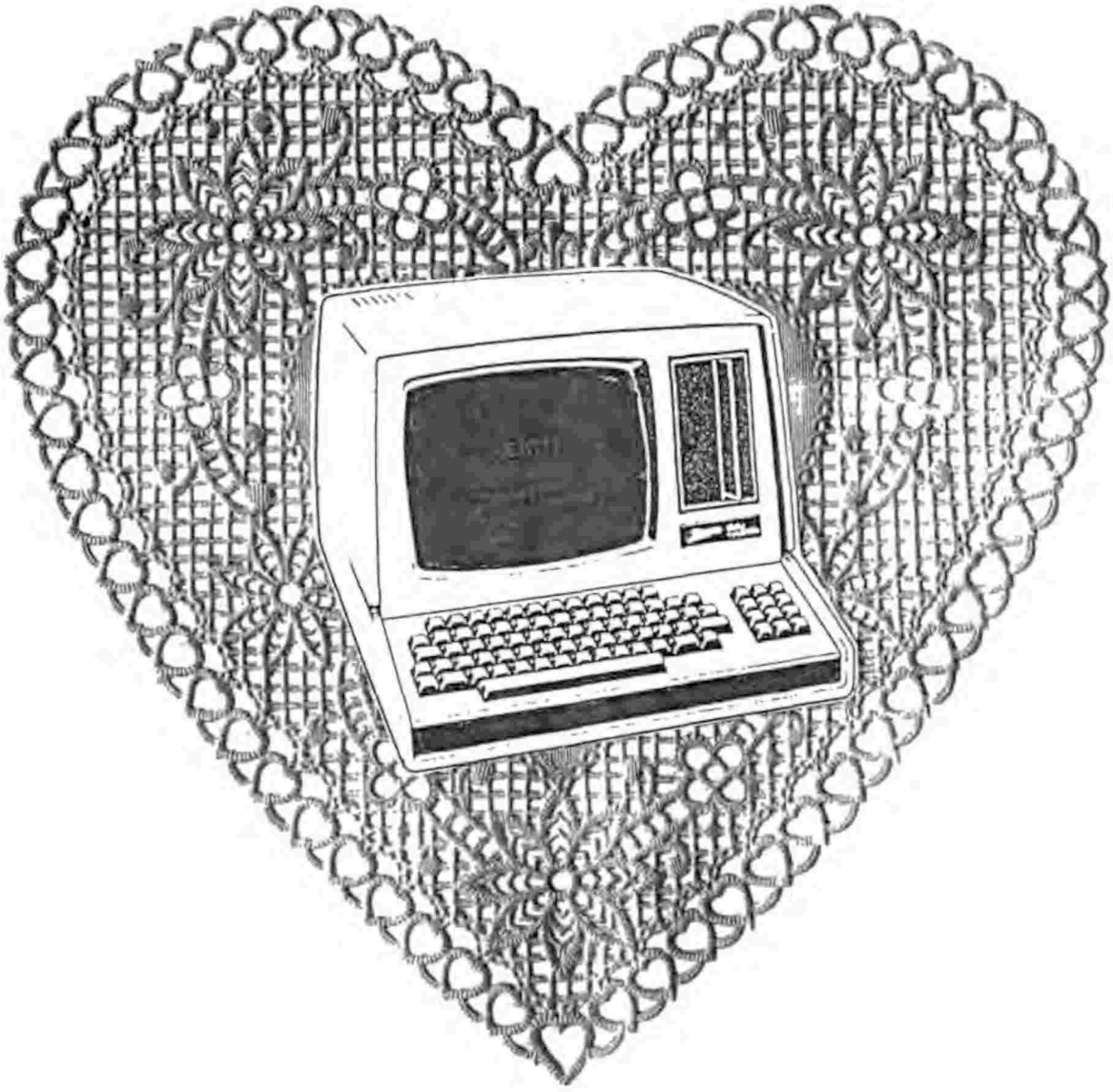
Saving Our HEATH Eight-Bit Machines!

SEBHC JOURNAL

Volume III, Number 6 $2.50 a copy, $17.50 a year January, 1989

OOPS! VOL. 3, NO. 7 FEB 89



Guess Who's OUR Valentine!

FOR HDOS LOVERS ONLY!

HDOS 2.0 ENHANCEMENT WITH A C/80 ROUTINE

by
Gary A Appel
1318 Old Abbey Place, San Jose, CA 95132

ABSTRACT

It is not generally known that the HDOS operating system supports use of control-character interrupt handlers for CTRL-A, -B, and -C keys. These handlers, if defined, are executed whenever the corresponding control character key pair is pressed. This article explains how to write and use specific interrupt handlers in conjunction with the C/80 compiler.

INTRODUCTION

Several years ago I decided it was time to buy a computer. Having grown up with Heathkits, the H89 was the obvious choice. I drove down to the local Heath store, and walked out with an H89 and the Heath Disc Operating System (HDOS). I later added CP/M to make use of the larger product base available, and eventually succumbed to MS-DOS and its' enhanced capabilities. But HDOS is still my favorite operating system.

HDOS has several advantages over CP/M. HDOS provides date stamping, loadable device drivers, interrupt-driven keyboard, and my all time favorite, control-Z. HDOS 3.0 has introduced the batch file (the one 'must' item missing from version 2.0) together with several other improvements. The combination of HDOS 3.0 and C/80 creates an environment where we can do almost anything.

Along with the interrupt-driven keyboard mentioned above, HDOS provides users with the ability to define three interrupt handlers for the control-A, control-B, and control-C characters. Any time one of these control characters is used, the appropriate handler is immediately executed. Programs can be written defining these handlers to provide instant interrupt handling to users to an extent not offered even under MS-DOS.

Unfortunately this capability has not been implimented to a large extent under HDOS. The main use in HDOS programs has been for the control C character to abort a currently executing command in system programs. And in order to write programs using these handlers, the programmer typically has no way to set up the interrupt handlers without resorting to writing in assembly language.

The HDOS 2.0 version of C/80 incorporates the control-C interrupt to provide an abort function. Not too exiting. Other than this, no control character interrupt processing is performed by C/80. And a CTRL-C abort may cause problems, if CTRL-C is entered while the display is in some unusual mode. (It's very difficult to make any sense of the display if, for example, everything is on the 25th line.) By adding a few functions we can enhance the capabilities of C/80 to handle CTRL-A, CTRL-B, and CTRL-C interrupts under HDOS 2.0, using them as was intended by the creators of HDOS.

THE HDOS CONTROL CHARACTER INTERRUPT ENVIRONMENT

When you enter a CTRL-A, -B, or -C character, HDOS attempts to execute a user-defined service routine (the 'handler'). If the handler address is zero (the default), execution will continue as if no interrupt had occurred. If a routine address has been provided to the operating system, HDOS will enter the service routine with the interrupted PC value and the interrupted PSW on the stack along with a return address into HDOS.

When the service routine has completed execution, control may be restored to the user program in one of two ways: Control may be returned to HDOS, in which case the BC, DE, and HL registers must have been saved and restored (the service routine is executed during interrupt time), or control may be passed directly back to the user program with a jump instruction.

Returning control to HDOS is simple, but somewhat restrictive. Returning control directly to the user program can be tricky. Because the interrupt may have occurred anywhere in the program code, including the C/80 libraries, or even some portions of the operating system, we have no way of knowing what may have been pushed onto the stack. We have to restore the stack to the condition expected at the destination point in the code. We shall later introduce a way to do this.

In the case of C/80, the start-up code installs address 'exic' into the control-C handler address. When CTRL-C is struck execution will immediately be transferred to point 'exic' in the run-time library. C/80 will first clear the console buffer and then perform a standard C/80 execute sequence. No return to the user program is ever executed.

We will first define several functions to support use of control-character interrupts using the C/80 compiler. We shall then present the source code used to create these functions.

