Saving Our HEATH Eight-Bit Machines!

SEBHC JOURNAL

Volume II, Number 3 $2.50 a copy, $15.00 a year October, 1987

This Reaction Solves Nothing!

WHY TRASH YOUR H/Z 8-BIT MACHINE

Just because "everyone loves that little tramp"?!

The *SEBHC JOURNAL*

Is Rapidly Becomming Your Only Source For
Authentic H/Z 8-Bit Help and Information!

RENEW YOUR SUBSCRIPTION OR START A NEW ONE

TODAY!

Send Name, Address, Computer Type, and Your
$15 Cheque or Money Order (1-yr subs) to:

The SEBHC JOURNAL
895 Starwick Drive
Ann Arbor, Michigan 48105

(Big Hint: Christmas Is Coming...SOON!)

--==<<8>>==--

SOFTWARE ARTICLE

Extending Useful CP/M Operating System Lifetime
With the Condensed CPM/DOS Package (c) 1987
by
William S Derby
Livermore, California

(The Derby CPM Utilities)

Despite its' many shortcomings, the CP/M operating system set the pace for early microcomputer system development. It is still the primary system used for 8-bit machines with a limited memory capacity. For various reasons, many of us are reluctant to part with our outdated though still capable 8-bit machines. Sadly, we have been left for the most part to fend for ourselves in a world dominated by bigger, faster and more modern microcomputers.

CP/M's shortcomings are even more emphasized when compared with features found in MS/DOS and other systems designed for machines with a much larger memory capacity. Expanded CP/M systems such as CP/M Plus or ZCPR3 offer a greater range of features, but they reduce the usable memory (by about 4K for ZCPR3); and they usually require extensive modifications and complicated installation procedures that are complicated even more if a system has already been modified to suit individual needs.

My approach to extending the CP/M operating system's useful lifetime has been to develop a package of programs with those features needed to overcome most of the frustrations I've experienced with the system, while still maintaining complete compatibility with the original CP/M 2.2 system. My condensed CPM/DOS package of programs--The Derby CPM Utilities--enriches a conventional CP/M environment in a simple and straightforward way without requiring modifications of the original system. The package is written in assembly language so as to keep the programs small and compact (6K bytes on disk). It was programmed and tested on a Heath/Zenith H89 computer, but it adheres completely to CPM/8080 conventions, so the condensed CPM/DOS programs should run in any CP/M environment that is compatible with Version 2.2 of CP/M.

Four programs comprise my package: SUB, SD, CMP, and COPY. Each provides a distinct capability in a form that is appropriate for but not limited to small machines with limited disk capacity. And all execute at speeds very close to optimum.

The SUB Utility Replaces SUBMIT

SUB replaces the SUBMIT facility. It emulates SUBMIT.COM for SUB files, but it also allows packing several SUB files into and invoking them directly or listing them from a single master file named SUB.BAT. In interactive mode SUB behaves much like the DOS command structure since it accepts CCP, COM, SUBMIT, or SUB.BAT commands without any qualification. SUB normally accepts multiple commands on a line, but this and other environment features such as allowing lower-case files can be enabled and disabled by special configuration commands.

In its simplest form, SUB is an exact replacement for SUBMIT; change its name to SUBMIT.COM and it could be substituted for SUBMIT with no noticeable differences; but when SUB's special features are invoked, capabilities that were not previously possible emerge. My two greatest frustrations with SUBMIT have been having to retype the SUBMIT qualifier for each command, and having to provide a separate file for each command sequence. Both these have been relieved by SUB.

Ideally, it should be possible to indicate a sequence of command line sequences on the same line without any qualification except a command sequence separator. This is exactly what is allowed by SUB; and when the interactive mode is used, it will return for the next command or command line sequence when the previous sequence is finished. The interactive mode is invoked by starting SUB without any commands or with a null command sequence. A subsequent null command sequence will terminate SUB. SUB operates by building a single $$$.SUB file containing all commands of all command sequences requested, plus an additional command to reinvoke SUB if the interactive mode is in effect.

The $$$.SUB file is processed by the CP/M system exactly as if it had been created from a single SUBMIT file - thus SUB operates within the framework of a standard CP/M 2.2 system. SUB is vulnerable to most of the same pitfalls as SUBMIT. For example, an illegal command, a missing COM file, or a deleted $$$.SUB file brings either to an abrupt halt. Chaining and XSUB commands are allowed by SUB, but both will normally cause the interactive mode to terminate at the end of the sequence. The interactive mode resumes after a chained command sequence only if it is reinvoked by a SUB null command sequence; and after an XSUB command only if SUB has been configured for XSUB resumption.

The SUBMIT facility is most useful when it is provided with a rich selection of command sequences tailored to suit individual needs. The standard CP/M SUBMIT facility requires each command sequence be contained in a separate SUBMIT file. On a machine with limited disk capacity, this tends to fill up the disks as well as the file index with several small files, often wasting at least 1K of disk space for files containing less than 100 characters. This problem is eliminated when all the most frequently used command sequences are combined in a single SUB.BAT file.

Within a SUB.BAT file, command sequences are identified by a line with the command sequence name preceded by a separator character. The SUB.BAT file is an otherwise normal ASCII file containing a collection of SUBMIT command sequences. Because it is a single file, it is easily maintained, and

CP/M ARTICLE Continued

requires only minimum disk and file index space. Maintainence of the file is facilitated by a SUB directive to echo a short list of the names of all command sequences in the SUB.BAT file, and by a directive to echo text of any indicated command sequence whether it is in the SUB.BAT file or in a normal SUBMIT file.

SUB handles $n sequences the same as SUBMIT and ^X sequences as described in SUBMIT's documentation. Blank lines, null commands, and the CCP directives (except DIR) with no parameters are ignored by SUB. SUB also excludes commands with more than 127 characters and sequences of commands with more than 2K characters as these are not processed correctly by SUBMIT and by the CP/M system.

SUB's adaptability is enhanced by a number of configuration commands which allow selection of the drive used for the SUB.BAT and SUB.COM files, selection of separator character used between command sequences, $$$.SUB file placement, continuation of XSUB command sequences, recognition of lower case characters, relaxation of error detection, and suppression of multiple commands on each line.

SUB inspects each command line to see if it is a normal CCP directive (such as DIR), name of a COM file, a normal SUBMIT command (but without the SUBMIT qualifier), a similar SUBMIT command for a sequence within the SUB.BAT file, or a sequence of any of these commands that are separated by separator characters ('/' character by default). Thus the SUB utility provides a rich selection of commands and allows maximum flexibility with minimal keyboard input.

The SD Utility Replaces STAT and DIR

SD replaces CP/M's STAT utility and prints a list of files, or of a subset of the files on a disk. For each file listed, it tells its size and number of unused sectors at the file's end. A summary of space used by the files follows the list. SD Directives allow listing four or less files per line in alphabetic or directory order by name or by extension. A user summary directive, as well as directives for changing a file's status are also provided.

