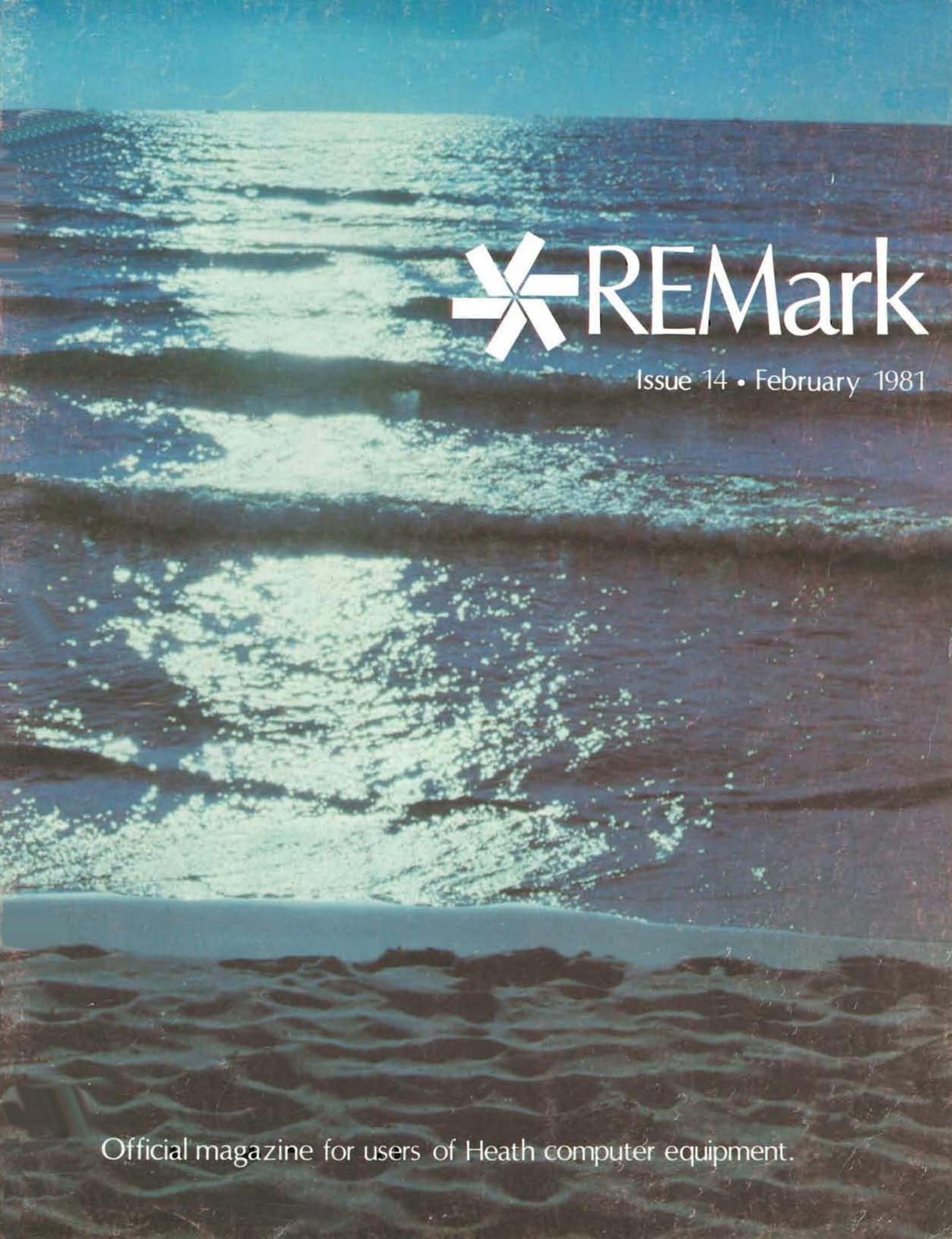


# REMark

Issue 14 • February 1981

Official magazine for users of Heath computer equipment.

# on the cover . . . .

Summer — Don't we wish!

Photo by Gerry Kabelman

# on the stack

>CAT




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

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

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

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

Send payment to:

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





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




REMark

# REFORMAT.HUG

>RUN REFORMAT.HUG

WELCOME TO HUG ADVENTURE 1981!

Now that we are entering a new year together as fellow Huggies, your Heath Users' Group would like to take this opportunity to briefly describe the changes that have occured in our operation and give you a hint of what you can expect in the near future. Of major importance is the division of "SOFTSTUFF" from HUG. Jim Blake, former Manager; Editor; Mentor and FRIEND of all Huggies, has been assigned the task of forming a software acquisition and development area. His SOFTSTUFF products will be of great benefit to all of us in the future, I'm sure! JB: will be sorely missed as it seems his creative ability has rubbed off on all of us to some extent. Please join with the HUG staff in wishing Jim the BEST for the future!

As you can see, the division mentioned above has caused some confusion around here. Fortunately, however, what has occured will be of great benefit to all members of HUG. We (the staff of HUG) are now capable of devoting all energies to the support and information exchange between "you guys"! Special feature items, such as a monthly MicroNET article with tutorials, will be included. Additionally, a monthly "goodie" (utility, game, etc.) will be placed on MicroNET for your retrival if you so desire. And, because we are publishing ten REMarks, we will be able to supply more "stuff"; from simple information for the beginner to complex information for the experienced programmer.

NAMES YOU SHOULD KNOW

All new HUG members should become aquainted with Nancy Strunk (NS:). She is the Office Wizard which takes on the large responsibility of handling new memberships, MicroNET additions, changes of address, and the general flow of all neat things which we recieve on a daily basis. She'll give you a hand with any problems, or get you to the right person if you require assistance. Your MicroNET Wizard is Terry Jensen. He can give you assistance with the operation, description, or answer general questions you may have. Terry will be known as "SYSOP" on the phone lines, and will do his utmost to keep your Heath Users' Group functioning smoothly in coming months. Patrick Swayne (PS:) probably requires little introduction. Pat, HUG's Assistant Editor, has contributed a wealth of information to past issues of REMark. One of his accomplishments include FOCAL-8 (both tape and a newly modified version on disk). His new documentation will be featured later as it allows newcomers to learn FOCAL via comparision with well known BASIC commands. One of HUG's very dedicated individuals is Gerry Kabelman. You should consider GK: as your main contact if you have any questions regarding the HUG Software Library. Gerry has been HUG's "up front" man at computer shows throughout the United States. He is dedicated to "quality HUG products and offerings". I mention myself as the last member of your HUG team. With the division of HUG from SOFTSTUFF, a new "whipping post" for the Heath Users' Group was required. My name is Bob Ellerton. I come to you from the Heath Customer Service Department where I was assigned to supervise the repair and testing of consumer computer equipment. I feel that HUG's responsibility to the membership is the continuous and accurate exchange of usuable information. It will be my primary objective during 1981 to make this happen! I hope each of you will take the time to get to know all of us. Remember, we are here to serve You!

Now that you have met your newly formed HUG team, I should mention how excited we are to get things underway. We have collected several excellent articles from fellow members contained in this issue. Watch for the monthly New Products section which will contain some of the latest developments by the membership. Also, HUG is "keeping our ear to the ground" for future developments from Heath Company, both hardware and software. We will try to keep you current on any info which may be interesting to "computer fanatics". Further, since we have close ties with Heath's Customer Service Department, we will attempt to bring you the latest in simple product updates which will keep your computers running smoothly. In turn, if you have any little "tips" for fellow members that you feel are worth while, pass them along to me for future issues of REMark!

By working together as a unit all of us can contribute to the continued growth and knowledge contained in YOUR Heath Users' Group. With the good things to come, HUG wishes all of you a Happy and Prosperous New Year!

BE:

# ET3400 Morse Code Transmitter

by Allen H. Wolach
Department of Psychology
Illinois Institute of Technology
Chicago, IL 60616

## Introduction

This article presents a program that makes it possible to type a message in English sentences and play it back in International Morse Code. An ET-3400/ETA-3400 microcomputer system with 4k bytes of memory is interfaced to a relay that can be used to operate a code practice oscillator or a radio transmitter. The NO and C contacts of the relay are substituted for the NO and C contacts of the telegraph key.

## Interfacing Circuit

Figure 1 shows a modification of the circuit described on pages 10-39 through 10-42 of the Heathkit manual Individual Learning Program in Microprocessors. If the relay in Figure 1 is rated at 5 volts and draws less than 500 ma, the entire circuit can be powered by the 5 volt power supply on the ET3400 microcomputer.

## Entering and Using Code

Enter the program from Tables 1 and 2 in memory locations 0BFA through 0D4C. The number in memory locations 0BFA (high byte) and 0BFB determines the rate of transmission. The higher the number, the slower the rate of transmission. This number can be empirically changed to obtain rates of code transmission that range from less than 1 word per minute (wpm) to over 40 wpm. Estimate the wpm by dividing the number of words in the message that is transmitted by the time (in minutes and fractions of a minute) it takes to transmit the words. Although the program is written in 6800 assembly language, it makes use of Tiny BASIC for entering the message. Start by typing

```
Do 1400
```

on the ET3400 key pad followed by a carriage return. Then type

```
G1C00
```

followed by a carriage return on the keyboard of the video terminal. The video terminal will respond with

```
HTB1 G1C00
:
```

indicating that the Tiny BASIC program has been entered. Then type a line number followed by the first line of the message, followed by a carriage return. For example, you could type:

```
10 THIS IS AN INTERNATIONAL MORSE CODE
   TRANSMISSION PROGRAM
```

After you enter the first line (followed by a carriage return), enter the next (higher) line number and type the second line of the message, for example:

```
20 THAT CAN SEND LETTERS, NUMBERS,
   PUNCTUATION MARKS, ETC.
```

Type a carriage return at the end of this line also and repeat the procedure to enter additional lines of the message. Type the % key followed by a carriage return after the final symbol in the last line of the message. Then type

```
BYE
```

followed by a carriage return to exit from Tiny BASIC. The display will show

```
MON:
```

which indicates that you have exited from Tiny BASIC. Now type

```
G0C00
```

followed by a carriage return to transmit the message. If you want to modify the message you can re-enter Tiny BASIC by typing B (no carriage return is required). Do not use G1C00 to re-enter or the message will be lost. The features of Tiny BASIC can be used to edit the message. For example, typing LIST followed by a carriage return will list the message on the video display. Typing a line number followed by a carriage return will delete that line from the message.

## How It Works

The code transmission program sends every symbol (letter, number, etc.) with one unit of time for a dit (dot) and three units of time for a dah (dash). One unit of time is inserted between letters or symbols that are not separated by a space. When a space is typed in the message, seven units of time are generated to indicate the space between symbols.

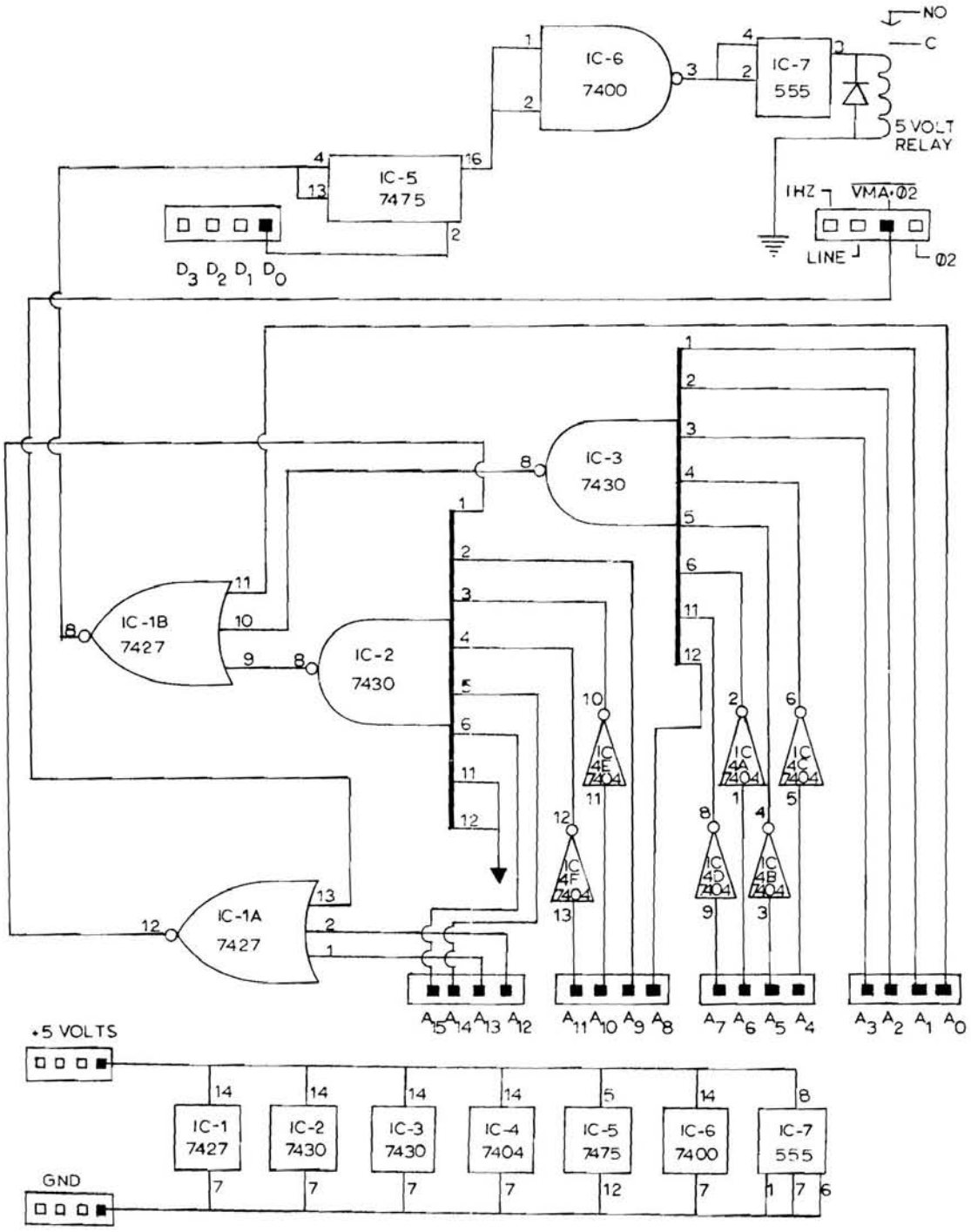



Figure 1. ET3400 Relay Driver Circuit

A carriage return, like a space, generates 7 units of time. The line numbers at the beginning of each line are ignored by the code transmission program. Whenever a % symbol is encountered in the message, control is returned to the monitor program. You can then return to BASIC by typing B.

