4MHZ OPERATION AND THE TRIONYX Z80

Prepared by

Tom Jorgenson

Software Wizardry, Inc.

January 5, 1982

TABLE OF CONTENTS

I. Introduction

This support package was developed under contract to Trionyx Electronics, Inc. for their new Z-H8 Z80A processor card. We feel that this card is the most advanced CPU on the market for the H8 to date and have tried to make the most of its advanced features - particularly the ability to toggle between clock rates "on the fly".

Most of the newer, (i.e. 179X-based), Heathkit controller cards handle disk I/O independent of the CPU clock rate, but the H17 controller design is, unfortunately, of an older discipline. This controller uses software timing loops which execute twice as fast at the 4 MHz clock rate and which require modification in order to run properly.

The thrust of this package is to modify the disk timing constants to properly duplicate the normal, (2 MHz), Heathkit timing periods and minimize soft errors when running under the 4MHz clock. Unfortunately, this is not quite as easy a task as it may appear since the Z80 environment adds several new variables we must contend with.

On first glance it would appear that merely doubling the loop timing values would be sufficient to properly operate at the higher clock rate, but in actuality there are three reasons why this method is unsatisfactory.

First of all, Z80 instruction execution time does not duplicate the time an 8080 takes to perform the same instructions, (even at the SAME clock rate). One or two instructions executed with this error make little difference, but our timing loops often perform such instructions many times.

The addition of wait states to memory I/O cycles on the Z80, (one of the very nice features of the Trionyx implementation), creates a deliberate slow-down of the Z80 execution periods at 4 MHz. This slow-down is not generated in a simple fashion since the idling time added is a function of machine cycles rather than either simple clocks or complete instruction periods.

Lastly, the Heathkit H17 device driver (HDOS) has some clock-dependent code within both the READ and WRITE routines when multiple-sector I/O is being performed. These routine must be modified or soft errors will result due to the WSC, (wait for sync character), loop timing out too quickly.

All three of these factors were taken into account when the new timing constants were calculated - and the results were substantiated in the laboratory with test equipment, (not merely by running them on a system). We feel we have duplicated the original H17 2 MHz timing with a very high degree of accuracy.

The SPEED utilities included in this package make use of the Z80 status port to determine the run-time clock rate and wait state settings in order to identify the precise constants required for proper operation in each particular case. There is no more accurate way to correct disk timing for 4 MHz operation.

A. Why is 4MHz Operation Desirable?

Fast as computers are, we users keep making more and more demands upon them - and more and more we keep finding ourselves waiting for the machines to complete their tasks.

This is pretty common during the assembly of large programs and execution of large BASIC programs on an interpreter, (the two operations in widest use).

Running a CPU at a higher clock speed, (in our case, double), will free up an awful lot of this time. The processor can now complete nearly twice as many CPU tasks in the same amount of time; and with the same final result.

Exactly how much time you will save depends upon where most of your time is being lost. Time spent on disk I/O will not change, since it is dependent upon the device rather than the CPU. Internal (i.e. in-memory) tasks will complete much faster. You can demonstrate this later on by setting up a bubble-sort under Microsoft Basic and executing it upon array data in memory. You'll be surprised at just how much of a difference there actually is!

B. What is Required for 4MHz Operation?

This package was generated specifically to support the Trionyx Z80 card, and can only be considered reliable if run on this card. The constants we use are not the same as those that would be required on any other Z80 board implementation - use of these routines therefore presupposes that they are run on the Trionyx card.

It goes without saying that these routines assume that the user has properly configured his Z80 card, especially in so far as the wait states are concerned. These utilities will interrogate the Z80 status port to determine the run-time settings of these jumpers; but it is the users' responsibility to verify that his memory cards are fast enough to operate with the wait states he has configured. We only mention this here because, in many cases, the first sign of running memory cards with too few wait states will be unreliable disk operation.