CP/M's DIR command doesn't indicate file attributes and won't list system files. STAT does list these, but it prints only one filename on a line. This quickly fills the screen, making it impossible to view a list of more than 20 files at a time because more just scroll off. Also, neither DIR or STAT allows any selection of the order in which files are listed.

SD solves all these problems by listing names and attributes (length, R/W/S status, and unused sector count) of all files indicated in the order requested. Files are normally listed four on a line, but fewer files on a line may be requested. SD may be configured to list files in any desired order and to list any number on each line by default.

Like STAT, SD can list all active and inactive users on a disk, and change a file's read/write/system status. Unlike STAT, SD correctly indicates used and free disk space even when a disk has been inserted since the last warm start. STAT directives not related to files have not been implemented in SD as they do not seem appropriate in a program designed for listing files.

To minimize keyboard input, SD interprets a null or blank file name or file extension as a '*' character. For example, ABC. becomes ABC.*, and .COM equals *.COM. SD's disk summary directives are slightly abbreviated from those of STAT. For example, 'SD A' summarizes active or inactive users and occupied or free space on drive A, while 'SD B:' lists names and and attributes of all files on drive B.

Finally, SD lists file names the way they are typed instead of separating the file name from its extension as in STAT and DIR. Here's an example of SD output (in alphabetical order, 3 files on each line):

```
BIOS.SYS  5K*4      PIE.COM  7K-4      SUB.BAT  1K-4
 CMP.COM  1K-0      PIP.COM  8K-6      SUB.COM  2K-0
COPY.COM  1K-0       SD.COM  2K-0    XSUB.COM  1K-2
 9 FILES  28K USED        62K LEFT
```

The CMP Utility Adds Comparison Capability

CMP compares the respective bytes of two indicated files up to the shorter file's end. It lists the first 30 (or more if indicated) differences in ASCII or hexadecimal. Directives allow comparisons in a binary mode, and the specification of a pattern byte to search for in the file (or in a file compared with itself). Number of bytes compared, differences found, and line number of the first difference are reported when the comparison is finished.

CP/M doesn't have comparison capability, and the various public domain and other programs I've used don't do very much along this line. On the other hand, the DOS comparison program requires that files to be compared are of equal length, and it indicates only the first 10 differences, using four lines to list each difference in hexadecimal form. CMP accepts files of different lengths and it lists five differences on each line. It reads alternate 1K blocks from each file, and indicates the first 30 differences found. CMP does not accept ambiguous file names, but the drive identifier is sufficient to indicate a second file with the same name as the first.

In ASCII mode (the default except for COM files), comparison is limited by the end of the shorter file, or by an ASCII 'EOF' character found in either file. Differences are indica-

CP/M ARTICLE Concluded

ted in ASCII (except for non-printing characters, which are shown in hexadecimal). When binary mode is chosen, comparison is limited only by the end of the shorter file; and all differences are indicated in hexadecimal. In the following example of the output from CMP:

```
2K:02A3  A 0D   2K:02A4  8 0A   2K:02A5 0D C    2K:02A6 0A D
7K    4 DIFFS ( 89  323  323)
```

Seven K-bytes were compared, 4 differences were found, and both files contained 323 lines. The first difference was at line 89 in both files; the first 2,723 = 2K + 2A3(hex) bytes compared exactly. The first difference was an A in the first file specified and a carriage return = 0D(hex) in the second file.

The COPY Utility Supplements PIP

COPY converts a DOS copy command line to a PIP command line and transfers control to PIP. It does not really provide any capability beyond the normal PIP command, but it accepts the more natural DOS copy command sequence. A directive exists to tell COPY to delete a file with the same name as the destination file to make more disk space available before copying the file.

Maybe it is just force of habit, but I find the DOS copy command sequence more natural than PIP's sequence. The COPY utility accepts any combination of the DOS or PIP copy directives appended to the source specification in either the DOS form (separated by a /) or the PIP form (surrounded by []'s); and the destination specification may be omitted if it is a file with the same name as the source on the default drive. When a COPY command line is preceded by a : character, the converted PIP command line is echoed to the console without being processed. Some examples of COPY command lines follow:

```
COPY A.B       becomes PIP d:=A.B (copies to default drive)
COPY A:B C       "     PIP d:C=A:B (both names unambiguous)
COPY *.* B:      "     PIP B:=*.* (source spec is ambiguous)
COPY B:C.D A:    "     PIP A:=B:C.D (unambiguous source name)
COPY E.F G.H/D   "     PIP d:G.H=E.F (destroys existing file)
COPY A:I.J+K.L   "     PIP d:I.J=A:I.J,K.L (concatenate file)
COPY M.N+B:P/B Q "     PIP d:Q=M.N,B:P[O] (concatenate files)
```

The Derby CPM Utilities Extend System Usability

These programs bring the CP/M system much closer to the 8-bit machine's maximum usefulness by not only taking less disk and memory space than the CP/M functions and utilities which they replace. Those H/Z users still willing continue with CP/M should find life much more pleasant when they use Derby's CPM Utilities.

In addition to these programs, I've nearly finished developing a replacement Command Line Editor for CP/M on the H89. This increases the size of the BIOS by 1/2K to 1K; it's incorporated into the BIOS (and the BDOS). One of its many features is it's ability to correct a command line without retyping the whole line. Stay tuned!

The Condensed CPM/DOS Package of programs and documentation is available from me (the author) for $12.00 (postage included in the U.S.) in standard H89 hard, or soft sector 48trk CP/M format. It's also available in standard 48trk formats of most other CP/M machines. Send your order--indicate preferred disk format--to: W.S. Derby, P.O. Box 2041, Livermore, CA 94550.

--::<<8>>::--

"How's *that* for user-friendly?"

A "PORTABLE" CP/M COMPILER

OR

Confessions of a Game Hacker

by

Tim Brengle

University of California
Los Angeles, CA 90024

This document describes Dungeon Definition Language, a meta-adventure specification language. It is designed primarily for the programmer who wishes to create a DDL "world" and secondarily for the programmer attempting to implement DDL on a new host machine.

There are many (many!) reasons for wanting and setting up a computer at home. Mine was: I needed a terminal which I could daily connect via phone lines to the mainframes at work. My rationale was that with a computer ("terminal", that is) at home, at least some of my time was available to spend with my growing family. I program for a living (mostly compilers and data-base management tools). Rather than spending late nights in the lab hacking away on my employer's mainframes, I could come home and at least SEE the kids. The truth of the matter is more complex, so a little personal history is necessary to clarify things.

I was the first of our family (which includes the famous Whitebeard himself) {Big Deal! His fame has spread all the way to his sons!--Al Brengle} to become involved with computers. It started with a Tic-Tac-Toe program I wrote in high school, and continues even today.