The look-up table in Table 2 shows the International Morse Code symbols and their equivalent keyboard representation. For example, typing the letter P will lead to the transmission of .--- which is the International Morse Code for P. All Morse Code symbols are not represented on the keyboard. The comments column of Table

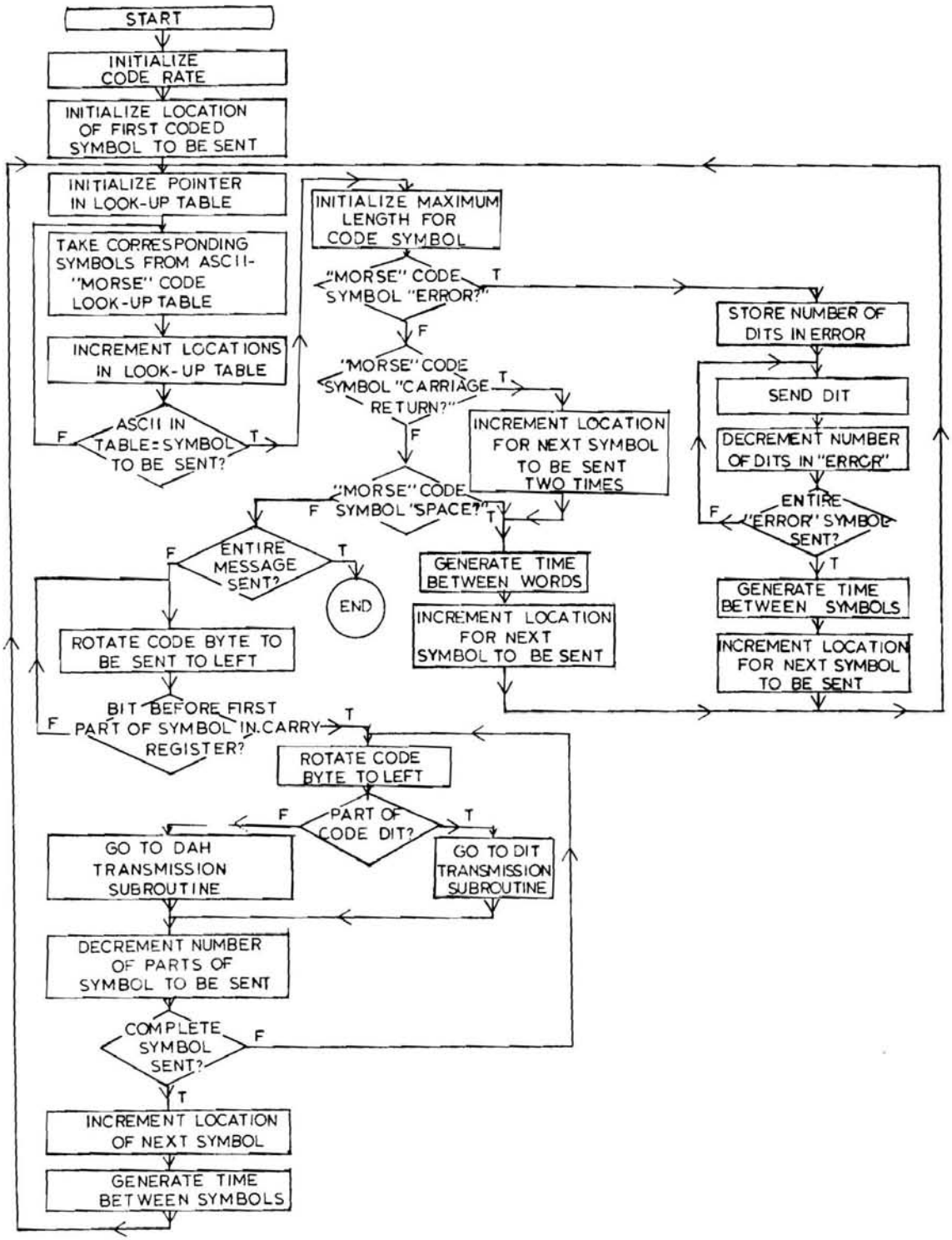



Figure 2. Morse Code Program Flow Chart

2 shows how these symbols can be typed. For example, typing ] will lead to the transmission of ...-.- which is the symbol for "end of work".

Memory locations ØCD1 through ØDØE (see Table 2) contain the ASCII representation of the symbols that are on the video terminal keyboard. Memory locations ØDØF through ØD4C contain the corresponding symbols that are used by the transmission program. All symbols in locations ØDØF through ØD4C, with the exception of the error symbol, are represented in the following way. A dit is represented by a Ø and a dah by a 1. The symbol is

represented in an 8-bit word that starts with right justified International Morse Code (1's and Ø's). For example, the Morse code for D is dah, dit, dit. This is represented in bits Ø, 1, and 2 by

```
dah dit dit
 1   Ø   Ø
```

A 1 is always placed in the bit to the left of the first code bit. Thus,

```
    dah dit dit
 1   1   Ø   Ø
```

becomes the representation of the letter D. Finally, the remaining bits at the left of the symbol are Ø's. The eight bits that represent D in the program are

```
 Ø   Ø   Ø   Ø   1   1   Ø   Ø
```

The program rotates the byte to the left until the first 1 is encountered in the carry register. This indicates that the next symbol is the first part of the code symbol that is to be transmitted. The program continues to rotate the byte to the left sending a dit if a Ø is placed in the carry register, and sending a dah if a 1 is placed in carry. It should be noted that the symbol for error (8 dits) is represented by ØØ.

Operation of the rest of the program is outlined in Figure 2. The program tests for the error symbol, carriage return, and space symbol before it transmits a symbol. If the symbol is an error symbol, 8 dits are transmitted. If it is a carriage return, the index register is incremented twice in order to avoid transmitting Tiny BASIC line numbers. Finally, it should be noted that the Tiny BASIC program places the ASCII text in consecutive memory locations starting at location Ø1Ø2. Every carriage return in the message is followed by the two byte line number for the next line of the message.

Table 1. Program Listing

```
Address Data     Label   Op code Operand Comments

ØBFA  21                 data            timing byte for code speed
ØBFB  FF                 data
ØBFC  11                 data            save index register
ØBFD  11                 data
ØBFE  Ø8                 data            dits in "error" that have been sent
ØBFF  Ø8                 data            number of parts of symbol sent
ØC00  86 Ø1              LDA A   #$Ø1    first half starting address of ASCII message
ØC02  B7 ØC 13           STA A   $ØC13
ØC05  86 Ø2              LDA A   #$Ø2    second half starting address of ASCII
ØC07  B7 ØC 14           STA A   $ØC14
ØC0A  CE ØC D1  GET      LDX     #$ØCD1  look-up table start
ØC0D  A6 3E     TABLE    LDA A   $3E,X   start of Morse Code portion of table
ØC0F  E6 ØØ              LDA B   $ØØ,X   start of ASCII portion of look-up table
ØC11  Ø8                 INX             increment position in look-up table
ØC12  F1 Ø1 Ø2           CMP B   $Ø1Ø2   current ASCII symbol to be sent
ØC15  26 F6              BNE     TABLE   no match, check next table location
ØC17  C6 Ø8              LDA B   #$Ø8    maximum length for Morse Code symbol
ØC19  F7 ØB FF           STA B   $ØBFF   location indicating all sent
ØC1C  4D                 TST A           determine if symbol is "error"
ØC1D  27 35              BEQ     ERROR   send Morse Code for "error"
ØC1F  81 99              CMP A   #$99    is current symbol a carriage return?
ØC21  27 44              BEQ     INCR    if so, increment symbol counter
ØC23  81 1F              CMP A   #$1F    is current symbol a space?
ØC25  27 25              BEQ     SPACE   execute space subroutine
ØC27  81 FF              CMP A   #$ØFF   is this end of message (%)?
ØC29  26 Ø3              BNE     BEG     no, continue
ØC2B  BD 14 ØØ           JSR     MON     yes, exit to monitor
ØC2E  49        BEG      ROL A           rotate to find beginning of code
ØC2F  7A ØB FF           DEC     $ØBFF   decrement no. of symbol parts
ØC32  24 FA              BCC     BEG     rotate until beginning found
ØC34  49        CODE     ROL A           rotate one bit of Code
ØC35  24 Ø5              BCC     DIT     no carry, go to DIT
ØC37  BD ØC 8E           JSR     DAHS    carry, go to DAH subroutine
ØC3A  2Ø Ø3              BRA     PARTS   decrement no. of parts
ØC3C  BD ØC 7D  DIT      JSR     DITS    execute DIT subroutine
ØC3F  7A ØB FF  PARTS    DEC     $ØBFF   decrement no. of symbol parts
ØC42  26 FØ              BNE     CODE    loop until all of symbol sent
ØC44  BD ØC 6F           JSR     INCR1   increment to next symbol
ØC47  BD ØC A8           JSR     TSYM    call time between symbols subroutine
```

```
0C4A  20 BE            BRA     GET     prepare to send next symbol
0C4C  BD 0C AF  SPACE  JSR     TWORD   call time between words subroutine
0C4F  BD 0C 6F         JSR     INCR1   increment to next symbol
0C52  20 B6            BRA     GET     get next symbol
0C54  F7 0B FE  ERROR  STA B   $0BFE   store number of bits for "error" (8)
0C57  BD 0C 7D  REPT   JSR     DITS    call DIT subroutine
0C5A  7A 0B FE         DEC     $0BFE   decrement error dits
0C5D  26 F8            BNE     REPT    loop until all dits are sent
0C5F  BD 0C A8         JSR     TSYM    allow time between symbols
0C62  BD 0C 5F         JSR     INCR1   increment to next symbol
0C65  20 A3            BRA     GET     get next symbol
0C67  BD 0C 6F  INCR   JSR     INCR1   increment to next symbol
0C6A  BD 0C 6F         JSR     INCR1   increment once more
0C6D  20 DD            BRA     SPACE   allow space between words
                *Subroutines start here
                *Increment to next symbol to be sent
0C6F  7C 0C 14  INCR1  INC     $0C14   increment low byte of symbol counter
0C72  26 03            BNE     AD1     if no carry, exit from routine
0C74  7C 0C 13         INC     $0C13   increment high byte of counter
0C77  39        AD1    RTS             return to main program
0C78  01               NOP             five unused program locations
0C79  01               NOP
0C7A  01               NOP
0C7B  01               NOP
0C7C  01               NOP
                *Subroutine to generate dits
0C7D  C6 01     DITS   LDA B   #$01    01 is byte to turn on relay
0C7F  F7 C3 03         STA B   $C30E   location of relay
0C82  BD 0C C2         JSR     TIME    generate a unit of time
0C85  C6 00            LDA B   #$00    00 is byte to turn relay off
0C87  F7 C3 03         STA B   $C30E   location of relay
0C8A  BD 0C C2         JSR     TIME    generate a unit of time
0C8D  39               RTS
                *Subroutine to generate dahs
0C8E  C6 01     DAHS   LDA B   #$01    turn relay on
0C90  F7 C3 0E         STA B   $C30E
0C93  BD 0C C2         JSR     TIME    generate units of time
0C96  BD 0C C2         JSR     TIME
0C99  BD 0C C2         JSR     TIME
0C9C  BD 0C C2         JSR     TIME
0C9F  C6 00            LDA B   #$00    turn relay off
0CA1  F7 0C C2         STA B   $C30E
0CA4  BD 0C C2         JSR     TIME    generate a unit of time
0CA7  39               RTS
                *Subroutine to allow time between symbols
0CA8  BD 0C C2  TSYM   JSR     TIME    generate units of time
0CAB  BD 0C C2         JSR     TIME
0CAE  39               RTS
                *Subroutine to allow time between words
0CAF  BD 0C C2  TWORD  JSR     TIME    generate units of time
0CB2  BD 0C C2         JSR     TIME
0CB5  BD 0C C2         JSR     TIME
0CB8  BD 0C C2         JSR     TIME
0CBB  BD 0C C2         JSR     TIME
0CBE  BD 0C C2         JSR     TIME
0CC1  39               RTS
                *Subroutine to generate units of time
0CC2  FF 0B FC  TIME   STX     $0BFC   save index register
0CC5  FE 0B FA         LDX     $0BFA   get timing value
0CC8  09        DELAY  DEX             decrement timing value
0CC9  01               NOP             waste some time
0CCA  01               NOP
0CCB  26 FB            BNE     DELAY   loop until timing value is zero
0CCD  FE 0B FC         LDX     $0BFC   restore index register
0CD0  39               RTS
```