If the user wishes to use the SPEED support utilities without modification, there is the additional requirement that he is running either HDOS 2.0 or before, (i.e. non-0 origin), or that he is running CP/M 2.2.02 with an unmodified BIOS, (i.e. he has not changed the locations of the constants by modifying the source code). Users capable of modifying their own BIOS are also capable of identifying the necessary changes to the SPEED utilities to make them function under the new system. Ordinarily this is as simple as changing the offset labels at the end of the SPEED routine.

In such cases where utilities or packages are designed to modify the users' BIOS automatically, (such as BIOS80 by Livingston Logic Labs), and in the cases of enhanced HDOS drivers; Trionyx will cooperate in-so-far as making sufficient information available to the author of the package to allow him to retain compatibility with this system. As a matter-of-fact this is already being done with both the LLL BIOS's and the Ultimeth Corporation Tandon device driver.

C. What are the Limitations?

Strangely enough, there ARE some minor drawbacks to operating your computer system at the higher clock rate. The most obvious limitation, (although some people actually find it desirable), is the execution of certain utilities at twice their normal rate.

Running games such as INVADERS at 4MHz has been described as thrilling by some persons - but this has the potential of becoming downright irritating in cases where you DON'T WANT the utility to complete in the faster time.

This is why the Trionyx Z80 card provides the ability to toggle between clock rates without stopping system operation. The user can come up in one clock rate and switch it to the other without ever reBOOTing his system.

Toggling clock rates is not so simple as to be easily incorporated into most assembly-language routines and this may be considered a limitation by some. The only easy way to toggle between clock rates is to come back to the command level (the PROMPT) and execute the SPEED utility. It is possible to incorporate the SPEED routines within an application utility, but we feel it is ill-advised since it carries a lot of code overhead and would support only the Trionyx system with only the standard operating systems.

Finally, adapting the SPEED routines to non-standard systems requires some knowledge of the operating system and some basic assembly-language familiarity. This is, unfortunately, unavoidable but also, (luckily), unusual.

II. HDOS Operation

HDOS is the simpler of the two operating systems to update to the new clock rate. This is because the HDOS timing constants are located in reserved memory locations where they may be easily (and safely) altered to the corrected values. These memory locations, (until now, at least), are fixed between HDOS revisions and the distribution drivers all utilize them when these timing loops are dependent upon the CPU clock.

Most users need not concern themselves with the details of how the 2 to 4 MHz changes are brought about but need merely to verify that their system is unmodified and of the correct revision to use the supplied utilities without modification. This support package is only necessary in cases in which the H17 ROM disk routines are being used - this is, (to the best of our knowledge), only on the H17-type drives and with the Ultimeth Corporation Tandon device drivers.

Those running without the use of these drivers can, most probably, avoid use of these utilities - except possibly in cases where they wish to use them only to toggle clock rates. (modifying the H17 routines when not used will not affect other system performance).

Please do not be concerned about the amount of information included here about how we are accomplishing this feat of magic unless you require this information for a non-standard implementation. It is provided purely to satisfy the needs of those who either prefer to customize their operating systems or just plain have 'a desire to learn'.

A. Necessary HDOS Modifications

The HDOS constants requiring modification for 4 MHz are as follows:

D.WRITA	040.112	Times the period during WRITE's from when a sector is located until the sync pattern is to be written.
D.WRITC	040.114	Times the period from when the sync pattern is first called to be written until the first pattern byte is output.

D.WHDA	040.123	Times the period from when a hard sector hole is detected until the WHD (wait for hole detect) routine returns to its caller.
D.WNHA	040.124	Times the period from when a no-hole is verified until the WNH (wait for no-hole) routine returns to its caller.
D.WSCA	040.125	Timeout counter for a missing sync byte.

These constant addresses are all defined in the EDCON.ACM common deck which is supplied on the distribution HDOS diskettes. The values they contain determine the H17 disk timing and must assume the values shown in Table I in the Appendix.

There are also two lines of code within the H17 READ and WRITE routines which require modification in order to avoid spurious, (although harmless), soft errors. The code in these two areas deals with multiple-sector reads and writes, and is dependent upon the system clock for inter-sector timing. These two routines are located at the READ2.4 label (address 034.130 in the H17 ROM) and at WRIT2.5 (address 034.377).