Probably one of the biggest turning points in my life occurred while I was in college working for my degree in abstract math; I found ADVENTURE! It was the original version, written entirely in FORTRAN. All its data and text was stored in core (yes, real core), and it was the most amazing thing I had ever seen. While working as a consultant in the computer center, I had the opportunity to make some modifications. Then a few more. And then just a couple more... You get the idea. My wife calls it the "just-one-more-compile, honey" syndrome. I was hooked for life. I also found out that I had a lot more fun writing and modifying games than playing them.

After leaving college I taught in industry for a while, learned a dozen or so [computer] languages, wrote compilers and data base management systems for various employers, more or less catching up with the state of the art in software development during the next few years. And I never lost my love for computer games, especially the adventure types.

Recently a friend gave me a copy of a strange and wondrous document. The cover sheet looks something like this:

A Brief Description of UCLA
Dungeon Definition Language (DDL)

Bruce Adler
Chris Kostanick
Michael Stein
Michael Urban

What choice did I have but to implement DDL? It wouldn't do to "waste" resources (or time) at work, so it would have to be an after-hours-at-home project. The next question was: On what machine? The answer: My trusty H-89 computer--I mean "terminal".

Ever wanted to write a compiler? Or just know something about how it's done? If so, keep reading. My dad, Al Brengle, Editor of the San Diego HUG's newsletter, convinced me {He said, "Dad, I think I've got a book in me."--a.b.} that I should expound upon my compiler-construction knowledge. The rest of this article and subsequent articles will tell you of my experience with DDL and its successors to illustrate and explain some techniques and considerations in the design and implementation of a compiler.

At the start, the first thing I did was estimate how long it might take. (I recommend that ALL programmers do this.) Past experience on the job taught me that a compiler isn't simple to write. Having made up my mind to work at home on the compiler, I was prepared to devote as much time as possible to the job but only after my two wonderful small children were in bed, and on weekends. This decision relieved me of any feeling of pressure to get the job done, which I might otherwise have had.

I next decided the compiler should be reasonably "portable". If I later wanted to sell it or put it in the public domain, it would be nice to have it run on as many machines as possible with a minimum of patching.

Then I considered the available implementation language choices. I decided that assembly wouldn't do, even though it produces very fast and small programs. Programs must be portable, easily debugged, and fairly understandable and readable at some future date. That left only the high-level languages.

Wait a minute! How can you write a *COMPILER* in a high-level language? Isn't that similar to trying to lift oneself by one's bootstraps? Not really; I already had available several compilers written by other computerists which generate code for my H89. And a compiler is really just a big program. Like most other programs, it really doesn't matter which language it's written if you have a translator for it.

COMPILER Continued

My available choices were: FORTRAN, BASIC, FORTH, LISP, Pascal, and C. I had (or could easily get) translators for all of these. (The set of all translators properly contains the union of the set of all compilers, and the set of all interpreters.) Let's now consider each in turn.

I would have to buy the FORTRAN compiler. FORTRAN doesn't allow recursion. One down.

BASIC is nearly universal on microcomputers, so would seem a good choice for portability. But it too doesn't permit recursion, also it tends to generate large "object" programs. One down.

I skipped FORTH because of personal aversion brought on by what I once saw actually happen: Person X approached person Y and showed Y a short listing. I was nearby, but even though I couldn't read the characters I could see there was only one line on the page. X challenged Y: "I bet you can't tell what this FORTH program does!" X was right, for Y looked at it for a half hour, then finally gave up. One more down.

What about LISP? I like LISP. I tend to design in LISP and then code in whatever language is handy. Interpreters and compilers are particularly easy to create quickly in LISP. I even had one of the best microcomputer LISPs then available. The only reason I had for not writing my program in LISP is that it wouldn't fit my H89's 64K memory. Sigh... Another one down.

Both the Pascal translators I have are interpreters. That wasn't bad (as I hope to show later!) but neither has a reputation for creating either fast or compact code. My only objection to Pascal is that really portable methods of string manipulation are ugly. One more down.

I have two different C compilers (BDS C and C/80) which generate reasonably acceptable code. Also both are fairly rapid. My final choice was the BDS C compiler over C/80, because the intermediate '.ASM file created by C/80 wouldn't fit on my (then) 90K hard-sectored disks.

(At the project's beginning, I programmed so that either compiler would be able to handle the code for a long time. Another aside: Those of you familiar with BDS C know that code written in it isn't all that portable. To this I plead: "I never said that I was completely consistent!" The only other excuse that I can offer is that BDS C compiles something like 3 to 5 times faster than C/80. With the Kres speed-up module which I got this last Christmas, AND running entirely from RAM disk, BDS takes about 7 minutes to compile the entire object program. Before the Kres mod and CDR Super RAM board, it took about 20 minutes.)

After getting my system running and debugged, I started work at Hewlett-Packard Company. There I discovered a man named Ross Cunniff had independently created a system very much like mine, but in UNIX C. In one of our "enhancement sessions" we came up with an idea (to be explained later) which radically changed the original DDL design. That, coupled with a host of other non-trivial changes, led us to rename it "The Adventure Definition Language". It still has some of the "flavor" of the original, but has evolved considerably.

Before discussing pertinent specifics of the ADL language, I shall clear up a few things about which I may have earlier been unclear.

First, I referred to some languages as being (or not being) recursive. While it is true that all of the languages that I mentioned do allow for recursion, some require you to handle most of the details yourself. The technique I planned to use for parsing is called "recursive descent". That means it'd be easiest to use a language which allows recursion "naturally". (I'll discuss parsing and the recursive-descent technique in more detail in a subsequent installment.)

Second, I said that I like LISP and tend to design in LISP and code in whatever language is handy. To elaborate, LISP does a lot of things for the programmer such as handling the symbol table (more on that later, too). The LISP environment is also very congenial for program development. Functions need not be declared (or even coded!) before they are called. The same is true for variables. Variables can also hold any value type (and the value types may change). Also most LISPs provide useful debugging features such as dropping into the interpreter upon an error occuring, allowing easy variable inspection, etc.

All this is true when actually programming in an LISP environment which I knew was not going to be the actual case. Why still design in LISP then? Because of a mindset (no, that is not a new telepathic Walkman!) common to lots of LISP programmers.

LISP programmers are accustomed to programming in top-down steps, usually twenty-four lines (one screen full) being considered good. They also break up the problem into parts of increasing difficulty, that is, start with the easier portions, get them out of the way, then break up the harder parts into small, easy to handle segments and plug each into the overall program at appropriate points. Actually, this type programming style can be applied to any language.

LISP encourages this approach by allowing you to code calls to functions that don't yet exist. When the not-as-yet-defined functions are called, you are dropped back into the interpreter and can verify that all is well up to that point.

I "just naturally" use LISP top-down design techniques and

COMPILER Continued

steps, usually twenty-four lines (one screen full) being considered good. They also break up the problem into parts of increasing difficulty, that is, start with the easier portions, get them out of the way, then break up the harder parts into small, easy to handle segments and plug each into the overall program at appropriate points. Actually, this type programming style can be applied to any language.

