Z-100 LifeLine
Since 1989

March 2022

#WEB

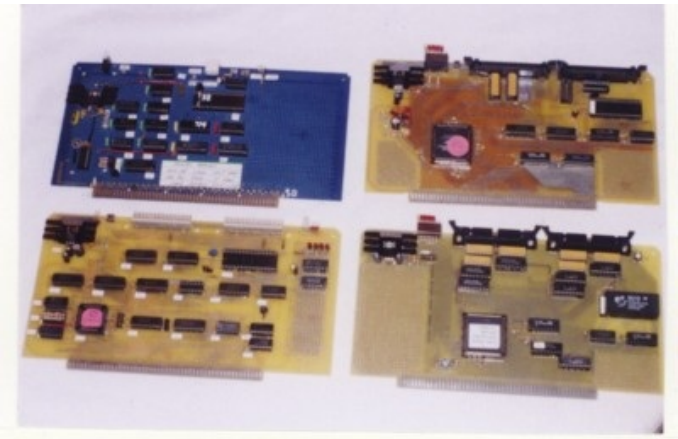
This article was first published in issue #109, February 2007

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# First Operational IDE Controller in a Z-100!

Finally, after years of work by the talented "*Z-100 LifeLine*" staff - Charles Hett, John Beyers, and myself, I get to test the latest version of the Z-100 IDE controller, version 3, which truly looks very promising.

In case you may have wondered why we were taking so long on development, and since this is really the first opportunity that I've had for showing you some pictures, I thought you might enjoy seeing the four IDE controller boards that we went through:



**Pix 1.**
**IDE Controller Boards**

The first board (top left) was our wirewrap board. Mine (pictured) has only the NVsRAM circuit and was described back in issues #94 and #97. Charles Hett had the full circuit on his board.

The second (bottom left) was our first prototype using a small Altera programmable integrated circuit, the medium-sized square in the lower left corner.

The third (top right) was our second prototype version using a larger Altera programmable chip (below and left of center) that replaced many of the smaller integrated circuits.

The fourth and hopefully last (bottom right) was our third prototype version. This one separated the left and right IDE connectors entirely and added several resistor packs. This one appears to have finally fixed our interference problems.

I thought the next logical step would be to see how someone would install the device in an actual Z-100.

So, using a version 2 controller with one of the drives that worked well with it, I decided on doing just that.

The first decision was which Z-100 model would be the hardest to convert?

The Z-120 model had a very adaptable drive tower that I've easily converted to hold several combinations of half-height floppy and hard drives and the plastic face plates were plentiful and easy to work with.

I have found the low profile Z-110 models were a bit less adaptable and harder to work with. But if you already had a unit with two half-height floppy drives, or with one half-height floppy and a full-height hard drive where the hard drive's faceplate was integral with the entire plastic faceplate, conversion is easy.

After much thought, I decided on an older Z-110 low profile model that sported two old full-height floppy drives with an aluminum faceplate that wouldn't serve any useful purpose anywhere else, and I just happened to have such a candidate on hand.

This model also did not have the usual hard drive power cables and these would not be necessary after the conversion either. However, as I had several Z-120 power supplies handy, for flexibility and testing purposes, I changed out the power supply and swapped the cases anyway.

So, follow along as we convert this old workhorse into a new, ready for anything, thoroughbred.

First, I dismantled the entire unit, carefully storing away the old full-height drives - you never know when someone might call looking for one of these old dinosaurs. Then I began with the power supplies. For some reason this old one had the connector for the MFM hard drive data separator card, but none for the Z-217 controller card. I had never seen that before. I wonder what it was meant to power?

Anyway, as I wanted the flexibility of being able to connect to old MFM hard drives if needed, I removed a newer model power supply from its Z-120 case, gave it a thorough cleaning and visual inspection for bad solder joints, and then installed it in the Z-110 power supply case.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Next, I cleaned and visually inspected the motherboard. Fortunately, it was a newer 5MHz motherboard with 768k RAM already installed using a RAMPAL kit, but it did not have all the chips deemed necessary by Heath/Zenith for 8 MHz operation. Undeterred, I slapped in a 24MHz oscillator assembly at U236 anyway. For those who may not recall, I've described this assembly way back in issue #78 and its use in several articles since.

