



# REMark

Issue 9 • February 1980

Official magazine for users of Heath computer equipment.

## COVER

"Your Editor returning from China."

## on the stack

>CAT

<b>Welcome to a New Decade</b> .....	3
Jim Blake	
<b>BASIC — "I Built My Own"</b> .....	4
Patrick Swayne	
<b>Software Review</b> .....	9
Fred Pospeschil	
<b>H19 and H89 Graphics in BASIC</b> .....	9
:HUG:	
<b>Single Drive HASL8 Assemble</b> .....	11
Jim Scott	
<b>Connecting Multiple RS-232 Devices in Parallel</b> .....	17
/AIWZ/	
<b>Consultant's Corner</b> .....	19
Staff	
<b>Buggin' Hug</b> .....	20
<b>KISS (Keep it Simple Stupid)</b> .....	28
Jim Blake	
<b>Million Dollar Check Writer</b> .....	29
Jim Tennant	
<b>ET-3400 Morse Code Reader</b> .....	30
Louis C. Graue	

## NOTICE CANADIAN MEMBERS

To avoid excessive shipping costs and to provide more expedient delivery to Canadian members, you may now place your orders for HUG software directly with Heath Canada at 1480 Dundas East Highway, Mississauga, Ontario Canada L4X 2R7.

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

	U.S. Domestic	Canada & Mexico	International
Initial	\$14	\$16	\$24
Renewal	\$11	\$13	\$18

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

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

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

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

HUG Manager and Editor ..... Jim Blake  
Graphics ..... Ron Hungerford  
HUG Secretary ..... Susan Gilfoyle  
Software Developer ..... Gerry Kabelman

Copyright © 1980, Heath Users' Group

HUG is provided by Heath Company as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Heath equipment. As such, little or no evaluation of the programs in the software catalog, REMark or other HUG publications is performed by Heath Company, in general and HUG in particular. The prospective user is hereby put on notice that the programs may contain faults the consequences of which Heath Company in general and HUG in particular cannot be held responsible. The prospective user is, by virtue of obtaining and using these programs, assuming full risk for all consequences.

REMark

# WELCOME TO A NEW DECADE

The cover shot on REMark 7 doesn't even hint that years ago there stood the famous Silver Beach amusement park. Families from all over the country would spend the weekend there building sand castles on the beach while the brush of their imagination painted colorful memories in their mind. From blocks away you could hear the cheers of children as they broke over the top of the rickety roller coaster and as you got closer, the familiar smell of the hot dogs, pop corn and the fresh lake breeze would fill the air. One day 'Old Chief', as he was known in the Carnie business, drove up to my house in his pickup with this big 8 foot picnic table and Bar-B-Q grill, seeking a home for the last remains of Silver Beach. We sat down on the table and he began to re-live the past 50 years in the amusement business. He didn't need a computer. He remembered everything. And back then, I didn't know a bit from a byte.

Today, however, the air is filled with the aroma of steaks on that old grill and to assure that they are cooked the exact amount of time I turn in the direction of the house and say 'TIME'. A pleasant voice acknowledges. (What a strange mixture of technology.) It is my H-99. It manages my life. Just moments ago, I received a letter from a friend on the other side of the world through the earth station concealed in the roof. Guests were announced at the front door by name since a unique code in everyones automobile was transmitted to the computer as they pulled in the driveway. Endless hours of television viewing waits in the family room on tiny disks. One only needs to select from the menu on the huge flat screen built into the wall. Of course, the household environment is precisely controlled, since the computer has collected weather information by satellite and adjusts heating and cooling requirements accordingly. One does not carry cash or credit cards, or worry about paying bills. All income and expenses are handled by the computer and there are no income tax forms to fill out. Greeting cards are automatically sent at the appropriate time, but no one has a mail box. Tonight, the house will be secured, alarms will be armed and monitored both locally and by a regional law enforcement agency. My H-99 communicates with other computers all over the world by satellite. In the morning, I need only to get out of bed and dress. The shower is ready and so is breakfast. I may not go to work, however, since I am linked to my associates by tiny optical fibers and their image appears on the 'phone' beside me. Yes, it is a new decade.

It's going to be a decade of excitement and innovation, technology and advancement. Ideas that seem pure fantasy now will become a part of our daily lives before the arrival of 1990. It will be a different kind of decade - hopefully, quieter and less turbulent than the '60's and '70's . But there will be changes - some very major ones. Living with shrinking energy supplies and resources, coping with inflation and unemployment and adjusting to changing attitudes and lifestyles will put our ability to cope to its toughest test yet.

Right now the slate is clean - there have been endless predictions but nothing has been written yet. No one really knows how the history of the United States will be written for the coming decade. And yet, each of us as individuals and as citizens of this country has the opportunity and the responsibility to contribute to that history through personal responsibility, conservation and achievement. That is the challenge of the '80's'... and those are my thoughts as I sit out here on that old picnic bench from Silver Beach...locked out of my own house. :JB:

# BASIC — “I Built My Own.”

By: Patrick Swayne  
290 Springdale  
Sebastopol, CA 95472

I have always enjoyed my H8 computer, from the time the UPS truck drove up with the kit, through the assembling, testing and learning stages, on to the branching-out stage of adding my own goodies to the H8. But all along I have envied the owners of some of those other micros who were able to use BASIC as soon as they turned on their machines, while I had to wait a minute or two for it to load from cassette. I don't use BASIC as much as most micro owners these days, being an assembly language fanatic, but it's nice to have for games, calculations, etc. So, I set out to put ROM BASIC in my H8.

My first attempt was to get another manufacturer's BASIC and adapt it to my H8. For some strange reason, no one who writes a decent BASIC thinks that the public has a right to the source code, even if it's sold for a price low enough the copying is not worth the trouble. I had to disassemble and decipher the BASIC I chose, and after many weeks of work, it was up and running on a home-made ROM board in my H8. But there were still problems. For one thing, the only way I could save a program on tape was to return to PAM-8, and use the DUMP button, after calculating the necessary addresses. Also, there were features I wanted, but didn't have, and features I had but didn't want.

I decided that the only way to get what I wanted was to do it myself. (I should mention at this point that Benton Harbor BASIC is not ROM-able. Except for PAM-8, all Heath/Wintek software makes extensive use of self-modifying code, which is a no-no in ROM!) Here are some of the goals I set for myself. My BASIC should have some kind of string handling capabilities. It should be able to access ports and memory and call machine language routines. And, hardest of all, it should fit in a mere 3k of ROM space. I imposed the 3k restriction because I wanted my BASIC to be ROM-able on the H8 without adding a ROM board (Even though I have one in my machine.) By making a few simple changes on the CPU board, it is possible to replace the one-k PAM-8 ROM with a 4k 2732 EPROM, with PAM-8 occupying the first k, and my BASIC in the other three. Later in this article, I will explain how to install a 2732 (Intel type) on the CPU board, and offer some hints on making a programmer.

Writing a BASIC interpreter is a formidable task, so I looked for a base to build upon. I found it in Li-Chen Wang's Palo Alto Tiny BASIC. This powerful little BASIC has more features crammed into 2k than I thought possible, and left me with one k for adding features. It is public domain software, and source listings are readily available. Others have used it as a base for their BASIC's, including Cromemco Control BASIC, Bally BASIC (for the Bally Professional Arcade), and, to a lesser extent, TRS80 Level One. The version of PATB that I used is the one in PCC's Reference Book of Personal and Home Computing, edited by Dwight McCade.

My PATB-derived BASIC is called Super Tiny-8, or ST-8. It is provided in a TED-8 produced assembly language listing, with conditional parameters that allow it to be assembled to a number of configurations. By changing a few numbers at the beginning of the listing, you can assemble it for use in ROM on

---

*.... The only way to get what I  
wanted was to do it myself.*

---

the CPU board at low memory, or for ROM use in high memory, or for RAM use starting at the usual 040100 for temporary storage and user programs, while the RAM version temporary storage starts at 054100, following the interpreter. If you decide to assemble ST-8 for high memory ROM use, you may change the starting address from the current E400H (344.000A) to any address above 8000H (200.000A). In addition to location, you have the option of assembling ST-8 for use with the H8-4 or H8-5 interface cards. ST-8 cannot work with more than one terminal port. Since I do not have the H8-4 card, the routine for that card is untested, but it should work. That routine is written for a 600 Baud terminal, but if you desire another rate, you may change the values BAUDL and BAUDH to the appropriate least and most significant divisor latch values as given in Table 2 on page 17 of H8-4 Operation/Service Manual. As supplied to HUG, ST-8 is set to assemble into the RAM version for use with the H8-5. It should be noted that the H8-5 is required for any version, since ST-8 uses cassette mass storage.

Super Tiny-8 is an integer BASIC, allowing numbers in the range -32767 to 32767. Numbers may also be expressed in hexadecimal format by preceding the number with a percent sign (%), in the range 0 to FFFF. ST-8 differs from standard BASIC in a few areas. It uses semicolons (;) to separate multiple statements on a line, and commas (,) to separate items in the same command. (example: INPUT A,B,C; LET D=5, E=7; PRINT F,G,H) As in Benton Harbor BASIC, relational operators may be used in any expression, but a true comparison evaluates to 1, not 65535. In ST-8, command words may be abbreviated.

## Variables

In ST-8 there are 52 variables, designated by the letters A through Z, and A0 through Z0. Variables may not be subscripted (no DIM statement), but there is a continuous array that is automatically dimensioned to use all of the memory space from the end of the user's program to the user-defined high memory limit. This array can be accessed in several different ways, making it very versatile. It can be accessed 16 bits at a time through \$(I), or one byte at a time through &(J), or as strings through \$(K). Strings may be up to 132 characters in length, with the length of the string N in &(K), and the actual string in &(K+1) through &(K+N). This array may also be accessed through POKE and PEEK. This all may seem confusing at this point, but it should become clearer through examples in the rest of this article, and through studying the sample ST-8 programs provided with ST-8. The array is used to replace the dimensioned variables of standard BASIC, for string manipulation, for data storage, and as extra variables if the regular variables are used up.

## Functions

ST-8 has 13 pre-defined functions. Some of them are designated by special symbols rather than words. They are:

ABS(X)	Returns the absolute value of X. It may be abbreviated A.
RND(X)	Returns a random number between 1 and X inclusive. Abbreviation: R.
SIZE	Gives the number of free bytes from the end of the user's program to the high memory limit. Like FRE(0) in standard Benton Harbor BASIC. Abbreviation: S.
SGN(X)	Returns 1 if X is positive or zero, and -1 if X is negative. No abbreviation.
AND(X,Y)	Boolean arithmetic is performed by functions rather than operators in ST-8. The BH BASIC statement Z = X AND Y would be stated Z = AND(X,Y), which returns the boolean AND of X and Y. No abbreviation.
OR(X,Y)	Returns the boolean OR of X and Y. No Abbreviation.
XOR(X,Y)	Returns the boolean exclusive OR of X and Y. No abbreviation.
PEEK(X)	Returns the contents of address X. Abbreviation: P.
IN(X)	Returns the data from input port X. No abbreviation.
LOC	Returns the absolute address of @(0), &(0), and \$(0). (LOC+10) would be the address of @(5), &(10), and \$(10).
-X or <X	Positions the printhead or cursor to the Xth column. It is like TAB(X) in B H BASIC, except that if the cursor is already past the Xth column, a carriage return without a line feed is sent before the cursor is moved.
↑X	Converts the ASCII code X to a string character. Like CHR\$(X) in B H BASIC.
:L	Returns the ASCII value of the string character L. Like ASC("L") in Extended B H BASIC.

## Arithmetic Operators

The following arithmetic operators are available in ST-8, and are listed in order of descending precedence.

*	Multiplication
/	Division
+	Addition
-	Subtraction

The \* and + symbols can also be used for logical (not boolean) AND and OR, respectively. Thus A\*B can mean A AND B, and A+B can mean A OR B.

## Relational Operators

ST-8 supports the following relational operators.

<	Less than
>	Greater than
=	Equal
#	Not equal (not <>, as in B H BASIC)
<=	Less than or equal to
>=	Greater than or equal to

For a logical (not boolean) complement, you may use 0=. 0=A can mean NOT A.

## Command Mode Statements

These commands can be used only in the command mode, not in a program.

LIST	LIST will list your entire program. LIST nn will list starting at line nn. LIST nn,n will list n lines starting at nn. If nn is not found, the next higher number is the starting place. LIST may be abbreviated L., or, since it is the first command in ST-8's command mode table, it may be abbreviated using a period only.
NEW	This command clears the program storage area, and sets LOC to its lowest value. It may be abbreviated N.
WIDTH s,h	ST-8 allows you to set a soft and hard terminal width.
NULL	With this command, you can instruct ST-8 to send nulls after each CR-LF, as required by some printing terminals.
LOCK	LOCK sets an address above which you cannot place program text or array elements.
CSAVE	This command saves your BASIC program text on tape. If you have data in the array area, and wish to save it on tape, you may POKE a new address into TXTEND.
CLOAD	CLOAD loads the program you have saved with CSAVE. It is a modified memory image loader, in that it always loads the file into the text area regardless of the addresses on the tape. This allows compatibility between ROM and RAM versions of ST-8.
SAVE	With this command, you can store a copy of your program in LOCKed memory.
LOAD	LOAD moves a program previously SAVED in LOCKed RAM into the user text area, and overwrites any program already there. LOAD p moves down a program that starts at page p. Abbreviation: LO.