LISP encourages this approach by allowing you to code calls to functions that don't yet exist. When the not-as-yet-defined functions are called, you are dropped back into the interpreter and can verify that all is well up to that point.

I "just naturally" use LISP top-down design techniques and so build code that is HIGHLY modular. That (usually) leads to more readable, modifiable, and understandable code. When I code my design in whatever language is appropriate, I do have to provide the subset of LISP function which I can use without defining--mostly symbol table handling. But onward to the ADL discussion:

Consider what appeared on my H-89 screen one evening:

```
You are standing outside the north entrance of a large
brick building. Inscribed above the doorway appears the
text: 'AARDVARK'S MUSEUM -- GATEWAY TO ADVENTURELAND'.
There is a coil of rope here.
There is a shovel here.
There is a carbide-flame lamp here.
There is a copy of a newspaper here.
> take rope then south
OK
You are in a large rotunda of an old museum. Doors lead
to the north, south, east, and west, and a narrow stairway
in the north-east corner of the room leads down.
There is a ball-point pen here.
There is a slip of paper here.
> take paper and pen east
OK
OK
You are in a dimly lit room containing an empty display
case. A portion of a vandalized sign above the case
reads:
'ARTIFACTS OF ANCIENT INDIA -- Several of these items,
including the sacred rhinoceros horn, the deadly ...'
The rest of the sign is unreadable.
To the west, you can look through a large door into the
rotunda of the museum. On the east wall of the hall there
is an outline of an arch.
> sign paper
In a blinding flash of light, a stone archway appears in
the east wall!
```

This scenario is familiar to those who've played ADVENTURE, ZORK or any other adventure-like games. It's exactly the thing for which ADL was created. ADL is designed to make the creation of adventure games easier. AND it was defined with portability as a major criterion. That is, ADL code created for an H89 should run on any other machine which has the ADL system, and it should run with no need of source-code changes.

Components of ADL:

There are five programs that make up the ADL system for the H-89: compiler, executer, debugger, message file encoder/compacter, and the message file dumper.

The compiler takes ADL source code and translates it to something which the executer can interpret. WHOA THERE! I thought you said this was a COMPILER!!

It is. It translates source statements into machine language. It just happens that it is not Z80 machine language. For reasons of compactness, ease of code generation and so on, the compiler produces code for a very simple stack machine. Then the executer simulates that stack machine. It is quite similar to the famous UCSD Pascal Compiler which produces p-code. (Don't panic. I will explain my reasons later!)

The debugger is really nothing more than a tool for looking at code and data after its' compilation into intermediate form.

The message file packer allows lots of text to be put into a comparatively-small space. By making some assumptions about the kinds of text that occur in ADL programs, it packs eight-bit characters into about 5.5 bits. (If anyone is interested in how ADL goes about doing that, tell me and I'll write an article about it.)

The message file unpacker decodes strings that the packer squished into readable form.

I'm writing mostly about the compiler (because that's where most of the fun stuff is), with references as necessary to the way that the executer works. (Contact me as described at the end of this article portion if you want more details.)

The ADL Language:

In general ADL is object and verb oriented. That is, most of the code in an ADL program has to do with specification of objects, and actions associated with verbs. A moment's thought about adventure games shows that this is also what the player is interested in. So it makes sense that a game should be written from this point of view.

Here's an explanatory quote from "A Brief Description of an Adventure Definition Language": "Using ADL, a programmer may specify Objects, Verbs to act upon those Objects, and Routines to describe the behavior of Objects and Verbs."

Conclusion, COMPILER Part 1, and LETTERS

In subsequent installments I shall concentrate on the syntax (form) and semantics (meaning) of the ADL routines, for they are what the compiler translates into code sequences. I'll refer to (and define) other pieces of the language as is necessary, but it is my intent that you should be able to understand the discussions without your having the above-mentioned document in hand.

Here's a preview of upcoming topics:

* What are the parts of a compiler?
* What is "recursive descent"?
* How does one read and create BNF?
* What *IS* BNF?
* What is a symbol table, and how are they used?
* An ongoing discussion of design trade-offs
* And more!

See you in the November SEBHC JOURNAL!

--=:<(8)>:=--

[Editor's note: This is part one of the 3-part article originally appeared in the San Diego, CA HUG's DUP & DUMP newsletter. It is presented here (in slightly edited form) with the permission of both its' author and DUP & DUMP, to whom we offer our most sincere thanks! You may contact the author either by writing us, or directly. His address and phone numbers are:

Tim Brengle	Residence 408-370-6853
961 Lovell Avenue	Office 415-477-5382
Campbell, CA 95014	hplabs!brengle via CSNET

Please keep all correspondence short and to the point.]

==========

LETTERS... LETTERS... LETTERS... LETTERS... LETTERS... LET

==========

Dear Mr. Geisler:

Words cannot express how delighted I am that someone has taken over supporting the original 8-bit Heath Users. Please accept my subscription (cheque enclosed) for one year. I was introduced to your publication by a fellow H-8 owner at work.

He lent me his back issues and I built the H19 Auto Key Repeat circuit submitted by Jim Holt (Vol I No 10). It works fairly well except that it will only repeat about ten times, then pause, repeat 10 times, pause, repeat 10, etc. By adjusting the pot for continuous repeating I get running keys during normal typing. I think the problem is a marginal C1, but haven't gotten around to trouble-shooting it yet.

I was surprised to find that the local Radio Shack had all the parts I needed for once; but I found that some of the part numbers [listed in Jim's article] didn't match. Also ethier there is an error in the keyboard pin numbers in the figures or there are two different connector types used on H19s. My connector has only 34 pins, and I found where the figures referred to pin 33, mine used pin 31; 36 in the figure worked out as pin 34 on mine.

In a JOURNAL back issue I saw a loan amortization program advertised for about $30. If any JOURNAL subscriber needs such a utility, I've enclosed the listing of my own CP/M MBASIC LOAN.BAS (which I worked out independently for my own use--avoids having to pay someone to do my calculations). [Listing on page 12.] With this program it is possible to schedule any desired interest rate, number of payments, or payment amount and get a hard-copy printout of the schedule. SEBHC JOURNAL subscribers can either copy this listing, or get it on disc from me for $6 in both CP/M and HDOS MBASIC form (hard-sector discs only). For just $3 I'll copy the program to their blank disc.

GARY S MELANDER, 460 Garrison Pl, Virginia Beach, VA 23452

[Thanks for the subscription, letter and MBASIC listing! The listing you had on the disc you sent us printed out ok, but you must have some different kind of text formatter working for you. When I tried to reformat the text of your letter and sample program printout to fit our two-column layout, my text processor (TXTPRO V4.1) refused to keep hardly any lines on screen! I finally was able to convince TXTPRO that it was ok to show me what was funny about your ASCII files. Guess what! Almost very word has one or more inverse video characters substituted for the word's last letter. This has also happened to me with discs which other authors also using non-dot-matrix printers have submitted. (I've also had this happen with files downloaded from CompuServe's HUG SIG!) It appears that printer justification control codes embedded in your text formatter's ASCII file are causing the problem. I hand-entered your letter and sample printout so that we could rush your important contribution into our subscriber's hands.--ed]

Dear Mr. Geisler,

Using an '89, I made some "flippies" to gain additional disc space with very satisfactory results. But at a recent local HUG meeting I was advised against doing this. The claim is that flipping a disc results in rapid disc wear at the hub, with the head getting fouled up with fine dust and debris. In addition, the disc's counter-rotation allegedly can result in disc warpage.