In both of these cases the secondary sectors are timed using the D.UDLY, (microsecond delay), routine instead of looking for the next hard sector hole - and in both cases we replace:

```
CALL D.UDLY          * 040.216
```

with:

```
CALL WHD             * 036.235
```

This does not affect system performance in any way, but allows operation of these two routines without dependency upon system clock (and without spurious 'Missing Header Sync' errors).

B. The HDOS SPEED Utility

The HDOS SPEED utility is an assembly-language program which will interrogate the Trionyx Z80 status port and set-up the necessary disk constants and sector I/O patches to enable accurate operation at the 4 MHz clock rate.

This utility consists of three parts: the source code (SPEED.ASM), the prologue file (PROLOGUE.SYS), and the command file (SPEED.ABS). Both the prologue file and the command file are actually derived from the same source code - the only difference is in whether the PROLOG label is EQUated to TRUE (for PROLOGUE.SYS) or FALSE (for SPEED.ABS).

If you install the PROLOGUE.SYS file on your system disk prior to BOOTing the disk it will automatically verify that a Trionyx Z80 card is present, determine the wait state settings, and modify the system to run at 4 MHz. During this process there will be advisory messages displayed on the console to verify that the operation has been performed (or NOT performed if a Z80 is not in the system).

The SPEED.ABS file is basically the same program as the above PROLOGUE file, except it is intended to be used as a command file and will request input from the operator to determine the operation it is to perform.

The format of the SPEED command line is simply:

>SPEED <Return>

Operation under any version of HDOS from 2.0 back is as simple as that! As long as the EDCON.ACM addresses are unaltered, this routine should work under any version of HDOS.

C. 4MHz with Special Disk Drivers

At this time we know of no disk driver which does not fall into either the class of 179X controllers, (like the H47 drives or the soft-sector controllers), or the class of those using the H17 ROM routines for the actual disk I/O, (like the standard SY.DVD driver and the Ultimeth drivers).

When we talk about Special Disk Drivers we are referring to device drivers which use software timing loops, (like the H17 drives - but not based on the 179X controllers), which are not enclosed within the H17 ROM itself.

In these cases it is the responsibility of the author of the device drivers in question, (or the user), to provide the necessary changes to maintain compatibility with the Trionyx Z80 card. Trionyx cannot maintain someone else's software code, but we do include a brief description of what is required here for someone who wishes to build-in this compatibility.

The first task of this individual should be to fully acquaint himself with the SPEED.ASM source code and this documentation. Properly written, his driver should, first of all, verify the presence of a Trionyx Z80 card, (via the Z80.TEI bit from the status port), next he should identify the current wait states and clock rate being operated, (again, from the status port), and finally he should modify his timing loops in accordance with this information and Table I in the Appendix.

The Table I values are calculated for use with PRECISELY the same timing loops as are part of the H17 controller. This includes JMP instruction timing, (via the RAM vectors), and EXACTLY the same code as in the timing loops right now. If these timing loops are modified in any way, the person authoring the new driver MUST recalculate these values himself directly from empirical data and use those values instead.

These calculations are of a non-trivial nature and require a solid understanding of the Z80 itself, micro-instructions, and hardware. We would advise against deviating from the standard Heathkit routines for this reason.

III. CP/M Operation

CP/M is a remarkably similar problem to HDOS since the disk routines in Heathkit CP/M are virtually exact copies of those used in the Siemens device driver. The most obvious differences under CP/M are the addition of one additional timing constant, (called READA), and the fact that the CP/M disk constants are relocated between different addresses by MOVCPM.

Due to slight differences in the disk timing loops in their CP/M BIOS; Heathkit found it necessary to add an additional delay loop to 'eat up' the extra time it found during sector reads. The addition of this loop allowed the CP/M disk constants to assume the same values as the HDOS constants, with the one exception of this additional loop counter. This is the origin of the READA value.