## Program Mode Statements

The commands can be used in the program mode, and all of them except RETURN can be used in the command mode.

- RUN** This command is valid only in the command mode in most BASICs, but in ST-8 it can be used within a program to execute another program that has been SAVED in LOCKED memory. RUN p runs a program at page p.
- STOP** Like RUN, STOP does more in ST-8 than in regular BASIC. IF a program in LOCKED memory is being run from another program, a STOP will return control to the calling program, but if the SAVED program ends without a STOP, control is transferred to the command mode. There is no END statement in ST-8. A program runs to the last line unless a STOP is encountered first. Programs used as subroutines may be nested. A program in the text area can RUN one in LOCKED memory which can RUN a third LOCKED program. A STOP in the third will return control to the second, and a STOP in that one will return control to the first program.
- LET** LET is used to make assignments to variables or array elements. LET A = 5 assigns the value 5 to the variable A. LET is optional, so A = 5 means the same thing as LET A = 5.
- FOR** FOR, TO, STEP, and NEXT work the same as in regular BASIC, except that the STEP must be an integer. If you leave a FOR loop using a GOTO, it remains active and uses some stack space unless a new loop using the same variable is encountered, or the old loop is returned to.
- IF** IF in ST-8 is different from other BASICs in that its argument does not have to be a comparison. Anything that evaluates to 0 is considered false, and anything else is considered true. The expression IF X PRINT X will cause the value of X to be printed if it is not zero. There is no THEN in ST-8.
- GOTO** Unlike regular BASIC, GOTO in ST-8 can be used in the command mode as well as in the program mode. If you don't want to run an entire program, you can GOTO the line where you wish to start. You can initialize variables in the command mode if necessary. GOTO may be abbreviated G.
- GOSUB** GOSUB may also be used in the command mode to try out subroutines in your program. In this case, RETURN gets you back into the command mode. GOSUB nn calls a subroutine at line nn, which must end with a RETURN.
- REMARK** This command is used to introduce a comment, as in standard BASIC. It may be abbreviated REM, with no period.
- POKE** In ST-8, one POKE can insert data into one address or a series of consecutive addresses. The format is POKE (A)=X,Y,..., where A is the address to start inserting data. The value X is placed at address A, Y is placed at A+1, and so on. An entire machine language routine can be placed in memory using one POKE statement, as was done in the "Alarm Clock" program (one of the demo programs sent to HUG with ST-8). Using LOC as a reference point, ST-8 can calculate JMP addresses within your machine language routine, making it relocatable. This also was done in the "Alarm Clock" program. The machine language routine is POKED into memory 100 bytes after the end of the BASIC text, which will be in different places for the RAM and ROM versions, but ST-8 calculates the correct address for the JNZ in each case. POKE may be abbreviated PO.
- OUT** This command is used to send data to a port. The format is OUT (P)=D, where P is the port to which the data D is sent.

- CALL** This is similar to standard BASIC's USR(X), but it is a command, not a function, and is more versatile. Its format is CALL A (B,C,...), where A is the address of the machine language routine you wish to call, and B, C, etc. are the parameters you wish to pass to the routine.
- PRINT** This command allows you to print numbers, quoted strings, and array strings. Quoted strings may be enclosed in either single or double quotes (' or "), but opening and closing quotes must match. ST-8 uses a format control to govern the printing of numbers. It is introduced by a pound sign (#), followed by an argument. If no control is specified, numbers are printed in decimal using 6 spaces. The line, PRINT 1+2 will print the number 3 preceded by 5 spaces. The arguments to the format control may be a number or one or two percent signs (%). A number signifies how many spaces will be used, while percent signs cause ST-8 to print the number in hex, using only the number of columns required for the number.
- INPUT** This command allows you to type in an expression or string to be used by the program. A variable or a two-byte array element may be used to save the expression. You may supply a string prompt, but if you don't, ST-8 uses the variables name or array element designation as a prompt.
- If you enter an illegal character in response to in INPUT, ST-8 DOES NOT CRASH OR RETURN TO THE COMMAND MODE AS WITH MOST BASICS. It prints an error message and repeats your last prompt.

## **SNYNTAX and Line Editing**

Programs in ST-8 consist of numbered lines, as in other BASICs. Line numbers may be in the range from 1 to 32767 inclusive. Each line may have up to 132 characters regardless of terminal width.

If ST-8 is assembled for use in RAM, running it is a simple matter of loading the tape and pressing GO. There are two cold start addresses in the H8-5 version, the second being for terminals requiring two stop bits. The warm start address is the beginning address +6(040106 in the RAM version). To run the ROM version, set the PC register to the starting address and press GO.

As I said before, ST-8 could be placed in a 2732 EPROM along with PAM-8. This is not an article on how to build an EPROM programmer, but I will present a few hints here. Programming any of the 5-Volt only EPROMS is done by first applying 25 Volts to the Vpp pin, and leaving it there through the programming session. Each location in the chip is programmed individually by holding the data and address lines steady for that location while the CE/PGM pin is held high for 50 miliseconds. The trick is to hold the data and address lines steady for that long. One way would be to run them through latches and implement the delay in software. Another way would be to put the processor on HOLD with a one-shot connected to the HOLD line on the bus, while the data and address lines are latched. The programming software would only have to move the data to the address of the EPROM programmer. A read operation may be performed after each write to verify the programming if desired, and the Vpp pin does not have to be lowered for that.

I hope H8'ers, H88'ers, and H89'ers will make good use of ST-8. I think it's a good way to make use of that unused area above PAM-8. I will be sending in more ST-8 programs from time to time, and I hope others will also.

**EOF**



# SOFTWARE REVIEW

By: Fred Pospeschil  
3108 Jackson Street  
Bellevue, Nebraska 68822

John Beetem accurately describes his program as an "easy to use text editor for the H8, designed for use with a video display terminal such as the H9 or H8. It includes most of the features of TED-8, but has better 'defaults', better pattern matching, and a better way to edit characters within a line. Tabs are handled in a way to make assembly language editing much easier. Lines are numbered which gives the user a better feeling of where he is in the file."

The editor is generally a functional duplicate of RED-8 with some notable and important differences. The most valuable of these are described below.

1. Within a line, it is a character editor meaning that you can easily insert or delete characters. When a character is inserted the rest of the line is moved to the right. The delete command deletes the character over the cursor and moves the rest of the text to the left. These capabilities are the ones which make Eddie Baby more efficient, from the users' point of view, than TED-8.

2. Free space remaining is shown, in decimal, on the front panel while the file is being loaded and while text is being added or deleted from the terminal. I found this to be a better approach than the "USE" command in TED-8.

3. The user has the option of using, or not using, line numbers. Again, this is a handy and easier method of working with the text file.

4. In general, the commands are much easier to use than those in TED-8. For example, to print the 11 lines beginning at line 102 simply enter "102P11" (which says to the program "beginning at line 102 print 11 lines").

5. User documentation is quite good. In comparison with some HUG products, such as PILOT, it is outstanding.

6. The program assumes a 16K system but the author has included instructions on how to easily change it, from the front panel, to use more or less memory, different prompt characters, etc.

7. Assembly Language Source Code is included and is fairly well commented although the general program flow and structure is left to the reader to figure out.

On The Negative Side:

1. It does not provide the 'PRINT,NN' option for spacing over the page folds.

2. It does not drive a printer through a 4-port serial board. It works perfectly if your printer is hooked in parallel with your terminal and an H8-5 board. The program does have a 'TRANSMIT' function which the author uses to send the text out to another computer. This module would be a logical place to provide 4-port serial board interface code.

3. The 'IGNORE' response to a tape CRC error does not appear to allow the data read into the memory prior to the CRC or sequence error to be retained. If you have such an error TED-8 can be used to recover most of the data on the bad tape.

4. You cannot smoothly stream a file larger than your memory through the computer as a single file (as you can with TED-8).

If you are going to do any amount of text editing, do yourself a favor and get a copy of this program. Although it has a few weak areas, it will greatly increase your productivity in preparing texts.

EOF

---

## H19 and H89 GRAPHICS IN BASIC

The H19 and H89 both utilize identical displays. The features of the display includes 33 special graphic characters (66 including reverse video), reverse video, cursor control, direct cursor addressing, a 25th line, insert/delete characters or lines, erasing features, and others.

A convenient way to use the graphic features is to create string variables equal to the terminal escape codes which are described in the operation manuals for the H19 and H89. (See the sample program listing on the next page).

For example:

```
E$=CHR$(27)
E$ now equals the character string
of the number 27 which is the ASCII
code equal to the escape key on the
terminal keyboard.
```

Another example:

```
E1$=E$+"E"
E1$ is equal to the erase page function
allowing an erase page to be done by
simply typing PRINT E1$
```

All escape codes may be setup in this format remembering to keep E\$ at the very beginning of the program to prevent having to retype the chr\$(27) for every variable.

A\$ is a list of all possible keys used for graphics.

NOTE: If using MBASIC the underline (\_) will delete the previous character, therefore a CHR\$(95) must be used in MBASIC.

F\$ and G\$ are used to enter and exit the graphics mode while P\$ and Q\$ are used to enter and exit the reverse video mode.

Lines 900 and 901 of the sample program will display all possible graphic characters on the screen. These two lines may be added at the end of a BASIC program to help in creating graphics in a program.

Another very important graphic feature is the direct cursor addressing, in this case Y\$. To use Y\$ a print statement must be used for example:

```
Print Y$;" ,D Center"
```

```
10 E$=CHR$(27): REM
20 E1$=E$+"E": REM
30 K$=E$+"K": REM
40 F$=E$+"F": REM
50 G$=E$+"G": REM
60 Z$=E$+"z": REM
70 P$=E$+"P": REM
80 Q$=E$+"Q": REM
90 X1$=E$+"x1": REM
100 Y1$=E$+"Y1": REM
110 X5$=E$+"x5": REM
120 Y5$=E$+"y5": REM
130 Y$=E$+"Y": REM
140 A$="abcdefghijklmnopqrstuvwxyz~`!\|>C": REM
150 PRINT E1$
900 PRINT F$;A$;G$:PRINT A$
901 PRINT F$;P$;A$;G$;Q$
```

```
10 E$=CHR$(27):E1$=E$+"E":K$=E$+"K":F$=E$+"F":G$=E$+"G":Z$=E$+"z"
20 P$=E$+"P":Q$=E$+"Q":X1$=E$+"x1":Y1$=E$+"Y1":X5$=E$+"x5":Y5$=E$+"y5"
30 Y$=E$+"Y":A$="abcdefghijklmnopqrstuvwxyz~`!\|>C"
40 PRINT E1$
900 PRINT F$;A$;G$:PRINT A$
901 PRINT F$;P$;A$;G$;Q$
```

The Y\$ is followed by a string of characters the first two characters determine the vertical and horizontal position while the remaining characters are what is to be printed. To determine the position where the characters in the previous example are to be printed, locate an ASCII table such as found in the H19 or H89 operation manual and lookup the decimal value for a comma (,) and the letter "D". The comma (,) is equal to 44 and the letter "D" is a 66. The vertical position is the ASCII decimal value of the first character minus 32, in this case the comma (,) represents the vertical position of the 12th line (44-32=12) from the top of the screen. The horizontal position is the ASCII decimal value of the second character minus 32, in this example the letter "D" represents 34 (66-32=34) characters from the left side of the terminal screen.

Another way to use the direct cursor addressing may be done by using the character string function (CHR\$). For example:

```
200 V=44:H=66
210 PRINT Y$;CHR$(V);CHR$(H);" Center"
```

This method allows usage of variables, but it will require more typing if a variable is not needed.

The above direct cursor addressing example prints the word "Center" in the center of the video display.

```
E$ = Escape key
E1$ = Erase Page
K$ = Erase to end of line
F$ = Enter graphics mode
G$ = Exit graphics Mode
Z$ = Power-up reset
P$ = Enter reverse video
Q$ = Exit reverse video
X1$ = Enable 25th line
Y1$ = Disable 25th line
X5$ = Cursor off
Y5$ = Cursor on
Y$ = Direct cursor addressing
A$ = Graphic characters
```

:HUG:

# Single Drive HASL8 Assemble

By: Jim Scott  
1724 Blakemore Rd.  
Richmond, VA 23225

In Issue 5 of REMark (p. 28), John Beetem described his modification to the HASL-8 assembler to allow it to write binary tapes even on a system with only one tape drive. His modification depended on certain locations in PAM-8 which, unfortunately, have changed with the newer software releases designed to support the H8-4 Four-Port Serial Interface. My modification, below, is designed for the latest release, HASL-8 #04.05.00. It requires at least 12K of memory.