Is a new disc drive the only way to expand disc space? I shall appreciate any comments on "flippies" from you or other members.

ANTHONY P MUSNICK, Broomall, PA 19008

[Flippies are ok IF they're *NOT* in constant use. Get a new

LETTERS Continued

another disc drive if you possibly can. There are any number of full-size, double-sided, 40trk Tandon or similar drives now advertised from seventy-nine to $129 in REMark, Sextant, Radio-Electronics, and other computer/electronics magazines. Consider that it is better to spend a few bucks to expand your drive capacity than lose even one disc full of vitally-important data files, not to mention the wear and tear which could kill your drive's head!--ed]

Dear Mr. Geisler,

Keep up the good work! I enjoy the SEBHC JOURNAL very much. I have let my subscriptions to BUSS, REMark and Sextant run out...NOT ENOUGH 8-BIT INFORMATION.

I have a problem that maybe you or one of your readers have experienced: I've had my H89 since 1981, use Video Scribe (H-DOS) and Wordstar (CP/M). After typing for an hour or so the keyboard hangs up if I type too fast. The only way around this is to re-boot. This has caused me to save my edited files after a few lines of typing.

I've also another problem which may be related to the first one: With OFF-LINE key up I powered up and got only one beep; no "H:". I tried SHIFT-RESET and got no beep, no "H:". But with OFF-LINE key DOWN, keyboard worked fine, displaying any character I typed.

I have the CDR controller with two half-height drives internally mounted, and the H17 controller with 3 external drives. Also I installed a modified HUG/Watzman Ultra (keyboard) ROM back in 1983. Perhaps one of these units is causing the trouble?

Any help shall be greatly appreciated!

JOHN McWILLIAMS, 1912 Morse St., Houston, TX 77019 -- Phone: 713-522-5975.

[Hmm! I've had similar problems with my H-19 terminal which runs my good old H-8. After lengthening the cooling slots to within 3/4-inch of the cabinet's back and installing an H89 cooling fan facing downward, the problem went away! This doesn't mean that yours has a similar cause, but it shouldn't be overlooked. It also is quite probable that your terminal board isn't being recognised by the CPU board either because of a broken wire in the interconnecting cable between them, or from corrosion buildup on connector pins at either end. This I've also seen in older H/Z89 and '90s. Just remove and replace the connectors a few times--it won't hurt anything. Now how about some of you other readers helping John? -- ed]

Dear Mr. Geisler:

I've been reading your excellent JOURNAL for several months plus all of the back issues and have learned a great deal, but unfortunately, I'm still very frustrated with my '89s. I purchased CP/M-80 GAMES DISC #0 and the HDOS "Programmer's CARE Package" Disc #0, both hard-sector 40trk. Thus far I've been unable to get either one to work because I am inexperienced in both CP/M and HDOS use. This is because all my computer software is in "package programs" (Autoscribe, etc.). And I don't know how to get beyond the BOOT command!

Recently I purchased a Z89-11 I/O card for use with a parallel dot-matrix printer (our Diablo 630 daisy-wheel printer is good, but...), and Micronics Technology's H89 2/4 Meg Speed Mod kit. I installed both, but haven't been able to use their accompanying software to activate them. I have CP/M 2.204, HDOS 2.0, and a BASIC-80 disc and manual, which also are usless to me until I learn something about them. I'd like to use my '89 for something more than just forms and word processing!

Earlier you had recommended HUG's MAPLE modem program which we bought and got working with a data network very quickly. It works great--except for printing, which we're still trying to figure out.

This may sound unbelievable to you and your readers, but I NEED HELP! I'm willing to pay for some expert time from someone to help me get started [computing]. If any reader in the Kankakee, Illinois area is willing to sell me a little of their time, please contact me at once! My address and phone number are below.

ANDY BRORSEN, 1235 Stanford Drive, Kankakee, IL 60901 -- Phone: 815-939-4930

[Wish I lived nearby, Andy! But perhaps one of our more than twenty IL subscribers lives close enough to you that they can drop by and help you "unscrew the inscrutible". Keep us all informed, and good luck! -- ed]

Dear Mr. Geisler:

I've read about your organisation in both Sextant and Family & Home Office Computing. I'd like to join the Society and receive the SEBHC JOURNAL. I own a Heathkit H89A which I built only a year ago. I'm still learning how to use it, so I'd love to read what others have to write about the machine, and perhaps share a few things I've learned with them. I also wouldn't mind knowing something more about the H8. I'm still amazed by what one can do with [only] eight bits and 64k. My cheque for $15 is enclosed.

KARL G RULING, Wilkes-Barre, PA 18702

[Welcome to you Karl; your subscription is now under weigh. I suggest you consider investing in a copy of Volume I. It has lots of good stuff about our favourite 8-bit devices which should be inspirational (if not entertaining) to you. And should you decide to write something for publication in the JOURNAL, by all means do so very soon. We're in constant need of fresh information! And your subscription will grow if we accept your submission. -- ed]

Dear Lenny,

I just received the JOURNAL and was surprised to find it marked "Your subscription has expired"! Somehow my subscrip-

LETTERS — WANT ADS — SURVEY RESULTS

tion shrank instead of growing after I'd submitted my article "AUTO KEY REPEAT for the H19/89" (I:10). Also, my name got mangled--somehow--from HolT to HolD! Can you get this fixed as soon as possible? Also, please explain JOURNAL policy on software submissions for inclusion on your "GAMES DISCs" as I have a few items from a while back you may find interesting.

JIM HOLT, Knoxville, TN 37919

[OOPS! We experienced an able-body office help shortage and yours was one of several mailing labels which were inadvertently marked "Expired" (one of my daughters was helping me get that issue out). It's all fixed now; check your label on this issue. Did you see the letter from another subscriber on page eight in connexion with your article? You two, get in touch! Now about your software: Sure, we'd like to take a look at what you've fashioned. But please make sure that any program you submit has been reasonably debugged, runs as you intended it to, and is well documented. If we think it is unusual enough and worthwhile to our readers, we'll include it on one of our discs and cut you in on the profits (if any) from sales. Also, first use our prices as a guide, then let us know what you'd like us to charge for your stuff. (We don't want to put people off by charging too much.) -- ed]

==========

FOR SALE... WANTED... FOR SALE... WANTED... FOR SALE... WANTED

==========

WANTED --
Need Z-37 high-capacity floppy disc system. Not necessary it has both drives. MARVIN F. ROBERTS, 4000 SW 28th St., Topeka, KS 66614-2207.