Table 2. Look-up Table

| Address | Data (ASCII) | Address | Data (Morse) | Comments | Morse code |
|---|---|---|---|---|---|
| 0CD1 | 21 | 0D0F | 00 | ! (error) | ........ |
| 0CD2 | 22 | 0D10 | 52 | " | .-..-. |
| 0CD3 | 23 | 0D11 | 28 | # | .-... |
| 0CD4 | 24 | 0D12 | 89 | $ | ...-..- |
| 0CD5 | 25 | 0D13 | FF | % (end of message) | |
| 0CD6 | 26 | 0D14 | 00 | & (error) | ........ |
| 0CD7 | 27 | 0D15 | 5E | ' | .----. |
| 0CD8 | 28 | 0D16 | 6D | ( | -.--.- |
| 0CD9 | 29 | 0D17 | 6D | ) | -.--.- |
| 0CDA | 2A | 0D18 | 00 | * (error) | ........ |
| 0CDB | 2B | 0D19 | 00 | + (error) | ........ |
| 0CDC | 2C | 0D1A | 73 | , | --..-- |
| 0CDD | 2D | 0D1B | 61 | - | -....- |
| 0CDE | 2E | 0D1C | 55 | . | .-.-.- |
| 0CDF | 2F | 0D1D | 32 | / | -..-. |
| 0CE0 | 30 | 0D1E | 3F | 0 | ----- |
| 0CE1 | 31 | 0D1F | 2F | 1 | .---- |
| 0CE2 | 32 | 0D20 | 27 | 2 | ..--- |
| 0CE3 | 33 | 0D21 | 23 | 3 | ...-- |
| 0CE4 | 34 | 0D22 | 21 | 4 | ....- |
| 0CE5 | 35 | 0D23 | 20 | 5 | ..... |
| 0CE6 | 36 | 0D24 | 30 | 6 | -.... |
| 0CE7 | 37 | 0D25 | 38 | 7 | --... |
| 0CE8 | 38 | 0D26 | 3C | 8 | ---.. |
| 0CE9 | 39 | 0D27 | 3E | 9 | ----. |
| 0CEA | 3A | 0D28 | 78 | : | ---... |
| 0CEB | 3B | 0D29 | 6A | ; | -.-.-. |
| 0CEC | 3C | 0D2A | 00 | < (error) | ........ |
| 0CED | 3D | 0D2B | 31 | = | -...- |
| 0CEE | 3E | 0D2C | 00 | > (error) | ........ |
| 0CEF | 3F | 0D2D | 4C | ? | ..--.. |
| 0CF0 | 40 | 0D2E | 00 | @ (error) | ........ |
| 0CF1 | 41 | 0D2F | 05 | A | .- |
| 0CF2 | 42 | 0D30 | 18 | B | -... |
| 0CF3 | 43 | 0D31 | 1A | C | -.-. |
| 0CF4 | 44 | 0D32 | 0C | D | -.. |
| 0CF5 | 45 | 0D33 | 02 | E | . |
| 0CF6 | 46 | 0D34 | 12 | F | ..-. |
| 0CF7 | 47 | 0D35 | 0E | G | --. |
| 0CF8 | 48 | 0D36 | 10 | H | .... |
| 0CF9 | 49 | 0D37 | 04 | I | .. |
| 0CFA | 4A | 0D38 | 17 | J | .--- |
| 0CFB | 4B | 0D39 | 0D | K | -.- |
| 0CFC | 4C | 0D3A | 14 | L | .-.. |
| 0CFD | 4D | 0D3B | 07 | M | -- |
| 0CFE | 4E | 0D3C | 06 | N | -. |
| 0CFF | 4F | 0D3D | 0F | O | --- |
| 0D00 | 50 | 0D3E | 16 | P | .--. |
| 0D01 | 51 | 0D3F | 1D | Q | --.- |
| 0D02 | 52 | 0D40 | 0A | R | .-. |
| 0D03 | 53 | 0D41 | 08 | S | ... |
| 0D04 | 54 | 0D42 | 03 | T | - |
| 0D05 | 55 | 0D43 | 09 | U | ..- |
| 0D06 | 56 | 0D44 | 11 | V | ...- |
| 0D07 | 57 | 0D45 | 0B | W | .-- |
| 0D08 | 58 | 0D46 | 19 | X | -..- |
| 0D09 | 59 | 0D47 | 1B | Y | -.-- |
| 0D0A | 5A | 0D48 | 1C | Z | --.. |
| 0D0B | 5B | 0D49 | 2A | [ (end) | .-.-. |
| 0D0C | 20 | 0D4A | 1F | space | |
| 0D0D | 5D | 0D4B | 45 | ] (end work) | ...-.- |
| 0D0E | 0D | 0D4C | 99 | carriage return | |

EOF

# DBMS (Data Base Management Systems)

How To Use Data Files in MBASIC

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

EDITOR'S NOTE: If you would like to know how to use data files in BASIC, read on. In this article, Doc explores the mysteries of sequential and random data files, records, fields, and buffers.

ABSTRACT

Fundamental concepts dealing with the items in the title above are discussed, and some of the fundamentals are demonstrated using three short MBASIC programs and one short data file. This article will be of most value to the Heath user who is new to HDOS (version 1.6) and MBASIC (version 4.7), or can serve as "review" for experienced MBASIC programmers.

DATA BASE

Frequently one sees references to DBMS (Data Base Management System). What is a data base? It is simply a program, or set of programs, which create and manipulate data (or, information) files. The data can be a mailing list, a Christmas card list, a 'library of music', an inventory, or whatever. It is just that and nothing more. The programs can be written in a low level language such as assembly language, or a high level language such as BASIC. The data files may be either sequential or random in type. Some data files may be created using a Text Editor such as HDOS EDIT or HUG's ED. (However, the sophisticated programmer rarely uses an EDIT program to create his data files.)

In all such lists of information, there are usually many different items that are logically grouped together. For example, in a mailing list, for each name, there is usually a street address, city, state, and zip code. These items are commonly kept together as a unit. Each unit contains the name and address of one individual. Such a collection of items is commonly called a RECORD.

A COLLECTION OF SUCH RECORDS IS CALLED A DATA FILE.

So, a data file consists of a number of records, and each record contains one or more items that are logically grouped together.

SEQUENTIAL DATA FILES

In a SEQUENTIAL data file, the items in one record are separated from each other by "delimiters". A frequent "delimiter" is the "carriage return-line feed". (Remember that in HDOS this is actually a "new line character", octal Ø12.) Another delimiter commonly used is any desired character which is not going to be used in the items. For example, a reverse slash (\).

However, we must also recognize that each record must be separated from the next record by a delimiter. If we use the "new line character" (RETURN) to separate various fields, how can we use the same delimiter to separate our records? This is not difficult. What one does is to determine at the beginning just how many fields are going to be in each record. Then, after x number of fields, x will be the record delimiter. For example, consider a record with just three fields (3 items):

```
Name
Address
CityStateZip
```

The name is separated from the address by ONE "new line character". The address is separated from the CityStateZip by ONE "new line character". The THIRD "new line character" occurs AFTER the Zip and can be considered to separate this record from the next record. In other words, every third "new line character" separates one record from the next record.

Another way of handling records, and the one which I use, is to separate the items (fields) IN a record using a reverse slash (\), and I separate the records with "new line characters". Such a data file might look like this:

```
Name1\Address\CityStateZip
Name2\Address\CityStateZip
etc.
```

So, my sequential data files consist of records with each record separated from the next by a "new line character". An individual record consists of several "fields" which are separated from each other by reverse slashes (\).

I have been asked many times why I keep my data files in this fashion. The answer is that I am used to this method, AND, this method will save disk space when one deals with disk files which are "Random Access".

Let us now consider how HDOS handles DATA FILES, both SEQUENTIAL and RANDOM.

HDOS and MBASIC (also HBASIC) ALWAYS move data to and from a diskette in a 256 byte aggregates. This is the MINIMUM amount of data that can be moved when the disk is accessed. When you open a file from MBASIC, you automatically create a 256 byte area of memory (called a buffer memory area) associated with THAT file. Note that 256 bytes (or characters) is also the length of a single sector on a disk. Suppose (using MBASIC) that you open a file for Input as file #1. Then you LINE INPUT X$. Next you print X$. Suppose your first "line" of data in the file was 60 characters, and each character was the letter "A". Suppose also, that the next "line" of data was again 60 characters and each character was the letter "B". Each line of data was separated from the next by a "new line character". In other words, when you create a file - say using EDIT or ED - you typed 60 A's, then hit RETURN, typed 60 B's and hit return. Now, after opening this file, and LINE INPUTting, then printing, you were rewarded with 60 A's being printed on the screen. But, HDOS did not just put those 60 characters (A's) in the 256 byte buffer; HDOS put a whole sector's worth of data in the memory buffer! Let's prove this!

THE READER IS NOW ENCOURAGED TO ENTER THE THREE ACCOMPANYING MBASIC PROGRAMS (TEST.BAS, TEST2.BAS, TEST3.BAS) AND TO SAVE EACH.

```
10 REM    TEST.BAS
20 OPEN "I",1,"TEST.DAT"
30 LINE INPUT #1,A$
40 PRINT A$
50 STOP
60 LINE INPUT #1,B$
70 PRINT B$
80 STOP
90 LINE INPUT #1,C$
100 PRINT C$
110 STOP
120 LINE INPUT #1,D$
130 PRINT D$
140 STOP
150 LINE INPUT #1,E$
160 PRINT E$
170 STOP
```

```
10 REM    TEST2.BAS
20 OPEN "R",1,"TEST.DAT"
30 GET #1
40 FIELD 1, 255 AS X$
50 PRINT X$
```

```
10 REM    TEST3.BAS
20 OPEN "R",1,"RTEST.DAT"
30 FIELD #1,64 AS A1$,64 AS B1$,64 AS C1$,64 AS D1$
40 A$=STRING$(60,"A")
50 B$=STRING$(60,"B")
60 C$=STRING$(60,"C")
70 D$=STRING$(60,"D")
80 LSET A1$=A$:LSET B1$=B$:LSET C1$=C$:LSET D1$=D$
90 PUT #1
100 CLOSE:END
```

Then he should create the test data file (Figure A) and name it "TEST.DAT", using EDIT or ED. Each line is exactly 60 characters long and 60*4=240 bytes which will fit into one sector on the disk. Exit your edit program and from the monitor prompt (>) enter - TYPE TEST.DAT - and hit RETURN. You will see displayed the contents of TEST.DAT. You can consider each line as being one record consisting of just one item. Each record is separated from the next record by a "new line character".

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
```

Figure A. TEST.DAT

Now load MBASIC and RUN "TEST.BAS". The first thing to happen is that you will HEAR the disk being accessed (TEST.BAS file is being opened and the data is transferred to the buffer memory). Immediately thereafter 60 A's will be printed on the screen, and the screen then informs you "BREAK IN LINE 50". The data was transferred to the memory buffer and printed to the screen in response to lines 30-40. However, you might assume that only that ONE line of data was transferred to memory and then printed out for you in response to line 30's LINE INPUT. NOT SO! Listen very carefully while typing "CONTINUE" and hit RETURN. You will be rewarded with a line of 60 B's and then a BREAK. Note that you did NOT hear any disk access!!!!!! There wasn't any! The line of B's was already in that 256 character memory buffer. Type "CONTINUE" again and again until you get a message "INPUT PAST END". Note that the disk was never accessed again! You will conclude (correctly) that one entire sector was transferred from disk to memory at the time of the initial single line input of line 30!

This is what ALWAYS happens during disk input. And, the same thing happens when you write to disk. Here, the information is stored in the 256 character buffer until about 256 characters are in the buffer (or when you close file), and then the whole buffer is written to disk at one time. Your hearing will verify this. Note that all of the above is "transparent" to the user when using SEQUENTIAL files, unless you are specifically looking for (and listening for) this phenomena.

So, you can now appreciate how HDOS (and MBASIC) handle a SEQUENTIAL file of data. Although several lines (records) of data are transferred to memory at one time, in response to a single LINE INPUT, only data from beginning of buffer to the first "new line character" is printed on the screen. This means that MBASIC looks for the "new line character" and it delineates this line (this record) from the next record for you when it encounters a "new line character". That is what happens behind the scenes, but as noted this is transparent to the user in SEQUENTIAL line handling. Also remember that the data on disk consisted of 'records', and each record was separated from the next record by a "new line character". You saw the results of the new line characters on the screen when you earlier displayed the entire data file. (If there had not been any new line characters in the file, the A's would have run into the B's and the B's into the C's, etc. (Note here that your software will insert a new line charater at the end of 80 characters on screen, or whatever line width your terminal is set for. However, with this file, we never approached the right margin of the screen, and therefore the separation of one line from the next was caused by the new line character which was IN the data file to begin with.)

SEQUENTIAL DATA FILE HANDLING SUMMARY

We have discussed how input-output from disk is always done in 256 byte aggregates. Individual records are frequently separated from one another by "new line characters", and, the new line characters are IN the data file. Fields within records are "delimited" by arbitrary characters (including "new line characters").

RANDOM FILES

Why consider all the above? Simply because you must understand that this buffer does exist and how it works, when you begin with RANDOM files. Here, although input and output also always occur in aggregates of 256 characters, MBASIC does not key on "new line characters" to separate one record from another. AND, delimiters within records are NOT required (although we will discuss how to use them, if desired, later). This is because individual FIELDS may be of FIXED length and you can define them using the FIELD statement. RECORDS in random files are ALWAYS of fixed length! The record length MUST be defined by the user in the FIELD statement.

In random file access, the user must manipulate the 256 byte buffer, and he does so with the FIELD statement. RUN TEST2.BAS now. Note that the entire contents of the buffer were printed out for you. This is because line 40 fielded the ENTIRE buffer (256 is an illegal octal number and so you must use 255 if you ever want to field the entire buffer). Now change line 40 to:

```
40 FIELD 1,60 AS A$,1 AS NL$,60 AS B$,1 AS NL$,60 AS C$,1 AS NL$,60 AS D$
```

Change line 50 to read - 50 PRINT A$:STOP

When you now RUN this program, it will print 60 A's. When program stops, you can use direct command PRINT B$; then PRINT C$, etc. (or write them into the program.)

Note that the fielding in line 40 includes 1 as NL$ between each "line". We are simply putting the "new line character" into this little field each time. Otherwise, we would be off by one character in each field after the initial 60 as A$.

Again, note with this program, that only ONE disk access was required to set the contents of the one sector into memory.

Next load program TEST3.BAS. This program simply creates a RANDOM data file similar to the file which we created in ED or EDIT above. But, that file was a SEQUENTIAL file. Now we wish to create the same file, except to have it in true RANDOM nature. After RUNning this program, exit MBASIC and, from the monitor prompt (>), enter TYPE RTEST.DAT and hit RETURN. You now see the contents of this random data file. Note that there are no "new line characters" in this data file. The software, as mentioned previously, has automatically inserted a new line charater at the right side of your terminal. If the terminal width was 256 characters wide, you would have ONE single line of data - 60 A's, 4 spaces, 60 B's, 4 spaces etc.. There are NO new line characters to separate the "records". When you use random access, MBASIC separates each record from the next by counting. For example, if you have fielded your data as 64*4 (and have done any necessary delimiting yourself within each record using say - a reverse slash), then MBASIC counts the first 64 characters as the first record, the next 64 as the next, and so on. (Remember, however, the LOF only gives you a sector number, and if you have more than one record in a sector then you must use the 2 FORMULAS (discussed in my article in issue 10 of REMark) to find any record from its'actual key record number.) Note that each group of 64 "characters" would include any characters which you might have used as delimiters. Note that the 4 spaces at the end of each 60 characters in the displayed data file were inserted at the end of each group of characters by the LSET statement.

We noted above that if one fields ONLY the RECORDS, and delimits the individual fields within the record with an arbitrary delimiter such as a reverse slash (\), that considerable disk space might be saved. This is because of random access fielding. IF you set up the fields for the various items in a record using the random access FIELD statement, you MUST allow enough space in EACH field for the MAXIMUM number of characters that you think will be needed for EACH field. For example, although the average length of a name might be 20 characters, you estimate that the MAXIMUM length of a name might be 30 characters. Therefore you must FIELD NAME$ AS 30. If NAME$ is less than 30, the LSET command which you use to move the name to the memory buffer will also pad the name to 30 with spaces. Similarly, you must allow for the MAXIMUM length of the address portion when you field it, etc. So, your field statement MUST allow for the MAXIMUM length of EACH individual item IF you FIELD the various "fields" (items) WITHIN each record. On the other hand, if you 'do your own' fielding (within a record) using an arbitrary delimiter such as a reverse slash (\), then you only have to worry about the MAXIMUM length of the WHOLE record. Since many times you might have a record with a short name, and a long address, these would CANCEL OUT as far as the TOTAL length of the record was concerned. I figured this out once and the savings can be considerable. Hence, I use my own delimiters within each record, and let MBASIC handle (through the FIELD statement) only the delineation of ONE RECORD FROM THE NEXT RECORD.

RANDOM FILE SUMMARY

Input-output to random data files is similar to sequential file handling in that data is always moved in 256 byte aggregates. However, records are delimited differently in a random file. Usually, there are NO "new line characters" to delimit records in a random file. (In a sequential file, the "new line character" is frequently used to delimit records.)

A case has been made for 'do-it-yourself' delimiting within individual random file records, rather than using the FIELD statement to do this for you.

FINAL SUMMARY

HDOS and MBASIC always move 256 byte (256 characters) aggregate of data to and from disk during EACH disk "READ" or disk "WRITE". A 256 byte buffer is always created whenever a file is OPENed, and this buffer is associated only with THAT named (and

numbered) file. The 256 bytes are moved from disk to the buffer and then parts of the 256 bytes can be accessed by the program as needed. Or, data is written into the buffer until about 256 bytes have accumulated, and THEN the whole 256 bytes are written to disk. This buffer and the movement of data to and from disk in 256 byte aggregates is present with both SEQUENTIAL and RANDOM data file handling. However, it tends to be transparent to the user in SEQUENTIAL file handling.

DATA FILES consist of one or more RECORDS. A RECORD consists of one or more FIELDS. The FIELDS of any given record are items of data information that are logically concerned with that record. FIELDS are separated from each other in SEQUENTIAL files by DELIMITERS. Sequential delimiters are either NEW LINE CHARACTERS, or an ARBITRARY character selected by the user, or a combination of these. Fields of SEQUENTIAL files may be of ANY length, and records of SEQUENTIAL files may be of ANY length. These are commonly said to be VARIABLE LENGTH RECORDS AND FIELDS. In contrast, RECORDS of RANDOM files are ALWAYS of a FIXED LENGTH (the length is selected by the user), and the length is defined in the Random File FIELD STATEMENT. If the FIELDS of any given record are defined in the FIELD STATEMENT also, then EACH of these fields MUST also be of a FIXED LENGTH (again the lengths of the individual fields are determined by the user). The LSET statement "pads", (with spaces), the fields and/or records, to the fixed lengths defined in the FIELD statement. HOWEVER, it is not necessary to define individual ITEMS of a RANDOM data RECORD in the FIELD statement, and the user may use his OWN delimiters WITHIN the records. This tends to cancel out some variable length fields as discussed above, thus saving disk space.

EOF

# Disk Care — Or Else

THE MAGIC EGG ... (continued)

The point of the MAGIC EGG in the previous issue of REMark was to call your attention to the little known facts about your operating floppy disks and how these disks are affected by the operating environment. DID YOU KNOW... that your disks could expand with changes in humidity, temperature, and age? DID YOU KNOW... that, because of the magnetic oxide which coats every usuable disk, the disk will not "come back" into shape after the expansion? DID YOU KNOW... that (and here's the biggy) the expansion will manifest itself as a possible DRIVE MALFUNCTION? Well, all of the above questions should be answered with a simple YES! If you didn't know, didn't care, etc.....READ ON.

The greatest enemy to your programmed disk is probably HUMIDITY. This "guy" actually causes the expansion. Oh! Before I forget, mylar, which is the material used to make the disk, is used quite commonly in home heating systems. Where? you ask. Mylar is often the sensing element for a humidity control system. It expands and contracts and is usually connected to some switching device to "kick in" a humidifier. (Enough of that.) Because our "programmed" disks are coated with oxide, the disk is not capable of returning or maintaining its' original shape. SO WHAT!?...When the disk expands the individual tracks become elongated and because the read/write head of the drive is fixed, we ask the drive to read an "egg" shaped "groove". How many of us have observed the tonearm on a record-player as it tries to follow a warped record?....Probably most of us! As you know, the tonearm "bobs and weaves" as it tracks the grooves of the record or....shall we say disk. Well, this is exactly what we are asking the "tonearms", or FIXED heads, of our drives to do during normal operation. Any drive or disk is required to operate in a "noisy" environment. By "noisy", I mean that the computer is radiating interference throughout the entire system. If, given the conditions of disk expansion, noise, humidity, temperature, etc., we now ask the head to read an egg shaped "groove", the results can be, to say the least, less than satisfactory.

WHAT CAN BE DONE to prevent disk expansion? First, the bad news. The disks now being manufactured WILL expand and we, the users, can do little or nothing about it. This means there is a good chance we will encounter disk (media...new word) problems in the future. All users must exercise as much care as possible when handling and storing our treasures. BACK-UPS are extremely important. It is unlikely that two disks will become unreadable at the same time (although it is possible...more on this later). Further, it is important to note the effect of speed (seek, step, etc.) on a particular disk. Some of you "oldtimers" that were with HUG when we released our original disks will remember the frustration when you were unable to get the disk to load. Our (HUG's) first releases were SET STEP 8 (or seeking at 8mS). Even if a particular drive could

pass TEST at 8mS, the disk may not have loaded anyway. In all probability, the reason the drive failed to read the disk was NOT because the drive had slowed, but, because the disk had expanded slightly or because the environment (computer system) contained more noise than the system used by HUG to make the copies. In any case ,...there YOU sit with a useless piece of software, and here WE (HUG) sit, the "bad guy." This situation was corrected when all future copies of software were run at 30ms (drive minimum standard)....Well ALMOST!

ALL IS NOT LOST.....
By now you probably are starting to get the idea! Running your drives too fast will give you media or disk problems more quickly than if you choose to go for the "MAX". Now we get to the GOOD NEWS. If you begin to encounter problems with your "DRIVES" such as "read" errors, "fatal system" errors, or a general lack of co-operation on the part of your drives, it is time to reexamine your media before taking other "drastic" steps. (It is well to keep in mind, that none of your disks, including ANY masters, are exempt from changes or expansion.) STEP 1. SLOW your drives down. Using SET, reset the step rate to 30mS. If this doesn't work, go SLOWER! 35mS, 40mS, etc. STEP 2. As a last resort, use HDOS STAND-ALONE to reset the speed if you are unable to "Boot" the system.

It is obviously best to make back-ups of your important software from the begining. However, if you find yourself (as I have) in the situations described above, and you need the additional speed, make a new copy of the disk you are having problems with. Copying the disk ensures what I will call a "ROUND TRACK" even on a disk which is egg shaped. Egg shaped disks ARE NOT BAD disks, they merely require a little extra attention on your part. If you want to save disk LABELS, make a copy on a GOOD scratch disk, then re-copy the same material back to the original disk that has the correct label.

BREAK... (Doc Campbell just called and offered a very excellent suggestion.... Before making copies, BE SURE TO REINITIALIZE THE DISK TO BE USED IN THE COPYING PROCEDURE. This will ensure the desired results will be obtained. Use INIT!.) When copying, it is good practice to reinitialize new or previously initialized disks.

LASTLY, magnetic media or disks that contain data DO have a "shelf-life". The media becomes less "charged" with age and if a refresh of old valued programs is not done you may very well find a blank the next time you want to run any "goodies" you have collected. Don't panic however, average shelf-life on quality disks is said to be about five years.

Now that you have reviewed the information above, RAISE YOUR HAND if you are having DRIVE problems!

SURE? (Y/N)

NOTE: Some folks have reported having problems with DYSAN diskettes purchased from computer stores, etc. The problem could be that they are getting 35 track diskettes. They make both 35 and 40 track diskettes with the number 107 on the label , so make sure the diskettes you buy have "40 tracks" written in the lower right corner of the label.

BE:

# Patch PATCH

(HDOS 1.6)

If you have an H8, you can patch the PATCH utility to allow system files to be patched. Three addresses are patched with the front panel, and two are done with PATCH. Here's how:

1) Boot up HDOS with a disk containing PATCH.ABS and enter PATCH as a command.

2) Return to PAM-8 (RTM/0) and use it to change the code at the following addresses:

A) 044.233 330/311
B) 044.354 330/311
C) 046.267 330/311

3) Press "GO" on the front panel.

4) Enter PATCH in response to FILE NAME?

5) Hit RETURN in response to PATCH ID?

6) Hit RETURN in response to PREREQUESITE CODE?

7) Enter 42231 in response to ADDRESS?
Change 312 to 303
Enter CTRL-D
Enter 46267 to ADDRESS?
Change 330 to 311
Enter CTRL-D
Enter CTRL-D to ADDRESS?

8) Hit RETURN in response to PATCH CHECK CODE?

9) Enter CTRL-D in response to FILE NAME?

10) THAT'S IT!

Thanks to Robert Pearce
504 Mc Coys Fork Rd
Walton, KY 41094

# New HUG Software

885-1092 RDT Debugging Tool $ 30.00