From experience, I have found that most 5 MHz newer motherboards can easily be pushed to about 10 MHz with the change of just a few chips. I changed the 8088, U211, to a 8MHz unit and the delay line at U149 from a 200ns unit to a 100ns unit (a 150ns unit would have been better, but I did not have one). I also changed the ROM chip at U190 to our new ROM v4.3 and remembered to change the jumper at J101 from 0 to 1 as required for the larger ROM chip.

**Note:** If you forget to change this jumper, no harm is done; you will just find that you won't have video.

Next, I cleaned and visually inspected the video board. It had 64K chips in all three banks and it was configured properly for color.

Next, I cleaned and visually inspected the floppy controller. It was an older model and obviously hand assembled, though very neatly done. As I was hoping to install a single 5-1/4" floppy and a single 3-1/2" floppy, I went ahead and completed the Barfield modifications as described in another article on the "*Z-100 LifeLine*" website.

Finally, I felt ready to take the unit on a test spin. As with most complicated systems, when powering up an unknown Z-100 for the first time, I find it is always best to reduce the computer to the bare minimum number of boards to eliminate as many variables as possible for trouble-shooting purposes. The reduced load also relieves pressure on the power supply, which may not have been on in quite a while, allowing the capacitors to reform properly before being placed in full service.

So, I installed the power supply, motherboard, and video card, and then attached the keyboard, power cable, and monitor.

Rechecking everything in place, I crossed my fingers and powered up. As expected, I got the initial "Primary Z207 Controller ERROR" because I did not bother installing any of the controller cards yet and the floppy controller was the default device (set by section 3 of the bank of switches at S101 on the motherboard). At the hand prompt, I selected the memory test and let it run. No problems.

I swapped out the 24MHz oscillator with a 28MHz oscillator and tried again. A short time later I got my first RAM error.

I rechecked the speed of the chips - 120ns, which was certainly fast enough, and I retried the memory test several more times.

Interestingly, on each run, a different chip was reported bad. OK, so I reasoned it was not the actual RAM chip that was bad. In cases like these, I normally go to the manual describing the HA-108 speedup and memory upgrade and try changing out the chips described in that upgrade.

In this case, I began with those chips around the memory and front of the motherboard and quickly found that swapping out the 74LS368 chip at U200 with the 8T98 recommended by the upgrade manual fixed the problem. This chip is in the CPU clock circuit and so could logically cause the random RAM errors.

I swapped out the 28MHz oscillator with a 30MHz oscillator and tried again. Everything was still OK.

Powering down again, it was time to try the floppy controller board. I installed the board and attached a 5-1/4" drive and powered back up. This gave me bad floppy controller errors.

The HA-108 upgrade mentions that at faster speeds we need to install a wait state jumper at J106-1, which is located beside U233A near the front right corner of the motherboard.

Installing this jumper eliminated the bad floppy controller errors.

Great, time to add the MFM hard drive system so I can boot the beast and run full diagnostics.

I have never found a Z-217 hard drive system that would not work properly at 10 MHz and today was no exception. The unit booted fine and I began to run diagnostics.

RAM was fine, but when I tried testing the floppy controller board, I got access errors, register errors, and head load errors!

Slowing to 8 MHz and swapping out Z-207 floppy controller boards gave me all kinds of differing symptoms, most of which I had expected.

To sum up, I have historically found that the most speed sensitive chip on the Z-207 card seems to be the controller chip at U22. The popular WD1797B controller chip is rated for 5 MHz and the WD1797B-02 controller chip is rated for 8 MHz. However, in practice, I've found some 5 MHz chips work to 10 MHz and some 8 MHz chips will not run at 8. If you are getting access, register, or read/ write errors, you need to slow down. The controller chip is generally the culprit and these chips can no longer be located for reasonable prices.

If swapping out the U22 controller chip won't fix the problems, then look to the chips mentioned by DIAG. Don't limit your trouble-shooting to those chips, as these are just suggestions and due to interaction with other chips on the board, the actual culprit could be elsewhere. You may need to swap out every chip on the board to find the problem.

Anyway, out of 5 floppy controller boards that I had handy (some of which already had known problems that I hadn't had time to troubleshoot yet), I finally managed to get all the boards to the same level of repair. I had to recalibrate three boards - one had a bad R1 potentiometer, another a bad U5 (74LS624) chip and the third just needed readjustment.