FOR SALE --
Heath 85-2731-1 16k MEMORY EXPANSION BOARD for H-89. Filled/w 4116 memory chips. Never used. $20

H89A CPU BOARD. Used briefly before being replaced by TMSI's H-1000 board. Like new, only 16k memory installed. $35

H89A TERMINAL LOGIC BOARD. Modified for 19200 baud communication with the CPU board. Purchased used, never installed. $20

VARIOUS SOFTWARE: dBASE-II, Peachtree General Ledger, Peachtree Accounts Receivable, Peachtree Sale Invoicing, CP/M BASIC 80, Peachtree Property Management, HDOS 2.0, HDOS BASIC Compiler, SUPERCALC, etc. All fully documentated and UNUSED. $25 for dBASE-II, all others $10 each. Call or write for more complete listing & available formats. JIM HOLT, 615-588-6967, 2109 Indian Hills Drive, Knoxville, TN 37919

FOR SALE --
At CHUGCON, our WIN89 20meg winchester hard disc, discounted by $100--that means you get it for ONLY $495! SEBHC JOURNAL readers, mention this ad if ordering before 30 November (You didn't make it to CHUGCON?!) and get the same discount. Also for sale, the WIN89 Interface Card with CP/M software for just $175 (HDOS software available shortly) which lets you install an IBM PC bus in your H/Z89 and permits use of Western Digital's Hard Disc controller. DARREL C PELAN, Micronics Technology, 444 Barbados Way, Niceville, FL 32578 -- Phone: 904-897-4257. * BE SURE TO MENTION THE SEBHC JOURNAL *

==========

RESULTS... SURVEY RESULTS... SURVEY RESULTS... SURVEY RESULT

==========

As promised, here are the results of our four-question survey of subscribers' opinion (97 readers responded--about 42%):

Question 1: Help defray cost of buying a laser printer?

YES [43] NO [55] Comment summary: Not really necessary, the Epson LQ-800 is doing just fine.

Question 2: Help pay for having a vendor laser typeset and print the JOURNAL?

YES [39] NO [58] Comment summary: Too many opportunities for mistakes, too expensive.

Question 3: Double JOURNAL price to raise money & pay authors rather than extend subscription by a year as compensation?

YES [48] NO [49] Comment summary: CON: Most authors should be happy to see their articles in print. PRO: You should be able to get more "professional" submissions.

Question 4: Charge vendors for ads to raise money to pay our authors and risk losing ability to fairly criticise vendors?

YES [12] NO [85] Comment summary: You're doing it right now; honest vendors have nothing to fear if they are selling honest goods and services!

The general opinion is: "Don't fix it if it ain't broke!" Surprisingly, many survey responses favored present editorial policy. The only "cliff hanger" was response to Question 3. Quality of articles we're now getting is very high--on a level similar to the old Dr. Dobb's Journal--why change? Thanks to you who took time to let us know what you think of YOUR newsletter.

--::<(8)>::--

TUTORIAL IN "C"

"C" The Light
by
Darrell C Pelan

[This is the first of four articles which give us a very good look at how to program in the C language. The rest will run in subsequent issues. You'll like them; I did! -- ed]

This month we examine a program which I wrote to take care of some annoying things that happen when you print out a cross referenced file generated by CREF.COM. The first is a form feed at the file's beginning which wastes a sheet of paper in my Prowriter. The next is tab spacing. I generally use the H19 tabs to set up columns for assembly language programs. If the tabs on the Prowriter are not set, textfile tabs are ignored. This results in a really compressed printout!

I wrote and compiled this program with Software Toolworks' C/80 V3.0 compiler. It's a good compiler but has a few bugs. The scanf function does not work correctly, and programs will occasionally start to re-execute rather than return to the system if exit() is not the last function called. That didn't happen in this program, but did in two others. Compiler Version 3.1 had these problems corrected. Now on to the program. This program layout is somewhat compressed to fit 2-column format. Usually I try to leave the comments on the same line as code they explain. Comments are anything between /* and */.

Define statements are used to make the program more readable. End-of-File (^Z) is defined as EOF. The compiler converts all references to EOF to decimal 26 at compile time. When you look at the source code though, all you have to remember is EOF for end-of-file rather than what 26 is. This is also true for the escape character called ESC. The only error checking in the program occurs with the use of the variables argc and *argv[]. Argc is an integer equal to the number of arguments in the command line. Argv is an array of pointers to characters. In C, character strings are implemented in single dimension arrays of characters, with the last byte equal to 0. So *argv[] is actually used as an array of pointers to strings. Remember that the command line includes the called program. Therefore argc will be one more than the number of arguments you want to see. In this case, I only want to print one program and test for (argc != 2). If argc is not 2 than PCREF prints a usage message and exits. An important side note here: == means a test for equity while a single = is an assignment operator. Looking for this error can save you a lot of debug time.

PCREF opens the file for reading, and the printer (LST:) for writing, using fopen(). A message is printed on the screen telling you that the program thinks all is well. The first character is always a form feed. It's read in but never used. Putc is used to send two bytes to the printer and sets it in compressed mode. The tabs are cleared using fprintf() which also sends two bytes. This illustrates that there are many ways to do the same thing in C. Fprintf() is then used again to set the Prowriter tabs to match H-19 tabs. A while() loop is used to read the file until the aEOF is detected. After the character is read and tested for EOF - while((c=getc(f1)) != EOF) it is sent to the printer with putc(). The integers f1 and f2 serve as channel numbers for communication with open() files or devices. Another important thing to note is (c=getc(f1)) is enclosed in (). This ensures that c is assigned a character before you test for EOF. PCREF then resets the printer to 12 CPI mode, closes the writing file and exits.

Listing 1 -- Print CREF

```
/*  PCREF - Prints a .prn created by cref.com to the
            C.Itoh printer.  The beginning form feed is
            deleted  and  the printer  is  set  to  the
            compressed print mode.                  */
/*          (c) 5 May 1984 - Darrell C. Pelan
           Public Domain use is expressly invited!!  */
#define EOF 26
#define ESC 27
#include <printf.h>

main(argc,argv)
int argc;
char *argv[];
{
int f1, f2, j;
char c;

if ( argc != 2 )
    {
    printf("\nUsage: pcref filename.prn\n\n");
    exit();
    }
f1 = fopen( argv[1], "r");
f2 = fopen("LST:", "w");
printf("\nPrinting cross referenced file %s\n",argv[1]);
c = getc(f1);           /* Get rid of Form feed */
        /* Put the printer in compressed mode  */
putc( ESC, f2); putc( 'Q', f2);
        /* Clear all privious tabs             */
fprintf(f2, "%c0", ESC);
        /* Set Prowriter tabs to match the H-19 */
fprintf(f2, "%c(009,018,027,036,045,054,063,072,");
fprintf(f2, "081,090,099,108,117.",ESC);
while( (c = getc(f1)) != EOF) /* Print file until end */
    putc( c, f2);
        /* Put the printer into 12 cpi mode   */
putc( ESC, f2); putc( 'E', f2);
fclose(f2);    /* Close the printer file */
}
```