Here at last is a really useful debugging tool for HDOS users. RDT (self-Relocating Debugging Tool) automatically moves itself up to high memory when it loads, allowing you to debug programs in the normal user memory area. It gives you control of all memory locations, ports, and 8080 registers. It lets you set breakpoints or single step through your program, and it prints out register values after each step while single stepping so you can trace every action taken by your program.

RDT can work in either the octal (split octal) or hex base, can convert between bases, and can add and subtract in both bases. It includes a mnemonic disassembler that prints out hex or octal addresses and data, mnemonics, and ASCII equivalents. RDT has printer commands that let you send memory dumps, disassemblies, etc. to a hardcopy device. Its disk commands let you load or save files anywhere in free user memory, and it even has cassette tape load and save commands (tape commands will not work on an H89 modified for ORG 0 CP/M).

RDT also makes an excellent patch utility, and can load user or system files for patching. You can patch in hex, octal, or ASCII, and you can check your work with the built in disassembler. Patched files can be given a different name and/or saved on a different drive, providing protection of the original.

RDT includes the complete source code and a printed user manual. It requires an HDOS computer system with at least 32k RAM (to allow sufficient space for debugging). With 48k, RDT can load MBASIC for patching or whatever.

885-1091 GRADE AND SCORE KEEPING $30.00

An all new grading and score keeping system is now available for the H89 or H8 (with H19 terminal) computer systems. This new disk product requires HDOS, MBASIC, 48k of memory and only one disk drive.

The system will maintain score sheets of up to fifteen grades for fourty different students or contestants and up to to twenty classes or events on a single disk.

A minor modification to the actual program will allow the grader system to measure any averaging event, i.e. temperature, speed, etc.

A new score sheet may be started at anytime and students may be added or deleted from the roster. This allows students to join a class late or dropout early. If a student does not complete, or have all scores, his total score is not figured into the class average.

Scores may be entered for all students at once or added individually at a later time. All scores may also be changed at anytime.

Weighting of all scores is available to allow more points for a final than for a weekly quiz. Weights are selectable from 0.1 to 9.9 times the valve of each score and may be changed at anytime to give emphasis on a particular score (final).

When quiz scores are entered a distribution (order) of the scores is provided, this may be either on the screen or a hardcopy may be used for plotting a class curve.

Other printouts include all student scores, with their averages and the class average. A final printout is also provided with the student totals and the class distribution.

885-1088 MBASIC GAMES DISK $20.00

The newest of HUG's games packages will include excitment for all! This disk, designed mainly for the "kid" in all of us, contains an excellent video graphics version of the ever popular LUNAR. This game requires that you land your lunar "module" at 3ft./sec. or suffer the consequences. On a sucessful landing, contact with Houston is established just before your "ship" is launched from the surface of the craterous moon.

Las Vegas style Blackjack is a game designed for two players. This game also features video graphics for dealing the cards. You have the choice of the amount of $ you wish and the amount of the bet up to $500.00 (table limit).

Two interesting games are included for the kids! However, both are fully documented to allow programmers to "play". NIT (Nelan Is Thirsty) and DTC (Deposit The Chair) are small adventure type games written for MBASIC. The reference files included with these games allow construction of your own "adventure"! Both games use video

(Vectored to page 31)

# HUG Products List

| Part Number | Description | | Selling Price |
|---|---|---|---|
| **CASSETTE SOFTWARE** | | | |
| MISCELLANEOUS COLLECTIONS | | | |
| 885-1008 | Volume I | Documentation | $ 9.00 |
| 885-1009 | Tape I | Cassette | $ 7.00 |
| 885-1012 | Tape II BASIC | Cassette | $ 9.00 |
| 885-1013 | Volume II | Documentation | $ 12.00 |
| 885-1014 | Tape II ASM | Cassette H8 Only | $ 9.00 |
| 885-1015 | Volume III | Documentation | $ 12.00 |
| 885-1026 | Tape III | Cassette | $ 9.00 |
| 885-1036 | Tape IV | Cassette | $ 9.00 |
| 885-1037 | Volume IV | Documentation | $ 12.00 |
| 885-1057 | Tape V | Cassette | $ 9.00 |
| 885-1058 | Volume V | Documentation | $ 12.00 |
| UTILITIES | | | |
| 885-1034 | Character Ed | Cassette H8 Only | $ 11.00 |
| 885-1035 | ED/ASM/DEBUG | Cassette H8 Only | $ 11.00 |
| PROGRAMMING LANGUAGES | | | |
| 885-1039 | WISE on Cassette H8 Only | | $ 9.00 |
| 885-1040 | PILOT on Cassette H8 Only | | $ 11.00 |
| 885-1045 | FOCAL Cassette H8 Only | | $ 11.00 |
| 885-1085 | PILOT Documentation | | $ 9.00 |
| AMATEUR RADIO | | | |
| 885-1027 | Morse8 Cassette H8 Only | | $ 14.00 |
| 885-1028 | RTTY Cassette H8 Only | | $ 11.00 |
| **HDOS SOFTWARE** | | | |
| MISCELLANEOUS COLLECTIONS | | | |
| 885-1024 | Disk I | H8/H89 | $ 18.00 |
| 885-1032 | Disk V | H8/H89 | $ 18.00 |
| 885-1044 | Disk VI | H8/H89 | $ 18.00 |
| 885-1060 | Disk VII | H8/H89 | $ 18.00 |
| 885-1062 | Disk VIII | H8/H89 (2 Disks) | $ 25.00 |
| 885-1064 | Disk IX | H8/H89 | $ 18.00 |
| 885-1066 | Disk X | H8/H89 | $ 18.00 |
| 885-1069 | Disk XIII | Misc H8/H89 | $ 18.00 |
| 885-1083 | Disk XVI | Misc H8/H89 | $ 20.00 |
| GAMES | | | |
| 885-1010 | Adventure Disk H8/H89 | | $ 10.00 |
| 885-1029 | Disk II | Games 1 H8/H89 | $ 18.00 |
| 885-1030 | Disk III | Games 2 H8/H89 | $ 18.00 |
| 885-1031 | Disk IV | Music H8 Only | $ 23.00 |
| 885-1067 | Disk XI | H8/H19/H89 Games | $ 18.00 |
| 885-1068 | Disk XII | MBASIC Graphic Games | $ 18.00 |
| 885-1088 | MBASIC Games Disk | | $ 20.00 |
| UTILITIES | | | |
| 885-1019 | Device Driver Disk H8/H89 | | $ 10.00 |
| 885-1022 | HUG Editor (ED) Disk H8/H89 | | $ 15.00 |
| 885-1025 | Runoff Disk H8/H89 | | $ 35.00 |
| 885-1043 | MODEM Heath to Heath H8/H89 | | $ 21.00 |
| 885-1050 | M.C.S. Modem for H8/H89 | | $ 18.00 |
| 885-1061 | TMI Load H8 Only | | $ 18.00 |
| 885-1063 | Floating Point Disk H8/H89 | | $ 18.00 |
| 885-1065 | Fix Point Package H8/H89 Disk | | $ 18.00 |
| 885-1075 | HDOS Support Package H8/H89 | | $ 60.00 |
| 885-1077 | TXTCON/BASCON H8/H89 Disk | | $ 18.00 |
| 885-1079 | HDOS Page Editor | | $ 25.00 |
| 885-1080 | EDITX H8/H19/H89 | | $ 20.00 |
| 885-1082 | Programs for Printers H8/H89 | | $ 20.00 |
| 885-1092 | RDT Debugging Tool H8/H89 Disk | | $ 30.00 |
| PROGRAMMING LANGUAGES | | | |
| 885-1038 | WISE on Disk H8/H89 | | $ 18.00 |
| 885-1042 | PILOT on Disk H8/H89 | | $ 19.00 |
| 885-1059 | FOCAL-8 on Disk H8/H89 | | $ 25.00 |
| 885-1078 | HDOS Z80 Assembler | | $ 25.00 |
| 885-1085 | PILOT Documentation | | $ 9.00 |
| 885-1086 | Tiny Pascal Disk | | $ 20.00 |
| BUSINESS AND FINANCE | | | |
| 885-1047 | Stocks H8/H89 Disk | | $ 18.00 |
| 885-1048 | Personal Account H8/H89 Disk | | $ 18.00 |
| 885-1049 | Income Tax Records H8/H89 Disk | | $ 18.00 |
| 885-1051 | Payroll H8/H89 Disk | | $ 50.00 |
| 885-1054 | SmBusPkg II 3 Disks H8/H19/H89 | | $ 60.00 |
| 885-1055 | MBASIC Inventory Disk H8/H89 | | $ 30.00 |
| 885-1056 | MBASIC Mail List H8/H89 Disk | | $ 30.00 |
| 885-1070 | Disk XIV Home Finance H8/H89 | | $ 18.00 |
| 885-1091 | Grade and Score Keeping | | $ 30.00 |
| AMATEUR RADIO | | | |
| 885-1023 | RTTY Disk H8 Only | | $ 22.00 |
| 885-1052 | Morse8 Disk H8 Only | | $ 18.00 |
| **H11 SOFTWARE** | | | |
| 885-1008 | Volume I | Documentation | $ 9.00 |
| 885-1033 | HT-11 Disk I | | $ 19.00 |
| **CP/M SOFTWARE** (version 1.43) | | | |
| 885-1201 | CP/M (TM) Volumes H1 and H2 | | $ 21.00 |
| 885-1202 | CP/M Volumes 4 and 21-C | | $ 21.00 |
| 885-1203 | CP/M Volumes 21-A and B | | $ 21.00 |
| 885-1204 | CP/M Volumes 26/27-A and B | | $ 21.00 |
| 885-1205 | CP/M Volumes 26/27-C and D | | $ 21.00 |
| 885-1206 | CP/M Games Disk | | $ 21.00 |
| **MISCELLANEOUS** | | | |
| 885-0017 | H8 Poster | | $ 2.95 |
| 885-0018 | H89 Poster | | $ 2.95 |
| 885-0019 | Color Graphics Poster | | $ 2.95 |
| 885-4 | HUG Binder | | $ 5.75 |

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

# Dynamic RAMs and the Z80 Adapter

In the October 1980 issue of Kilobaud MICROCOMPUTING magazine I described an adapter circuit that plugs into the H8 CPU board and upgrades it to a Z80. This circuit works fine with Heath memory boards and other Heath periferals, but it does not work with some of the dynamic RAM boards made by other companies. In this article, I will describe a modification that allows the adapter to work with the Trionyx and DG-32 dynamic RAM boards. The fix may also apply to other dynamic RAM boards, but I do not have data on them. NOTE: The Trionyx mod is for early boards without recent Trionyx fixes. Their Z80 fix may work with my circuit.

On the Trionyx and DG-32 boards there is a circuit that ORs the memory read and write lines to produce a signal that initiates synchronous refreshing of the dynamic RAMS. However, the write pulse from a Z80 is too short to allow both refreshing and writing to occur. The Z80 produces a refresh signal which can be used to refresh dynamic RAMs, but the Trionyx board is set up for use with an 8080 and has no provision to use that pulse. The DG-32 board can be strapped to use the refresh pulse, but we will present a fix that will work with both boards.

The Z80 produces a memory request signal during both reads and writes that is long enough to refresh the Trionyx and DG-32 boards. The only drawback is that it is produced twice during an opcode fetch, once for the fetch and once for the refresh output. If we blank it during the refresh output, we can use it for synchronous refresh of the RAM board.

Figures 1 and 2 show circuits that produce a memory request signal that is blanked during refresh time. The Figure 1 circuit produces a positive output, and the other one produces a negative output. Use the positive output circuit for the Trionyx board, and the negative one for the DG-32 board. The gates are drawn according to their function in these diagrams, so that U2 in Figure 1 is shown as a negative AND gate even though it is a NOR gate.

You can use bus pin 18 to send the output from the memory request circuit to the RAM board. That pin is normally grounded on the CPU board by a strap (the lowest of four straps near the edge connector). Remove the strap and connect the output of the circuit in Figure 1 or 2 to the hole nearest the edge connector.

Figures 3 and 4 show the circuits that produce the synchronous refresh signal on the Trionyx and DG-32 boards. The output of the circuit on your board should be disconnected and replaced with the signal on bus pin 18. The best way to make the change is to remove the output chip (U52 or U8), bend out the output pin, and replace the chip. Then connect a wire from the pad under the pin you just bent out (on the back side of the board) to the pad connected to the edge connector for pin 18. The board should now operate correctly with the Z80 adapter.

PS:

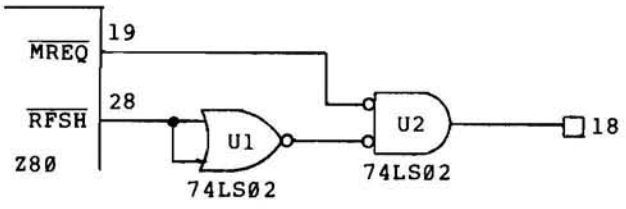



Figure 1. Positive Memory Request

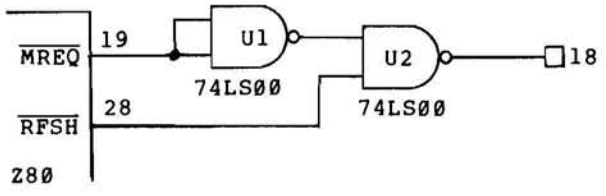



Figure 2. Negative Memory Request

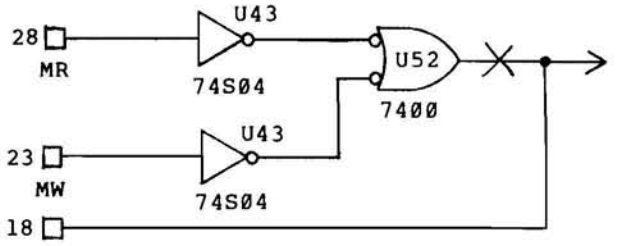



Figure 3. Modification to Trionyx Board

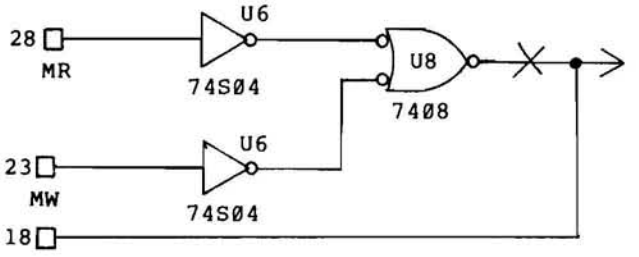



Figure 4. Modification to DG-32 Board

# QSE (Quick Simple Editor)

by Robert Pearce
504 McCoy's Fork Road
Walton, KY 41094

The intent of QSE is to provide a quick simple editor program in BASIC for small file changes. It uses a simple commom sense command format so that its use does not become a learning experience.

Commands may be written out in full or abbreviated to the first character. Trailing delimiters may be omitted. The program will add three delimiters to each command.

Lines are numbered from 1 to N to allow the BASIC line number to be changed (provided a BASIC program is being edited). However, the line numbers do not become part of the saved file.

A new file may be created using QSE by specifying 'NEW' as the file name. Remember when you save it you must give a file name at that time.

With the exception of the 'INSERT', 'APPEND', 'REPLACE' and 'DELETE' commands, the line is displayed after each command.

QSE Command List

APPEND (A) Add new line(s) starting at the end of the file.

BOTTOM (B) Set the line pointer to the bottom of the file.

CHANGE (C) Change string 1 to string 2 on the present line displayed.
Ex. C /STRING1/STRING2/

DELETE (D) Delete the present line.

END (E) End the QSE session. This command may be combined with the save command to save and end in one command.
Ex. E
Ex. E/S Save and End.
Ex. E/S/SY1:QSE.DOC Save file as 'Sy1:QSE.DOC' and end.

FIND (F) Starting at the present line, search the file for the first occurrance of the string1. display the line when it is found.
Ex. F /STRING1/

INSERT (I) Starting at the present line, add line(s) to the file. This command is terminated by entering a 'RETURN' on a new line.

LIST (L) The command 'LIST' is used in severel forms as shown below:
Ex. L List the entire file.
Ex. L/N Display line 'N'.
Ex. L/N/M Display lines 'N' to 'M',
Ex. L/* Display the current line.

NEXT (N) Display the next file line.

PRINT (P) Print the entire file on device LP:.

Q (Q) Change the delimiter. Initially the delimiter will be set to a '/'
Ex. Q/* The delimiter is now an asterisk.

REPLACE (R) Replace the present line.