I had to slow to 6.3 MHz to get all the controllers to work, but at 8 MHz, two boards still reported Head Load errors. Without going into circuit theory, Head Load is only used for eight inch drives and is set by resistor R18 and capacitor C48 attached between pins 1 and 2 of U15, a 96LS02 chip.

It seems that on these two boards the timing is just off enough to cause errors about 20% of the time at 8 MHz. I've tried replacing the capacitor on one board, but the old capacitor tested as good as the new one and replacement with the new one did not eliminate the problem.

Likewise, in-circuit testing (while removing U15) showed the resistors and the capacitors were at their correct values. So, the fault is either the resistor/capacitor timing is just outside the desired value or the testing done by DIAG is affected by the faster computer speed and incorrectly reports an error. I will have to investigate this further later.

A more prevalent error, affecting all 5 of my boards at 8 MHz or greater, reports that power is not on at the drive with the failure rate increasing with any increase in CPU speed.

As DIAG was designed to run at 5 MHz and possibly may run properly at 8 MHz, I am convinced that this may be a problem with DIAG, as the controller reads and writes to the drives just fine, even if the controller reports a power failure on every pass. I will have to look into this further also.

Anyway, after much testing, I found a suitable floppy controller card that works fine at 8 MHz, if we disregard the floppy drive power failure errors. I will leave this computer running at 8 MHz.

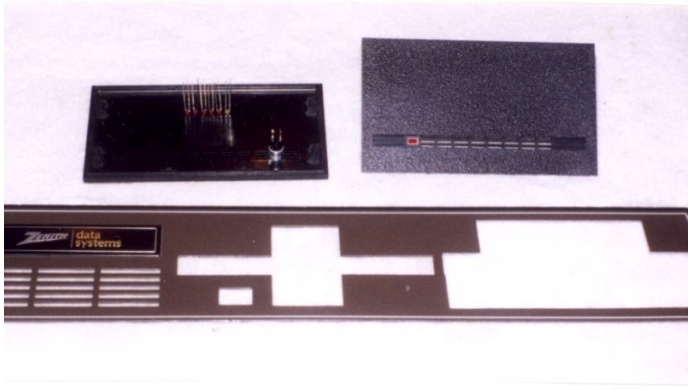
Now it is time to figure out how you want to configure your computer to accommodate one or more IDE drives. In my case, I felt one IDE drive would be sufficient. The one that worked with this particular prototype card was only 264Mb. But considering that the most popular MFM drive that I used was 40Mb, 264Mb would be much more than I would ever need. The newer IDE controllers will be able to handle four such drives.

However, one consideration that you must think about is drive back-up. On a Z-100, how does one intend to back up 264Mb of data? The obvious solutions of today's PC's - the CD or DVD - will not be available on a Z-100, at least not in the foreseeable future.

Tape - if you can still find such a unit? I didn't like that idea even when I just wanted to backup 20Mb of data.

No, I would think that backing up one IDE drive with a second unit would be the answer. So, you may want to figure on a minimum of two drives, if you are concerned about critical data. One could still be MFM, but why hang on to that dinosaur. At the low cost of used IDE drives, just use two of them.

On my unit, a proof of concept prototype, I did not need a second drive (and this particular card did not like two drives anyway). However, as we go through the installation, you will see that there is plenty of room for adding a second drive right over the first. With the floppy controller (upgraded) to control a single 5-1/4" and a single 3-1/2" drive and these mounted directly over one another, I have a large bay remaining to contain two and possibly even three IDE drives, if I so desired.




**Pix 2.**
**Z-100 Low-profile Faceplate**

Pix 2 shows the partially modified drive bay cover of the Z-100. Originally looking like the center drive area, the right bay has already been modified to accommodate the over/under configuration of the 5-1/4" and 3-1/2" drives. This aluminum faceplate was a real monster to cut out - hours of hacksawing and filing. The newer plastic faceplates are much easier to modify.

The upper left black rectangle shows the back of the modified hard drive faceplate that will cover the left (unmodified) drive bay. The upper right shows the front of an unmodified hard drive faceplate.

