should be done in black ink on plain, white paper. It should be neat and clear and should contain no lettering less than the size of this printing. Disks will be returned to the author at the meeting following publication of their contents. EOF.

THE HIDDEN CP/M CLOCK
by Steve Leache

Approximately a year and a half ago, I noticed a message on the >CHUG bulletin board about a mysterious CP/M clock which supposedly lurked within the Heath BIOS source code. Interesting stuff, but since my H-89 and I were inexperienced, I only made a mental note of this message and returned to generating syntax errors and all the things computer novices usually do.

However this past winter, the need for some type of clock became apparent to me as I was entertaining some Cub Scouts during a den meeting. The program I had written counted in large block letters, Ten, Nine, Eight...etc. until at Zero, a rocket ship appeared and blasted off. (The rocket is from a REMARK issue years ago.) I wanted the H-89 to sit quietly for 15 minutes or so without calling attention to itself, and then during the meeting spring to life with the rocket program. What I did was precede the program with a series of nested delay loops to eat up the 15 minutes. Had I been able to sample a real time clock ticking away somewhere in memory, I could have replaced my crude delay loops with a more sophisticated peek at the clock.

I thought of the year old message, and began...

Initially things were surprisingly simple and understandable. I found the equate for TOD (time of day) in BIOS.ASM, changed it from false to true, did a MAKEBIOS to create a new BIOS.SYS and using the following MBASIC programs, I had a running clock.

```
5 'SETS THE CP/M BIOS CLOCK
1Ø PRINT:PRINT"ENTER HOUR (ie. Ø8)";
2Ø HOUR=VAL(INPUT$(2))
25 IF HOUR>24 THEN PRINT" Incorrect entry ":GOTO 1Ø ELSE PRINT HOUR
3Ø PRINT"ENTER MINUTE           ";
4Ø MINUTE=VAL(INPUT$(2))
45 IF MINUTE>59 THEN PRINT" Incorrect entry ":GOTO 3Ø ELSE PRINT MINUTE
5Ø PRINT"ENTER SECOND           ";
6Ø SECOND=VAL(INPUT$(2))
65 IF SECOND>59 THEN PRINT" Incorrect entry ":GOTO 5Ø ELSE PRINT SECOND
7Ø CLOCK=(PEEK(1Ø)*256)+PEEK(9) 'POINTER TO CLOCK IN RAM
8Ø POKE CLOCK-9,HOUR
9Ø POKE CLOCK-1Ø,MINUTE
1ØØ POKE CLOCK-11,SECOND
11Ø 'CLOCK-8,-7, AND -6 CAN BE USED FOR DAY, MONTH, AND YEAR
12Ø PRINT:PRINT"     CLOCK NOW SET"
13Ø END
```

```
5 'READS AND PRINTS THE CP/M BIOS CLOCK
1Ø CLOCK=(PEEK(1Ø)*256)+PEEK(9)
5Ø HOUR$=STR$(PEEK(CLOCK-9))
```

```
6Ø MINUTE$=STR$(PEEK(CLOCK-1Ø))
7Ø SECOND$=STR$(PEEK(CLOCK-11))
8Ø PRINT:PRINT HOUR$;" HOURS ";MINUTE$;" MINUTES ";SECOND$;" SECONDS
9Ø END
```

```
5 'REALTIME ROUTINE KEEPS TIME WHILE WAITING FOR A KEYSTROKE
1Ø CLOCK=(PEEK(1Ø)*256)+PEEK(9)
2Ø HOUR$=STR$(PEEK(CLOCK-9))
3Ø MINUTE$=STR$(PEEK(CLOCK-1Ø))
4Ø SECOND$=STR$(PEEK(CLOCK-11))
5Ø CHOICE$=INKEY$
6Ø IF CHOICE$<>"" THEN 9Ø
7Ø IF SECOND$<>LASTVALUE$ THEN PRINT CHR$(7);:LASTVALUE$=SECOND$
8Ø IF VAL(SECOND$) MOD 1Ø THEN 2Ø
9Ø PRINT HOUR$;" HOURS ";MINUTE$;" MINUTES ";SECOND$;" SECONDS
1ØØ GOTO 2Ø
```

Then things got interesting...