```
00010 REM       QUICK SIMPLE EDIT (QSE)              This program is in B H BASIC
00020 CLEAR :DIM R$(161),B$(2):PRINT CHR$(12);"QSE"
00030 LINE INPUT "Enter Full File Name ";F$
00040 IF F$="NEW" THEN I=1:GOTO 90
00050 OPEN F$ FOR READ AS FILE #1:FOR I=1 TO 160
```

```
00060 R=CIN(1):IF R=0 THEN CLOSE #1:GOTO 90
00070 LINE INPUT #1,;R$:R$(I)=CHR$(R)+R$:NEXT I
00080 CLOSE #1:PRINT "Incomplete File Load"
00090 N=I:R$(N)="EOF":R$(0)="TOP":D$="/":P=0:IF F$="NEW" THEN 290
00100 LINE INPUT "QSE ";A$:A$=A$+D$+D$+D$
00110 FOR I=0 TO 2:J=MATCH(A$,D$,1)
00120 B$(I)=MID$(A$,1,J-1):A$=MID$(A$,J+1):NEXT I:B=ASC(B$(0))
00130 ON 91-B GOTO 100,100,100,100,100,470,440,390,290,490,500,100,450
00140 ON 77-B GOTO 330,100,100,280,100,100,250,230,220,190,180,520
00150 GOTO 100
00160 PRINT P;R$(P):IF B<> 76 THEN 100
00170 RETURN
00180 P=N:GOTO 160
00190 S=MATCH(R$(P),B$(1),1):IF S=0 THEN PRINT "Not Found":GOTO 100
00200 R1$=MID$(R$(P),1,S-1):R2$=MID$(R$(P),S+LEN(B$(1)))
00210 R$(P)=R1$+B$(2)+R2$:GOTO 160
00220 FOR I=P TO N:R$(I)=R$(I+1):NEXT I:R$(I)="":N=N-1:P=P-1:GOTO 160
00230 IF LEFT$(B$(1),1)="S" THEN B$(1)=B$(2):GOSUB 400
00240 END
00250 FOR I=P TO N:S=MATCH(R$(I),B$(1),1):IF S=0 THEN NEXT I
00260 IF S=0 THEN PRINT "Not Found":GOTO 100
00270 P=I:GOTO 160
00280 IF N=160 THEN PRINT "QSE Table Full":GOTO 100
00290 LINE INPUT "? ";R$:IF R$="" THEN 100
00300 IF B=82 THEN R$(P)=R$:GOTO 100
00310 FOR I=N TO P+1 STEP -1:R$(I+1)=R$(I):NEXT I:N=N+1
00320 P=P+1:R$(P)=R$:GOTO 280
00330 IF B$(1)="" THEN FOR I=1 TO N:P=I:GOSUB 160:NEXT I:GOTO 100
00340 IF B$(1)="*" THEN B=0:GOTO 160
00350 IF VAL(B$(1))>N THEN B$(1)=STR$(N)
00360 IF B$(2)="" THEN P=VAL(B$(1)):B=0:GOTO 160
00370 IF VAL(B$(2))>N THEN B$(2)=STR$(N)
00380 FOR I=VAL(B$(1)) TO VAL(B$(2)):P=I:GOSUB 160:NEXT I:GOTO 100
00390 GOSUB 400:GOTO 100
00400 IF B$(1)<>"" THEN F$=B$(1)
00410 IF F$="NEW" THEN LINE INPUT "Enter File Name ";F$
00420 OPEN F$ FOR WRITE AS FILE #1
00430 FOR I=1 TO N-1:PRINT #1,R$(I):NEXT I:CLOSE #1:PRINT "Saved":RETURN
00440 P=0:GOTO 160
00450 p=p+1:IF P>N THEN P=N
00460 GOTO 160
00470 P=P-1:IF P<0 THEN P=0
00480 GOTO 160
00490 D$=B$(1):PRINT "Delimiter is ";D$:GOTO 100
00500 OPEN "LP:" FOR WRITE AS FILE #1:FOR I=1 TO N-1
00510 PRINT #1,R$(I):NEXT I:CLOSE #1:GOTO 100
00520 P=N-1:GOTO 280
```

EOF

# Hex Output for ASM

This fix causes the HODS assembler ASM issue #104.04.00 to print numbers in the address and data columns in hex instead of octal. Our thanks go to Charles Floto, editor of BUSS -- The Independent Newsletter of Heath Co. Computers for letting us publish this information.

This patch can be done either with DUMP (from HUG part no. 885-1062) or with PATCH as modified in the article PATCH PATCH in this issue. We will present the method for doing it with dump first and include an octal dump of the patch for PATCH users.

Because ASM can be anywhere on your disk, you will have to use DUMP in the file mode first and locate the track and sector containing the relative address 0F24H. Then use DUMP in the disk mode to patch starting at that point as follows:

```
ADDRESS FROM    TO          CODE
  24    C5      F5  NUMO    PUSH    PSW     SAVE BYTE
  25    Ø6      ØF          RRC             MOVE HIGH NIBBLE DOWN
  26    Ø3      ØF          RRC
  27    A7      ØF          RRC
  28    17      ØF          RRC
  29    17      CD          CALL    HEXO    PRINT HIGH NIBBLE
  2A    17      A5
  2B    F5      31
  2C    E6      F1          POP     PSW     RESTORE BYTE
  2D    Ø7      E6  HEXO    ANI     ØFH     ISOLATE NIBBLE
  2E    C6      ØF
  2F    3Ø      6C          ADI     9ØH     CONVERT TO ASCII
  3Ø    77      9Ø
  31    23      27          DAA             A - F CAUSE CARRY
  32    F1      CE          ACI     4ØH     ADD WITH CARRY
  33    Ø5      4Ø
  34    C2      27          DAA              AND ADJUST
  35    AØ      77          MOV     M,A     STORE ASCII DIGIT
  36    31      23          INX     H       GET NEXT BYTE
  37    C1      C9          RET
  38    C9      (not used)
```

When you have entered this patch, the assembler will print in hex, but it will still put a period between the upper and lower halves of an address, which is not usually done in hex notation. To remove the periods, locate relative address ØCE3H and patch in three NOP's there (three zeros).

If you would like to do the patch with modified PATCH, here is an octal dump of the patch, which starts at absolute address 61234 (split octal).

```
Ø61.234  365 Ø17 Ø17 Ø17
Ø61.24Ø  Ø17 315 245 Ø61 361 346 Ø17 3Ø6
Ø61.25Ø  22Ø Ø47 316 1ØØ Ø47 167 Ø43 311
```

To patch out the period in addresses, replace the data with zeros in three addresses starting at 57133. Be sure you have a back-up copy of the assembler before you make these patches in case you do something wrong or in case you need an octal assembler some day.

EOF

## Type-Ahead Buffer

The values given in REMark #11 for HDOS type ahead buffer are not correct. I have listed the correct value below. You will note that I use the SCALL dispatch address at .UIVEC 7 to determine the address for HDOS this address remains constant whereas S.SYSM (Ø4Ø 32Ø Ø4Ø 321) will change if overlays are present to point to the start of the overlay rather than the start of the HDOS resident code.

J.J. Thompson
281 Warren Ave.
Kenmore, NY 14217

Type Ahead Buffer for HDOS 1.6

| Location | Octal | Decimal |
|---|---|---|
| HDOS Starting Address (low byte) | Ø4Ø Ø62 | 8242 |
| HDOS Starting Address (high byte) | Ø4Ø Ø63 | 8243 |
| Offset to line counter byte | Ø12 14Ø | 2656 |
| Offset to Queue tail pointer (low byte) | Ø12 144 | 266Ø |
| Offset to Queue tail pointer (high byte) | Ø12 145 | 2661 |
| Offset to Queue head pointer (low byte) | Ø12 146 | 2662 |
| Offset to Queue head pointer (high byte) | Ø12 147 | 2663 |
| Offset to pointer to buffer start (low byte) | Ø12 15Ø | 2664 |
| Offset to pointer to buffer start (high byte) | Ø12 151 | 2665 |
| Offset to pointer to buffer end+1 (low byte) | Ø12 152 | 2666 |
| Offset to pointer to buffer end+1 (high byte) | Ø12 153 | 2667 |
| Offset to buffer start | Ø12 157 | 2671 |
| Offset to buffer end | Ø12 323 | 2771 |

# MicroNET SYSOP

Ever since I arrived at HEATH, I had been hearing of a MicroNET and a bulletin board which had something to do with HUG. Well, now after seeing and using the HUG Bulletin Board through the services of MicroNET, I have come to realize what a "nifty-neat" way of transferring data; be it information, games or just a means of "chatting" to other users.

With all the changes taking place here at HUG, I became the most likely candidate for undertaking the task of watching over the HUG Bulletin Board. MNET has been pushed aside in the past when other priority matters came up.... however, Bob has informed me that this will not be the case any longer. From now on nothing will be set aside.... I have to stay at work until everything is finished (oh, well). As long as I have to be here I might as well enjoy it.... Right?

That is what I am about to do, however, I need your help. What do you want as a user of the HUGBB? How can HUG make itself more useful through MNET? If you have any constructive ideas, suggestions or comments let me know.... but first I better explain what we are going to try within our time and work frame while remaining realistic to our other obligations.

First, we intend to be monitoring MNET at least three times a week (especially Mondays and Fridays). New users of the HUGBB will not have to wait for several days to be recognized by the system. Questions concerning HUG can be answered without a long wait for a reply by mail. Please note however that the answers may be brief.... but to the point. Remember Users, HUG is not HEATH and many questions concerning HEATH we cannot answer. We will do our best to relay any information, within our means.... but NO PROMISES!!

Our second objective is to provide a new software "toy" by way of MNET once a month. This may be anything from a game to a utility.... whatever happens to be in our library at that time that seems appropriate.

Finally, we will be providing this space in REMark for the MNET user. We will explain what new "toy" we have placed on the HUGBB for the month. Questions concerning MNET and the HUGBB will be brought to your attention as well as other HUG members.

BEGINNERS on MicroNET:

In the next issue of REMark we will begin to explain the "how, when, what, and where's" of MicroNET and the HUGBB.

Now it is time to ask a favor of you.... when you are directing your message(s) to HUG, we would ask that you key in the word SYSOP for us to do a <S>ELECTIVE RETRIEVAL. Jim Blake, JB, Bob, Bob Ellerton, Nancy, HUG, and SYSOP have all been used in the past as key words which makes it very difficult to monitor all of these key words (and sometimes others). Many times you, the user, would key in several words which did make it more likely that we would "see" your message(s), however, can you see how it would create more "worries" besides the time it takes to monitor each key word? So, from this point in time we would appreciate if all users would direct their messages to SYSOP.... this will be a benefit to all of us in the "long run". We thank you very much for your cooperation.... in return we expect a quick turn-around within HUGBB.

Please remember, MicroNET is far from "bug" free.... so between the "bugs", housecleaning, maintenance and generally running around trying to figure out "what went wrong", we will be doing our best to achieve these goals (no comment please). Seriously, I am looking forward to using MicroNET and the HUGBB as an effective way of serving you.... I hope you feel the same.

SYSOP <TLJ>

## H11 Owners — Too Much Time?

Tom Kneisel reported difficulties with the LTC on his H11. He indicated that the clock had been gaining a few seconds per minute. Heath's standard "cure" for this particular situation was the installation of a 470 ohm resistor at location R114 of the power supply. (Heath Consultants tell us that it may be necessary to drop to 150 ohms in some situations. Further, a change of R114 all the way from 8200 ohms to 150 ohms would not always correct the situation.) Tom investigated the problem and found noise spikes and additional "garbage" on the BEVNT line (probably caused by line interference). He added a .001 uf capacitor from Pin 6 of IC103(B) to Ground by paralleling R108 which goes to the input pin of this IC.

Tom writes "The LTC now keeps good time, or at least as good as the power company allows!" --- THANKS for the info Tom. I'm sure our H11 users will appreciate your efforts.

BE:

# The Piece Of Paper

by John Beetem
856 Allardice Way
Stanford CA 94305

The following program produces a "shaggy dog story" when it is run. It will run with any version of Heath BASIC, including version 5 (first cassette BASIC). WARNING: Be prepared to do a lot of reading when you run this program.

```
01000 PRINT "          The Piece of Paper"
01010 PRINT
01020 PRINT
01030 B1=0:B2=0:B3=0:B4=0:B5=0:B6=0
01040 PRINT "   I was walking down the street one day, came"
01050 PRINT "along a piece of paper; couldn't read it, so I"
01060 PRINT "took it to ";
01070 IF B1 THEN 1110
01080 PRINT "the bartender.  He said 'What happened?'"
01090 PRINT "So I said:"
01100 B1=1:GOTO 1040
01110 PRINT "you.  So he said 'Hey, let me read it!'"
01120 PRINT "So he read it, got real mad, and threw me out the"
01130 PRINT "door.  ";
01140 B1=0
01150 IF B2 THEN 1190
01160 PRINT "Then this old lady came along, and she said"
01170 PRINT "'What happened?'  So I said:"
01180 B2=1:GOTO 1040
01190 PRINT "So then the old lady said 'Hey, let me read"
01200 PRINT "it!'  So she read it, got real mad, and screamed"
01210 PRINT "'HELP! MURDER! POLICE!!'  ";
01220 B2=0
01230 IF B3 THEN 1270
01240 PRINT "So then this cop came"
01250 PRINT "along, and he said 'What happened?'  So I said:"
01260 B3=1:GOTO 1040
01270 PRINT "So then the cop said 'Hey,"
01280 PRINT "let me read it!'  So he read it, got real mad, and"
01290 PRINT "threw me in front of ";
01300 B3=0
01310 IF B4 THEN 1350
01320 PRINT "the judge.  The judge looked"
01330 PRINT "at me and said 'What happened?'  So I said:"
01340 B4=1:GOTO 1040
01350 PRINT "you.  So then the judge said"
01360 PRINT "'Hey, let me read it!'  So he read it, got real"
01370 PRINT "mad, and threw me into a cell.  ";
01380 B4=0
01390 IF B5 THEN 1430
01400 PRINT "Well, I had a cell-"
01410 PRINT "mate, and he said 'What happened?'  So I said:"
01420 B5=1:GOTO 1040
01430 PRINT "So then my cellmate"
01440 PRINT "said 'Hey, let me read it!'  So he read it, got"
01450 PRINT "real mad, and started beating me up.  ";
01460 B5=0
01470 IF B6 THEN 1520
01480 PRINT "The guard ran"
01490 PRINT "over to the cell and said 'What happened?'"
01500 PRINT "So I said:"
01510 B6=1:GOTO 1040
01520 PRINT "So then the"
01530 PRINT "guard said 'Hey, let me read it!'  So he read it,"
01540 PRINT "got real mad,";
01550 REM The end of the story is encrypted so as not to spoil it
```