**Note:** To cover the old drive slot completely, the 5-1/4" drive must be positioned precisely 5/8" above the bottom of the bay. I used wood spacers 5/8" x 3/4" x 5" cut from common 3/4" stock on a table saw. The drive combination then completely eliminated the odd looking original bay cover. The middle bay will be completely covered with the faceplate of a full-height hard drive no longer being used.

These full-height faceplates should be easy enough to locate nearly anywhere that handles old drive equipment. If you have difficulty, I have plenty.

It is hard to see in this picture, but the left drive cover has already been modified and shows where the breakout switch and drive LEDs for the NVsRAM and up to four IDE drives are located. Pix 6 has a better, front view.

**Pix 3.**
**Rear of Modified Drive Bays**


**Pix 4.**
**Front of Drive Bay**

Pix 3 shows the rear of the drive bays. Pix 4 shows the front with the drive faceplate in place.

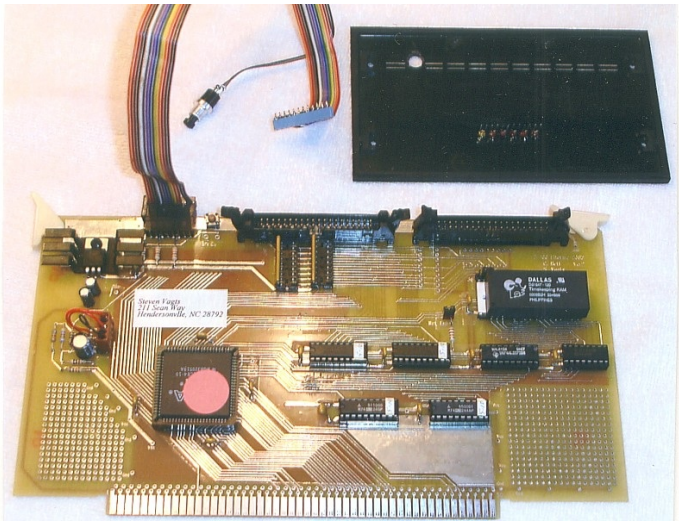
The 5-1/4" drive is mounted on 5/8" wooden spacers to fit in the cutout of Pix 2. The 3-1/2" drive is mounted over the 5-1/4" drive using the lower drive's flat upper shield.

The ribbon cable runs from the Z-207 Floppy Controller (not pictured) to the 5-1/4" drive and then the 3-1/2" drive.

**Note:** The 5-1/4" drive (configured as drive A:) does not need a terminator resistor in this arrangement. Most 3-1/2" drives are purchased configured as drive B: and cannot be configured any other way. They depend upon a twist in the PC's ribbon cable to become drive A:. Termination in a 3-1/2" drive is also integral to its circuitry and cannot be removed.

As every Z-100 configuration will probably be different and the LED drive lights are not necessary for operation, no attempt is being made to devise a cable to mount IDE drive lights anywhere on the Z-100's exterior. For trouble-shooting purposes, each board will be shipped with a bank of drive LED's mounted on the connector attached to the top of the IDE Controller Card (see Pix 1) and a breakout switch is already mounted on the board to the right of the IDE connectors.

However, for this computer, I devised one possible cabling solution you might consider.

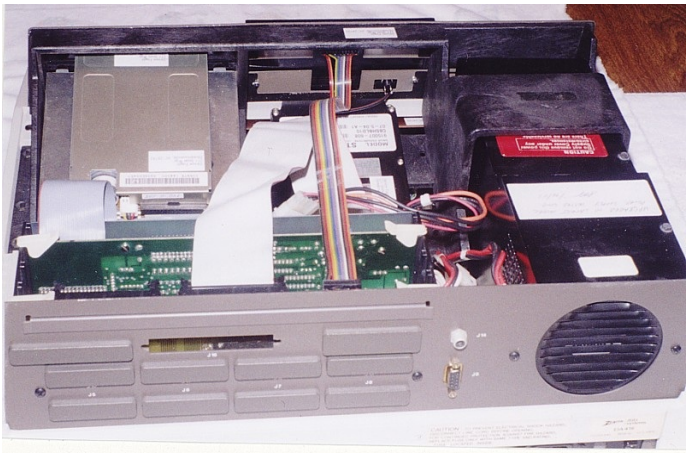


**Pix 5.**
**IDE Controller Board**

Pix 5 shows the IDE Controller Board with the drive LED ribbon cable connected at the LED connector at the top of the card. The other end of the ribbon cable ends in a SIP connector that slips onto the leads of the LED's installed in a row on the drive faceplate.