I have the 2+4 mhz. Najarian module in my machine and my new BIOS with the TOD did not support it. I had to run it thru the programs on the Najarian source disc to further modify the BIOS for 4 mhz. operation. Nifty, except after the additional Najarian code was inserted, my new BIOS.SYS didn't work! (A point of interest here - one of the Najarian mods is to set the BRKKEY EQU true (break key) in the CP/M source code. You can do this whether you have the Najarian device, or a stock Heath unit. With the break key active, disc swaps under CP/M can be accomplished with this one key rather than the more cumbersome Control-C we have been using.) (Also note - change the FDHDD equate to 4Ø to allow the head to settle a little more if you have encountered numerous BDOS errors.) Anyway, why didn't the new BIOS work in operation with the Najarian mod? Thoughts of port conflicts and coding errors clouded my mind.

After a great deal of experimentation I realized the machine would work with...

(1) TOD ---	(2) H-17 DRIVER -----------	(3) H-37 DRIVER -----------	(4) NAJARIAN MOD ------------
-->NO	YES	YES	YES
YES	-->NO	YES	YES
YES	YES	-->NO	YES
YES	YES	YES	-->NO

But not with...

YES	YES	YES	YES

The problem obviously was size of the BIOS. CP/M only allows a specific area in ram for the BIOS to load, and my new creation was too big. I got to work with my editor. (You must use an editor of the Wordstar variety. Editors like Pie which work only from ram won't have enough room to load the entire BIOS.ASM file.) First I eliminated the sign on message. (Now when my machine boots up, it simply says 64k.)

Experimentation proved that I needed a still smaller BIOS, so my new BIOS.ASM went under the Wordstar knife still again. I shortened the already cryptic error messages even further and finally succeeded. (Thank goodness for Heath's Submit Makebios, as after each attempt a new BIOS had to be constructed.)

I now had my 4 mhz. operation, using both my hard and soft sector drives and my clock as well.

If only it had been accurate...

The clock is in software, so I expected it would lose time stopping with each disc access, but this devil lost one second every minute just sitting there running. I recalled an article in REMARK several years ago by Pat Swayne. The article included a calibration factor for the HDOS CK.DVD (A clock device driver by Dale Lamm). What Pat did was to let the clock run for a while, and then caused it to skip a few seconds to correct itself. I chose not to use his exact method. I reasoned if my program was waiting for the time to be exactly 9:ØØ o'clock and the clock skipped over a few seconds, I might not see 9:ØØ for another 24 hours. After studying his code, I did pattern my modification after some of his ideas. However, my modification keeps accurate time without ever skipping a second. Using your editor, add the following code to BIOS.ASM

(The newly inserted code is designated by the *)

```
CP/M BIOS.ASM     (Page 97 of BIOS.ASM listing)
---------------------------------------------------
              IF TOD
   NDAYS      DB        31,28,31 etc.
              ENDIF
 * DSAVE      DW        Ø
 * CALFAC     DW        Ø6ØØH
 * MYCNT      DW        Ø
   TODVAL     DB        Ø,Ø,Ø,Ø,Ø,Ø
   EVTCTR     DW        Ø
   DLYMO:     DB        Ø
   DLYH:      DB        Ø
   DLYW:      DB        Ø
   CLOCK      SHLD      HSAV
              POP       H
              SHLD      RETSAV
              PUSH      PSW
*             XCHG
*             SHLD      DSAVE
              LXI       H,CTLPRT
              MOV       A,M
              OUT       H88CTL
              INX       H
              MOV       A,M
              ORA       A
              JZ        CLKØ
              OUT       H8CTL
   CLKØ:      LHLD      TICCNT
              INX       H
              SHLD      TICCNT
```

```
    *       LHLD    MYCNT
    *       INX     H
    *       SHLD    MYCNT
    *       XCHG
    *       LHLD    CALFAC
    *       XCHG
            MOV     A,L
            ORA     A
            JNZ     CLKRET
    *       MOV     L,D
    *       SHLD    MYCNT
            IF      TOD
            MOV     A,L
            RAR
```

End of modifications for Page 97.