```
01560 REM for the poor sucker who had to type this in.
01570 X=13
01580 FOR I=1 TO 45
01590 READ Y
01600 X=X+X+Y-32
01610 PRINT CHR$(X);
01620 NEXT I
01630 PRINT
01640 STOP
01650 DATA 38,65,-52,-88,-136,83,-94,-65,-74,-168,77,-85,-124
01660 DATA -47,16,25,16,44,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01670 DATA 0,0,0,0,0,52,-64,-43,-74,37,-28,-56
01680 END
```

The Piece of Paper

I was walking down the street one day, came along a piece of paper; couldn't read it, so I took it to the bartender. He said 'What happened?' So I said:

I was walking down the street one day, came along a piece of paper; couldn't read it, so I took it to you. So he said 'Hey, let me read it!' So he read it, got real mad, and threw me out the door. Then this old lady came along, and she said 'What happened?' So I said:

I was walking down the street one day, came along a piece of paper; couldn't read it, so I took it to the bartender. He said 'What happened?' So I said:

I was walking down the street one day, came along a piece of paper; couldn't read it, so I took it to you. So he said 'Hey, let me read it!' So he read it, got real mad, and threw me out the door. So then the old lady said 'Hey, let me read it!' So she read it, got real mad, and screamed 'HELP! MURDER! POLICE!!' So then this cop came

EOF

# H19/H89 Screen Control

by William B. Rothman

If one uses lots of direct cursor addressing, as well as other commands to the VDT (Video Display Terminal), to generate a decent display with an H19 or H89, one will find that apparently random characters appear from time to time on the screen. This is rather annoying, not to say ugly. The cause of this "mung" on the screen can be traced to the setting of the "width" of the terminal in the HDOS device driver to 80 columns, which appears to be the default setting used by Heath. This setting causes HDOS to output a CR/LF upon reaching column 80, which is normally not noticed, since the terminal is printing and receiving carriage returns regularly from programs and users. But direct cursor addressing and other commands to the terminal depend upon having characters transmitted in an unbroken sequence. Should HDOS interject a CR/LF in the middle of a sequence, the command does not function correctly and the characters which are transmitted after the CR/LF are printed as normal ASCII characters instead of being treated as nonprinting commands. The fix to the problem is to SET the terminal driver to a width of 255 characters, which effectively eliminates the CR/LF injection.

I would also recommend that one sets the DIP switch on the VDT board to wrap at the end of lines. Although this change is not as critical, it is often of advantage to the user to have the screen in line wrap mode. An example is the printing by the SoftStuff C.P.S of periods for each line sent over the modem. With the terminal in line wrap mode, one can be assured of seeing a dot for each line, as opposed to wondering what is happening after the eightieth line is transmitted. Further, the two parts, the terminal width and line wrap, are complimentary and should be set in unison.

EOF

# Modifications to HDOS 1.6

By Jack Thompson
281 Warren Ave.
Kenmore, NY 14217

Recently several modifications to HDOS 1.6 have been published which will allow automatic bootup into HDOS without the need to type spaces, or to enter the date if it has previously been entered. I have added modifications to make control S a toggle so that you don't have to alternate between control S and control Q to control text scrolling on the screen. I also have modified HDOS so that control P will toggle the printer on and off with all terminal output also going to the printer when it is toggle on. To make these changes you will need the program DUMP.ABS which is on HUG's Disk VIII (HUG P/N 885-1062). It is recommended that to make these changes you SYSGEN a fresh disk and remove all non-system files, then copy the file DUP.ABS, which is also on HUG Disk VIII, to the disk, make the changes to this disk and then use it as a master to make other system containing disks using the DUP program. (NOTE: Two drives are required for DUP.) This is necessary as some of the changes will not be passed by the normal SYSGEN procedure, in HDOS 1.6.

CAVEATS: Once these changes have been made you will not be able to use the CHECK or IGNORE functions of the boot routine. The control P modification can only be made if the console is on the H8-4 I/O board as the routines for the H8-5 are overwritten. Control P will no longer be avaible for clearing the control O function. The console output routine will no longer replace form feed characters with a series of line feed characters, and to get proper operation of your printer you should set TT: for NOTABS unless your printer understands tabs. When using the control P routine with a printer that buffers its input before printing (such as the H-14 or TI-810), it is possible to have unprinted characters left in the buffer which may print the next time the printer is used normally with a printer driver. To avoid this problem it is recommended that the printer be turned off and back on before using it with a printer driver.

Modifications to HDOS 1.6 made using the program DUMP.ABS from HUG's Disk VIII (HUG P/N 885-1062).

1) Mod to eliminate the need to type spaces. Use the baud rate divisors for the power up baud rate of your terminal. If you change the baud rate after boot-up the terminal will have to be reset before subsequent re-boots.

Track 0 Sector 4

| Location | Old Value | New Value |
|---|---|---|
| 54H | 3A | 21 |
| 55H | 08 | Divisor LS |
| 56H | 20 | Divisor MS |
| 57H | F5 | C9 |

| Baud Rate | Divisor LS | Divisor MS |
|---|---|---|
| 110 | 17H | 04H |
| 300 | 80H | 01H |
| 600 | C0H | 00H |
| 1200 | 60H | 00H |
| 2400 | 30H | 00H |
| 4800 | 18H | 00H |
| 9600 | 0CH | 00H |
| 19200 | 06H | 00H |

2) Mods to eliminate the need to type a "CR" to initiate a boot after baud rate has been determined, and to elimate printing the word BOOT.

Track 0 Sector 0

| Location | Old Value | New Value |
|---|---|---|
| 10H | CD | C3 |
| 11H | 2D | 22 |
| 12H | 25 | 23 |
| A2H | CD | 00 |
| A3H | 2D | 00 |
| A4H | 25 | 00 |
| A5H | 42 | 00 |
| A6H | 4F | 00 |
| A7H | 4F | 00 |
| A8H | 54 | 00 |
| A9H | 80 | 00 |

3) Mod to bypass the need to type a "CR" if the date has been previously entered into the system.

Track 2 Sector 0

| Location | Old Value | New Value |
|---|---|---|
| 4FH | 20 | 3A |
| 50H | A8 | A0 |
| 76H | C3 | C9 |
| 8DH | 29 | 7F |

4) Mod to eliminate printing of the message "Type spaces to determine BAUD RATE" when rebooting the system. Instead issue a command to clear the screen of an H19/H89.

Track 2 Sector 9

| Location | Old Value | New Value |
|---|---|---|
| 6BH | 54 | 00 |
| 6CH | 79 | 00 |
| 6DH | 70 | 00 |
| 6EH | 65 | 00 |
| 6FH | 20 | 00 |
| 70H | 73 | 00 |
| 71H | 70 | 00 |
| 72H | 61 | 00 |
| 73H | 63 | 00 |
| 74H | 65 | 00 |
| 75H | 73 | 00 |
| 76H | 20 | 00 |
| 77H | 74 | 00 |
| 78H | 6F | 00 |
| 79H | 20 | 00 |
| 7AH | 64 | 00 |
| 7BH | 65 | 00 |
| 7CH | 74 | 00 |
| 7DH | 65 | 00 |
| 7EH | 72 | 00 |
| 7FH | 6D | 00 |
| 80H | 69 | 00 |
| 81H | 6E | 00 |
| 82H | 65 | 00 |
| 83H | 20 | 00 |
| 84H | 42 | 00 |
| 85H | 41 | 00 |
| 86H | 55 | 00 |
| 87H | 44 | 00 |
| 88H | 20 | 00 |
| 89H | 52 | 00 |
| 8AH | 41 | 00 |
| 8BH | 54 | 1B |

5) Mod to allow Control S to toggle screen scolling.

Track 2 Sector 6

| Location | Old Value | New Value |
|---|---|---|
| A6H | 7F | FF |

6) Mod to allow Control P to toggle the printer on and off. the values in this modification are the Heath's default LP port E0 hex (340 octal) and Baud rate of 4800.

a) Mod to make Control P toggle a Printer enable flag.

Track 2 Sector 6

| Location | Old Value | New Value |
|---|---|---|
| A0H | FE | FF |
| A1H | 00 | 08 |

b) Mod so that the printer port and the CRT port will both be initialized on Boot-up (these values are noted in parenthesis).

Track 1 Sector 4

| Location | Old Value | New Value |
|---|---|---|
| 33H | 3A | 21 |
| 34H | E3 | 18(Baud Divisor LS) |
| 35H | 20 | 00(Baud Divisor MS) |
| 36H | FE | AF |
| 37H | 01 | D3 |
| 38H | CA | E1(Printer port +1) |
| 39H | 5B | 3E |
| 3AH | 29 | 10 |
| 3BH | 3E | D3 |
| 3CH | 81 | E4(Printer port +4) |
| 3DH | D3 | 3E |
| 3EH | FB | 80 |
| 3FH | D3 | D3 (same) |
| 40H | FB | E3(printer port +3) |
| 41H | D3 | 7D |
| 42H | FB | D3 |
| 43H | D3 | E0(printer port) |
| 44H | FB | 7C |
| 45H | 3E | D3 |
| 46H | 40 | E1(printer port +1) |
| 47H | D3 | 3E |
| 48H | FB | 03 |
| 49H | 3A | D3 |
| 4AH | D7 | E3(printer port +3) |
| 4BH | 20 | DB |
| 4CH | E6 | E0(printer port) |
| 4DH | 08 | 3E |
| 4EH | 07 | 6E |
| 4FH | 07 | CD |
| 50H | 07 | 2B |
| 51H | 07 | 00 |
| 52H | F6 | DB |
| 53H | 4E | E4(printer port +4) |
| 54H | D3 | E6 |
| 55H | FB | EF |
| 56H | 3E | D3 |
| 57H | 15 | E4(printer port +4) |
| 58H | D3 | 00 |
| 59H | FB | 00 |
| 5AH | C9 | 00 |

c) Mod to allow output to the printer if flag is set.

Track 2 Sector 7

| Location | Old Value | New Value |
|---|---|---|
| 67H | FE | F5 |
| 68H | 0C | 3A |
| 69H | CA | DA |
| 6AH | 92 | 20 |
| 6BH | 36 | E6 |
| 6CH | F5 | 08 |
| 6DH | 3A | CA |
| 6EH | E3 | 82 |
| 6FH | 20 | 36 |
| 70H | FE | DB |
| 71H | 01 | E6(printer port +6) |
| 72H | CA | E6 |
| 73H | 82 | 10 |
| 74H | 36 | C2 |
| 75H | DB | 70 |
| 76H | FB | 36 |
| 77H | E6 | DB |
| 78H | 01 | E5(printer port +5) |
| 79H | CA | E6 |
| 7AH | 75 | 20 |
| 7BH | 36 | CA |

| | | |
|---|---|---|
| 7CH | F1 | 70 |
| 7DH | D3 | 36 |
| 7EH | FA | F1 |
| 7FH | C3 | F5 |
| 80H | 8C | D3 |
| 81H | 36 | E0(Printer port) |

d) Mod to the relocation table so that the above routines will be properly relocated when HDOS boots up.

Track 3 Sector 7

| Location | Old Value | New Value |
|---|---|---|
| E3H | 6A | 6E |
| E5H | 73 | 75 |
| E7H | 7A | 7C |
| E9H | 80 | 00 |
| EAH | 36 | 27 |

EOF

# DUP Update

The DUP program (from HUG part no. 885-1062) has a bug in it that has been brought to our attention lately. This bug causes system disks that have been made by DUPing onto a new, unused disk to show soft errors when stat is run. It usually only shows up on systems using Seimens drives. This bug is in DUP version 2.0 -- it has been corrected in 2.1. If you have version 2.0, use PATCH to make the following changes to DUP.ABS. Note: These patches over-write the code that prints the word "Done" at the end of copying.

| Address | New data |
|---|---|
| 042.302 | 061 |

```
047.031  042 115 053 052 112 053 044
047.040  042 112 053 076 012 224 302 323
047.050  046 315 235 036 072 242 040 323
047.060  177 373 052 112 053 054 046 000
047.070  042 112 053 076 050 275 312 123
047.100  047 072 135 050 247 304 064 050
047.110  076 001 315 053 000 315 171 040
047.120  303 176 046 315 372 052 000
```

These patches are for the .ABS file as supplied on the HUG disk. It contains some routines that were left out of the assembly source on the disk, due to lack of space. Those routines control the H8 leds to display the current track being copied, etc. Do not insert the above patches into an .ABS file made from the disk source. Here are the patches for the source, if you wish to re-assemble it. Under MISC EQUATES at the beginning of the source, insert this definition:

```
DLY     EQU     53A
```

A few lines below the label DUP, change the line

```
        DB      12Q,'DUP - Version 2.0'
```

to

```
        DB      12Q,'DUP - Version 2.1'
```

Under the label IDS3, before the line SHLD BUFPTR, remove the following two lines:

```
        LDA     R.DVCTL
        OUT     H17DSK+3
```

A few lines lower down, before the EI instruction, insert these lines:

```
        CALL    WHD
        LDA     R.DVCTL
        OUT     H17DSK+3
```

Under the label TKDONE, before the line CALL R.MAI, add these lines:

```
        MVI     A,1
        CALL    DLY
```

This completes the patches. There may be a misprint in your source. If under the label SAFETY you find the line SCALL .VER, change it to SCALL .VERS.

PS:

---

HUG BUGS:

In the article "Build Your Own EPROM Programmer" (REMark issue #12), on page 21, the line that reads:

```
        MOV     A,H     Subtraction
```

should read:

```
        MOV     A,D     Subtraction
```

Jim Gillogly, 2520 Chard Ave., Topanga, CA 90290 was the author of "Set HDOS Stand-Alone" on page 13 of REMark issue #12.

Is-My-Face-Red dept: We get goodies from all over, and sometimes don't check the source closely enough. The program in REMark issue #12, page 15 was first published in Byte, Feb. 1979, page 154. Our appologies to Byte.

If you use ED.COM on CP/M Version 2.2 as supplied by Heath, any lower case letters you type in will be converted to upper case (not while typing, though) unless you give the command -UC and then use lower case letters to invoke the commands (for example, "i" for "insert").

# Non-Heath Products for Heath Computers

Whitesmiths C for HT-11

An HT-11 version of the Whitesmiths C compiler and library is now available from Dr. Paul W. Abrahams, Consulting Computer Scientist, 214 River Road, Deerfield, Massachusetts 01342. Whitesmiths C supports all facilities of the complete C language, including bitfields and defined types. The run-time library is more complete and systematic than that described in B. Kernigan and D. Richtie, The C Programming Language, Prentice-Hall, 1978, which is the accepted reference for C. The compiler operates in three sequential passes and then calls the assembler; it can replicate itself in 20k words of memory under HT-11. A binary subliscense for the system can be obtained for $640, which includes the compiler, run-time library, installation and operating instructions, the users' manual and system interface manual, and the distribution diskette.

The Software Toolworks

The Software Toolworks offers the following programs for H8 and H89 computers using HDOS. Unless otherwise specified, all programs will run in 32k of ram. You can order them from The Software Toolworks, 14478 Glorietta Drive, Sherman Oaks, CA 91423. Add $2.00 per order for first class postage. California orders add 6% sales tax. Overseas orders add an additional $1.50 per program for air mail.

FULL SCREEN EDITOR. PIE 1.5 text editor uses the H19 or H89 screen as a window into a file. Cursor motion keys allow changes to be typed anywhere on the screen. Function keys perform character and line insert and delete, string search, move and copy single or multiple lines, and scrolling of text in the window. Powerful macro capability provides search and replace, more. For H89 or H8+H19. PIE 1.5 Editor: $29.95
(PIE is a registered trademark of Programma International Inc.)

TEXT FORMATTER. By Dr. Jim Gillogly. Performs fill and justification of text. Page numbering, headers, footers, hanging indents, centering, underlining. Inclusion feature allows insertion of other files into document; supports resetting disks. Combine with PIE 1.5 (above) for word processing. TEXT: $34.95.

C COMPILER. C/80 compiler for a large subset of the C programming language. Supports character and 16-bit integer data, pointers, arrays and strings, macros, data initialization, a full complement of arithmetic and logical operators, and all C control statements. Lacks structures, pointers to pointers, and long and floating point data types. Includes source for sample utility programs, and standard C library. Documentation includes language summary that complements Kernighan and Ritchie's The C Programming Language (not included). C/80: $39.95.

FULL SCREEN GRAPHICS EDITOR. ED-A-SKETCH is a tool for the creation and editing of pictures and graphics using the H19 or H89 graphics character set. Cursor motion keys position the cursor at any point on the screen, allowing typing of regular or graphics characters in normal or reverse video. Area operations may be performed on any rectangular area of the screen, such as erase, fill with a character, invert video, and pick up and move. Eight disk formats make displaying saved pictures easy from BASIC, MBASIC, assembler, C, or any other language. Special "relative mode" saves picture segments for display at any screen position. By Gail Halverson. Requires H8-H19 or H89. ED-A-SKETCH: $29.95.

MACRO ASSEMBLERS FOR Z80 AND 8080. UVMAC is an absolute macro assembler. 8080 version accepts same source files as ASM, but includes macro capabilities. Z80 version accepts full Z80 instruction set (Zilog mnemonics). Both support XTEXT, IF, listing control, etc., and produce .ABS files. Selectable octal or hex listing. UVMAC (specify 8080 or Z80): $29.95.

MODEM AND FILE TRANSFER PROGRAM. REACH turns the H89 into a remote timesharing station. H89 acts as a dialup terminal, transfers files between H89 and remote computer, and spools from remote computer to H89 printer. Operates at speeds up to 9600 baud, full or half duplex. Supports communication between two H89's, XOFF-XON protocol for IBM equipment. Requires serial I/O port. For H89 only. REACH: $19.95

LISP INTERPRETER. Based on the INTERLISP dialect, LISP/80 offers over 75 built-in functions, operations. Comes with a simple editor, file librarian, and formatted expression print routine, all written in LISP, and a 32 page manual. Includes two artificial intellegence programs: a guessing game that learns as it plays, and a simple version of the famous ELIZA psychiatrist program. By Walt Bilofsky. Requires 40k. LISP/80: $39.95

FILE COMPRESSION AND ENCRYPTION. Two program package saves disk space and provides security for sensitive data. PACK uses Huffman coding to compress files, saving 25-50% on text and program source. CRYPT takes a user-provided password and employs a sophisticated Tausworth-Lewis-Payne based cipher algorithm to protect files against unauthorized readers. Written by Dr. Jim Gillogly. PACK and CRYPT both for: $24.95.

FLIGHT CONTROLLER GAME. By Dr. Jim Gillogly. AIRPORT tests your skill as an Air Traffic Controller. Your radar screen maps the moving planes as you direct them along air lanes and into and out of airports. Adjust length of game to increase challange. Requires H89 or H19. AIRPORT: $19.95

CHAMPIONSHIP CHESS PLAYER. MYCHESS, best micro chess game in the 1979 ACM North American Computer Chess Championship, and winner of the 1980 West Coast Computer Faire over such opponents as Sargon 2.5 and Atari, is now available under HDOS (Z80 only). It plays varying openings from a "book" of over 850 moves. Optional tournament time control. Can display suggested best line of play, record moves on a printer. By Dave Kittinger. Requires H89 or Z80 CPU, 40k. MYCHESS: $34.95

The BUSS Directory

The BUSS Directory is the most complete list available of suppliers of hardware and software for Heath computers (H8, H88/H89, H11). It is available from BUSS: The Independent Newsletter of Heath Co. Computers, 325 Pennsylvania Ave. S. E., Washington, DC 20003 for $7.50. It is also included free with a subscription to BUSS ($18 for 12 issues, $25 for 18 issues, $30 for 24 issues. Overseas airmail: 12/$25; 18/$35; 24/$45). Subscriptions can start with the current issue or available back issues (about 15).

SYSMOD

SYSMOD is a program that modifies HDOS 1.6 for easier operation without any patching on your part. Here are some of the changes it makes: Commonly used commands may be abbreviated, for example, CAT SY0:, CAT SY1:, etc. becomes C0, C1, etc.; DISMOUNT SYn becomes Dn; RESET SYn becomes Rn; RENAME becomes REN; DELETE becomes DEL; TYPE becomes TY; COPY becomes CO. PIP is made RAM resident in the command mode so commands that use it are faster. A new LAB command lets you change disk labels. You can suppress the date request on boot-up. For ordering information, contact Jim Teixeira, 62 Churchill St., Sudbury, MA 01776.

# Local HUG News

HUG has recieved information from some of the local user clubs. If your club has not been mentioned, or if the club is sponsoring a special activity please let us know as quickly as possible (at least two months in advance) so that we can "hold the presses" for coming issues of REMark. Send your "data" to the HUG Manager c/o The Heath Users' Group.

THUG (Toronto Canada) located at the Heath Electronic Centre, 1478 Dundas Hwy. East, Mississauga, Ontario L4X-2R7 meets on the last Thursday of every month except July, August, and December from 7:00 PM to 10:00 PM. Interested individuals should contact Keith Boone or Bill Smith. [Phone (416) 277-3191]

SPOHUG (Spokane Heath Users' Group) produces a monthly news letter for members. For additional details, individuals should contact Charles Ballinger, RFD 1 Box 676, Spokane, Washington. Chuck can be contacted directly by calling (509) 448-9727.

The San Diego Heath Users' Group meets on the first Wednesday of the month at 7:00PM. Meetings take place at the Parkway Junior High School, 9009 Park Plaza Drive, La Mesa, California. For further details write San Diego Heath Users' Group, 12202 Kingsford Ct., El Cajon, CA. 92021.

THUG (Toledo, Ohio) has general meetings at individual members homes on the last Sunday of every month. Meetings are held at 8:00PM. Additional information can be obtained from the President of this group, John F. Priebe, 4804 Mt. Airy Road, Sylvania, Ohio 43560. John can be reached at (419) 882-3626. He informs us that they also publish a monthly newsletter.

CCCC (Champaign County Computer Club) supports East Central Illinois. The mailing address for further information is CCCC, c/o Dickson Lum, 1618 Twining Drive, Rantoul, Illinois 61866. Direct contact with CCCC by calling Dick at (217) 893-8916.

HUGO (HUG Ottawa) meets every Wednesday night at 8:00 PM at the Ottawa Heathkit store, 866 Merivale, Ottawa, Ontario K1Z 5Z6. The president is Brian Fultz.

An H11 users' group is being formed in the southern California area. Those interrested should contact Dr. Mike DiGirolamo at (714)886-4766 (office) or (714)793-1470 (home), or Martin Greenberger at (213)937-2377 (office) or (213) 645-6428 (home).

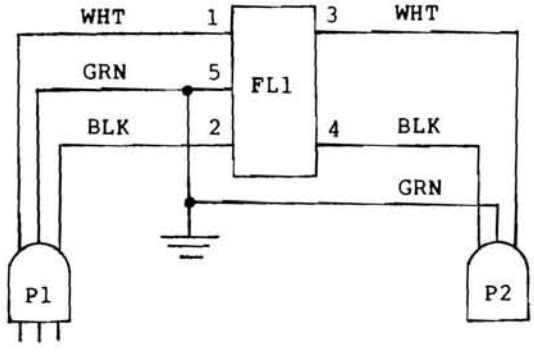
# BUGGIN' HUG


Many of you have returned interesting articles and "hints" for inclusion in REMark only to find your material missing from the next printing. We, at HUG, are now establishing a "Sort" method to allow printing of as much material as possible. Since we compile material for print roughly a month in advance, it is necessary to have some patience when expecting your "neat stuff" to appear. Further, if we recieve more than one letter on the same topic, we may combine input to produce a larger article to cover several possibilities. As a suggestion, you should indicate the type of information and a general catagory your documentation falls under (i.e. BASIC tricks, tips for other users, utilities, problems and cures, etc.). When you supply data in this fashion, we can quickly put the information in a REMark folder for printing in future issues with like material to everyones advantage!

Thanks to all! BE:

Dear HUG,

I have put together a circuit to reduce TVI (television interference) generated by my H89. Using this circuit, I've reduced the TVI by about 75%. The circuit is shown below.




The parts required are:

FL1 Heath part no. 150-90
P1 AC Male 3-prong plug
P2 AC Female 3-prong plug

Mount the filter in a suitable case (I used a 4.75" by 2.56" by 1.4" case, Radio Shack #270-222). With this size case, the lugs on FL1 must be bent straight (be very careful not to break off the lugs). Use 6" or less wire on P1 and P2. For best results, use a metal case.

Robert Juranek
9411 Marguerite Apt. 8
Plymouth, MI 48170

Editor's Hint: Mount the filter in a Heathkit Multi-outlet Box (catalog no. HD-1360 or HD-1274), and filter the lines from all your computer equipment. It's not quite as effective as using a separate filter for each piece of equipment, but it still helps a lot. There is room to mount the filter on the bottom plate of the outlet box. You can strip the insulation back further on the line cord to provide the extra wire needed.

PS:

Dear Gerry,

I found two bugs in my Check Validation Program for ET3400 Tiny BASIC that you printed in REMark. The changes are listed below. The first bug I found involves the printing of a zero when the cents field in the balance is less than 10. The second concerns the entering of negative amounts less than one dollar. Since you cannot enter a minus zero, you must enter the cents with a minus sign. Example: enter minus one dollar as -1,00 and minus ten cents as 0,-10.

Changes to the program. Add these lines:

```
    35 IF C<0 GOTO 50
    125 IF Y<10 GOTO 205
    205 PR"        BALANCE  $";X;".0";Y
    210 GOTO 25
```

Change line 80 to read:

```
    80 IF Y=0 GOTO 205
```

George I. Brown, C.D.P.
2428 Eck Drive
Raleigh, NC 27604

Editor's Note: I also had to make these changes. Change line 100 to read:

```
    100 IF Z<0 GOTO 125
```

Add this line:

```
    155 IF Y<10 GOTO 205
```

Remove line 70. Note that this program cannot handle a negative balance.

PS:

(Vectored from page 16)

graphics to display a "magic map" that allows you to track your movement as you play.

Bowling is a display of the computers capability to keep score for a typical game of random "rolls" of the bowling ball. The game handles up to five players and uses video graphics to simulate the normal score sheet used by most bowling establishments. Guess is another random game where players are asked to choose a number between 1 and 100. The computer then tells the player if the number selected was higher or lower. Video graphics, averaging, and totals are all components of this package.

ROBOT CHASE is an extensive graphics game which includes "electric fences", a transportation device, and lots of action. Skill levels are selected by the player from 1 to 9. As you become accomplished at "getting away" from the robots, the skill level automatically advances. Also, if you fail, the skill will decrease for your next attempt.

This game package is fun for all, even the programmer! The Games are designed for an H-8, with the H-17 and H-19, or the H-89 using MBASIC (48K RAM req.).

EOF

ATTENTION ADVENTURERS!!! Are you missing a point....Refer to REMark for the results you need. (That's a clue folks!)

# Congratulations! to Larry Towner

Larry Towner of Mt. Laurel, New Jersey was selected as the winner of the HUG's latest software contest. Larry submitted the Transaction Data Management System or, TMS. Selection of a winner for the contest was extremely difficult as each of the programs indicated high skill levels by the individual entrants. It was indeed an honor to assist in the review and selection process.

Larry's TMS package was chosen primarily because he had supplied excellent documentation (more like a book) on the operation of his data management technique. The information supplied with Larry's program would enable the experienced programmer to formulate his own "package" with little left to the imagination. TMS (condensed) will be featured in a future issue of REMark for our mutual benefit. Look for TMS as a coming HUG disk product.

I must say, all of the individuals that contributed programs for this contest should be commended for their efforts. HUG will be producing some of your products, either by an addition to the software library or as featured articles in REMark. We sincerely thank everyone!

BE:

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

CUT ALONG THIS LINE

# HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPIATE BOX AND RETURN TO HUG

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

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

# ROCKET

This programs causes a "rocket" to rise from a silo and then blast off on your H19/H89 screen. It will run in either B H BASIC or MBASIC (see the instructions in the listing.

```
10                      REM     ****** ROCKET VER. 2.00 *****
20 E$=CHR$(27):         REM     Escape Key
30 E1$=E$+"E":          REM     Clear Screen
40 P$=E$+"p":           REM     Enter Reverse Video
50 Q$=E$+"q":           REM     Exit  Reverse Video
60 F$=E$+"F":           REM     Enter Graphics Mode
70 G$=E$+"G":           REM     Exit  Graphics Mode
80 Y$=E$+"Y":           REM     Direct Cursor Addressing
90 X1$=E$+"x1":         REM     Enable  25th Line
100 Y1$=E$+"y1":        REM     Disable 25th Line
110 Y5$=E$+"y5":        REM     Turn Cursor On
120 X5$=E$+"x5":        REM     Turn Cursor Off
130 U$=CHR$(95):        REM     Make U$ = Underline For MBASIC
140 PRINT E1$X5$Y$"7 "
150 PRINT TAB(39)F$"  `   "
160 PRINT TAB(40)P$"r "U$Q$
170 PRINT TAB(39)P$"r---"U$Q$
180 PRINT TAB(39)P$"c    f"Q$
190 PRINT TAB(39)P$"` U `"Q$
200 PRINT TAB(39)P$"` S `"Q$
210 PRINT TAB(38)P$"r` A `"U$Q$G$
220 PRINT X1$Y$"84 Hit RETURN to launch rocket";
230 :
240 REM                 REMOVE next TWO lines if using Benton Harbor BASIC
250 A$=INPUT$(1):       REM             One character input for MBASIC
260 GOTO 280:           REM             By-passes PAUSE when in MBASIC
270 PAUSE:              REM             One character input for BH BASIC
280 REM                 ****** Build Rocket and Raise from Silo *****
290 PRINT Y1$Y$"7F";
300 FOR I=1 TO 24:PRINT TAB(39)F$"`````":FOR J=1 TO 25-I:NEXT J:NEXT I
310 PRINT TAB(39)"}```|":PRINT TAB(40)"````"
320 PRINT TAB(40)"}`|":PRINT TAB(41)"`"          Thanks to  Rick Comito
330 FOR I=1 TO 24:PRINT G$TAB(23)" ":NEXT I                62 Haviland Ave.
340 PRINT Y5$                                              Lynn, MA 01902
```

---

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085





**885-2014**