In our application, (i.e. 4 MHz), however, these routines do not actually execute in the same period of time - but actually in a slightly different period of time. We could have taken the Heathkit approach, once again, but chose instead to duplicate each timing loop individually from the original Heathkit timings. This is why the CP/M constants are slightly different than those used under HDOS.

A. CP/M Modifications

The CP/M constants requiring modification for 4 MHz are as follows:

WRITA	06EAH+REL	Times the period during WRITE's from when a sector is located until the sync pattern is to be written
WRITC	06F2H+REL	Times the period from when the sync pattern is first called to be written until the first pattern byte is output.
WHDA	08CBH+REL	Times the period from when a hard sector hole is detected until the WHD (wait for hole detect) routine returns to its caller.
WHNA	08D6H+REL	Times the period from when a no-hole is verified until the WNH (wait for no-hole) routine returns to its caller.

READA	08EBH+REL	Compensates for the difference between HDOS and CP/M timing to allow these constants to assume nearly-identical values.
WSCA	08F7H+REL	Timeout counter for missing sync byte.

If you are using an unmodified Heathkit 2.2.02 BIOS you needn't concern yourself with these values further. The SPEED utility will automatically locate them and install the appropriate corrections. If, however, you have found it necessary to modify your BIOS you should identify the new addresses where these values are stored and modify the SPEED routine to reflect these changes. (this can be done from a .PRN listing of the BIOS assembled with an origin of 0000H). The addresses given are for the offsets of these values from the start of the standard Heathkit BIOS (revision 2.2.02).

Since CP/M performs disk I/O on a single-sector basis, there are no multiple-sector I/O changes necessary as there were with the HDOS system.

In addition to disk constant modification, (as with HDOS), many CP/M utilities use their own internal timing loops, and these require modification in order to be run at 4 MHz as well. The routines in question are: MOVCPM5.COM, FORMAT.COM, DUP.COM, and the BIOS pre-loader.

MOVCPM, FORMAT, and DUP all contain their own disk routines, (similar to the H17 ROM routines), and do not function via BIOS calls. The BIOS pre-loader requires modification only when the Z80 is set-up to BOOT directly into 4 MHz. If the user merely wishes to BOOT up at 2 MHz and run AUTO.COM to toggle to 4 MHz, the pre-loader changes are not necessary and should not be installed.

Modification of the three utilities, (MOVCPM, FORMAT, and DUP), requires familiarity with DDT.COM, (from the CP/M distribution disk). The BIOS pre-loader changes require a suitable disk-modification program, (such as DUU.COM from the CP/M Users' Group). Once these changes are installed in these utilities they WILL NOT function properly at 2 MHz anymore! For this reason it is suggested that the user merely use the SPEED command file to toggle down to 2 MHz prior to using one of these utilities, and therefore avoid the effort, (and confusion), resulting from having two sets of utilities in use.

There are some individuals, however, who will prefer to

operate at 4 MHz all the time and will prefer to put forth the effort these changes involve. For these persons the methods follow.

The first step is to identify your hardware configuration, (number of wait states you have selected). Armed with this information, refer to Table II in the Appendix to determine the necessary constants to be installed in these programs.

Table III properly identifies the memory addresses within each routine to be changed, (or in the case of the BIOS pre-loader, the disk addresses).

Note that the BIOS pre-loader is a special case in that you must be certain to have the correct constants installed within it to match your hardware during the BOOT process, or the BOOT will probably not complete properly. If you are going to BOOT directly into 4 MHz, (i.e. without AUTO.COM doing the clock-rate toggle), you must be certain to always change the pre-loader constants BEFORE attempting a BOOT under a new hardware configuration.

B. The CP/M SPEED Utility

Just as with HDOS, the CP/M SPEED utility will make the necessary modifications to the operating system to facilitate operation at the 4 MHz clock rate.

In the case of CP/M, however, the SPEED utility is very dependent upon the version of CP/M in use. This is because the CP/M disk constants are located by their normal (offset) position within the BIOS. If SPEED is executed with either a modified BIOS or the wrong revision, the results will be indeterminate, (and dangerous). Be sure to modify SPEED to match your operating system if it falls into one of these two classes.