SETTING A FLAG

The simplest way to incorporate control character interrupt handlers into a C/80 program is to write a service routine which simply sets a flag whenever a control character is pressed. The C/80 program can test the flag as required, and take appropriate action. While this may seem to defeat

FOR HDOS LOVERS...continued

the use of interrupts, it does provide you with an ability to interrupt the program even if the program never looks at the keyboard. For instance, examine the following code:

```
do
{   calculations   }
while (!done and !control_C_Flag);
```

This calculations loop will continue to execute until done unless you enter the CTRL-C character, causing an abort at the end of the current iteration--crude, but effective. We require two functions to accomplish this level of handling:

```
set_int(which)    Used to enable the routine.

tst_int(which)    Used to test the flag.
```

Where 'which' is the character 'A', 'B', or 'C', corresponding to a desired control character.

The function 'set_int' is called initially to set up the service routine, and to clear the flag. It may be called at any later time, with the effect of clearing the flag.

The function 'tst_int' is called to determine the flag's state. It returns a '1' if the control character has been struck, otherwise returns a '0'. It also resets the flag before returning. The above loop now becomes:

```
set_int('C');

do
{   calculations   }
while (!done and !tst_int('C'));
```

The CTRL-C character will abort the calculations loop if it's entered. These functions could be used in this way to follow the HDOS practice of aborting a command whenever you enter CTRL-C, so long as the 'tst_int' function is executed regularly.

THE LONG JUMP

In many cases we would like a control-character interrupt to force the program to resume execution at some pre-defined location. In order to do this we must have the ability to mark this location, and later jump to it. This is provided by the long jump.

Two functions have been defined in standard C to enable long jumps. These functions are the 'setjmp' function, and the 'longjmp' function. C/80 doesn't support long jumps so we'll have to write our own.

The 'setjmp' function is used to "mark" a point in the program. At any later time and at any other program location a jump may be executed to that point in the program, continuing execution. A value of zero is returned by the 'setjmp' function after it executes.

The 'longjmp' function is then used to force execution to continue at the "marked" point in the program. The long jump appears to resume execution within the 'setjmp' function, which now returns a non-zero value which was passed to 'longjmp'. This returned value can be used to determine where the long jump originated. Consider the following example:

```
if (value = setjmp(&here))
    printf("Long Jump number %d/n", value);
if (value == 20)
    exit();

for (;;)
     longjmp(&here, value+1);
```

When first called, 'setjmp' returns a value of zero, which is stored in the variable 'value'. Because the result of the expression is zero, the 'printf' statement will not be executed. It should not be executed, because a long jump has not occurred. And, since value does not equal 20, the exit will not be executed. Control then passes to the 'infinite for' loop.

The only statement contained in the 'for' loop is a long jump back to the location marked with the 'setjmp' function. But 'longjmp' returns a value of 'value+1', which equals '1', on the first execution of 'longjmp'. Although the value of 1 is returned by 'longjmp', it appears to be returned by the 'setjmp' function. The returned value is assigned to 'value' and 'printf' is executed, flagging a successful long jump.

Again the exit function is not executed, and control passes to the 'infinite for' loop which executes longjmp with a value of 2. This loop is repeated twenty times, printing out the value of 'value' as it is assigned values from 1 to 20. Finally, with 'value' = 20, the exit function is executed and the program terminates.

The infinite 'for' loop is actually not required, as the body of the loop is executed only once. But it does serve to show that a long jump can be used to exit from a loop at any time.

The pointer '&here' passed to both 'setjmp' and 'longjmp' functions is a pointer to an "environment block". This is a structure which is used to store values of certain CPU registers when the setjmp function is executed. The long jump is accomplished by having 'longjmp' restore these values to the CPU registers, in affect allowing the program execu-

FOR HDOS LOVERS...continued

tion to continue just where it was when 'setjmp' returned, but returning a non-zero value which is passed to 'longjmp'. In our case we need only restore the program counter's contents and the stack pointer, and then return a value which is left in the HL register.

There are a few restrictions to our version of 'setjmp'. The 'setjmp' function must be executed before attempting a long jump to that location. If 'setjmp' had not yet been executed, the destination point would not yet be defined. A program crash would almost certainly result.

The function which called 'setjmp' must still be active when 'longjmp' is executed. If the function has already returned, the stack will no longer be valid, including what should be the return address from the function, and a crash will again most likely occur.

Finally, we require that 'setjmp' not be used in an expression but rather as an expression by itself. This is necessary, because any evaluation of the expression before execution of 'setjmp' may not be restored during 'longjmp' execution, and even if it is, the values of any variables may have changed.

Note that the long jump is little more than a global version of the GOTO statement, and should be treated with the same caution as GOTO statements are. Don't use it unless there's a good reason.

USING THE LONG JUMP

Now that we have examined the long jump functions, we can use them for control-character interrupt processing. As mentioned above, the long jump is very similar to the way a GOTO statement works, but is of a global nature. The GOTO statement's destination must be within the same function as the GOTO statement. The long jump is not so restrictive. So long as we have not left the function in which the destination was defined using 'setjmp', we can jump to that destination from anywhere. The only requirement is that 'longjmp' functions be somehow passed to the environment structure. We may execute the long jump from within the same function, or from a called function, at any depth, and even from a separately compiled function.

Our interrupt service routine can use a long jump to resume program execution at some pre-defined point. Consider the following code:

```
static  abort_menu();
main()
{   /* main */
    setjmp(&restart);
    install('C', abort_menu);
    choice = master_menu();
    switch(choice)
    {
        case 0:
            sub_menu_0;
            break;
        case 1:
            sub_menu_1;
            break;
        etc.....
    }
}   /* main */
abort_menu()
{
    longjmp(&restart, 1);
}
```

I have introduced a new function, called 'install'. Its' purpose is to install the address of a control character interrupt handler into HDOS (The .CTLC System Call, or SCALL). Once installed, striking a selected control character will result in execution of the interrupt handler. In this code segment we have set up the function 'abort_menu' as the interrupt handler. Our handler will simply execute a long jump back to the function which apparently displays a master menu, and waits for the users' choice, returning it as a value.

Once the user has selected a choice from the master menu, he is dumped into a 'sub_menu' function, and whatever is required beyond there. But he can enter a CTRL-C at any time and will immediately be returned to the master menu. That can be pretty useful.

INDEPENDENT HANDLERS

The interrupt handler does not have to execute a long jump, but instead can execute and return, which will in fact be a return to the operating system. In this case HDOS will take care of the cleanup and transfer back to the point where the program was interrupted by a control-character input. HDOS requires than you must first save the CPU registers (except AF which has already been saved) before executing any code, and restore them before returning. (Remember, the service routine is executed at interrupt time). This can be done using the '%asm' directive. Not portable? Neither is HDOS's control-character interrupt. At any rate I don't see much use for a handler which simply returns, except perhaps to display a help screen.

THE FUNCTIONS

Listing 1 is a listing of functions written to provide control-character interrupt handling under HDOS. These functions were compiled as a separate module under C/80, and assembled using Microsoft's M80. I strongly recommend using

FOR HDOS LOVERS...continued

this assembler when developing programs. Debugged functions can be added to a library and linked with the main program thus greatly reducing compile-and-assemble time.

The three globally-accessible functions are 'set_int', 'tst_int', and 'install'. These functions have already been defined above. Also included in this module are three local functions 'ctrl_A', 'ctrl_B', and 'ctrl_C'. These are the actual interrupt handlers for the control-character interrupts. The interrupt flags are contained in a three-element array.

The 'set_int' function simply returns the interrupt flag and installs the appropriate handler function. Although the handler need be installed on only the first execution, redundant installation every time the function is called causes no problem.

The 'tst_int' function returns the value of the appropriate flag, and resets it if required.

The 'install' function interfaces directly with HDOS. The desired control character to install is passed as an 'A', 'B', or 'C' character, which is mapped to a binary one, 2, or 3 in the HL register. This value is placed in the A register (the H register contains zero), and the address of the handler function is left in the HL register with the 'addr;' statement. System call 41Q is then executed, installing the handler, and the function returns whereupon HDOS recognizes the corresponding control-chacacter interrupt.

The actual handlers simply set the appropriate flag and return. Examine the compiler output and you'll see that only the HL register is affected by the assignment statement. We must therefore save and restore only the contents of the HL register during handler execution. Remember that handlers are executed during interrupt time.