**Note:** Referring to Pix 6, the drive LED's are configured so that a yellow LED indicates data is being read from the NVsRAM. The rest of the LED's are red and indicate data being written to the NVsRAM, and read/write operations to IDE drives connected as J1 master, J1 slave, J2 master and J2 slave, in that order. The last two wires from the ribbon cable are connected to a separate, normally open, pushbutton, breakout switch also installed on the faceplate. The switches are available from Radio Shack, part #275-1547.


**Pix 6.**
**Exterior Closeup of**
**Modified Drive Faceplate**



**Pix 7.**
**Interior Connections**

Pix 7 shows the interior of the finished configuration.

With the hardware modifications completed, we turn our attention to the software issues.

## Programming the NVsRAM on the IDE Controller Board.

The heart of the boot up operation on the IDE Controller Board is the NVsRAM chip, also commonly referred to as the EEPROM to remain consistent with the terminology used previously to describe the EEPROM on the "*Z-100 LifeLine*" SCSI Controller Card.

NVsRAM is the acronym for NonVolatile Static Random Access Memory. This is memory that can be programmed and then an internal battery in the chip will hold this programming in memory until the chip is reprogrammed or erased.

Programming the chip is not difficult, but for frequent changes, it can be a pain, so I developed a couple of batch files to make programming somewhat easier. These, or something similar, will be distributed with the Controller Boards.

First off, memory in the NVsRAM is limited. Depending upon the model, it can be less than 64K, 128K, 256K, 512K, or more than 1Meg. With the growing needs of Z-DOS boot files for space, it was becoming increasingly difficult to fit all the desired files on a standard 360K, 5-1/4" floppy and the 1 meg NVsRAM with a clock was going to require additional circuitry on our board. So, due to space and circuit requirements, we settled on a 512K NVsRAM with an internal clock to best suit our needs.

Also, as it was desirable to minimize the space required by the boot files, I thought it best to use a handy compressing utility called PKLITE to compress the files that could be.

As it turns out, everything fit comfortably on the 512 NVsRAM, with plenty of room for additional files, but in case you find yourself pressed for space for your own requirements, let's pretend that we need to do it anyway.

Compacting or compressing IO.SYS and MSDOS.SYS will realize the following savings:

| File: | Normal: | Compressed: |
|---|---|---|
| IO.SYS | 32915 | 26344 bytes |
| MSDOS.SYS | 37376 | 27683 bytes |

Once compressed, the files will still execute, but cannot be changed. A full review of PKLITE can be found in issue #93 of the "*Z-100 LifeLine*".

PKLITE can also be used on most of the other files that you may want to place on the NVsRAM if you need even more space.

**Note**: Compressing IO.SYS does create one problem that may not make it worth all the effort. When creating bootable floppy disks using the FORMAT A:/s/v command, the compressed system files will be transferred to the newly created floppy disk. If you then run DRIVECFG on the new floppy disk, you will get the error:

*Version mismatch with file!!!!*

This is because during compression all the DRIVECFG data will have been moved from where it should be located.

The first batch file, PACK_IO.BAT, is used to compress IO.SYS. **PACK_IO.BAT** contains:

```
@ECHO off
ECHO This utility uses PKF00.COM to PKLITE
     IO.SYS and creates IO-PK.SYS.
ECHO.
ECHO NOTE: Before using this utility, run
     DRIVECFG and CONFIGUR to ensure
ECHO that IO.SYS is configured properly.
     The PKLITE version of IO.SYS
ECHO cannot be changed.
ECHO.
ECHO When completed, copy IO-PK.SYS to
     a bootable drive as IO.SYS.
ECHO.
ECHO Use {CTRL}-{C} to exit, or
PAUSE
COPY IO.SYS IO.COM/y >nul
PKLITE IO.COM >nul
PKF00 IO.COM
if exist IO-PK.SYS DEL IO-PK.SYS
REN IO.COM IO-PK.SYS
```

**Description**: ECHO thru PAUSE is just a helpful reminder about what the batch file does. As PKLITE cannot operate on a system (.SYS) file, the name is changed first. PKLITE is then used to compress IO.COM and PKF00 is used to adjust the start of the new program at offset 100h instead of offset 00h. Finally, if IO-PK.SYS exists, it is deleted and the compressed IO.COM is renamed to IO-PK.SYS.