CP/M BIOS.ASM   (Page 99 of BIOS.ASM listing)

```
    CLKR2:  POP     PSW
            LHLD    RETSAV
            PUSH    H
    *       LHLD    DSAVE
    *       XCHG
            LHLD    HSAV
            EI
```

End of modifications for Page 99.

It all boils down to this...

Get your BIOS.ASM and modify the code for TOD, BRKKEY, and FDHDD. Add the code for the calibration factor, and do a SUBMIT MAKEBIOS. If you have both drivers, and the 2-4 mhz Najarian module as well, you must also edit out all the excess text and run the BIOS.ASM through the Najarian source for modification before you do a SUBMIT MAKEBIOS.

After you have a running system...

Try the following program, CALFAC.BAS. It will return the calibration factor to you as the most significant bit (MSB) of DW, along with its address in memory.

```
1Ø 'CALFAC.BAS      FINDS THE BIOS CLOCK CALIBRATION FACTOR
2Ø CLOCK=(PEEK(1Ø)*256)+PEEK(9)
3Ø PRINT, "CALFAC DW     MSB=";HEX$(PEEK(CLOCK-14))
4Ø PRINT,"ADDRESS       ";CLOCK-14
```

The Ø6 at CALFAC DW is the factor which causes my unit to keep accurate time. (Remember, a software clock will always stop while a disc is accessed.) Check your system against a watch to see if the clock is accurate for you. If not then at the MBASIC prompt, type the following one liner.

```
            POKE((PEEK(1Ø)*256)+PEEK(9)-14),n
```

Where n is the value you want to try at CALFAC. (Such as 07 or 05)

Change the BIOS.ASM to reflect your new CALFAC DW value and once again do a SUBMIT MAKEBIOS to create your personal version.

This project, while seemingly straight forward to someone with a good knowledge of CP/M, is admittedly somewhat more than trivial to a novice. Nevertheless, an attempt to accomplish this project will provide the novice with a better understanding of CP/M, the Heath MAKEBIOS method, and add another useful tool to his programming workshop. EOF.

GEMINI-10X PROBLEM: THE RIBBON JUMPS OUT
(A probable cure, which seems to work so far!)

by

Jerome H. Horwitz

I have had a Star Micronics Gemini-10X printer for about a year, now. In the last few months, a rather disconcerting problem has arisen: When using fan-fold paper, the ribbon would usually jump the track (come up out of its slot in front of the print head) when printing across the edge of a sheet. Apparently, the fold would rub against the ribbon sufficiently if the print pins were operative to actually drive the ribbon up out of its position. The problem seemed worse if the ribbon was near or at one end. Usually the ribbon would get very slack on the side of the print head which was ahead of its travel. Not only would the ribbon jump out from in front of the head, but it also usually jumped one or both spool guides adjacent to the platen roller.

The Gemini office in Boston advised the following cures, which seem, at least so far, to work:

1) Remove both ribbon spools. Beneath each spool is a small "C" type retaining ring around the spool shaft. Carefully remove the ring (do not lose it--the spool platform it retains is under spring tension and will pop up). Lift the platform off the shaft and set aside. Under the platform is a spring, which is compressed by the platform when it is in place. Remove the spring and stretch it to increase the tension. I suggest adding about 3/8 or 1/2 inch to its unloaded length. Replace the spring and platform. Press down on the platform and carefully replace the retaining ring. (Three hands would help here!)

2) Remove the ribbon from its track between the print head and the ribbon guide (the shiny metal piece between the print head and the platen roller). Grab the right and left edges of this guide with a finger on each hand and bend toward you to permanently change its shape. You don't need too much of a bend --the idea is to put increased pressure on the ribbon to hold it against the print head. If you can visibly tell that you have bent the guide, that is probably enough.

3) Remove the print head (two screws, one on either side of the head) and carefully turn it so the part where the ribbon passes is pointing up. Put a cloth below the head to catch any debris from the