Listing 2 is a listing of 'setjmp' and 'longjmp' functions. 'Setjmp' saves the PC and SP values into the environment block, the address of which was passed in the function call. Upon entry to 'setjmp' the stack contains the return address (which is where 'longjmp' will resume execution) and the environment-block address. The return address is first retrieved from the stack and saved, and then the value of SP plus 2 is saved as the stack location. The statement 'blk;' is used to obtain the environment-block address.

'Longjmp' restores the saved value of SP, places the return value in HL, and executes a jump to the instruction following the 'setjmp' call instruction. We use the 'clibrary' function "h." to retrieve the return value from the stack. We can't do this with a C statement, because we have disturbed the stack with our PUSH instruction. Upon returning, SP is set two bytes too high. This will be taken care of by a POP instruction supplied by C/80 following the 'setjmp' call. It's used to pop off the address parameter passed to 'setjmp'. The BC, DE, and AF registers are undefined, and therefore did not need to be saved and restored.

The standard definition of 'longjmp' is that 'longjmp' cannot return a value of zero. As implemented here, the return value is not tested, and a zero value may be returned. Listing 3 is the header file "longjump.h", used to define the environment block. It must be included in the source code for 'setjmp' and 'longjump' functions, and in any programs or modules using them.

After compiling these modules, they are linked in with any program using them. The link process can be simplified by adding them to a library. This can be done for HDOS by copying the .rel files and libraries over to CP/M using an HDOS to CP/M file transfer program. The relocatable modules can then be added to the libraries as desired using the CP/M librarian. The finished libraries are then copied back over to HDOS using a CP/M-to-HDOS file transfer program. It's awkward, but saves a lot of time later. I have incorporated all C/80 functions including the long and floating point libraries into a library called 'clib.rel', and my own functions into a library call 'mylib.rel'.

CONCLUSION

These control-character interrupt functions I have presented should allow you to write C/80 programs recognizing user interrupts which employ CTRL-A, CTRL-B, and CTRL-C characters. The control-C interrupt can be used to abort commands in process as implemented in many system programs. The longjump functions can be incorporated in programs in conjunction with interrupt functions or by themselves, and simply implement a global form of GOTO statement.

I hope this article proves useful to you programmers still working with HDOS, and will bring some of HDOS's benefits into C/80 programs.

THANKS

A special thanks to Bill Parrott for bringing us HDOS 3.0!

: :

LISTING 1 STARTS AT TOP OF PAGE 6

: :

FOR HDOS LOVERS...continued

LISTING ONE

```
/* interpts.c:  Control Character Interrupt Handlers 11/28/87

The functions in this module are used to provide interrupt
handling capabilities for CTRL-A, CTRL-B, and CTRL-C inter-
rupts in a C/80 program.  Public routines defined in this
module are:

    set_int:    Set up Interrupt Handler.
    tst_int:    Test for Interrupt.
    install:    Install Interrupt Handler.

The set_int and tst_int functions are used as a pair to trap
control character interrupts in order to set and test a flag.
These routines are useful for determining if the user has
typed a control character during processing; for example, to
abort a lengthly loop.  In this case, the loop would test for
the interrupt flag once during each loop.

The install function is used to install a user function as
the control-character interrupt handler.  The user must
insure that the BC, DE, and HL registers are preserved.  The
function must return to insure that the stack is not disrup-
ted.  Execution of the program will resume where the inter-
rupt occured.                                                */

static    char    flag[3];          /* Interrupt Flags */

static    int     ctrl_A();         /* Interrupt Handlers */
static    int     ctrl_B();
static    int     ctrl_C();

set_int(code)                  /* Enable Interrupt Handler */

   char  code;                       /* Control Character */

/*  This function will set up the interrupt handler by
    notifying HDOS of the handler address.  It also
    resets the interrupt flag, and can be used whenever
    it is desired to reset the flag.  The code passed
    should be the letter 'A', 'B', or 'C', in upper or
    lower case.                                          */

{                                                /* set_int */

   static  int i;                    /* Used to index flag */
   static  (*loc)();     /* Uset to Store Handler Address */

   i = toupper(code) - 'A';

   flag[i] = 0;                     /* Reset Interrupt Flag */

   switch (i)           /* Set up Correct Handler Address */
   {
      case 0:

         loc = ctrl_A;
         break;

      case  1:
         loc = ctrl_B;
         break;

      case  2:
         loc = ctrl_C;
         break;
   }

   install(code, loc);                 /* Install Handler */

}                                                /* set_int */

tst_int(code)                        /* Test for Interrupt */

   char  code;                       /* Control Character */

/*  This function is called to find if the specified
    interrupt has occured.  If it has, a value of 1
    will be returned.  If the interrupt hasn't occured,
    a value of 0 will be returned.  The code passed
    should be a letter 'A', 'B', or 'C', in upper or
    lower case.  The flag is reset before the function
    returns.                                             */

{                                                /* tst_int */

   static  int   i;                  /* index into flags */

   i = toupper(code) - 'A';

   if (flag[i])                              /* Interrupt ? */
   {    flag[i] = 0;      /* Yes.  Reset flag and return */
    return(1);
   }

   return(0);                             /* No Interrupt */

}                                                /* tst_int */

install(code, loc)           /* Install Interrupt Handler */

   int  code;                        /* Control Character */
   int  (*loc)();         /* Pointer to Interrupt Handler */
/*  This function will call HDOS to set up the inter-
    rupt handler.  The address is first saved in local
    storage so that it can be loaded into the HL
    register without disturbing the A register.  The
    HDOS System Call is then executed to set up the
    address of the interrupt handler.    [continued on pg 7]
```

FOR HDOS LOVERS...continued

[continued from pg 6]

```
    Installing a NULL address (0) will disable the
    interrupt.                                          */

{                                               /* install */

    static  int  i;
              /* Code to pass to System Call (1, 2, or 3) */
    static  int  (*addr)();
                  /* Pointer to Interrupt Handler, Local */

    addr = loc;                                 /* Save Address */
    i = toupper(code) - '@';                     /* 1, 2, or 3 */

    i;                      /* Move Code (1, 2, or 3) into A */

#asm

    MOV  A,L

#endasm

    addr;                                   /* Address into HL */

#asm                                        /* And Call HDOS */

    RST  7
    DB      41q

#endasm

}                                               /* install */

/*  The following three functions are the default
    interrupt handlers for the three control-character
    interrupts.  Each function will set the appropriate
    flag to indicate that an interrupt has occured.     */

static      ctrl_A()                    /* Set Control A flag */

{                                               /* ctrl_A */

#asm

    PUSH HL                                 /* Must Save HL */

#endasm

    flag[0] = 1;                            /* Set the Flag */

#asm

    POP  HL                                 /* And Restore HL */

#endasm

}                                               /* ctrl_A */

static      ctrl_B()                    /* Set Control B flag */

{                                               /* ctrl_B */

#asm

    PUSH HL

#endasm

    flag[1] = 1;

#asm

    POP  HL

#endasm

}                                               /* ctrl_B */

static      ctrl_C()                    /* Set Control C flag */

{                                               /* ctrl_C */

#asm

    PUSH HL

#endasm

    flag[2] = 1;

#asm

    POP  HL

#endasm

}                                               /* ctrl_C */
```

= =

LISTING TWO

```
#include "longjump.h"

setjmp(blk)
    struct  env_blk  *blk;
                        /* Environmental Block Address */

{

    blk;            /* Get Address for Environment Data */
```

[continued on pg 8]

FOR HDOS LOVERS -- Concluded

[continued from pg 7]

```
#asm

XCHG                            /* Save Address in DE */

                        /* Get Address of Next Instruction */

    LXI     HL,0            /* Address is Last on Stack */
    DAD     SP

    MOV     A,M         /* Move it into Environment Block */
    STAX    DE
    INX     DE
    INX     HL
    MOV     A,M
    STAX    DE
    INX     DE
    INX     HL

    MOV     A,L                 /* Save Stack Position */
    STAX    DE
    INX     DE
    MOV     A,H
    STAX    DE

    LXI     HL,0                            /* Return 0 */

#endasm

}



longjmp(blk, val)

    struct  env_blk  *blk;  /* Environmental Block Address */
    int     val;                        /* Return Value */

{
    blk;                /* Get Address of Environment Data */

#asm

     XCHG                           /* Save Address in DE */

     LDAX    DE          /* Set up Jump to Old PC Address */
     INX     DE
     MOV     L,A
     LDAX    DE
     INX     DE
     MOV     H,A
     SHLD    jmploc+1

     LDAX    DE                     /* Get Old SP Value */
     INX     DE
     MOV     L,A
     LDAX    DE
     MOV     H,A
     PUSH    HL                 /* And Save it on Stack */

     LXI     HL,4                 /* Get Return Value */
     DAD     SP
     CALL    h.##
     XCHG                                /* Save it in DE */

     POP     HL                 /* Restore Old SP Value */
     SPHL

     XCHG                    /* Return with Passed Value */

jmploc: JMP     0

#endasm

}
```