Similarly, a second batch file, PACK_DOS.BAT, is used to compress MSDOS.SYS. **PACK_DOS.BAT** contains:

```
@ECHO off
ECHO This utility uses PKF00.COM to PKLITE
     MSDOS.SYS and creates DOS-PK.SYS.
ECHO.
ECHO NOTE: The PKLITE version of DOS.SYS
     cannot be changed.
ECHO.
ECHO When completed, copy DOS-PK.SYS to
     a bootable drive as MSDOS.SYS.
ECHO.
ECHO Use {CTRL}-{C} to exit, or
PAUSE
COPY MSDOS.SYS DOS.COM/y >nul
PKLITE DOS.COM >nul
PKF00 DOS.COM
if exist DOS-PK.SYS DEL DOS-PK.SYS
REN DOS.COM DOS-PK.SYS
```

**Description**: Again, ECHO thru PAUSE is just a helpful reminder about what the batch file does. As before, the name is changed first. PKLITE is then used to compress DOS.COM and PKF00 is used to adjust the start of the new program to offset 100h. Finally, DOS-PK.SYS is deleted and the compressed DOS.COM is renamed to DOS-PK.SYS.

The last batch file, **EPROMPGM.BAT**, is used to actually program the NVsRAM on the IDE controller card. EPROMPGM.BAT copies all the files that you wish to transfer to the NVsRAM and, for my purposes, contains:

```
ECHO off
ECHO This file assumes that IO.SYS and
     MSDOS.SYS have been packed to
ECHO IO-PKSYS and DOS-PK.SYS with the
     PACK_IO.BAT and PACK_DOS.BAT and
ECHO are located in the \EPROM directory
     with this batch file.
ECHO Place COMMAND.COM in the \EPROM Dir.
ECHO Place all other files to be copied
     to the IDE EPROM in \EPROMPGM.
ECHO All files presently on the EPROM will
     be deleted!
ECHO Type{CTRL}-{C} to exit or
PAUSE
ECHO Deleting all files on the EPROM...
DEL K:*.*
ECHO Copying COMMAND.COM...
COPY COMMAND.COM K:/v
ECHO Copying IO-PK.SYS to K:IO.SYS
COPY I-PK.SYS K:IO.SYS/v
ECHO Copying DOS-PK.SYS to K:MSDOS.SYS
COPY DOS-PK.SYS K:MSDOS.SYS/v
ECHO Copying all files from L:\EPROMPGM to K:
COPY L:\EPROMPGM\*.* K:/v
ECHO Calculating Checksum and saving to
     NVsRAM.
CHKSUMEP
ECHO.
ECHO Running EPRDFILE to save programming in
     backup file, EPTEST.DAT.
EPRDFILE
ECHO  EPROM programming completed.
ECHO  Remember to run CHKSUMEP after making
      any other changes to the EPROM.
```

**Description**: As before, ECHO thru PAUSE is just a helpful reminder about what the batch file does. The COPY commands copy the system files to the NVsRAM, followed by all the files from an \EPROMPGM directory on the IDE drive.

**Note**: The usable size of the NVsRAM is presently limited to 507K. So ensure the files in the \EPROMPGM directory do not exceed this amount, less the size of the system files.

CHKSUMEP is automatically run to calculate the checksum and place that on the NVsRAM.

EPRDFILE is run to save a data file EPTEST.DAT on the default drive as a data backup (more on this later).

Once completed, additional files can be copied to the NVsRAM manually, but remember to run CHKSUMEP again and last to validate the programming.

I also recommend running EPRDFILE again to save the programming in a backup data file. Upon rebooting, the NVsRAM is ready for use.

With the batch files in place, we are nearly ready.

We are used to booting to a hard drive or a floppy drive that has the DOS commands in a directory on the same drive.

Now, with the NVsRAM on the IDE Controller controlling the boot process, we either place all the DOS commands in the NVsRAM or have some means to have any request for a DOS command refer to the right drive. In the past, this had always been controlled by use of the PATH command in the AUTOEXEC.BAT file.