The purpose of this modification is to allow a single-tape-drive system to assemble programs whose origin is below the HASL high-memory address. (Most programs would probably fall into this category.) The unmodified HASL cannot assemble such a program directly into memory because it would write over the assembler itself; it cannot create a binary tape on a single-drive system because, except for very small programs, HASL starts writing output (binary) tape records before it has read the last input (source) tape records.

The modified HASL stores the binary file in memory above the high memory address. When the assembly is finished, the source tape is dismounted and the data stored in high memory is dumped onto a binary tape.

This modification eliminates the only need for two cassette recorders with the H8. TED-8 and BASIC are probably easier to use with one drive than with two. And using just one drive eliminates the possibility that head-alignment differences will make it difficult for one drive to read what another has written.

The listing contains instructions for installing and using the modified HASL.

- \* This program, when loaded with a properly configured
- \* and modified HASL-8 #04.05.00, will allow assembly
- \* of large programs with any origin, and will produce
- \* a binary copy on tape even on a system with a
- \* single cassette drive.
  
- \* This program is based on a similar one by John
- \* Beetem, published in "REMark" issue 5, designed for
- \* an earlier issue of HASL-8.
  
- \* Installation Instructions:
- \* 1. Load distribution copy of Hasl-8 #04.05.00, and
- \* configure with high memory = 19199. (NOTE:
- \* 19199 = 113.000 - 1.)
- \* 2. Save a copy of this configured assembler on tape.
- \* 3. Load this copy back in, and write down the final
- \* address displayed on the H8 front panel when the
- \* load is finished.
- \* 4. Change locations 044.317-044.320 from 125,044 to
- \* 030,113.
- \* 5. Change locations 065.202-065.206 from 323,370,303,
- \* 347,002 to 303,004,113,000,000.
- \* 6. Dump a copy of this configured, modified
- \* assembler to tape, as follows:
- \* A. Reset memory display to what you wrote down
- \* in step 3.
- \* B. Set up recorder to record.
- \* C. Press "DUMP".
- \* D. Do not rewind tape when finished.

- \* 7. Enter the following program into memory, using the H8 front panel.
- \* 8. Dump a copy of this program to tape, right after the copy created in step 6, as follows:
  - \* A. change PC (program counter) to 040.100.
  - \* B. Change location 040.000-040.001 to 000,113.
  - \* C. Set memory display to 113.112.
  - \* D. Set up recorder to record right after the assembler copy created in step 6.
  - \* E. Press "DUMP".
- \* Usage Instructions:
  - \* 1. Press "LOAD" twice, to load both the assembler and the program below.
  - \* 2. Reply "Y" to both questions, "BINARY?" and "BINARY TAPE?".
  - \* 3. When the assembler stops after the second pass, remove the input tape and set up the recorder to record the binary copy; then press "GO". (NOTE: The front panel lights will not display anything in particular as the binary copy is being written.)
  - \* 4. When the binary copy has been written, you may press "GO" if you wish to restart the assembler, or you may change PC to 113.030 and press "GO" to write another binary copy.
  - \* 5. If you reply "N" to "BINARY TAPE?", then when the assembler finishes you must set PC to 040.100 before pressing "GO" if you want to restart the assembler.

```

                ORG      113000A

POINT          DW      AREA
COUNT        DW      0

```

\* Simulate Dumping of one Byte.

```

DMPSIM        EQU      *
              PUSH     H          SAVE REGISTERS.
              LHLD     POINT     SAVE BYTE IN AREA.
              MOV      M,A
              INX      H          ADD 1
              SHLD     POINT     TO POINTER.
              LHLD     COUNT     ADD 1
              INX      H          TO COUNTER.
              SHLD     COUNT
              POP      H          RESTORE REGISTERS.
              JMP      002347A    PAM-8 WILL COMPUTE CHECKSUM AND
                                  RETURN TO ASSEMBLER.

```

- \* Dump tape stored in memory onto object tape. This
- \* program is run after the assembly is finished.
- \* DE = # of bytes left.

```

DMPAREA EQU *
        LHLD COUNT DE = COUNT + 2.
        XCHG
        INX D
        INX D
        LXI H,AREA HL POINTS TO AREA
        MVI A,21Q TURN ON TAPE.
        OUT 371Q
DAL EQU *
        IN 371Q WAIT FOR TRANS. READY.
        ANI 1
        JZ DAL
        MOV A,M GET BYTE.
        INX H ADD 1 TO POINTER.
        OUT 370Q OUTPUT BYTE.
        DCX D SUBTRACT 1 FROM COUNTER.
        MOV A,D COUNT = 0?
        ORA E
        JNZ DAL NO, CONTINUE WITH NEXT BYTE.
        OUT 371Q YES, TURN OFF TAPE.
        HLT STOP, AND WAIT FOR "GO".
        LXI H,POINT REINITIALIZE POINT.
        LXI D,AREA
        MOV M,E
        INX H
        MOV M,D
        INX H REINITIALIZE COUNT.
        SUB A
        MOV M,A
        INX H
        MOV M,A
        JMP 040100A

```

\* Dump area - this area must be big enough to save  
the entire object tape.

```

AREA DB 0
      END 040100A

```

EOF

## NEW DDDEF.ACM FOR HDOS VERSION 1.5

```

DVDDEF SPACE 4,10
** DEVICE DRIVER EQUIVALENCES.

DVDFLV EQU 307Q DEVICE DRIVER FLAG VALUE
      ORG PIC.COD STARTS AT PIC CODE AREA

DVD.DVD DS 1 MUST BE DVDFLV, FLAGS TO HDOS AS DRIVER
DVD.CAP DS 1 DEVICE CAPABILITY FLAG
DVD.MUM DS 1 MOUNTED UNIT MASK
DVD.MNU DS 1 MAXIMUM NUMBER OF UNITS
DVD.UFL DS 8 UNIT SUB-CAPABILITY FLAGS FOR UNITS 0-7
DVD.SET DS 1 = DVDFLV IFF DRIVER WILL TAKE SET OPTIONS
      DS 24 RESERVED, MUST BE 0
DVD.STE EQU * ENTRY FOR 'SET' INVOCATION

DVD.ENT EQU 2000A DRIVER ENTRY POINT (MUST BE MULT OF 256)

```

:HUG:

# PROLOGUE FOR MBASIC

Load everything run MBASIC and user programs

HEATH ASM #104.05.00  
17-Dec-79 Page 1

```

*
042.200          XTEXT  ASCII
042.200          XTEXT  HOSDEF
031.136          $TYPTX EQU   31136A
030.252          $MOVE  EQU   30252A          ; Use Romsub
*
*          Get greedy
*
042.200          ORG    USERFWA
042.200          START  EQU    *
042.200 076 000    MVI    A,0          ; Get Overlay 0
042.202 377 010    SCALL  .LOAD0
042.204 076 001    MVI    A,1          ; Get overlay 1
042.206 377 010    SCALL  .LOAD0
042.210 041 362 042 LXI    H,ATDVD      ; Get all the device drivers
042.213 377 062    DB     SYSCALL,.LOADD
042.215 041 356 042 LXI    H,LHDVD      ; Might need the H14
042.220 377 062    DB     SYSCALL,.LOADD
042.222 041 346 042 LXI    H,LDDVD      ; And the Diablo
042.225 377 062    DB     SYSCALL,.LOADD
042.227 041 352 042 LXI    H,LTDVD      ; And the TI-810
042.232 377 062    DB     SYSCALL,.LOADD

*          Set up the stuff on the stack
042.234 041 000 000 LXI    H,0
042.237 071        DAD    SP
000.012          SET    PROBE-MENU+1
042.240 021 366 377 LXI    D,-.
042.243 031        DAD    D
042.244 371        SPHL          ; Push down the stack

042.245 001 012 000 LXI    B,PROBE-MENU+1
042.250 021 366 042 LXI    D,MENU
042.253 315 252 030 CALL   $MOVE          ; Move the stuff into the STACK space

042.256 041 017 043 LXI    H,DRIV          ; (HL) = 'SY1:'
042.261 377 202    SCALL  .MONMS
042.263 041 024 043 LXI    H,DRIV1        ; (HL) = 'SY2:'
042.266 377 202    SCALL  .MONMS          ; Mount syl: and sy2:

042.270 041 000 043 LXI    H,MBASIC        ; (HL) = MBASIC
042.273 377 040    DB     SYSCALL,.LINK  ; Run MBASIC
042.275 315 136 031 CALL   $TYPTX
042.300 007 105 162 DB     BELL,'Error... unable to execute MBASIC',ENL

042.343 257        XRA    A
042.344 377 000    SCALL  .EXIT

042.346 114 104 072 LDDVD  DB     'LD:',0
042.352 114 124 072 LTDVD  DB     'LT:',0
042.356 114 120 072 LHDVD  DB     'LP:',0
042.362 101 124 072 ATDVD  DB     'AT:',0

042.366 040 115 105 MENU  DB     ' MENU/F:5',0 ; Execute user program called MENU
*          ; and tell MBASIC give us five channels
042.377          PROBE  EQU    *-1
043.000 123 131 060 MBASIC DB     'SY0:MBASIC.ABS',0
043.017 123 131 061 DRIV  DB     'SY1:',0
043.024 123 131 062 DRIV1 DB     'SY2:',0

043.031 000        END    START

```

PROLOGUE.SYS

00177 Statements Assembled  
42416 Bytes Free  
No Errors Detected

:JB:

# HDOS DISK LABEL CHANGER

By: M. Duttweiler

```

TITLE ' LABEL CHANGER'
STL ' INSTRUCTIONS'
XTEXT SYDEF
XTEXT SYDEF
LABEL ORG USERFWA

*** LABEL CHANGER PROGRAM
*
* THIS PROGRAM ALLOWS THE USER TO EXAMINE/ALTER THE
* VOLUME LABEL FOR THE SPECIFIED DRIVE.
*
*
* SYNTAX IS: LABEL DEV: NEW LABEL
* LABEL = THE PROGRAM NAME
* DEV: = THE DEVICE & UNIT NUMBER
* NEW LABEL = OPTIONAL NEW LABEL
*
*
* EXAMPLES:
*
* 1. LABEL SY0:
* (EXAMINE THE VOLUME LABEL FOR SY0:)
*
* 2. LABEL SY1:ASSEMBLY DEVELOPEMENT
* (CHANGES THE LABEL FOR THE VOLUME IN SY1: TO
* "ASSEMBLY DEVELOPEMENT")
*
*
* NOTE: THE VOLUME LABEL IS CURRENTLY WRITTEN
* ON TRACK 0, SECTOR 9 (HDOS VERSION 1.0
* AND 1.5). SHOULD THIS BE CHANGED IN THE
* FUTURE, THE LABELS "TRACK" & "SECTOR"
* MUST ALSO BE CHANGED.
*
*
* WRITTEN BY M. DUTTWEILER 30-SEP-79
*
* WITH THANKS TO B. WATZMAN AND J. BLAKE FOR INFO
* CONCERNING TRACK & SECTOR ACCESSING AND COMMAND
* STRING ACCESSING.
*
*
* TRACK EQU 0 LABEL TRACK
* SECTOR EQU 9 LABEL SECTOR
* STL 'MAIN'
* EJECT
** READ COMMAND STRING
*
* LXI H,1 GET STACK ADDRESS+1
* DAD SP
* MOV A,M GET 1ST CHARACTER
* INX H
* CPI 'S' 'SY'?
* JNZ LABSYN
* MOV A,M
* INX H
* CPI 'Y'
* JNZ LABSYN IF SYNTAX ERROR
* MOV A,M GET UNIT NUMBER
* INX H
* STA DEFALT+2 STORE ASCII UNIT #
* SUI '0'
* STA UNITNO SAVE FOR POSSIBLE WRITE
* MOV A,M CHECK FOR COLON
* INX H
* CPI ':'
* JNZ LABSYN IF NOT PRESENT
* PUSH H SAVE STRING ADDRESS