- -

LISTING THREE

```
/*  longjump.h: Environmental Block Descriptiong  6/22/88  */

struct     env_blk

    {
     int   *old_PC;         /* Saved Program Counter Value */
     int   *old_SP;           /* Saved Stack Pointer Value */
    };
```

--=:<<8>>]]=:--

[EDITOR'S NOTE: We sincerely thank Gary Appel for submitting this highly-useful article--and on a disc, no less! We do hope that other JOURNAL subscribers who use HDOS will follow Gary's lead and send us their HDOS items, likewise as plain-vanilla ASCII files on disc. One caution: To save your editor's time and possibly avoid mistakes, set your left margin to column ZERO and your right to column SIXTY ONE, then type in your text or listings. We don't like to complain, but Gary's listing was set up to use the entire page width and it took lots of time and effort to compress it to fit our page layout of two text columns wide and with a center gutter 4 spaces wide. We hope that we didn't introduce any errors in Gary's three program listings. Notice that in most cases where there are remark/comment lines or blocks that we tried to uniformly place them toward the right side of our 2 columns. This isn't C standard format, but it was the best we could do under the circumstances! Also note that we DID NOT test these HDOS routines; the originals came to us on a CP/M disc and we haven't had time to convert 'em!]

READER'S MAIL BOX

Dear Mr Geisler,

My heart nearly skipped a beat momentarily when I read in the December JOURNAL that CIS charged more than double for connect fees at baud rates higher than 300 during business hours. Since I am frequently off work during the day in the middle of the week, I occasionally log on between 9am to 5pm Mondays through Fridays at 1200 baud. Had I screwed up doing my homework?!

Not really. A quick rate-check on 9-Jan-89 revealed prime daytime AND evening rates are $6 an hour through 450 baud, $12.50 an hour at 1200 and 2400 baud. At 4800 baud, rates are $32.50 an hour, prime time days, $44 an hour evenings. In other words, the rates are the same from 300 to 1200 baud, higher rates apply during the day for 4800 and 9600 baud. Speeds above 1200 baud are not available in all locations.

CIS also have a 30-cents an hour surcharge. If you must go through Telenet or Tymnet there's an additional $12 an hour charge during the day and $2 an hour nights throughout the contiguous U.S.A.

Hours designated prime or daytime service are 8am through 7pm weekdays. Standard evening service is 7pm to 5am weekdays, all day Saturdays, Sundays and announced Compuserve holidays. The period between 5am and 8am weekdays is "as available" during which time they perform system maintenence.

Joe Katz has done an excellent job running the Zenith forum [on CIS] since HUG's departure. But there is precious little useful information for [us] H8/H89 users.

While I have your attention, does anyone have an address for the Xebec Company? I picked up one of their 1410 controllers and would like to get a manual for it.

CHARLEY BEGIN, 5424 Weaver Road, Cheyenne, WY 82009

[I must apologise for being so far out of date in connexion with CIS charge rates! Although I do sign on the service now and then--usually is on weekends or evenings--I haven't kept up with their rate structure. I'm signed up with GENie also, and I tend to get the two service's rates confused since they're both charged to my American Express card... (Ah, the curse of "plastic money"!) I do thank you for the CIS price structure update information, and I'm sure our readers do also. And Joe's doing as well as can be expected as SYSOP, but I'm quite sure he'd welcome any information us 8-bit users are able to supply to him. You other readers, help Charley find Xbec's address, will you please?! -- ed]

Dear Mr Geisler:

I am in need of the schematic diagram for Heath's H47 disc controller used in the H8. If any of the [JOURNAL] readers has one I shall appreciate it very much if they would share it with me. Thank you for your assistance.

CHARLES E F LISS, 9607 Columbia, Redford, MI 48239

[Ok readers, help ole Charley out! -- ed]

Hi Len,

Thanks to Mario Davidson for his "zap" to allow testing the speed of drives other than SY0:. I'm sure it will come in handy many times in the future.

After reading Allie Lingo's and Karl Ruhling's letters about decreasing the number of sectors that DIRECT.SYS occupies, I realised that there probably are some newer HDOS users out there who might be interested in some of the patches [for HDOS] which were brought out early in the game. Here are some I have, and maybe other readers can dig out some more. I don't remember where I got them so can't give anyone credit. You will have to use some type of disc-dump utility to make these patches. I used DUMP.ABS from HUG disc #885-1062 for my hard-sector drives. (HUG no longer lists this utility and it may not be available.)

-- Speed up booting of HDOS 2.0 sysgened discs --

1 - Eliminate bootup date request.
Change TRACK 2 SECTOR 0 BYTE 4C (HEX)
from CD
to C9

2 - Eliminate showing HDOS version and issue number.
Change TRACK 1 SECTOR 4 BYTE 90 (HEX)
from CD
to C9

3 - To change delay after the ACTION? <BOOT> message.
Change TRACK 0 SECTOR 2 BYTE 2C (HEX)
from 3C (30 seconds)
to ?? (number of seconds * 2)

-- Directory sector allocation changes --

To decrease number of directory sectors allocated on a hard-sector disc I patched INIT.ABS. I set it for 4 sectors and have never run out of directory space. I have also used this patched INIT:ABS for my SIGMASOFT hard disc and soft-sectored disc and haven't encountered any problems. (I suspect that the SIGMASOFT driver ignores my patch and does his own thing.) I don't know what this patch will do with other drives such as the H37s, H47s, etc.

Use PATCH.ABS to make the change.

ADDRESS	PATCH	NUMBER OF SECTORS ALLOCATED
061364	074 to 076	
061356	074 to 003	4 sectors (85 files max)
"	or 005	8 sectors (173 files max)
"	or 006	10 sectors (217 files max)
"	or 010	14 sectors (305 files max)

If PATCH asks for a PATCH ID and PREREQUISITE CODE then you'll have to do something else first; that is, use the code I've supplied below.

PATCH-PATCH

This allows you to make patches to [.ABS] program files without a PATCH ID and PREREQUISITE CODE. You must then keep track of your modifications so that you don't make patches on top of patches! This patch is from REMark (May 1982). Run PATCH.ABS and reference the below example run. Your entries are shown inside parenthesis (don't enter the parenthesis!).

More MAIL BOX

This patch must be made to a fresh unpatched copy of PATCH.-ABS from your distribution disc, otherwise the PREREQUISITE CODE won't match.

```
>(PATCH)
PATCH Issue #50.06.00

Filename? (PATCH)
Patch ID? (IFOJIC)
Prerequisite Code? (IF8EIADPGEFFCF)

Address? (042231)
042231 : 312/(303)
042232 : 244/(^D)              (Type CTRL-D)
Address? (042263)
042263 : 247/(257)
042264 : 304/(^D)
Address : (044055)
044055 : 076/(303)
044056 : 000/(354)
044057 : 377/(047)
044060 : 046/(^D)
Address? (^D)
Patch Check Code? (DLMIAGPD)

PATCH Issue #50.06.00

File Name? (^D)
```

If anyone has problems making these patches they can write or call me and I'll try to help.