CP/M MBASIC PROGRAM LISTING

```
5 REM          LOAN.BAS        by Gary S Melander
100 FOR X=1 TO 24
110 PRINT
120 NEXT X
130 DIM DT$(12)
140 DT$(1)="JAN"
150 DT$(2)="FEB"
160 DT$(3)="MAR"
170 DT$(4)="APR"
180 DT$(5)="MAY"
190 DT$(6)="JUN"
200 DT$(7)="JUL"
210 DT$(8)="AUG"
220 DT$(9)="SEP"
230 DT$(10)="OCT"
240 DT$(11)="NOV"
250 DT$(12)="DEC"
260 PRINT "This program will calculate loan payments based
 upon:"
270 PRINT
280 PRINT "     # of Monthly payments and"
290 PRINT "     Annual Percentage Rate (APR)"
300 PRINT
310 PRINT:PRINT "ENTER INITIAL PRINCIPAL ";:INPUT A#
320 PRINT "ENTER NUMBER OF MONTHLY PAYMENTS ";:INPUT N
330 PRINT "ENTER THE ANNUAL PERCENTAGE RATE (APR)";
340 INPUT R2
350 R1#=R2/12
360 P#=A#/((1-(1+R1#/100)^-N)/(R1#/100))
370 PRINT:PRINT "PAYMENTS WILL BE ";:PRINT USING
 "$$######.##";P#
380 PRINT "DO YOU WANT TO CHANGE PAYMENT AMOUNT? ";
390 A$=INPUT$(1):PRINT A$
400 IF A$="N" THEN 420 ELSE IF A$="n" THEN 420
410 PRINT "ENTER PAYMENT AMOUNT DESIRED ";:INPUT P#
420 PRINT "AMORTIZATION SCHEDULE DESIRED? ";:A$=INPUT$(1):
 PRINT A$
430 IF A$="N" THEN 930 ELSE IF A$="n" THEN 930
440 INPUT "DATE OF FIRST PAYMENT (DD-MMM-YY)";FP$
450 DT$=FP$
460 FOR X=1 TO 12
470 IF MID$(FP$,4,3)=DT$(X) THEN 500
480 NEXT X
490 PRINT "INVALID DATE!":GOTO 440
500 PRINT "ENTER HEADING DESIRED ON PRINTOUT "
510 INPUT AN$
520 PRINT "PAUSE BETWEEN PAGES (Y or N)?";
530 ST$=INPUT$(1):PRINT ST$
540 PRINT "READY PRINTER -- THEN PRESS <RETURN>";:
 A$=INPUT$(1):PRINT A$
550 GOSUB 1010
560 B=1
570 C#=A#
580 I1#=0
590 I#=C#*(R1#/100)
600 IF P#>C#+I# THEN P#=C#+I#
610 P1#=P#-I#
620 C#=C#-P1#
630 IF C#<.01 THEN C#=0
640 LPRINT TAB(10);DT$;:LPRINT TAB(20);
650 LPRINT USING "###";B;
660 LPRINT TAB(25);
670 LPRINT USING "$$#######.##";P#;:LPRINT TAB(37);
680 LPRINT USING "$$#######.##";P1#;:LPRINT TAB(49);
690 LPRINT USING "$$#######.##";I#;:LPRINT TAB(61);
700 LPRINT USING "$$#######.##";C#
710 I1#=I1#+I#
720 B=B+1
730 X=X+1
740 IF X>12 THEN X=1 ELSE GOTO 780
750 DT=VAL(RIGHT$(DT$,2))
760 DT=DT+1
770 DT$=LEFT$(DT$,7)+RIGHT$(STR$(DT),2)
780 MID$(DT$,4,3)=DT$(X)
790 IF C#<.01 THEN 880
800 LC=LC+1
810 IF LC<50 THEN 590
820 LC=0
830 LPRINT CHR$(12)
840 IF ST$="N" THEN 860 ELSE IF ST$="n" THEN 860
850 PRINT "PRESS <RETURN> WHEN READY FOR NEXT PAGE";
 Q$=INPUT$(1)
860 GOSUB 1010
870 GOTO 590
880 LPRINT:LPRINT TAB(10);"INITIAL LOAN   = ";:LPRINT
 USING "$$#######.##";A#;:
890 LPRINT " AT ";R2;"% APR"
900 LPRINT TAB(10);"TOTAL INTEREST = ";:LPRINT USING
 "$$#######.##";I1#
910 LPRINT TAB(10);"TOTAL PAYBACK  = ";:LPRINT USING
 "$$#######.##";A#+I1#
920 LPRINT CHR$(12)
930 PRINT "ANOTHER RUN? ";
940 A$=INPUT$(1)
950 PRINT A$
960 IF A$="N" THEN 1070 ELSE IF A$="n" THEN 1070
970 FOR Y=1 TO 24
980 PRINT
990 NEXT Y
1000 GOTO 310
1010 LPRINT TAB(25);AN$
1020 LPRINT
1030 LPRINT TAB(13);"DATE";TAB(20);"PMT";TAB(25);"AMOUNT";
 TAB(41);"PRINCIPAL";TAB(51);"INTEREST";TAB(64);"BALANCE"
1040 LPRINT
1050 LC=0
1060 RETURN
1070 END
```

Sample-run printouts on page 14...

Listing 2 -- SAMPLE CONSOLE SESSION WITH LOAN.BAS:

```
Upon initialization of LOAN.BAS:

     This program will calculate loan payments based upon:

          # of Monthly payments and
          Annual Percentage Rate (APR)


     ENTER INITIAL PRINCIPAL ? 1000
     ENTER NUMBER OF MONTHLY PAYMENTS ? 12
     ENTER THE ANNUAL PERCENTAGE RATE (APR)? 9.5

     PAYMENTS WILL BE $    87.68
     DO YOU WANT TO CHANGE PAYMENT AMOUNT? N
     AMORTIZATION SCHEDULE DESIRED? Y
     DATE OF FIRST PAYMENT (DD-MMM-YY)? 01-JAN-88
     ENTER HEADING DESIRED ON PRINTOUT
     ? EXAMPLE AMORTIZATION SCHEDULE USING LOAN.BAS
     PAUSE BETWEEN PAGES (Y or N)?N
     READY PRINTER -- THEN PRESS <RETURN>
     ANOTHER RUN?N

     OK
```

Listing 3 -- PRINTOUT FROM ABOVE SAMPLE RUN:

EXAMPLE AMORTIZATION SCHEDULE USING LOAN.BAS

DATE	PMT	AMOUNT	PRINCIPAL	INTEREST	BALANCE
01-JAN-88	1	$ 87.68	$ 79.77	$ 7.92	$ 920.23
01-FEB-88	2	$ 87.68	$ 80.40	$ 7.29	$ 839.83
01-MAR-88	3	$ 87.68	$ 81.03	$ 6.65	$ 758.80
01-APR-88	4	$ 87.68	$ 81.68	$ 6.01	$ 677.12
01-MAY-88	5	$ 87.68	$ 82.32	$ 5.36	$ 594.80
01-JUN-88	6	$ 87.68	$ 82.97	$ 4.71	$ 511.83
01-JUL-88	7	$ 87.68	$ 83.63	$ 4.05	$ 428.19
01-AUG-88	8	$ 87.68	$ 84.29	$ 3.39	$ 343.90
01-SEP-88	9	$ 87.68	$ 84.96	$ 2.72	$ 258.94
01-OCT-88	10	$ 87.68	$ 85.63	$ 2.05	$ 173.31
01-NOV-88	11	$ 87.68	$ 86.31	$ 1.37	$ 86.99
01-DEC-88	12	$ 87.68	$ 86.99	$ 0.69	$ 0.00

```
INITIAL LOAN    = $  1000.00 AT  9.5 % APR
TOTAL INTEREST = $    52.20
TOTAL PAYBACK  = $  1052.20
```

==========

OR SALE... WANT ADS... FOR SALE... WANT ADS... FOR SALE... WAN

==========

FOR SALE --

Sixteen (yes, 16!) Z90s with Z89-37 controllers, all in excellent condition for $300, each or make offer on the lot. When ordering, tell George you saw it in the SEBHC (pronounced SEB HIK) JOURNAL. George J Sellers, 1033 Bishop Walsh Rd., Cumberland, MD 21502 -- Phone 301-722-7950

SOFTWARE CLEARANCE --

Only one copy of each still in stock...while they last:

Item		Item	
#4 - CP/M BASIC 80,	$25	#12 - CP/M BASCOM,	$30
#8 - CP/M CBASIC,	$30	#14 - CP/M SPELL,	$30
#11 - CP/M Magic Wand,	$25		

Shipping not included. When ordering, say you saw it in the SEBHC (pronounced SEB HIK) JOURNAL. Darrell C Pelan, Micronics Technology, 449 Barbados Way, Niceville, FL 32578 -- Phone 904-897-4257

$525 H/Z-89 HARD DISC SPECIAL --

[_] WIN89 Interface Card, including parallel port, software, installation and operating instructions & parts, only $175!

[_] HALLOWEEN SPECIAL! Complete WIN89 Hard Disc Package ONLY $525 through 15 Nov., '87. When ordering, say you saw it in the SEBHC (pronounced SEB HIK) JOURNAL. Darrell C Pelan, Micronics Technology, 449 Barbados Way, Niceville, FL 32578 -- Phone 904-897-4257

==========

COMING ATTRACTIONS... COMING ATTRACTIONS... COMING ATTRACTIONS

==========

Next month's SEBHC JOURNAL will have a comparison software review of the CompuMagic Utility Package (CP/M), UTILISET V1.1, some common CP/M utility-like functions, and several ZCPR-3 utilities. If there's room (and time), we may also present a comparison review of CompuMagic's SEARCH and O'Neil Software's Electra Find utilities.

And we'll present the second installments of our C-language articles--which started in this issue--in November's JOURNAL.

There also will be some public domain software listings we finally have unscrewed (thanks to a vital hint from Al Brengle of White Beard Computer Consulting) we'd downloaded from the CompuServe HUG SIG library nearly a year ago! (Thanks for the help, Al!)

Hope to get the November issue to you before Thanksgiving!

SEBHC JOURNAL

DISCLAIMER

Reviews, editorial references, and advertisements in the SEBHC JOURNAL should not be taken as authorative endorsements of any products or services. Opinions expressed in the JOURNAL are based on the individual's experiences and shall not in any way be considered as official endorsment or certification, nor do they reflect intensive technical analysis as might be provided by a professional testing firm. Although we do not knowingly publish fraudulent materials, we shall not be held liable for any damages arising from purchase or use of any product. People having complaints about goods or services purchased from our advertisers are urged to send us written notification of their specific complaints so that we may take any action which we deem appropriate. *Caveat emptor!*

Detach before filling out & mailing

The Subscription & Order Blank

Name_____

Mailing Address_____

_____City_____

State_____ZIP_____
If not USA, Country_____

Phone number(s)_____

H/Z Computer Model(s)_____

Modem? Yes [__] Baud_____ No [__]

HDOS 2 [_] 3 [_] CP/M ver _____

Favorite Language(s)_____
_____

Computer used mainly for_____
_____

One Year Subscription [__] $15.00

Bound Set, Volume I [__] $22.50

Vol II Back issues @ $2.50ea
#1[_] #2 [__] $_____

CP/M GAME DISC #0 hard sect
$7.96 [__] soft $6.96 [__] $_____

HDOS 2.0 Programmer's CARE
Package Disc #0 hard sector
$3.66 [__] soft $3.00 [__] $_____

========

Order Total $_____

I have enclosed my cheque [__] or money order [__] in U.S. funds for the above. I realise back issues are posted by 3rd-class mail, and may be slow in reaching me.

Please make cheque or money order payable to L.E. Geisler. Thank you.

Tell All Your Friends About The *SEBHC JOURNAL!*

The SEBHC JOURNAL's Back Page

Society and Journal Policies

* The SEBHC JOURNAL is published twelve times a year and strains to get mailed around 22nd of a month. Editorial deadline: 20th of any month.

* All advertising is printed free of charge. Vendors will please submit B&W "camera-ready" ad copy, 7" wide by 9" high (one frest page/issue) no later than 15th of month in which it's scheduled to appear. All Society members are entitled to run one free--fresh--250-word Want Ad a month.

* Subscriptions are US$15/year in Canada, Mexico, the USA and its possessions (all others $25/yr) and start in month following receipt of application. Please make cheques or money orders payable to L.E. Geisler. Single back-issue copies by special order--allow 5 weeks for processing.

* Subscribers are automatically Society of Eight-Bit Heath Computerists members. Members' ID# and expiration follows the name on mailing label. The three member classes are: REGULAR (H/Z 8-bit user), ADVERTISING (one vote/vendor), ASSOCIATE (non-8-bit computerist, library, etc.). REGULAR members can vote and hold any Society office. ASSOCIATE members cannot hold office or vote in Society elections. The Society's official meeting coincides with HUG's annual conference; place and time announced in each July JOURNAL issue.

* The SEBHC JOURNAL is composed, edited and published by L.E. Geisler at 895 Starwick Drive, Ann Arbor, MI 48105. Phone 313-662-0750, 9am - 6pm Eastern Time, Monday thru Friday. Off hours, try 313-769-6052 (home).

SEBHC Journal
895 Starwick Drive
Ann Arbor, MI 48105

==========
== BULK RATE ==
== U. S. POSTAGE ==
== PAID ==
== PERMIT No.624 ==
== ANN ARBOR, MICH ==
==========