```

```

**      ENSURE THE UNIT EXISTS BY ATTEMPTING TO READ DIRECTORY
*
XRA     A                CHANNEL = 0
LXI     D,DEFAULT        (DE) = DEFAULT BLOCK
LXI     H,FILNAM         (HL) = FILE NAME BLOCK
SCALL   .OPENR           ATTEMPT TO OPEN FILE
JC      SYERR            IF DISK NOT EVEN INITTED!
XRA     A                CLOSE FILE
SCALL   .CLOSE
JC      SYERR

**      READ LABEL SECTOR
*
LXI     H,SECTOR*256+TRACK   TRACK 0, SECTOR 9
LXI     D,BUFFER            BUFFER ADDRESS
LDA     UNITNO              (A) = UNIT NUMBER
LXI     B,1*256            ONE SECTOR ONLY
CALL    DKREAD              READ SECTOR
JC      SYERR              IF READ ERROR

**      READ NEW LABEL STRING
*
POP     H                  (HL) = STRING ADDRESS
LXI     D,BUFFER+17        (DE) = LABEL ADDRESS
MOV     A,M                CHECK FOR NULL LABEL
INX     H
ORA     A
JZ      LABOUT             IF SO, PRINT CURRENT LABEL
MVI     B,61               MAX COUNT = 60
LAB1    STAX D              STORE CHARACTER
INX     D
DCR     B                  TOO MANY CHARACTERS?
JZ      LABOVFL            IF SO
MOV     A,M                GET NEXT CHARACTER
INX     H
ORA     A                  END OF LABEL?
JNZ     LAB1               IF NOT
MVI     A,' '              FILL REMAINING LABEL
LAB2    STAX D              WITH SPACES
INX     D
DCR     B
JNZ     LAB2

**      WRITE NEW LABEL SECTOR
*
LXI     H,SECTOR*256+TRACK   TRACK 0, SECTOR 9
LXI     D,BUFFER
LXI     B,1*256
LDA     UNITNO              (A) = UNIT NUMBER
CALL    DKWRITE
JNC     LABXIT             IF NO ERROR, EXIT

**      READ/WRITE ERROR
*
SYERR   MVI     H,$LF        PRINT ERROR
SCALL   .ERROR
JMP     LABXIT             EXIT

**      PRINT CURRENT LABEL
*
LABOUT  PUSH    D            SAVE LABEL ADDRESS
XCHG
MVI     A,60               FIND END OF LABEL
CALL    $DADA.
MVI     M,$LF+200Q        STORE END OF LINE
POP     H                  RESTORE LABEL ADDRESS
SCALL   .PRINT
*      JMP     LABXIT        EXIT

```



```

**      LABEL EXIT ROUTINE
*
LABXIT  XRA      A
        SCALL   .EXIT

**      SYNTAX ERROR
*
LABSYN  CALL    $TYPTX
        DB     'SYNTAX ERROR!', $LF+200Q
        JMP    LABXIT

**      LABEL OVERFLOW
*
LABOVFL CALL    $TYPTX
        DB     'TOO MANY CHARACTERS IN LABEL (60 MAX)!'
        DB     $LF+200Q
        JMP    LABXIT

BUFFER  DS      256                SECTOR BUFFER
UNITNO  DS      1                  UNIT NUMBER
DEFAULT DB     'SY0',0,0,0
FILNAM  DB     'DIRECT.SYS',0 NO DIRECTORY -- NO LABEL!

        STL    'TRACK & SECTOR ACCESSING'
        EJECT
        LON   C
        XTEXT H17TSA

        END    LABEL

```

EOF

## CONNECTING MULTIPLE RS-232 DEVICES IN PARALLEL

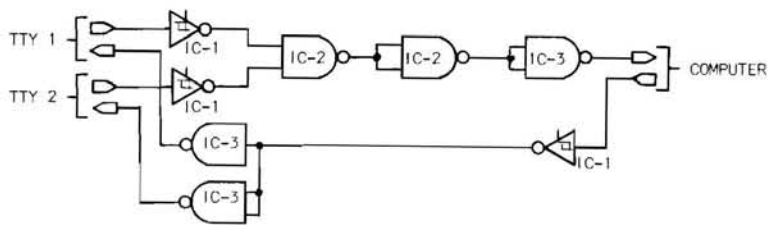
By: /AIWZ/

Have you ever wanted to connect your RS-232 terminal AND your printer to your modem at the same time? Do you want to have several terminals in the house simultaneously connected to the console port? If you tried connecting RS-232 devices in "parallel", you probably realized your mistake as the resulting puff of smoke began to clear. Courage, adventurer. There is a safer way, and it shall be revealed forthwith.

Combining two or more RS-232 signal sources poses 2 problems. First, the output levels go from +12V to -12V. Naturally, if two paralleled devices go to opposite states, the extreme current surge produced will probably damage one of the driving devices (hence the puff of smoke).

The sources must therefore be electrically isolated. Second, the signals must be logically combined so that each will have access to the single output (logical "OR" function). Thus when all other sources are idle, a single active source will appear to be the "only" source to the computer or modem.

There is only one problem in connecting several RS-232 receivers together. Since all receivers on line will get the same signal, there will not be a current conflict in this part of the system. The driver may not be able to handle the multiple loads, however, and a separate driver must be used for each terminal. Now, we will satisfy these requirements with an example circuit.



CHIP	TYPE	HEATH P/N
IC1	75189	443-795
IC2	7400	443-1
IC3	75188	443-794

FIGURE 1

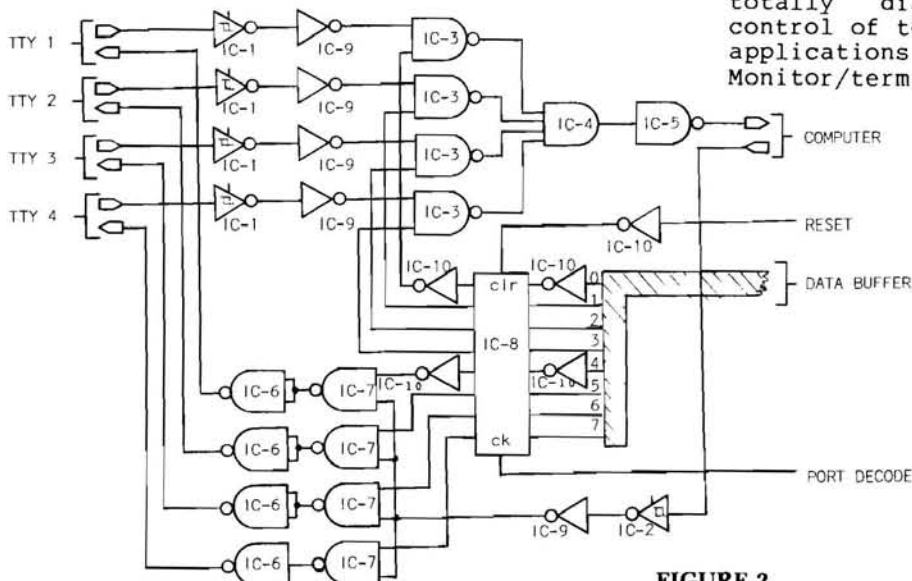
Figure 1 shows a circuit which could be used to allow two terminals to act as console to the same computer, or connect a terminal printer to a modem. IC1 is a standard level converter which produces an inverted TTL output from an RS-232 input. Since RS-232 levels are low when they are idle, the inverted signals will be normally high. By combining these with a logical AND function, the output will follow either input as long as the other remains idle. IC2 provides the AND function (actually Inverted NAND) using a common chip. IC3 then converts back to RS-232 levels and inverts the signal to restore positive logic. In the receiver circuit, IC1 converts and inverts the signal as before, and a separate stage of IC3 restores the logic and drive levels for each terminal.

This circuit could be expanded to accommodate any number of terminals, but only one terminal can be "active" at a time. If a character was typed on two terminals simultaneously, the logic AND would combine the two characters producing a third (incorrect) character. The incorrect character would echo on both terminals, and (hopefully) one operator would allow the other to delete the mistake and continue typing.

This "typist conflict" could be avoided by using one-shot's to detect the first incoming signal and inhibit all other inputs for a period of time, but there is a better and more versatile method. Why not let the computer be capable of selecting a particular terminal for input? While we're at it, we can also let the computer select a terminal or terminal group for output.

Figure 2 shows our basic circuit expanded to accept four terminals, with a latch to select the individual inputs and outputs. IC1 & IC2 provide the incoming level translations, IC3 & IC4 provide the logical combining plus selective gating for input, and IC5 & IC6 provide level translation for output. IC7 allows selective gating for output, IC8 is the gating latch, and IC9 & IC10 provide signal inversion where needed.

As shown, the lower four bits of the latch select which of the terminals is allowed to provide input, and the upper four bits select the terminals to receive output. Since there is a separate bit for each terminal and each function, any combination of inputs and outputs can be selected. For instance, all terminals may be sent a common output, but only one would be allowed to input; or, a single terminal may be enabled, and the others totally disabled. By having program control of terminal selection, many unique applications become possible. Monitor/terminals throughout a building



CHIP	TYPE	HEATH P/N
IC1, 2	75189	443-795
IC3, 7	7400	443-1
IC4	7421	443-50
IC5, 6	75188	443-794
IC8	74LS243	443-805
IC9, 10	7404	443-18

FIGURE 2

could be supplied with individual information updates. Four players could play Multi-Spacewar as individuals or in teams of two. There are many other possible applications, and you will perhaps invent some of your own.

Now, a little more detail about the latch. The clock should be triggered by a port decode signal, so the select code will be accepted when it is written to a particular I/O port (H8-7 owners should use the IOW line). The latch inputs will be connected to a data buffer to read the data from the computer buss (Data In 0-7 on H8-7). To assure that communication can take place after power-up, the latch should be preset from the system's reset line. To accomplish this, the control lines to the console (TTY1) are inverted before and after the latch. This has no net effect on the program control; but when the latch is cleared, TTY1 will be enabled, and all other terminals will be disabled. Since the latch clears on a low signal, the system reset line is also inverted.

The composite circuit could be addressed easily by a BASIC program. Let's assume the port decoded is port 0. The console (TTY1) could be addressed by "OUT 0,17" (17 is decimal for 00010001). TTY2 would be "OUT 0,34". All terminals could be enabled for output, with only TTY1 for input by "OUT 0,241". Other conditions can be set by issuing the "OUT" command with the decimal equivalent of the desired binary pattern.

When programming under assembly, it is preferred that the incoming characters be cleared whenever a new input condition is selected. This can be accomplished in HDOS by "SCALL .CLRCO". This would prevent an incomplete line typed by the previously selected terminal from preceding the new input.

Now that you can connect and control multiple terminals, share your new applications with HUG.

EOF

(continued on next page)



In this column of REMark, the Heath Technical Consultants provide answers to the most commonly asked questions on Heath computer products, both hardware and software.

- Q: How can I list a program on my printer while using Microsoft BASIC?
- A: The "SAVE" command will write the program in memory to any device. For the line printer, the syntax is: 'SAVE "LP:",A'. You must specify "A" (ASCII), or the output will be unreadable. Benton Harbor BASIC users can do the same thing by typing 'REPLACE "LP:"'.
- Q: I removed jumper "J" from my H-11 power supply so the line time clock would operate, but it gains about 5 seconds every minute. Obviously, that is less gain than would occur if the system was expecting 50 Hz AC; so what could be the cause?

### HDOS DISK RESET PROGRAM

Dismount SYn: and ask user for another diskette

HEATH ASM #104.05.00  
17-Dec-79 Page 1

```

042.200          XTEXT  HOSDEF
042.200          ORG    USERFWA
042.200          EQU    *
042.200 041 210 042  START LXI    H,DRIVE      ; HL = point to desired drive
042.203 377 204      SCALL .RESET           ; Ask user for diskette (Get New)
042.205 257          XRA    A              ; Clear (A)
042.206 377 000      SCALL .EXIT           ; Return to monitor. Issue prompt

042.210 123 131 062 DRIVE DB    'SY2:',0   ; Reset SY2:

042.215 000          END    START
: HUG:

```

(continued from previous page)

A: Some H-11 power supply boards have a pair of foil runs that make incorrect connections. These runs connect R111 & R118 to pins 7 & 5 of IC103, and the correct arrangement is shown on the schematic. If you find that R111 connects to pin 5 on IC103, your board will require correction. Without cutting foils, repair can be effected by disconnecting one lead of R111 & R118, and re-installing them in the opposite holes. The resistors will form an "X" pattern, one crossing over the other. The "top" resistor may need a lead extension to reach the hole.

Q: How can I get my H-11 disk BASIC to accept lower case characters?

A: The BASIC can be PATCHed to allow lower case. Make sure that you have the current version of HT-11 (890-6-2).

Then, use this procedure:

```
.R PATCH
FILE NAME-- SY:BASIC.SAV
*546/ 20000 60000
*E
```

The user responses are underlined. In addition to BASIC.SAV, this patch would also apply to XBASIC.SAV and BASIC.FIS. Note that once this patch is implemented, BASIC will not understand commands typed in lower case.

Q: I have a TRS-80 computer and would like to use the H-14 line printer with it. What special information do I need to make these work together?

A: Since there have been a lot of requests for this, the Heath Technical Services department has developed a special application note. This note is available free of charge from the Heath parts department. Order part number 597-2152.

Q: Will you tell me how to get my treasure back from the Troll, and how to get out of the Repository?

A: No.