BOB OLSON, 24450 Kirby Street #146, Hemet, CA 092343, phone 714-926-5146

[Hey, Bob! I tried your INIT.ABS patch on my H37 system disc and it works "peachy-keen"! Because 80trk dsdd soft-sector discs have a tremendous number of sectors available compared with their hard-sector counterparts I set INIT.ABS for eight sectors. Pat Swayne's REDUCDIR.ABS on ssdd 80trk discs makes DIRECT.SYS at 12 sectors, plenty space for a disc full. It'll bring a 40trk sssd or dssd hard-sector directory down to four sectors, and I've yet to run out of directory space with that size. Methinks eight sectors is probably a very good 80trk hard or soft sector compromise. Thanks for your input! - ed]

Dear Sir or Madam,

Would you please send me information about the SEBHC JOURNAL and supscriptions thereto. I read about it in the December, 1988 Computer Shopper.

I recently purchased an H89 computer with Magnolia Microsystem's controller, dual external drives and an H25 printer. There is NO documentation for any of my hardware, so I'm looking for information, tips, other sources of useful data and add-ons.

DAVID E YOUNG, 402 Houghton Street, Ontonagon, MI 49953

[Well Dave, you came to the right place! If you can't find it in the JOURNAL many of our other subscribers will probably be able to help you out (they're very good at that). We sent you a sample of the December '88 JOURNAL and hope you'll become a subscriber BEFORE any of our readers get in touch with you. And our thanks to the Computer Shopper and Mr George Ewing for having so kindly mentioned us. -- ed]

Dear Sir:

I recently purchased a Hayes-compatible external modem for my H89 computer. I'm looking for help in obtaining programs which will allow me to use this modem with my computer. Also I'm looking for a source for a hook-up cable, or the proper information on how to construct one.

Additionally, I'd like information on the correct way to be able to log onto the electronic bulletin boards and [what] sources of information [about them] which are available. Any help you can give me shall be greatly appreciated.

JOE SIEDLER jr, 2900 Emerson Ave So, Minneapolis, MN 55408; phone 612-824-7755

[Thanks for writing us Joe. We need to know if you received an instruction manual with your modem and if it has any information in it about what type interface cable and connector(s) should be used between it and your computer. Most modems use a modified form of RS232 DTE connector (type D) at the computer end. Sometimes they use a similar type D-connector for the modem end or a DIN connector-to-type-D RS232 arrangement. My two modems use different connector types, one a RS232-to-RS232 D-connector ribbon cable, the other uses a short DIN-to-RS232 D-plug interface cable. I suggest you get in contact with HUG and order a copy of their MAPLE CP/M Modem APplications Effector, p/n 885-8012, or the HDOS version, p/n 885-8005. The last price I saw for it was $35, but don't quote me on that! It's a good package--but it does have a very steep learning curve. To learn how to log onto a commercial Bulletin Board--or any BB for that matter--you should get their instruction manual (if one's available). You can get onto CompuServe Information Service (CIS) by buying HUG's package (don't remember the catalogue number!) for maybe $29. HUG's phone number is 616-982-3463. This package brings you a contract form, a temporary password, some user instructions, and a one-hour "free" on-line session. "Way back when" I bought my CIS package from Radio Shack for $29+MI sales tax. They may still have that around under the original RS catalogue number of 26-2224. Worth checking out. By the way, we found your name, address and other data in our subscriber's database but the date you signed up date is missing. Would you please let us know what that was? -- ed]

Dear Leonard,

Thought you might be interested in some experimenting I've been doing with my H90 and [the new] 3.5-inch 720k disc drives. The 3.5-inch drive duplicates an 80trk double-sided

More MAIL BOX

drive when connected to the H90. Since this is the same as a 5.25" 80trk drive I'm not sure as to how practical it might be. [Don't fret about that; I'm sure we ALL can come up with many plausible reasons for buying smaller drives. -- ed]

I tried two different brands of drives connected to the external jack of my Z89-37 soft-sector controller and tested them with the TESTH37 routine (on HUG's soft-sector support disc). Both drives made three passes with no errors. As of now I haven't done any extended testing with user-type programs.

I used a Mitsubishi MF353B which came mounted in a 5.25" frame and set the drive-select jumper to DS1 (the unit's numbered 0-3) so that HDOS would use the drive as SY1:. Other jumpers left ON were MM, IS, and DC. [Similar to 5.25" drive unit designations. They're obviously interchangeable with older 1/2-height Mitsubishis excepting size difference. - ed]

I also used a Toshiba ND-354A kit. It contained a FDD4408 drive and ND354KU hardware kit. It was necessary to change the drive from the 3.5" face to a 5.25" face and frame before mounting. I set the drive-select jumper to DS2 (this unit's numbered 1-4) so that HDOS would recognise it as SY1:. It has a jumper marked RY which should also be on. The 5.25" to 3.5" adaptor board has a jumper on it which should be set to 'AT' on older boards or 'A' on newer boards. This jumper disconnects edge-connector pin 34. The instruction manual said that the 'AT' position was the connected mode, but the board was silk-screened backwards and an addendum [correcting this] was packed with the adaptor board. I believe this kit's number is now ND532. It has an ND352KU hardware kit and an FDD4206, FDD4208, or FDD4210 drive. Each drive is hard-wired on its' board for different functions of pin 34. Since we don't need this pin it shouldn't matter, but it does make a big difference if you're using it with a PCAT-style computer.

I checked the schematics for both the H88-1 hard-sector and Z89-37 soft-sector controller boards. Neither of them uses edge-connector pin 2 (alternate signal) or pin 34 (Disc Changed or Ready output) so even if they can't be disconnected as with the Mitsubishi drive we shouldn't expect any problems. I haven't the slightest idea how these drives would work with outside vendor controllers--such as Magnolia's--if used with 3.5" drives. [Maybe some of our other readers can give us a clue?! -- ed]

I also tried a 1.44-meg, 3.5" Toshiba FDD4603 drive, and as suspected, it wouldn't work unless it was forced into Low Density Mode and had a low-density disc inserted. These 1.44-meg drives require a high-density controller to work properly. You could GO BROKE VERY FAST buying the new 1.44-meg discs as they cost between $4 to $8 EACH--that's $45 to $80 a box of ten! Of course, these are 80trk double-sided and hold as much data as two dsdd 5.25" discs, but the price is still rather steep. MIE Micro just sent me an ad listing their 720-k 3.5" discs at 79 cents each in lots of 25; including lables, shipping and handling this comes out to 99 cents each. On the other hand, I can get 5.25" dsdd Nashua discs at the local department store for $5.99 a box of ten, and even cheaper from bulk vendors [see above MEI ref].

Since our H/Z 8-bit drive standards don't even come close to 3.5" units, switching to these smaller drives might not be too useful, but it's nice to know that it can be done.

BRIAN L HANSEN, 315 Roast Meat Hill Road, Killingworth, CT 06417

[Yup, why spend more on floppies than you have to. Of course the 3.5" floppy disc is less prone to damage, DOES take up far less space, and if you can use the 1.44mB format it gives more storage, but look at all that WORK and COST! If you bill yourself your own labor at, say $10 an hour, the overall expense will nearly total what you'd pay for a decent 20mB hard disc, controller and software. This assumes you'll get a decent trade-in allowance on your old 5.25" drives. Frankly, I'm not even considering 3.5" drives. Darrell Pelan just sent us an ad (see p 12) offering H89 hard-disc upgrades from $419. Why fool around with 3.5" unless you find it is REALLY necessary? I'm not knocking your efforts in the least. Personally I feel that half-height 5.25" drives are going to be around a lot longer than those monsterous old 8" clunkers and 5.25" discs will be the standard of data exchange for a very long time to come. PLEASE keep on writing this kind of good stuff! -- ed]

Dear Len,

I'm passing on my solution to a nasty little bug which consumed the better part of my day to fix. [As in "render immobile"? -- ed] My H89 started to die at odd intervals and sometimes wouldn't boot at all even though the terminal seemed fine (displayed properly what was typed when off-line). I wiggled all the wires and it appeared that the serial cable connecting the terminal logic board (TLB) and CPU was bad. A quick cable swap killed THAT theory. Then I swapped CPU cards; same problem! My next step was to take all the goodies out of the '89. Out came the H-37, serial I/O, and WIN89 card. Same problem. The only thing left was the TLB. Since the terminal portion seemed to work, I figured it might be the 8250 serial IC. A quick replacement showed that I had a worse spar--but when I bowed the pins on the original chip in order to re-install it the problem went away. Bowing the 8250 pins must have fixed an intermittent contact problem between chip pins and socket contacts. Everything is [now] back [and] working fine.