This utility consists of three parts: the source code, (SPEED.ASM), the AUTO file (AUTO.COM), and the command file (SPEED.COM). These files bear the same relationship to each other and the operating system as the three HDOS files previously mentioned.

If you wish to automatically toggle from the BOOT rate of 2 MHz to 4 MHz using AUTO.COM you must first run CONFIGUR.COM from the distribution CP/M disk to inform the CP/M system that it is to BOOT into an AUTO file. Refer to the CP/M documentation for details on operating in this manner.

Once the AUTO option is properly set in CP/M and the AUTO.COM file is installed on the BOOT disk during reboot, AUTO will behave in the same fashion as the HDOS PROLOGUE.SYS file. It will determine the current operating parameters of the Z80 card and install the necessary timing constants to enable reliable 4 MHz operation of the system. Once again, it will display advisory messages; and toggle the clock, (unless already operating at 4 MHz). No operator input will be required.

The SPEED.COM file will behave similarly, except it will advise the operator as to the current operating characteristics of the Z80 card, and allow the operator to toggle the clock rate in either direction.

The format of the SPEED command line is:

A>SPEED <Return>

Operation under CP/M 2.2.02, (with unmodified BIOS), is as simple as that! These routines will have to be modified to correctly operate under other CP/M versions or if the BIOS has been altered.

C. 4MHz with Modified BIOS

It is assumed that, if the user has modified his BIOS, that he can also install the corrected disk constants located in the Appendices with a little additional information. If any question arises, please refer to the CP/M documentation regarding generation of a modified BIOS (from source code).

Refer back up to Section A and the labels identified there for modification -- these are the labels to locate, (and alter), in your modified BIOS source code. The values to change them to are identified in Table II in the Appendix and assume unique values determined by the wait states installed in the system.

Modification of the wait state settings also requires similar modification of these constants in the BIOS.

These modified constant values are calculated for use with PRECISELY the same timing loops as the normal H17 controller. If these timing loops are altered in any way, these values must be recalculated from empirical data.

Such calculations are non-trivial and require a thorough

knowledge of the Z80, micro-instructions, and hardware. We therefore advise against any variance from the standard H17 controller physical disk routines.

V. In Case of Trouble

We have tried to present a thorough discussion of the problems, and methods supported under the Trionyx Z80 CPU card and this support utility package -- but problems can, (and do), arise.

The first step in identifying where the system went awry is to verify that the system hardware is properly configured for the system capabilities. If there is any doubt about the system memory being fast enough to handle the current wait states, (even if it SEEMS to be operating OK at 2 MHz), try adding a wait state or two, (and whatever changes, if any, are necessary to the software as well to support these), and try again.

Verify that the correct version of HDOS or CP/M is running on the system -- as we mentioned before, this package is really intended for use with HDOS 2.0 and CP/M 2.2.02 and may require modification for use on any other system. The BIOS must be unmodified or the user will have to change the constant values in source code, (or the SPEED utility offsets), to accomodate the changes as well.

Re-read the documentation on both the Z80 card and this support package to insure that the proper steps are being taken and there is no doubt that you are using these tools in the right fashion.

Finally, failing the above, contact Trionyx with the details of your plight. Remember that troubleshooting a system via phone is one of the most difficult of tasks in this business, and please be prepared with as solid a description of the problem as you can to help simplify the process, (for everyone concerned).

Thank you for your support, we hope you will enjoy your new Trionyx Z80 system as much as we all do!!