EOF

## BUGGIN' HUG



Dear Jim:

The H11 Software reference manual on XBASIC states that XBASIC is identical to regular H11 BASIC except for PRINT USING and string arithmetic. This is not true. XBASIC reads data files in a different way. Regular BASIC will read multiple numbers on a file line, but XBASIC will ignore the blank spaces and run the numbers together. I wrote the attached routine to read the lines as a string and separate the numbers when XBASIC must be used with files having multiple numbers on a line (XBASIC will write the numbers just fine)! The routine reads N numbers on a line. If N is 10 or larger, B( ) must be dimensioned.

```
680 FOR I=1 TO N
690 INPUT #2:B$
692 GOSUB 870
694 FOR I1=1 TO 4
696 S(I,I1)=B(I1)
698 NEXT I1
870 D$=""
875 I2=0
880 FOR I1=1 TO LEN(B$)
890 C$=SEG$(B$,I1,I1)
900 IF C$=" " THEN 930
910 D$=D$&C$
920 GO TO 970
930 IF D$="" THEN 970
940 I2=DI2+1
950 B(I2)=VAL(D$)
960 D$=""
970 NEXT I1
980 RETURN
```

Very truly yours,

William A Manly  
Graham Magnetics Inc  
Sub of Carlisle Corp  
1300 Summitt Suite 500  
Ft Worth TX 76102

Dear Sir:

I am very much interested in obtaining an indexing program written in BASIC. I am a novice programmer and as yet do not understand just how to accomplish this. I am primarily interested in indexing several books in my library.

Example: Smith, John 2,3,5,19,27

I can do a simple sort which will put the names under each other, however, I would like the page numbers to follow the name.

I am wondering if you might include such a program in REMark in the near future. I think this might be a useful program for many people.

I would appreciate it if you would announce that there is now a San Jose area Heath Users Group which meets every 1st and 3rd Wednesday of the month, at the Heathkit Electronic Center and 2350 S. Bascom Ave., Campbell CA.

The group has been meeting several months now and we welcome others in the area to join us.

Sincerely,

Gerlene York  
1992 Borchers Dr.  
San Jose CA 95124

Dear HUG,

I finally received issue #7 of 'REMark' here in Hawaii. It arrived (of course) only yesterday and two days after my complaint that I did not have it. The bulk mailing to Hawaii is S L O W .....

That is not, however, what I am writing about this time. I have found two items regarding the H8 that may be of some interest to the readers. The first is that the "LIST" command works for your printer even though this is not documented in the HDOS manual for BASIC. The sequence is:

- (1) Open "BP:" for write as file #1
- (2) List #1, or list #1, (IEXPl)\_, (IEXP2)
- (3) Close #1

This routine will allow you to list a hardcopy of the program without having to get out of BASIC, do a listing in HDOS, and then returning to BASIC to load up and start over again. Note that "BP:" is my device driver for my baudot printer. You will have to address your printer driver by its appropriate name.

The second item that I have is a hardware fix that has been buggin' me since I bought my H8 almost two years ago. Perhaps it will pertain to some of the other folks. Hawaii's atmosphere is extremely salty and corrosive. My H8 has "enjoyed" little glitches that have caused it to dump whatever it was supposed to be doing and go off on its own. The problem was very disconcerting to say the least, but I could never find a cause.

Even cleaning all of the contacts on the mother-board only seemed to be of temporary help. The offer in issue #6 of the new 5-volt regulator socket seemed to be the answer. After I installed them, things went really well for about a month and then it was back to the same problem all over again, but at least I now had the final clue that I needed. It appears that the metal on the leads of the regulator chips is combining either with the salt air or with the silicon heat sink compound (or both) to form a greasy, black oxide on the leads of the 5-volt regulators. There are two possible solutions:

- (1) Solder the lead-in wires directly to the chip (after insuring that the regulator is still putting out +5 volts plus or minus 1/4-volt).
- (2) Clean the leads on the regulator with a "pink pearl" brand eraser. This is the soft, pink eraser that school children buy. Don't buy anything harder as you will stand a good chance of destroying the plating on the leads as well as the oxide. When the eraser gets covered with the oxide, rub it on a piece of scrap paper to remove the oxide and continue until the eraser is clean after rubbing the regulator leads - it may take 3 or 4 passes. Spray the regulator socket with a non-residue contact cleaner and then re-install.

I used the second method above and it has worked miracles for my former "jumpy" computer. It hasn't dropped a single bit in well over two months and since my entire machine (H8/H9/H17) is powered 24 hours a day, I consider the problem completely resolved.

This same procedure (#2) indicates that there is some type of oxidation on the connectors on the mother-board. These connectors, though, are apparently plated and the oxidation is not nearly as bad as it is on the regulator pins which appear to be "tinned" for ease of soldering. I am now attempting to obtain the part numbers\* for the same type of connector and pin with gold plating so that I can change all of my mother-board pins and individual card connectors to the gold plated type. Heath may have recognized this problem already since

it appears that all connectors and pins in the new H89 are of the gold plated variety. This will probably be an expensive proposition, but I want to reduce the possibility of any failure to as low a level as practical. Some amount of "preventive maintenance" as suggested above should, however, preclude the largest majority of problems.

One last note and I'll return to computing. I have seen most of the systems on Oahu, Hawaii and even some from the neighbor islands and Midway. It has been my experience that considerably fewer problems are occurring in those systems that are left on constantly then those systems whose owners turn them off and then back on when they are ready to use it. The electrical "spikes" generated at turn-on are the culprits. I have installed General Electric movistors on all of my units to protect them against powerline surges and I leave the system on to prevent internal power supply surges. It apparently works since I have experienced zero chip failures while other users have had (collectively) well over a half a dozen various types of chip failures in the last year. This may be sheer luck, of course, but I doubt it. The Godbout memory board (I have two of them) is an excellent board, but appears to be fairly vulnerable to this type of turn-off/on failure. I have, again, never had a problem with either of the boards while other users report intermittent failures after "firing up". They can usually agree that the memory was working without apparent fault when the system was turned off. Perhaps a reader survey could shed some further light on the matter.

I hope that this information can be of use to someone else. I enjoy 'REMark', hope that it will be able to be published more often, and that I can get my issues sent to Hawaii via 1st class mail. I think that you guys are doing a good job and I appreciate it.

Sincerely,

Gerry Cramm  
2545a Lawrence Place  
Kailua HI 96734

Dear HUG,

I have discovered a subtle error in H11 DOS regarding the IF END # statement in BASIC that I felt could be of help to you and your readers. This soft error was the cause of the loss of much of my hair over the past month because it was inconsistent but pernicious.

H11 users are advised that programs using the IF END #n statement should be set up so that this statement may only be true ONCE for each open file. I found that when otherwise constructed, the program could fail to realize a second time that the file was in fact empty and generate unwanted (even bizarre) additions to the file. For those situations where it is necessary to check for the end of the SAME file more than once (and where it may be at the end for more than one of the tests), readers are advised to rewrite the program using variables as flags for the IF statement.

A sample program and output demonstrating this flaw is enclosed.

```
10 OPEN "TEST" FOR OUTPUT AS FILE #1
20 PRINT #1:"THIS IS THE FIRST LINE"
30 PRINT #1:"THIS IS THE LAST LINE"
40 CLOSE #1
50 OPEN "TEST" FOR INPUT AS FILE #1
60 FOR I=1 TO 100
70 IF END #1 THEN 100
80 INPUT #1:Q$\PRINT Q$
90 Q$=""
100 IF END #1 THEN 130
110 INPUT #1:Q$\PRINT Q$
120 Q$=""
130 IF END #1 THEN 160
140 INPUT #1:Q$\PRINT Q$
150 Q$=""
160 NEXT I
170 PRINT "END OF ENTRIES"
180 END
```

```
THIS IS THE FIRST LINE
THIS IS THE LAST LINE
THIS IS THE FIRST LINE
THIS IS THE LAST LINE
END OF ENTRIES
```

Douglas H. McNeill MD  
Poynette Family Practice Center  
PO Box 115 - 330 N Main St  
Poynette WI 53955

Dear Mr. Blake:

I just received issue 7 (yesterday) and really appreciate your efforts. I am a novice H11A, H9, H14, and H27 owner and could use some help. Let's consider these things:

1. Put a short blurb at the heading of each article in HUG saying what system the article is about. Most do this, but I am keeping a file of all H11 related articles and some I don't understand as yet. For instance, page 13 doesn't say if patch is H8 or H11 related. Page 12 I can figure out by looking at the listing (menu) and I now know that HDOS is H8 related. Page 25 is even a greater problem. The bootup program I don't understand at all, but I can see that if it is for the H11, I will want to implement it as soon as I can get some help and figure it out.
2. I use my H11A, 32K, H14, and H27 for business purposes and have written a lot of programs in BASIC for that purpose. Others have done the same as shown by pages 9 and 10. I would like to communicate with Dr. McNeill but hesitate to do so not knowing if he is prepared to share his efforts. How about a mailing list of members who are eager to share (I am) their H11 software or at least a note whether an author is willing to share his efforts in each article?

Keep the H11 software and tips coming.  
We need all you can give.

Sincerely,

Bob Case  
Box 246  
Summersville MO 65571

Dear Heath H11 User:

For some time we have been wanting to have a way of communicating with other H11 users in our general geographic area. Recently we were given a list of those who have purchase H11 computers through the Heathkit Electronic Center at Frazer, PA, and we are trying to set up a meeting which would include all interested persons who would respond to such a possibility.

Our idea is to get together for a "get acquainted" dinner somewhere in the general area of the Heathkit Store. We would have a chance to get to know each other and find out the various applications in which the H11 is being used. After the dinner, and a general rap session, we would go over to the Heath Store and have a chance to look at some varieties of applications software which people would bring with them to the get together.

We would also have a chance to decide if we would wish to set up an organization

through which we could benefit each other (helping solve hardware/software questions, etc.), and also determine if we would want to have regular meetings.

If this is of interest to you, please respond by writing or calling any of us listed below.

We have extremely powerful computing equipment - perhaps through this we will be able to better put this power to practical use.

Sincerely,

W. Douglas Wilkens (Acting Secretary)  
Dimension Five, Inc  
24 N. Third St  
Womelsdorf PA 19567  
(215) 589-2546

Fil Laser (Acting Treasurer)  
TELE Publications  
158 Penn Ave  
W. Reading PA 19611  
(215) 374--3155

Dr. William Campbell (Acting Chairman)  
Fox Hill Farms  
249 Smith Bridge Rd  
Elam Glen Mills PA 19342  
(215) GL9-3218

Dear HUG,

Here is a short program for REMark. It is designed for the use of those who have a H27 floppy with the H11 computer who have had in the past the H10 paper tape system. It allows one to use the disk for an absolute loader program and any paper tape program written in that format, such as Focal-8K. After the first section is completed, any paper tape can be read by putting it in the reader and typing 'R ABSLDR' on the terminal. After the FOCAL-8K is read in and the rest of the procedure is done, FOCAL can be brought in and used as before. A control C will not bring back the keyboard monitor, and HT 11 must be restarted normally, or the register 4 routine here can be used.

PROCEDURE TO PUT PAPER TAPE PROGRAMS  
ONTO H27 DISKETTE.

Start HT 11 normally and set in the date if necessary. Then use the keyboard monitor deposit routine to insert the absolute loader into diskette storage.

Type:

D 37500 10706 24646 10705 62705 112 5001 12716 0  
D 37520 6016 103402 5016 403 6316 1001 10116 5000  
D 37540 4715 105303 1374 4715 4767 74 10402 162702  
D 37560 4 22702 2 1441 4767 54 61604 10401  
D 37600 4715 2004 105700 1753 0 751 110321 770  
D 37620 16703 152 105213 105213 105713 100376 116303 2 60300  
D 37640 42703 177400 5302 207 12667 46 4715 10304  
D 37660 4715 303 50304 16707 30 4767 177752 4715  
D 37700 105700 1342 6204 103002 0 700 6304 61604  
D 37720 114 0 12767 352 20 12767 765 34  
D 37740 167 177532 16701 26 12702 352 0 0  
D 37760 0 0 0 0 0 0 765 177550  
D 40 37500

SAV SY:ABSLDR.SAV 0-37776

This puts the absolute loader program on disk where it may be used for any paper tape program using up to 8K memory. Put the FOCAL 8K tape in the reader and type 'R ABSLDR'. When the tape is finished, type control D, then hit the break key to get into ODT. Enter the following routine by typing the first five numbers followed by a /. Ignore the data that is displayed and enter the data listed.

24400/nnnnn 12700  
24402/nnnnn 0  
24404/nnnnn 12701  
24406/nnnnn 24430  
24410/nnnnn 12702  
24412/nnnnn 24500  
24414/nnnnn 12022  
24416/nnnnn 20001  
24420/nnnnn 0  
24400G           Wait here, then type  
R4/165264        Return key  
165242G         Wait for monitor re-entry

D 24400 12700 0 12701 24340 12702 24500 12220 20001  
D 24420 1401 774 137 4254  
D 40 24400

SAV SY:FOCAL.SAV 0-51140

R. A. Huntsinger  
19765 Los Gatos Road  
Saratoga CA 95070

Dear Mr. Blake:

Me again!

When I have a few minutes, I dip into the library and look for a helpful strategy or subroutine for you and REMark readers. I found in my archives something that I think you will enjoy.

Enclosed find the code for a text editor in BASIC for use within programs. I have used this subroutine for slight alterations in a data field, i.e. insurance records, and find it to be both fast in execution and fast in operator time, for few keystrokes are required to change a small segment of an otherwise satisfactory alphanumeric field. The reduction in re-entry of correct data has helped to cut down on the introduction of new typographical errors.