Please note that I've enclosed a short advertisement copy. I'm pleased to announce that the price on the WIN89 is now down to an all-time low of $419. And I'm now offering the faster ST-125 with the WIN89 for just $479.

DARRELL C PELAN, Micronics Technology, Suite 159 54 Dalradia Road, Montgomery, AL 36109; phone--205-244-1597, BBS--205-244-0192

[more]

Speed and Power for your H/Z89!

Micronics Technology has the H-89 upgrades you need. Increase your operating speed to 4 MHz and disk space to 20 Mega bytes.

SPEED MOD - Software selectable speed, 2 or 4 MHz. The assembled and tested version includes a Z80A, speed card, and software. The kit includes all parts and software, you supply the solder. The software supports Heath, Magnolia, and CDR CP/M systems, plus Heath HDOS. Assembled and Tested: $34.95

WIN89 - 20 Mega Byte disk drive for the H-89. The hard disk drive is bootable (3 times faster than H-37). The drive is configured for two 8 meg drives and one 4 meg drive. Source code is provided for the BIOS changes and most utilities. The WIN89 adds a PC buss to your H-89 so you can take advantage of the low prices for PC hard disk drives. The interface card mounts on the memory side and is compatible with Heath, Magnolia, and CDR CP/M systems. The Standard ST-225 is 20 meg hard disk with an average access time of 65 ms. The ST-125, available for slightly more, has an average access time of 40 ms. HDOS support will be available by March 89. ZCPR is provided to help manage your increased disk space. We also sell the interface card with software separately. You supply the hard disk drive.

Assembled and Tested with ST-225	$419
Assembled and Tested with ST-125	$479
Interface Card	$175
External Option	$125

Perfect Money ($19.95) - Calculate all your loan costs, number of payments, or any other loan variable. Also handles balloon payments.

Paycheck ($39.95) - Provides an optimized data base environment to handle your payroll. Prints checks, handles federal and state taxes, tips and much more.

MT Accountant ($19.95) - Organizes your financial records by account and date. You can edit, delete, add, or browse through each record in the database. Reports subtotal each account, maintain a running subtotal, and print the grand total. You specify which accounts are credits and debits.

Perfect Printer ($19.95) - Electric typewriter program for your computer. Uses your printers special features and has both immediate and buffered modes.

ORDER by writing to the above address or calling the above number (6-8 PM M-F, 9-12 Sat, or leave a message). We accept checks, VISA, and MC. Shipping is $2 except for the WIN89 ($7). Alabama residents add 7.5% tax. You can also call our 24 hr BBS at 205-244-0192 (300/1200/2400 8N1). Ask for our free catalog. The BBS features a CP/M file section with UNARC, a program to uncompress files using the popular ARC format. I will gladly add an HDOS section if folks upload the files.

Micronics Technology
Suite 159, 54 Dalraida Road
Shipping Address: 410 Bellehurst Drive
Montgomery, AL 36109

Voice: (205)-244-1597
BBS: (205)-244-0192

MORE MAIL BOX

Dear Leonard,

I received CP/M GAME DISC #0 very promptly. [But] it seems at this point in the life of H89ers that it shouldn't be necessary to write on the [disc] sleeve in red ink "PLEASE TYPE READTHIS.1ST BEFORE RUNNING THIS SOFTWARE". How many novices are joining the H/Z 8-bit users? Then as I recall I had to re-do it [READTHIS.1ST] anyway with PIE to remove embedded EDITOR processing before I could print it!

My MAIL LADY did not appreciate your comment above my name on the November JOURNAL [that is]:

"ATTENTION POSTAL CARRIER:
The addressee HAS PAID FOR THIS ITEM. PLEASE DELIVER IT!"

That really puts it in the category of much of the junk mail [which] you and I receive.

[Incidentally,] why is it assumed that everyone uses the same word processor or editor?! I recently obtained NZ.COM from Alpha Systems Corporation, and the gibberish in the .DOC file when LSTed on my H125 was unbelievable. It took a day's work to make it printible.

[Note] there are no individual .DOC files on DISC #0 which is too bad. Luckily I obtained from you one of the last copies of Vol I.

I have typed in Lee Hart's description of ACES from I:6 pp 15 & 16, but not the illustration at the bottom of p15. It's on the enclosed disc as ACES2.SB and ACES2.ASC. If someone uses SPELLBINDER (this was Henry Fale's favorite) then the .SB will print ok. The other file is ASCII and should cause no problems. Feel free to add it to the disc. Without it, or a copy of the original JOURNAL [article] the game is [almost] unplayable.

I have yet to explore NZ.COM, and am wondering if I really need it, but like LUCIDATA PASCAL, I'm collecting things before they disappear.

I have a number of old hard-sector discs which someone may have for the UPS charges. I never kept one with a bad sector.

Although this was composed and printed using SPELLBINDER, it is on the disc in ASCII.

FRED A ROSE md, 4206 Ruby Place, Bellingham, WA 98226; phone 205-671-5495 (voice or TDD)

[We have had to resort to the message on our disc sleeves because we're now getting subscribers who've recently bought "previously-owned" H/Z 8-bit machines (usually WITHOUT software and documentation). We're averaging complaints from one of every 3 disc buyers saying they can't get a .BAS file to run (without understanding they must FIRST load MBASIC)! And Fred, the only commands in READTHIS.1ST are just two printer form feeds so that the otherwise plain-vanilla ASCII file prints out nicely on two 60-line pages. That's PROCESSING?! Also, we didn't mean to offend your female Postal Carrier! We've had to resort to this otherwise well-intentioned notice to grab the attention of certain OTHER Postal Carriers who routinely trash (throw away, that is) other subscriber's JOURNALs because they see our BULK-MAIL rate imprint. We use bulk mail rather than submitting to the mindless and endless nonsense of becoming a "non-profit Inc." just to obtain the cheaper "newspaper" rate. Please relay this to your Postal Carrier! Of course you're correct in saying that individual .DOC files would enhance GAMES DISC #0. But please understand that they would take up too much room on the single-sided HARD-SECTOR version of GD#0; we'd have to issue that version on TWO discs. Our way saves the purchaser a buck. Of all the GD#0s sold we have received only one other complaint about the lack of adequate documentation (MAIL BOX, III:5). I find your opinion of plain text editors somewhat confusing. We use Text Processor V4.1 here (both HDOS and CP/M) and find we can read any ASCII files it produces on the terminal screen without problems. TXTPRO.ABS ASCII "readme"-type files will print out cleanly on *any* H14 or H25 printer, exactly the same as on the terminal. The only files we can't read on our H19 and H89 terminals are those {#@*&!%!!} "processed" Word Star and Magic Wand abortions we occasionally get. That's why we prefer that everyone submits articles in PLAIN-VANILLA ASCII FORMAT. We take care of all the necessary editing and formatting for our pages. If your H125 printer doesn't recognise the universally-employed CTRL-L (form feed) symbol embedded in READTHIS.1ST, something about it is VERY much different than most other printers. Even our "venerable" Smith-Corona daisy-wheel printer nonchalantly accepts embedded form feeds! Check your H125 out and see why it doesn't like 'em. And thanks for your generous offer of "free surplus hard-sector discs". I'm sure one or more of our other readers'll be contacting you Real Soon Now. -- ed]

Dear Len,

I'm FINALLY settled, and here's my "renewal". (It's been so long that I've probably been lost in or by your system!)

Retirement from the USAF is GREAT!! No [more] blue shirts, 2:00am panic calls, unannounced exercises...and I even get to choose my tie colour every morning!

[Is there] any chance of having my subscription start with last July's issue where it expired? If not, I'll order the back issues after we've closed on our "new (to us) home in April.

Is your ad policy still the same? I have some pretty good programs that are just about ready [to market]. And are you still looking for articles? Any particular topic(s)?

