chores, such as reconfiguring high memory. Congratulations to everyone for superb work. Every member is really a winner when the next issue of HUG Software is published!

ARTIFICAL INTELLIGENCE

At the PC'78 Computer Show in Philadelphia in August, perhaps thousands played a multi-user version of adventure on the H8 system . . . this same game cleverly squeezed down to reside in less than 24K by Gordon Letwin is now available to HUG members on disk for $10 plus postage and handling.(Michigan residents add 4% sales tax.) This single user version will keep H8 system owners busy all winter long! A sample (edited) run is on page 8.

To get your copy, use the green order form. Include $10 check or money order and part number . . . the part number is 885-1010.

RENEWAL TIME . . . ALMOST

Although many of you applied for membership as early as last November and December, no applications were actually processed until February. You can determine your renewal date by examining the last six digits of your ID number. A renewal form is on page 31. Of course, if you have submitted a program to the Library and it was accepted, your membership will be automatically extended and you will be notified accordingly.

and that is . . .the last word :JB:

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

885-2004