V. Appendices

Table I: The HDOS Timing Constants

D.WRITA	040.112	D.WNHA	040.124
D.WRITC	040.114	D.WSCA	040.125
D.WHDA	040.123		

1. 2 MHz operation (always)

D.WRITA = 20 D.WNHA = 20
D.WRITC = 16 D.WSCA = 00
D.WHDA = 20

2. 4 MHz - No wait states (1W & 2W Set to Logic 0, [on])

D.WRITA = 46 D.WNHZ = 46
D.WRITC = 47 D.WSCA = 164
D.WHDA = 48

3. 4 MHz - 1 Wait on M1 (1W only Set to Logic 1, [off])

D.WRITA = 40 D.WNHA = 40
D.WRITC = 41 D.WSCA = 137
D.WHDA = 41

4. 4 MHz - 1 Wait on M1, 1 Wait on MREQ (2W only Set to Logic 1, [off])

D.WRITA = 35 D.WNHA = 35
D.WRITC = 36 D.WSCA = 122
D.WHDA = 36

5. 4 MHz - 2 Waits on M1, 2 Wait on MREQ (1W & 2W Set to Logic 1, [off])

D.WRITA = 28 D.WNHA = 28
D.WRITC = 29 D.WSCA = 101
D.WHDA = 28

Table II: The CP/M Timing Constants

WRITA	06EAH+REL	0675	WNHA	08D6H+REL	08E2
WRITC	06F2H+REL	067D	READA	08EBH+REL	08F7
WHDA	08CBH+REL	0807	WSCA	08F7H+REL	0903

1. 2 MHz (always)

WRITA = 20	WNHA = 20
WRITC = 16	READA = 48
WHDA = 20	WSCA = 80

2. 4 MHz - No waits (1W & 2W Set to Logic 0, [on])

WRITA = 45	WNHA = 46
WRITC = 45	READA = 107
WHDA = 46	WSCA = 151

3. 4 MHz - 1 wait on M1 (1W only Set to Logic 1, [off])

WRITA = 40	WNHA = 40
WRITC = 39	READA = 93
WHDA = 40	WSCA = 135

4. 4 MHz - 1 wait on M1, 1 wait on MREQ (2W only Set to Logic 1, [off])

WRITA = 35	WNHA = 35
WRITC = 35	READA = 83
WHDA = 35	WSCA = 121

5. 4 MHz - 2 waits on M1, 2 wait on MREQ (1W & 2W Set to Logic 1, [off])

WRITA = 28	WNHA = 28
WRITC = 28	READA = 67
WHDA = 28	WSCA = 101

Table III: 4MHz CP/M Utility Modifications

1. The following disk addresses apply to the BIOS pre-loader, modifying these constant must be made directly to a 5" standard Siemens floppy disk (10 hard sector), if required:

Track	Sector	Offset within Sector	Constant	Old Value
2	13	0014H	WHDA	14H
2	13	001FH	WNHA	14H
2	13	0034H	READA	30H
2	13	0040H	WSCA	50H

Track number is counted started with track 0, sector number is counted starting with sector 1. Sectors are as would be shown by DUU.COM, (as though there were 128 bytes per sector.

2. The remaining changes apply to the standard CP/M utilities and must be applied to these utilities if they are to be operated at 4 MHz. These addresses are specified as they will appear when the utiities are loaded under DDT.COM.

FORMAT.COM:

There are three special constants here, which always change as shown:

Address	Old Value	New Value	
04B2H	01H	02H	Pre-WSP delay (header)
04CFH	01H	02H	Pre-WSP delay (data)
057EH	01H	02H	Pre-WSP delay

These addresses should assume values based upon the constant table.

Address	Label	Old Value
05DBH	READA	30H
05E7H	WSCA	50H
06E4H	WNHA	14H
06F1H	WHDA	14H

DUP.COM:

There are three special constants in DUP as well which should always (at 4 MHz) assume the values shown:

Address	Old Value	New Value	
0603H	01H	02H	Pre-WSP delay (header)
0623H	01H	02H	Pre-WSP delay (data)
076CH	01H	02H	Pre-WSP delay

The following locations should assume the values identified in the constant table:

Address	Label	Old Value
088AH	WNHA	14H
0897H	WHDA	14H
0952H	READA	30H
095EH	WSCA	50H

MOVCPM5:

The following locations should assume the values identified in the constant table:

Address	Label	Old Value
2394H	WNHA	14H
239FH	WHDA	14H
23B4H	READA	30H
23C0H	WSCA	50H