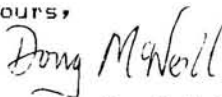
```

5000 REM:SBR- EDIT ENTRY AND VERIFY
5002 REM: INPUT WITH I$, EXIT WITH I$ CORRECTED
5004 PRINT "USE A HYPHEN (-) FOR CHARACTERS TO RETAIN IN FAULTY LINE"
5006 PRINT "USE J (CAPITAL M) TO DELETE A CHARACTER"
5008 PRINT "USE ! TO DELIMIT ADDITIONS TO LINE; E.G.: !THESE ADDED!"
5010 PRINT "END CORRECTIONS WITH ^ (CAPITAL N) IF BALANCE OF LINE O.K."
5012 PRINT "ENDING LINE WITH RETURN ALONE SHORTENS LINE"\PRINT
5014 Q$=I$\PRINT " ";I$\PRINT ":";\INPUT #0:R$
5016 PRINT \I$=""\IF R$="" THEN 5046 \IF Q$="" THEN 5044
5018 G5=LEN(R$)\FOR I=1 TO G5
5020 B1$=SEG$(R$,I,I)\IF B1$<>" " THEN 5026
5022 IF I=G5 THEN 5024 \PRINT (" " MUST BE LAST CHARACTER)\I$=Q$\GO TO 5014
5024 I$=I$&SEG$(Q$,I,LEN(Q$))\GO TO 5046
5026 IF B1$<>"!" THEN 5036 \G6=1
5028 IF I+G6<G5+1 THEN 5030 \PRINT ("!" MUST END ADDITIONS)\I$=Q$\GO TO 5014
5030 G7$=SEG$(R$,I+G6,I+G6)\IF G7$="!" THEN 5032 \I$=I$&G7$\G6=G6+1\GO TO 5028
5032 G7=POS(R$,"!",1)\R$=SEG$(R$,1,G7-1)&SEG$(R$,POS(R$,"!",G7+1)+1,LEN(R$))
5034 G5=G5-G6-1\GO TO 5020
5036 IF B1$<>"-" THEN 5038 \I$=I$&SEG$(Q$,I,I)\GO TO 5042
5038 IF B1$<>"[" THEN 5040 \GO TO 5042
5040 I$=I$&B1$
5042 NEXT I\GO TO 5046
5044 I$=R$
5046 PRINT "NEW VERSION: ";I$\PRINT "HIT RETURN TO ACCEPT IT, ANY CHARACTER ";
5048 PRINT "TO REJECT"\INPUT #0:Q4$\IF Q4$="" THEN 5050 \I$=Q$\GO TO 5004
5050 RETURN

```

Regards for your holiday season.

Yours,



Douglas H. McNeill, M.D.

Dear Sirs:

I would like to express my appreciation to HUG and Mr. Irvin Hoff for making available the SFO 8.4 program generating system (885-1035) on cassette. I had about given up learning to program in assembly language until I tried the SFO 8.4 approach. If you don't have a disc, the time consuming edit-assemble-debug routine of the Heath software can get discouraging. The SFO 8.4 lets you write an assembly program, assemble it, run it, and make all the corrections in one piece of software and I sure recommend it to your members.

One thing missing is the provision to use the H-14 printer, and to illustrate the usefulness of the co-resident editor/assembler I wrote the software patch using the SFO 8.4 and I had never written an assembly language program before. Some of your readers may benefit from my efforts.

Very truly yours,

Chester H Brent  
149 Indian Hill Drive  
Greeneville TN 37743

```

PRINT      CALL INIT      SET UP I/O & PRINTER
WAIT       IN   0345      IS I/O READY?
           ANI  040
           JZ   WAIT      IF NOT WAIT
           IN   0346      IS PRINTER READY?
           ANI  020
           JNZ WAIT      IF NOT WAIT
           MOV  A,B       GET NEXT CHARACTER
           OUT  0340      PRINT IT
           RET           RETURN TO MAIN PROGRAM
INIT       MVI  A,0200    SET DLAB
           OUT  0343      LINE CONTROL REGISTER
           MVI  A,030     SET BAUD 4800
           OUT  0340      DATA REGISTER
           MVI  A,0       NO INTERRUPT
           OUT  0341      INTERRUPT REGISTER
           MVI  A,3       8 BITS
           OUT  0343      LINE CONTROL REGISTER
           LXI  H,073/000
           MVI  M,0       TO BYPASS INIT PROCEDURE
           INX  H
           MVI  M,0
           INX  H
           MVI  M,0
           RET
END        PRINT

```

Dear Manager:

HUG TEXT EDITOR A A1 CINCINNATI

I am writing to ask you to list in the next issue of REMARK, our very popular courses on microcomputer interfacing and Digital Electronics, a new course will be offered for the first time. This new program is an exciting and unique course which will teach users how to interface and use a popular home computer in instrumentation and automation systems.

Workshop: Computer Interfacing and Programming for Instrumentation and Control. March 17, 18, 1980.

Workshop: 8080-8085-Z80 Microcomputer Interfacing, Design and Software. March 12, 13, 14, 1980.

Workshop: Digital Electronics for Instrumentation and Automation. March 10, 11, 1980.

These are hands-on workshops with the participants having the opportunity to retain the equipment. For more information, contact Dr. Linda Leffel, CEC, Virginia Tech, Blacksburg VA 24061 (703-961-5241).

David G. Larsen  
Virginia Polytechnic Inst & State Univ  
Dept of Chemistry  
Blacksburg VA 24061

Dear Jim,

Just received my copy of REMark issue #8 and I was pleased to see my letter included. Lots of fine, new ideas. As I have said before, I look forward to every issue. HUG is absolutely essential to all Heath computer owners.

This letter has several purposes and I will separate the issues for clarity:

1. I had hoped there would be something in issue #8 elaborating on the Cincinnati HUG MOD to the HUG Text Editor, but alas, nothing. So I am enclosing a paper on that subject.

2. I finally got around to installing Jim Buszkiewicz's mod to the H8-2 (issue #5, p. 21). It works beautifully. A paper re the mod and a minor improvement are enclosed.

3. MBASIC is the "CAT's MEOW". Best 100 bucks I have spent since I started with the H8. That, coupled with ver. 1.5 of HDOS, is going to be hard to beat in any micro. I am enclosing a paper with some notes concerning these systems which may be of general interest.

The best seasons greetings to you and the staff at HUG. I still am amazed at the many things you get done with so few people. Hope that 1980 is a good year for you all.

I don't have much assembly language experience (I'm on lesson three in the programming course.), but I just had to have the hardcopy change that the Cincinnati HUG presented in REMark issue #7, P. 18. SO I started in blindly and after quite a few problems, I came away with a working "ED2". The changes are well worth the effort but I hope you can avoid the problems that I encountered.

First off, putting DB lines up front with the EQU lines is apparently a no no. At least when I did it I got an ABS of 59 sectors (wow) that bombed as I loaded it. To fix this I put the DB lines somewhere where they won't be pointed at by the PC. I entered them just before HAVNAM.

The fact that I had a different version of ED than did Cincinnati didn't offer too much of a problem at first. G was already used for Gizmo, so I used J. But nothing worked. All the commands did the wrong thing. Then I discovered that Cincinnati forgot to tell us about DSPIBL where HCMD, JCMD, XCMD and YCMD must be substituted for CMDERR in the proper places in that table.

This helped, but still the new mod wouldn't work. So I traced the problem to the lack of CSTS in my version of the ED. I couldn't even find a close, possible, typing error that looked like CSTS. As a last resort, I inserted a RET as the second line in PRTOU1. This lets the printer work but does not give me CNTRL-S,O control of the printer from the keyboard of the terminal. (I hope one of you experts can give me the correct coding for PRTOU1.)

A couple of other minor changes were made. LPS EQU was changed to 12 for my H9. Changing the CPI line in SEARL3 to 040H allows the "AT" sign to be the wild card instead of the ASCII bar which isn't on the H9. MVI C,0AH adn CALL JOUT were added in CRCNV to make my expandor line-feed upon CR.

Of course I still have the CSTS problem and there is a bug with H and CNTRL-C -- the LEDs don't reset. And you cannot recover from a cassette dump or load which is improperly executed. But, all in all, it was well worth the effort and frustration. I learned alot and now have a good "ED2".

My hat is off to the Cincinnati HUG for their spade work and inspiration I got from their article.



## Keep it Simple Stupid

# KISS

You are right. Many of the magazines, including this one, have been ignoring the user that just got his system on the air, no small achievement in itself. So, in this column we will assume almost nothing and try to show you how to do things and explain it as we go along. The task before us today is to write a mailing list program so that we can show the computer widow that computers do more than play Startrek. All we hope to accomplish in this installment, is write a program that will ask the user for the NAME, STREET ADDRESS, CITY AND STATE, ZIP, PHONE, BUS. PHONE AND NOTES. Then save the information on tape. This program is written for cassette BASIC 10.02 or later. At the conclusion of this piece, we will show you how to convert it to the disk system.

We have decided that there will be a maximum of 100 names. We will call the information associated with each name a 'RECORD'. And since that sometime during the manipulation of this list, we will have all 100 RECORDS in memory at one time, we need to tell BASIC of our intentions so that we do not compete for the same memory space. We do this with the DIMension statement. Since we are only dealing with strings (and not numeric values) we enter at line...

```
110 DIM A$(100,6),T$(6).
```

This sets aside string space for 101 names along with 7 'elements' (address etc.) per name. Also, since we will use the words NAME, ADDRESS, CITY, ZIP, PHONE, BUS. PH and NOTES several times, we will stick them in an array (T\$) to conserve program space. Now READ the DATA into reserved 'string space' (invisible mail boxes).

```
120 FOR I=1 TO 6:READ T$(I):NEXT I
130 DATA "NAME","ADDRESS","CITY STATE, ZIP","PHONE","BUS. PHONE","NOTES"
```

You may want to insert a STOP at line 140 and RUN. Then when you see the prompt, ask BASIC to PRINT T\$(2) see what you get. Try T\$(10)!

OK, Lets' add some names to the mailing list.

```
2000 N=N+1:PRINT:REM Set up a counter called 'N' and PRINT A BLANK LINE
2010 PRINT T$(1);:LINE INPUT " : ";A$(N,1):REM PRINT "NAME :" on the screen
    and plop the first name into 'mailbox A$(1,1)'
2030 IF LEN(A$(N,1))=0 THEN N=N-1:GOTO 200:REM Check to see if user is done.
2040 FOR I=2 TO 6:REM now print address, city state etc.
2060 PRINT T$(I);: LINE INPUT": ";A$(N,1): REM and file away in memory.
2070 NEXT I:REM keep going until we get to "NOTES"
2080 GOTO 2000:REM bump the counter and get next name.
```

There... we have input the names, addresses of persons until the user hits return when asked to enter the name. When the length of the string a\$(n,1) is 0 then the program will branch to 200 where we will give the user some options. Let's do that part now.

```
200 PRINT : LINE INPUT "LIST ALL NAMES, ADD MORE NAMES, OR STOP";A$
205 REM Print a blank line and ask for his choice
208 REM Now see what he wants to do
210 IF A$="LIST THEN 1000
220 IF A$="ADD" THEN 2000
230 IF A$="STOP" THEN 3000
295 REM If he doesn't know what to do, give him another chance.
300 PRINT:PRINT: PRINT "I DON'T UNDERSTAND THAT ":GOTO 200
```

```
1000 REM He wants to list all the names. set up a counter
1020 FOR I = 1 TO N : REM N should equal how many names we have. Right?
1030 FOR J = 1 TO 6: REM we need to use the headings again
1040 PRINT T$(J);":":TAB(14);A$(I,J):REM Line it up in neat columns.
1050 NEXT J:NEXT I: REM keep going til done
1060 GOTO 200: REM All done!
3000 REM He wants to quit!
3010 PRINT : PRINT"TYPE 'FDUMP' NOW TO SAVE THE DATA ON TAPE."
3020 STOP
```

Until next issue, practice with 'FLOAD', 'GET ' and 'PUT'. However, add one more line to your program. At line...

```
100 IF N>0 THEN 200:REM Why do this?
```

Another question. Why should you load the program and type 'CONTINUE' instead of 'RUN'?