TOM BOHON, P O Box 293, Olympia, WA 98507

[Hey Tom, welcome back! We've been wondering if you had fallen off the edge of the world. In reply to your questions in categorical order: Yes; yes; yes; you name it, we'll consider it! And if it's on "How-2 Use HDOS 3.0 Without Pain or Strain", we'll be delighted! All our readers are not blooming experts, most of them like "duffer-type" stories (Dr Rose please note). Do write about HDOS 3.0 Real Soon! -- ed]

MORE MAIL BOX

::

* INTRA-SUBSCRIBER CORRESPONDENCE OF NOTE *

::

TO: Rick Swenton

Dear Rick,

The purpose of my note to the SEBHC JOURNAL was to find out if my reaction to NZ-COM is shared by anyone. I am all for promoting the self-installing version of Z3.4, with is neatly done and likely to expand the user base considerably, but I think a little controversy wouldn't hurt. But personally I am interested in the logical underpinnings of the system, and I guess that there should be others at least as interested in dissasembling an alarm clock than in looking for what time it is.

The self-installing versions of ZCPR are geared toward a standard CP/M 2.2 system. Rather than just tack an advanced operating system onto a plain-vanilla CP/M base, I submit it pays to examine how the available hardware may impact [sic] your actual use of the operating system. It doesn't matter--for example--if you use a RamDisk, that the requested utility is CP, RCP or disc-resident, especially now that type-4 relocation is available. I've not fully evaluated yet how I'm going to integrate Z-3.4 with my H89, but I'm working on it. It is not that I cannot do with Z-3.4 what Z-3.3 allowed, but that with Z-3.4 I should maybe do it differently because of its' added capabilities.

I do have NZ-COM running, but am not currently using it because I had to revert to the original Heath BIOS, which does not include the revisions [which] I made for 3.3. Many decisions [which] NZ-COM makes for you may conflict with what I have in mind, and it will take some time to sort it all out. Let me expand a little bit on my previous Z implementation(s):

-- My system is an H89a/MTR-90, with Z89-37 controller and H88-3 I/O. I did install Anapro's 6MHz mod and a CDR Super Ram 89. The CDR RamDisk, in addition to the system's 64k memory provides 15-63k banks which all share an upper 1k global memory with system memory. At cold boot the RamDisk is initialised from a floppy drive [and] then substituted for drive A:, and all floppy drives are [then] re-assigned. I did incorporate this procedure into the Heath BIOS, and a Warm Boot from RamDisk has been substituted for the original floppy re-boot.

-- The Heath 2-step BIOS-loading procedure removes the restrictions imposed on the size of a standard CP/M BIOS. My BIOS.SYS file contains of the System Segments, and an additional LDR.COM from STARTUP is no longer required. All Heath BIOS Logical and Physical I/O Device routines are transferred to the IOP segment, retaining only required drivers. All drivers provided in Heath's BIOS remain available by just loading a different IOP at any time. Not only do I gain the flexibility offered by a substitution of IOPs, but the IOP code may even be smaller than the required by any idle driver which must alwasy be present in the original Heath BIOS.

-- The configuration definition (contained in the BIOS Header Section) and the clock control are access extensively (but not only) by the device drivers. The IOP structure lends itself to insertion of this Header in a buffer following the Internal Name, which makes it available to the IOP drivers, and likewise allows it to be changed by switching Segments. The BIOS--being assembled with an embedded IOP--configuration definition, and clock control are also tied in with the other sections of the BIOS referring to it.

I think that none of the above is incompatible with Z-3.4, but it's not obvious right now if and how I can make NZ-COM understand that the System Segments (including ENV) are already there. I must confess that I have not yet assimilated all of the User's Manual, so maybe it is there somewhere and I may still become another convert. I thank you for pointing out that NZ-COM loads the approved DUMMY IOP which is well defined in the "ZCPR3 and IOPs Tutorial" by R Conn. Having a valid IOP in place at Cold Boot, I have no use for the DUMMY and would rather disassemble the load, than have to come back and restore what NZ-COM has destroyed. Where did you find out about this?!

I am suspicious about the NZBIO.ZRL; my RamDisk code is also intercepting BIOS calls, and handles them differently if they address [either] the RamDisk or the floppies. I really cannot use anything in the operating system I don't have source code for, in order to make sure I understand the intent and full implications of it. No doubt that the challenges to adapt NZ-COM are greater than ever before!

I've heard of a possible bug in PREL for large files, but haven't so far encounted it. I'd me grateful if you could provide me with that comment by Peter Haas. I never followed 'Buss' but could try to see if anybody around here may have it. Is GENSPR devoid of this problem? Given the revival of .PRE files, I'd better be ready.

According to Sage, the only dirrerence between PRL (=PRE) and SPR files is in the object code's ORG. When relocating, the offset must be adjusted for an SPR file. I'm sure it is no accident that PRE files are preceeded by an empty page containing the code length in postions xx1=xx2. This is also the format in which the CCP-BDOS-BIOS is stored inside MOVCPMxx. I wonder where this structure originated and what other applications are using it? It seems to me that the blank page is intended to receive a relocation routine, exactly the way type-4 utilities are implemented, and that the first byte (xx0h) is intended for a LD dd,nn instruction which would make the code length available to the relocating loader by putting it in a register.

I recently got the Z-3.4 source and this, together with the articles in TCJ by Bridger Mitchell, Jay Sage, Joe Wright and the as-yet unassimilated sections of the NZ-COM manual is a lot of material to go through. I'm aware of the publicised features of ZSDOS and would use it immediately if I could get it with a well-commented source. Meanwhile, I stay with the CP/M BDOS which I have now thoroughly disassembled.

The MAIL BOX is Empty!

I do appreciate your articles in the SEBHC JOURNAL and just hope that others will come out of the woodwork and join in a dialogue. Now that commercial considerations have subsided in the 8-bit world it may be easier to present approaches on the basis of their technical merit. This, together with the openess of our systems and of the operating system software, is a major assett in the effort to keep our systems alive. It behooves us to make the most out of our H89s!

LUDO VAN HEMELRYCK, 16514 14th Ave SE, Mill Creek, WA 98012

TO: Ludo Van Hemelryck

Dear Ludo,

Of course you are right when you say that your CDR RamDisk and your BIOS and banked memory mods may be in conflict with what NZ-COM may force upon you. But now that you know how to go about getting the source for ZCPR 3.4 and possibly adding ZSDOS, you may be in for some more marathon system-development sessions and sleepless nights. I know what it's like!

You mentioned Heath's two-step BIOS loading process. Very few people really understood how versatile this "non-standard" BIOS loading method is. When I ran a manual ZCPR3 system, I had also included all the system segments in the BIOS.SYS file as you did.

Regarding the IOP under NZ-COM you are not "forced" to a particular size. It can be any size from zero to as much memory you wish to sacrifice. Of course, the size is incremented by one record, 128 bytes at a time. You should also investigate possible use of the NZBIO for your BIOS header applications. You could even define IOP space and load nothing there. Just allocate the space and delete file NZIOP.ZRL or create your own NZIOP.ZRL. The IOP initialisation code to patch the BIOS jump vectors is in the IOP itself (even the dummy iopl, NZIOP.ZRL).

Since you asked where to find ALL the pertinent information, I looked in my NZ-COM binder and found a considerable amount of information which was not released with NZ-COM but rather was available free on the Z-Nodes. I have included both paper and disc files as appropriate to facilitate even more sleepless nights for you!

On the subject of PRL files I am certainly no expert, but here is what I determined over time: BIOS.SYS is really a System Page Reloctable (SPR) file. The addresses in the file are originated at 0000H and the file contains a bit map which points to those bytes in the oject code which are address-dependent and would need to be changed if the location of the program in memory changed. SPR files must be loaded on an even page boundry. A PRE file generated by PREL.COM is really an SPR file. On the other hand, a Page Relocatable (PRL) file is the same thing except the object code is originated at 100H. To be honest, I'm not sure why use of one should be preferred more than the other. I suspect that it may have had something to do with the old MP/M operating system. Since a particular (COM) file could be invoked by any number of users in a multi-user system and each user was assigned their own block of memory, the (COM) file would have to have the ability to run anywhere in memory. Hence, the PRL file took the place of what we know as the COM file. But again, I could be all wet on this one!

By the way, you mentioned the blank page at the beginning of the SPR (PRE) file. The BIOS.PRE file created by PREL.COM will get the BIOS Loader installed in the first (blank) page (256 bytes) by the third invocation of MAKEBIOS (MAKEBIOS B:3 B:).

I've converted my Heath BIOS to Z80 opcodes and I assemble to a REL file with SLR180. It only takes about 30 seconds. Then I use Digital Research's LINK (which came with RMAK in CP/M+) to generate the BIOS.SPR file. I rename BIOS.SPR to .PRE and execute MAKEBIOS B:3 B: and I'm done.

You could also use the SLRNK linker to create the two BIOS.HEX files from the REL ffile and use PREL.COM (or GENSPR.COM, which I included on the disc) to generate the SPR (PRE) file. This eliminates one complete assembly from the old MAKEBIOS batch file.

I feel as you do about getting others to come out of the woodwork. There is so much work [still] to do with our H8s and H89s and so little time [in which] to do it.

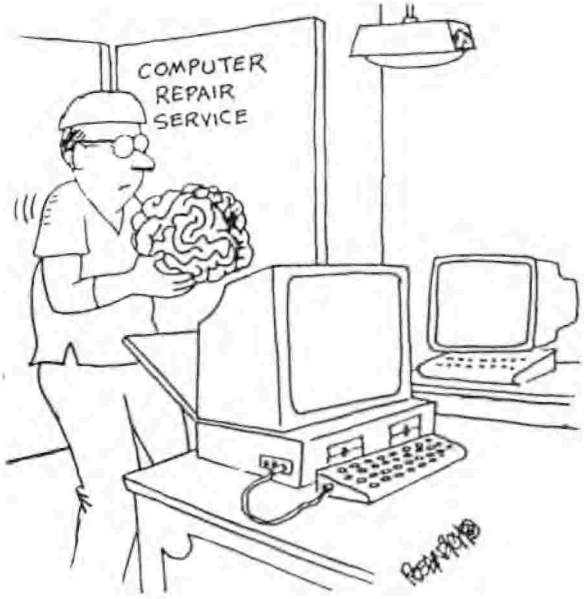
Since Heath/Zenith released HDOS 2.0 source code into the public domain, possibly we could get them to release source code for the CP/M utilities such as CONFIGUR, FORMAT, the cold-start and BIOS loaders.

I was talking with my peesee-consultant friend the other day and he told me his feelings about computers in general. He said that some day, we (the Ricks and Ludos and Jay Sages, etc.) will have to migrate over to the peesee, which by then will be a dinosaur. After the rest of the world discards them as worthless we will take those orphans into our homes, nurse them back to health, and finally write all those utilities and new operating-system enhancements which [they] are now lacking. And we will make the peesee do things which nobody believed possible. It's rather like what we're today doing with the H89!

RICK SWENTON, 106 Melinda Lane, Bristol, CT, 06010-7176

[Pretty heady stuff, you two! How about supplying us some vendor names & address, etc., for sources of those assemblers and other nifty items? I have an orphan ZCPR operating system which I'd like to try but have no idea of how to change it from H/Z90-only to H89-plus-soft-sector operation. You're quite right about the peesee eventually suffering the fate of our most-favourite Z80 machines, but since the original ibm peesee began its' life as an indifferently-smart terminal (later kludged up as a computer) don't you feel it'd be better to concentrate on the far superior multiple-CPU-types of "peesee-compatible" machines (H100s, etc.)? They're much preferrable over the generally-distastful jerking-and-blinking of the peesee's CRT display caused by the seriously overworked, solitary CPU chip! -- ed]

FILENAME: SMILES



As soon as you make something idiot-proof, along comes another idiot.

FIRST LAW OF MONEY DYNAMICS:

A surprise monetary windfall will be accompanied by an unexpected expense of the same amount.

"The bad news, Mr. President, is that an 11-year-old kid in Fort Wayne broke into the Pentagon computer—the good news is that he conquered Asia."

LETTERS POLICY

Our READER'S LETTERS/MAIL BOX is set up as an open forum for the free exchange of H/Z 8-bit computer information between all subscribers and readers of the SEBHC JOURNAL. We ask all correspondants to keep their letters reasonably concise and preferrably around 250 words maximum length (about six screens, 24 lines/screen of ASCII text). The JOURNAL shall exercise its' right to condense letters exceeding this recommended maximum unless that might destroy their intent or meaning. In such cases we shall contact the writer.

The SEBHC JOURNAL RESERVES THE RIGHT TO REFUSE TO PRINT any letter containing profanity, derogatory racist, or sexist remarks, specific political or libelous statements of any nature directed toward any individual or organisation. The JOURNAL will not knowingly publish malicious fabrications, lies, or distortions of fact, but will take appropriate legal action against any individual(s) uttering them.

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/Prov_____

Zip/PO Code _____Country_____

Phone number(s)_____

H/Z Computer Model(s)_____

Oper Sys: HDOS Ver____ CP/M _____

Computer used mainly for _____

_____

Renewal [_] New Subs [_]	$17.50
Non-USA [_] 1st class [_]	$25.00
Softcover Vol I [_] 22.50	_____
Softcover Vol II [_] 22.50	_____
TWO-VOLUME SET [_] 40.00	_____
CP/M GAME DISC #0[_] $6.96	_____
hard sector [_] $7.96	_____
HDOS "Programer's CARE Package" Disc #0--Soft $3.00	_____
Hard sector [_] $3.66	_____
HDOS GAMES DISC #1	_____
soft sector [_] $3.00	_____
hard sector [_] $3.66	_____
WordStar H/Z19/89 Keypatch	_____
soft sector [_] $12.50	_____
hard sector [_] $13.50	_____
TXTPRO DEMO Disc [_] $2.50	_____
HDOS hard-sector [_]	
CP/M soft-sector [_]	

=> Please pay in US DOLLARS by Cheque or Money order <= Total $_____

NOTE--From 1-Dec-88, only bound copies of Volume I and Volume II as above. Only $40 for Two Volume set of I & II! All Current Volume III back issues at $2.50 ea. Please allow 6 weeks for delivery of bound & back issues.

The SEBHC JOURNAL's Back Page

Society and Journal Policies

* The SEBHC JOURNAL is published once a month and strives to be mailed by the 20th of a month. Editorial copy deadline is the 10th of every month (weather & holidays permitting).

* Subscriptions: $17.50/year in Canada, Mexico, USA and its' possessions. First Class and Foreign are US$25/year. Subscriptions start the month following order receipt. Please make cheques or money orders payable to L E Geisler, NOT "the JOURNAL" or "SEBHC". Single back-issue copies are available at $2.50 each. See order blank for bound volume discounts.

* Subscribers are automatically Society of Eight-Bit Heath Computerists members. Member's subscription number and expiration follows their name on mailing label. The three member classes are: REGULAR (voting H/Z 8-bit user) ADVERTISING (one vote/vendor) and ASSOCIATE (non-8-bit computerist, library, etc.). REGULAR members can hold any elective Society office. ASSOCIATE members cannot hold office or vote. The Society's official yearly meeting place and time is announced every July in the JOURNAL. Advance registration of US$25 for each attendee is required by May first, please.

* All advertising is printed Free Of Charge. Vendors: Please do submit your B&W "camera-ready" ad copy, 7"w x 9"h (1 page to an issue) no later than the 10th of month in which it's scheduled to appear. All Society members can run one new free 250-word (maximum) Unclassified Want Ad every month.

* All subsribers/members are urged to submit their H/Z-oriented computer articles on disc in standard ASCII format rather than as hard copy. If a word needs to be emphasised or italicised please insert these symbols PRECEEDING the word: [EMPH] for emphasise, [ITAL] for italics. We'll return your disc after copying it and will gladly copy any SEBHC JOURNAL software disc onto it. Note: We can't pay authors but we do extend their subscription another year for a published article.

* 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. Other times: 313-769-6052 residence.

--==<<[[8-BIT POWER!]]>>==--

SEBHC Journal

895 Starwick Drive
Ann Arbor, MI 48105


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

ATTENTION POSTAL CARRIER:

Addressee HAS PAID FOR THIS ITEM. *PLEASE DELIVER IT!*

Alex M Bodnar jr [216.6.91]
C-33 Conowingo Circle
Oxford, PA 19363-1463