:JB: EOF

# MILLION DOLLAR CHECK WRITER

By: Jim Tennant  
Box 7176  
Ketchikan, AK 99901

```
010 DIM U$(19),T$(9),Y$(8),E$(6),W$(2):REM - VOCABULARY
020 FOR A=1 TO 19:READ U$(A):NEXT A:REM - UNITS WORDS
030 FOR A=1 TO 9:READ T$(A):NEXT A:REM - TENS WORDS
040 FOR A=1 TO 3:READ Y$(A):NEXT A:REM - DENOMINATION TAGS
050 FOR A=0 TO 6:READ E$(A):NEXT A:REM - ERROR MESSAGES
060 B=0:C=0:D=0:E=0:F=0:W$(1)=" ":W$(2)=" ":REM - C=COMMAS, D=DECIMALS
070 :REM          FOOL-PROOF KEY ENTRY
080 LINE INPUT "INPUT AMOUNT: $ ";A$
090 IF ASC(A$)=48 THEN A$=RIGHT$(A$,LEN(A$)-1):GOTO 90:REM - NO LEAD ZEROS
100 IF LEN(A$)=0 THEN A$="0":REM - MAKE ONE LEAD ZERO
110 FOR A=1 TO LEN(A$):N=ASC(MID$(A$,A,1)):IF N=63 GOTO 530:REM - PRINT ERROR
120 IF N=44 THEN C=C+1:E=A:IF C>1 THEN Z=1:GOTO 60:REM - E = COMMA PLACE
130 IF N=46 THEN D=D+1:B=A:IF D>1 THEN Z=2:GOTO 60:REM - B = DECIMAL PLACE
140 IF N=45 OR N=47 OR N<44 OR N>57 THEN Z=3:GOTO 60:REM - NUMBERS ONLY
150 NEXT A:IF A>11 THEN Z=4:GOTO 60:REM - Z = ERROR CODE
160 IF ASC(A$)=46 THEN A$="0"+A$:B=B+1:REM - ADD LEAD ZERO
170 IF B=0 THEN B=A:A$=A$+"":F=1:REM - F = 'NO MORE' FLAG
180 IF E=1 THEN Z=1:GOTO 60:REM - COMMA CANNOT BE FIRST ENTRY
190 IF E>0 THEN A$=LEFT$(A$,E-1)+RIGHT$(A$,LEN(A$)-E):IF B>E THEN B=B-1
200 IF E>7 THEN Z=5:GOTO 60
210 IF LEN(A$)>B+2 THEN Z=6:GOTO 60
220 IF LEN(A$)<B+2 THEN A$=A$+"0":GOTO 220:REM - FILL DECIMAL PLACES
230 :REM          WORD STRING GENERATOR
240 Y=1:N=VAL(LEFT$(A$,LEN(A$)-3)):IF N=0 THEN X=1:W$(Y)=" ZERO":GOTO 370
250 Y=0:GOTO 400:REM - X = DIGIT CURSOR
260 IF B-X=3 THEN Y=2:REM - Y = OUTPUT STRING SELECTOR
270 IF INT((B-X+1)/3)=(B-X+1)/3 GOTO 300:REM - TEST FOR 2-DIGIT PROCESSING
280 D=VAL(MID$(A$,X,1)):IF D=0 GOTO 400:REM - START 1-DIGITER HERE
290 GOTO 360
300 D=VAL(MID$(A$,X,2)):IF D=0 THEN X=X+1:GOTO 370:REM - START 2-DIGITER HERE
310 IF D<20 THEN X=X+1:GOTO 360
320 D=VAL(MID$(A$,X,1)):IF D=0 GOTO 340
330 W$(Y)=W$(Y)+T$(D):REM - ADD TENS WORD
340 X=X+1:D=VAL(MID$(A$,X,1)):IF D=0 GOTO 370
350 W$(Y)=W$(Y)+"-"
360 W$(Y)=W$(Y)+U$(D):REM - ADD UNITS WORD
370 IF ASC(RIGHT$(W$(Y),1))>32 OR ASC(Y$(7-B+X))>32 GOTO 390
380 W$(Y)=LEFT$(W$(Y),LEN(W$(Y))-1):REM - ELIMINATE DOUBLE SPACE
390 W$(Y)=W$(Y)+Y$(7-B+X):REM - ADD TAG
400 Y=X+1:IF X<B GOTO 260:REM - NOT LAST PASS
410 IF F=1 THEN X=X+1:W$(Y)=W$(Y)+Y$(7-B+Y):GOTO 430:REM - OBEY F-FLAG
420 W$(Y)=W$(Y)+RIGHT$(A$,2)+Y$(7-B+X)
430 IF (LEN(W$(1))+LEN(W$(2)))>46 GOTO 450:REM - MAKE TWO LINES
440 W$(1)=LEFT$(W$(1),LEN(W$(1))-1)+W$(2):W$(2)=" ":REM - COMBINE LINES
450 :REM          PRINTER OUTPUT DATA
460 Y=1:IF B<5 THEN PRINT TAB(65)A$:GOTO 480
470 PRINT TAB(65) LEFT$(A$,B-4);", ";RIGHT$(A$,LEN(A$)-B+4)
480 FOR A=1 TO (46-LEN(W$(Y)))/2:PRINT "*";:NEXT A:A=A-1
490 PRINT W$(Y);:FOR A=A TO 1 STEP -1:PRINT "*";:NEXT A:PRINT
500 IF LEN(W$(2))>1 AND Y=1 THEN Y=2:GOTO 480
510 PRINT : PRINT : GOTO 60
520 :REM          PROGRAM MEMORY WORDS
530 PRINT "IMPROPER ENTRY -- ";E$(Z):Z=0:PRINT :GOTO 60
540 DATA "ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NINE","TEN"
550 DATA "ELEVEN","TWELVE","THIRTEEN","FOURTEEN","FIFTEEN","SIXTEEN"
560 DATA "SEVENTEEN","EIGHTEEN","NINETEEN","TEN","TWENTY","THIRTY","FORTY"
570 DATA "FIFTY","SIXTY","SEVENTY","EIGHTY","NINETY"," HUNDRED "," THOUSAND "
580 DATA " THOUSAND "," HUNDRED "," AND "," AND ","/100 ","NO MORE "
590 DATA "NO ERROR DETECTED","COMMA MISUSE","DECIMAL POINT EXCESS"
600 DATA "NON-NUMERIC CHARACTER","DIGIT EXCESS","$1 MILLION IS THE LIMIT"
610 DATA "TOO MANY DECIMAL PLACES"
```

EOF

# ET-3400 MORSE CODE READER

By: Louis C. Graue  
624 Campbell Hill Road.  
Bowling Green, OH 43402

This short program makes it possible for the ET-3400 to copy Morse Code off the air and print it out ticker tape fashion on the LED's. The program also regenerates the code it has received and this can be fed to an oscillator.

## OPERATION INSTRUCTIONS

First wire the PIA to the trainer exactly as pictured in Figure 10-78 on page 10-99 of the ET-3400 course manual. Then on the breadboard, set up the 567 tone decoder circuit shown in Fig. 1. The input to the 567 goes across the receiver speaker and the output is connected to the PIA pin A0. Don't forget to connect ground. Next, enter the program and tune in a CW signal so that the LED on the 567 circuit is blinking. The decoded Morse should then start to flow across the ET-3400 LED's.

The speeds which can be received are controlled by the number entered at address 0100. The 00FF you have entered should enable perfect copy of speeds from 15 to 25 wpm. If the copy is not good and the code sounds faster use the keyboard to change the contents of 0100 to a smaller number. I use 00A0 to copy 35 to 40 wpm and 00EE to copy 13 wpm. Any speed can be copied by setting the proper delay constant.

If you wish to hear the code regenerated by the computer then attach an oscillator across ground and PIA pin B0.

If you wish to check your sending technique then connect your key or keyer across ground and PIA pin A0. If your technique is good then a perfectly formed message will flow across the LED's. You can easily spot letters which you do not send well or any improper spacing.

## HOW THE PROGRAM WORKS

The flow diagram of Fig. 2 will help to explain how the program works. There are two important problems to solve when writing a program to decode Morse, timing and a lookup table algorithm.

Timing is important for determining whether a key down is a dot or a dash and whether a key up is an element space, a character space or a word space. Self-adaptive adjustment procedures for changes in code speed add to the complexity and length of the program. They work very well with machine sent code but my experience indicates that the simple procedure used in this program works better with hand sent code.

A reference number called T is stored at 0102 and four times this number is stored at 0103. Adjustment for speed is obtained by changing the delay constant at 0100. A check of the input for a change in key status is made, then a counter is bumped and a delay before the next check is determined by the delay constant. When a change in key status is detected, the counter is compared to T. For a key down to key up change, if the comparison shows the counter is less than T, then a dot is stored, otherwise a dash is stored. For key up to key down change, if the comparison is less than T an element space is assumed, if the comparison is more than T but less than 4T a

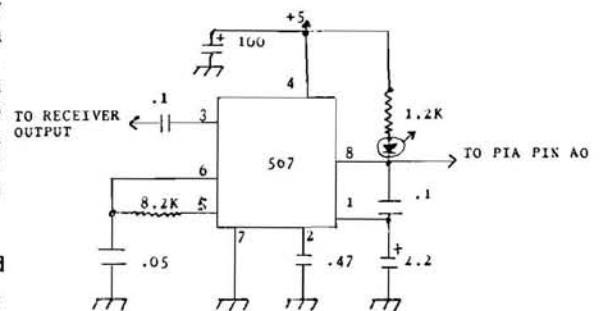


FIGURE 1

character space is assumed, and if more than 4T a word space is printed. Checking the input at millisecond rather than microsecond intervals also considerably reduces the probability that some random noise will interfere with the decoding process. The lookup table algorithm uses the number stored in 0104 as the second half of the address in the lookup table that contains the 7 segment code to be printed. The procedure starts with a 01 stored at 0104, if a dot is to be stored the number is shifted to the left, if a dash is to be stored the number is shifted to the left and incremented. Thus if a C has been received the binary number will be 00011010 or 1A Hex. Ignore the first 1 and read the rest of the ones as dahs and the zeros as dits. Thus it reads dah dit dah dit. Notice that the address 031A contains 4E which is the 7 segment code for C. The 031A is loaded into the index register and then index addressing is used to load accumulator A with the 4E from that address which is then printed by the display routine.

When entering the program it is a good idea to fill all unused addresses in the lookup table with zeros so that any errors are printed as spaces.

This program was written for the ET-3400 with the ETA-3400 accessory attached. It can be easily changed to fit on the ET-3400 alone by putting the main program on page zero and the lookup table on page one. The delay constant will be quite different because of the slower cpu speed but can be determined experimentally.

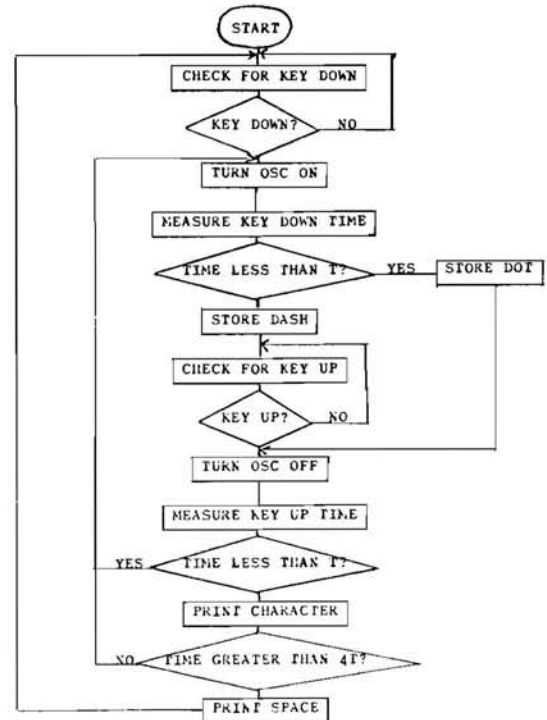


FIGURE 2

Morse Code Reader Listing starts on the next page.

## MBASIC SORT

By: Wm. Campbell  
249 Smith Bridge Rd.  
Elam Glen Mills Pa 19342

```

10 REM                               Filed as SMM.BAS
20 CLEAR(10000):DEFINT A-Z:DIM A$(300)
30 LINE INPUT "Type name of file to be sorted (Ex. SY1:OLD.DAT). ";M$
40 OPEN "I",1,M$:OPEN "O",2,"SY1:SORT.DAT"
50 IF EOF(1) THEN 70
60 LINE INPUT #1,A$(I):I=I+1:GOTO 50
70 C=I:B=I
80 C=INT(C/2):PRINT C:IF C=0 THEN 160
90 D=1:E=B-C
100 F=D
110 G=F+C:IF A$(F)<=A$(G) THEN 140
120 SWAP A$(F),A$(G):F=F-C:IF F<1 THEN 140
130 GOTO 110
140 D=D+1:IF D>E THEN 80
150 GOTO 100
160 FOR X=0 TO I:PRINT #2,A$(X):NEXT X:CLOSE #1:CLOSE #2:END
  
```

EOF

# ET-3400 MORSE CODE READER LISTING

ROUTINE ET-3400  
 PROGRAM MORSE CODE READER  
 BY K8TT

DATE 12/5/79  
 PAGE 1 OF 7  
 CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	OP CODE	OPERAND	COMMENTS
INITIALIZATION					
0100	00 FF				DELAY CONSTANT
0102	40				TIME FACTOR
0103	A0				4xTIME FACTOR
0104	03 01				CODE STORAGE
0106	CE 00 04		LDX	#0004	A SIDE IN
0109	FF 80 00		STX		
010C	CE FF 04		LDA	#FF04	B SIDE OUT
010F	FF 80 02		STX		
WAIT FOR KEY DOWN					
0112	B6 80 00	WAITD	LDA	A	CHECK FOR KEY DOWN
0115	84 01		AND	A#01	MASK TO BIT 0
0117	26 F9		BNE	WAITD	NOT DOWN, GO CHECK AGAIN
TIME KEY DOWN					
0119	B7 80 02	TIMED	STA	A	OTHERWISE, TURN OSCILLATOR ON
011C	C6 00		LDA	B#00	RESET TIME COUNTER
011E	8D 4F	CHECK1	BSR	DLY	GO WAIT,BUMP CNTR,& CHECK INPUT
0120	26 17		BNE	DOT	IF KEY UP GO STORE DOT
0122	F1 01 02		CMP	B	OTHERWISE, IS TIME LESS THAN T?
0125	25 F7		BCS	CHECK1	IF SO, GO WAIT,BUMP CNTR,& CHECK INPUT
STORE DASH					
0127	0C		CLC		OTHERWISE, STORE A DASH
0128	79 01 05		ROL		BY LEFT SHIFT OF CODE STORAGE
012B	7C 01 05		INC		AND PLACING I IN BIT 0
012E	25 33		BCS	SPACE	NOT PROPER CODE GO PRINT SPACE
WAIT FOR KEY UP					
0130	B6 80 00	WAITU	LDA	A	CHECK FOR KEY UP
0133	84 01		AND	A#01	MASK TO BIT 0
0135	27 F9		BEQ	WAITU	IF NOT UP GO CHECK AGAIN
0137	20 06		BRA	TINEU	OTHERWISE, GO TURN OSCILLATOR OFF
STORE DOT					
0139	0C	DOT	CLC		STORE DOT
013A	79 01 05		ROL		BY LEFT SHIFT OF CODE STORAGE
013D	25 24		BCS	SPACE	IF NOT PROPER CODE, PRINT SPACE
TIME KEY UP					
013F	B7 80 02	TIMEU	STA	A	TURN OSCILLATOR OFF
0142	C6 00		LDA	B#00	RESET TIME COUNTER
0144	8D 29	CHECK2	BSR	DLY	GO WAIT,BUMP CNTR,& CHECK INPUT
0146	27 D1		BEQ	TIMED	IF KEY DOWN GO TURN OSCILLATOR ON
0148	F1 01 02		CMP	B	OTHERWISE IS TIME LESS THAN T?
014B	25 F7		BCS	CHECK2	IF SO, GO WAIT,BUMP CNTR,CHECK INPUT
PRINT CHARACTER					
014D	FE 01 04		LDX		IF NO, GET CODE STORAGE ADDRESS
0150	A6 00		LDA	A 0,X	GET DISPLAY CODE FROM LOOKUP TABLE
0152	BD 02 07		JSR	DISPLY	PRINT IT
0155	86 01		LDA	A#01	RESET CODE STORAGE
0157	B7 01 05		STA	A	
CHECK FOR WORD SPACE					
015A	8D 13	CHECK3	BSR	DLY	GO WAIT,BUMP CNTR & CHECK INPUT
015C	27 BB		BEQ	TIMED	IF KEY DOWN GO TURN OSCILLATOR ON
015E	F1 01 03		CMP	B	OTHERWISE, IS TIME LESS THAN 4xT?
0161	25 F7		BCS	CHECK3	IF SO, GO WAIT,BUMP CNTR, CHECK INPUT
PRINT SPACE					
0163	86 00	SPACE	LDA	A#00	OTHERWISE PRINT A SPACE
0165	BD 02 07		BSR	DISPLY	
0168	86 01		LDA	A#01	RESET CODE STORAGE
016A	B7 01 05		STA	A	
016D	20 A3		BRA	WAITD	GO WAIT FOR KEY DOWN



ROUTINE DELAY DATE 12/5/79  
 PROGRAM ET-3400 MORSE CODE READER PAGE 4 OF 7  
 BY K8TT CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	CP CODE	OPERAND	COMMENTS
DELAY					
016F	FE 01 00	DLY	LDX		GET DELAY CONSTANT
0172	09		DEX		DECREMENT IT
0173	26 FD		BNE	DLY	NOT ZERO? DECREMENT AGAIN
0175	5C		INC	B	OTHERWISE, BUMP TIME COUNT
0176	B6 80 00		LDA	A	CHECK INPUT
0179	84 01		AND	A#01	MASK TO BIT 0
017B	39		RTS		RETURN TO CALLING PROGRAM

ROUTINE DISPLAY DATE 12/5/79  
 PROGRAM ET-3400 MORSE CODE READER PAGE 5 OF 7  
 BY K8TT CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	CP CODE	OPERAND	COMMENTS
0207	B7 02 38	DISPLY	STA	A	PUT CHARACTER AT END OF STRING STORAGE
020A	CE 02 32		LDX	#0232	SHIFT CONTENTS OF
020D	A6 01	SHIFT	LDA	A,0X,01	STRING
020F	A7 00		STA	A,X,0	STORAGE
0211	08		INX		BACK
0212	8C 02 38		CPX	#0238	ONE
0215	26 F6		BNE	SHIFT	STEP
0217	CE 02 2A		LDX	#022A	TRANSFER
021A	A6 08	MOVE	LDA	A,X,08	STORED
021C	A7 00		STA	A,X,00	STRING
021E	08		INX		TO
021F	8C 02 30		CPX	#0230	OUTPUT
0222	26 F6		BNE	MOVE	STRING
0224	BD FC BC		JSR	REDIS	MOVE FIRST CHAR TO "H" DISPLAY
0227	BD FE 52		JSR	OUTSTR	DISPLAY THE STRING
022A					022A to 022F RESERVED FOR OUTPUT STRING
0230	80				DECIMAL POINT TO END STRING
0231	39				RETURN TO CALLING PROGRAM
0232					0232 to 0238 RESERVED FOR STRING STORAGE

ROUTINE LOOKUP TABLE DATE 12/5/79  
 PROGRAM ET-3400 MORSE CODE READER PAGE 6 OF 7  
 BY K8TT CPU TYPE 6800

ROUTINE LOOKUP TABLE (contd) DATE 12/5/79  
 PROGRAM ET-3400 MORSE CODE READER PAGE 7 OF 7  
 BY K8TT CPU TYPE 6800

ADDRESS	CONTENTS	LABEL	CP CODE	OPERAND	ADDRESS	CONTENTS	LABEL	CP CODE	OPERAND
0302	4F			E	0319	36			X
0303	0F			T	031A	4E			C
0304	06			I	031B	3B			Y
0305	77			A	031C	49			Z
0306	15			N	0320	5B			5
0307	76			M	0321	33			4
0308	5B			S	0323	79			3
0309	3E			U	0327	6D			2
030A	05			R	032F	30			1
030B	3F			w	0330	5F			6
030C	3D			D	0331	01			-
030D	07			K	0338	70			7
030E	7B			G	033C	7F			8
030F	1D			O	033E	73			9
0310	17			H	033F	7E			0
0311	2B			V	034C	65			?
0312	47			F	0355	0C			.
0314	0E			L	0373	10			,
0316	67			P					
0317	3C			J					
0318	1F			B					

EOF

## MBASIC STRING FINDER

This program will search for any given string in a text file created by the editor such as my list of albums. The target name 'John', for instance would return all files containing that name.

Written in MBASIC by:  
Robert Kaytor  
628-18 Street  
Brandon MB.

```
10 CLEAR 200
20 PRINT CHR$(12)
30 PRINT TAB(15) "ALBUM FILE AND RECALL PROGRAM"
40 PRINT TAB(15) "BY ROBERT KAYTOR..."
50 PRINT TAB(15) "WRITTEN...12-MAY-1979..."
60 PRINT :PRINT
70 PRINT "DO YOU NEED INSTRUCTIONS (Y OR N)":INPUT YNS
80 IF YNS <> "Y" THEN IF YNS <> "N" THEN 70
90 IF YNS="Y" THEN 100 ELSE 170
100 PRINT CHR$(12)
110 PRINT TAB(10) "THIS PROGRAM IS DESIGNED TO LIST AND"
120 PRINT TAB(10) "SEARCH FOR ALBUMS BY TITLE,ARTIST,OR"
130 PRINT TAB(10) "CLASSIFICATION OF MUSIC,OR ANY OTHER"
140 PRINT TAB(10) "DATA WHICH MAY BE USED IN YOUR FILES."
150 PRINT :PRINT :PRINT
160 FOR X=1 TO 2500:NEXT X
170 PRINT CHR$(12)
180 PRINT TAB(20) "LIST OF OPTIONS AVAILABLE"
190 PRINT
200 PRINT TAB(10) "1-GENERAL LISTING(ALL ALBUMS ON FILE)."
```

```

530 IF INSTR(A$,B$)>0 THEN PRINT A$
540 IF INSTR(A$,B$)>0 THEN FLAG=1
550 GOTO 510
560 PRINT:PRINT
570 IF FLAG <> 1 THEN PRINT "SORRY, THAT SELECTION NOT ON FILE."
580 CLOSE #1
590 PRINT:PRINT
600 INPUT "WANT TO LOOK FOR ANOTHER SELECTION (Y OR N)";YNS
610 IF YNS="Y" THEN PRINT CHR$(12)
620 IF YNS <> "Y" THEN IF YNS <> "N" THEN 600
630 PRINT:PRINT
640 IF YNS="Y" THEN 470 ELSE 170
650 END
660 REM
670 J=0
680 PA=1
690 GOSUB 1060
700 L1=0
710 REM ALTERNATE TERMINAL ROUTINE
720 OPEN "O",1,"AT:"
730 OPEN "I",2,"SY1:ALBUMS.DAT"
740 IF EOF(2) THEN 830
750 IF L1=53 THEN PA=PA+1
760 IF L1=53 THEN GOSUB 1000
770 IF L1=53 THEN L1=0
780 LINE INPUT #2,A$
790 J=J+1
800 L1=L1+1
810 PRINT #1, USING "###.";J;:PRINT #1,A$
820 GOTO 740
830 PRINT #1,
840 PRINT #1,
850 PRINT #1,
860 PRINT #1,"YOUR LIBRARY CONSISTS OF";J;"ALBUMS."
870 GOTO 650
1000 REM:
1010 REM SUBROUTINE TO PRINT PAGE HEADINGS....
1020 FOR X=1 TO 11
1030 PRINT #1,
1040 NEXT X
1050 CLOSE #1
1060 REM LOOK UP DATE
1070 OPEN "O",1,"AT:"
1080 DATE$=""
1090 FOR X= 8383 TO 8391
1100 DATE$=DATE$+CHR$(PEEK(X))
1110 NEXT X
1120 PRINT #1,TAB(8)DATE$,TAB(32)"ALBUM LISTING",TAB(56)"PAGE #";PA
1130 PRINT #1,
1140 IF PA=1 THEN CLOSE #1
1150 RETURN

```

EOF

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

## LP.SYS CORRECTION

The 'cut and paste' method misplaced 8 lines of code. On page 31 of REMark 8. The 8 lines begin at '.ASECT '. Take those 8 lines (down to and including 'LPCQE: .WORD 0') and place them on page 30 AHEAD of label '.ENABL LSB'. Apologize for that.

## MORSE8 IC

The 14538 IC's used in the Morse Code send/receive package (885-1027) has been hard to find by some. Heath parts now has it in stock as part number 443-916. \$2.60.

## SOFTWARE CONTEST

Ok... Summer fun-time is over and it's time to get the computer back on line. So, let's have another contest. This one is for H8's and H88/H89's. And the prize is a \$250 gift certificate redeemable at any of the Heathkit Electronic store or by mail order. The value of the program will be based on: Useability. Program stability. Ease of use. completeness of work. Quality of documentation. The winner is determined by a panel of qualified judges and all decisions are final. Submit the program on cassette or diskette. The documentation may be typewritten, but we prefer it on tape or disk. Deadline for entry is April 15. The winner will be notified immediately and announced in the next .UPDATE or REMark.

## AUTHORS NOTE

We encourage the members to contribute articles of interest to be included in this magazine. It will be a big help to us if they are submitted on cassette or diskette and if using RUNOFF, include the raw text. Thanks.

## SOURCE

We had plan to review SOURCE (see page 86 of January issue of Interface Age) with you this issue but they've had our 100 bucks for over a month now, and we don't have anything but 3 unanswered phone calls from them. Therefore, we can't tell you our ID number. But read the mail and look us up.

## MEETINGS and CLUB NOTICES

### SALT LAKE CITY

Wanna start sumpin? Call Richard Crawford at 801-272-4552 or write him at 5150 Gurene Dr. Zip 84117.

### REDWOOD CITY, CA

BAHUG Meets the second Tuesday of each month at 7:00 PM at the Heathkit Electronics store at 2001 Middlefield Rd. (I look forward to meeting with you in March :JB:)

### CLEVELAND OH

Local users meet once a month at the Heathkit store on 28100 Chagrin Blvd. Call 216-292-7553 for details.

### NEW JERSEY

No requirements... No dues. Meet with other computer jocks on the 3rd Monday of the month at the Fair Lawn store. (35-07 Broadway at 8PM)