For example, the AUTOEXEC.BAT file in my computers generally looked like:

```
ECHO Off
ASGNPART 0:Z-DOS4 E:
PATH=E:\;E:\DOS
SET ZDIR=/F
PROMPT $P$G
```

With this simple AUTOEXEC.BAT file executed during boot up, any DOS command on the command line would be directed to look in the root directory of the MFM hard drive unit 0, in the partition labelled Z-DOS4, with the drive letter E: assigned to it and if not found there, it would look next in the \DOS directory.

As you may recall, DRIVECFG assigns drive letters to the various floppy and hard drives on the computer using Z-DOS v4.

On my computers, for standardization among the many different computer configurations that I have around the house, I generally assign the first four drive letters, A thru D to the floppy drives; E: thru H: to the MFM hard drive partitions; I reserve drive letter I to be mapped as an imaginary drive by assigning it to map drive A:; and finally, DRIVECFG automatically assigns the next available drive letter to the EEPROM on either the LLSCSI board or the IDE board, depending on if either is detected.

The next available drive letter is then assigned to any drive and their partitions found on either board.

This has worked out well for me in the past.

The IDE drive, however, on my computer is assigned the drive letter K: by DRIVECFG.COM. And the NVsRAM (referred to as EEPROM) on the IDE card, is assigned drive letter J:

Using the AUTOEXEC.BAT configuration above, however, now generates an error if the MFM hard drive is not attached.

*Physical Sector Zero is not readable,*
*unit does not exist.*

So, we need to configure the AUTOEXEC.BAT file on the NVsRAM to refer to the proper drive for executing DOS commands. If there will not be any MFM hard drive system on the computer, change the AUTOEXEC.BAT file to:

```
ECHO Off
PATH=J:\;K:\;K:\DOS
SET ZDIR=/F
PROMPT $P$G
```

If you intend to leave an MFM hard drive in the system, but will be booting to the NVsRAM as the primary drive, you may wish to use something as complicated as:

```
ECHO Off
ASGNPART 0:Z-DOS4 E:
PATH=J:;E:\;E:\DOS;K:\;K:\DOS
SET ZDIR=/F
PROMPT $P$G
```

Obviously, the object is to let the computer know where to find the files you wish to run.

**Note**: The IDE driver will automatically recognize two partitions on each of up to four IDE drives and assign them drive letters. Any additional partitions on drives will require the use of an IASNPART utility that will operate in a manner similar to ASGNPART.

**Note**: All the above drive configurations have assumed that four partitions have been set aside for MFM hard drive use. If you do not intend to ever worry about an MFM hard drive again, just delete those drive letters using DRIVECFG. The IDE driver will adjust the IDE drive letters accordingly.

We have one last complication to address, and I saved it for last.

Back up in the programming batch file, you may have noticed that the drive letters were changed from the ones that we had assigned automatically. The IDE driver is configured to add a different drive letter to the NVsRAM when it is being programmed, and this causes the IDE drives to bump down to the next available drive letter after this new drive is listed.

For example, the normal IO.SYS boot screen will list the drives as configured by DRIVECFG and others found as:

```
A: LOW Den.  48tpi 5 1/4"  34 Z207p 0
B: DUL Den. 135tpi 3 1/2"  34 Z207p 1
C: LOW Den.  48tpi 5 1/4"  34 Z207p 2
D: LOW Den.  48tpi 5 1/4"  34 Z207p 3
E: Fixed Disk Partition       Z217p
F: Fixed Disk Partition       Z217p
G: Fixed Disk Partition       Z217p
H: Fixed Disk Partition       Z217p
I: Imaginary Drive mapped to A:
J: EEPROM on IDE  LifeLine Board  s
K: ST3290      FAT16    261MB LIDEp 0
```

As one of several protection methods against accidently programming, or inadvertently changing the NVsRAM programming, we felt it best to only enable programming the NVsRAM by loading a separate driver, EPROMDSK.SYS during the boot process.

Therefore, CONFIG.SYS, which we will describe in great detail in a moment, has a separate section that is enabled to load this driver only when we specifically ask for it.

So, when we want to program the NVsRAM, the NVsRAM is actually enabled for programming and the IO.SYS boot screen becomes:

```
    A: LOW Den.  48tpi 5 1/4"  34 Z207p 0
    B: DUL Den. 135tpi 3 1/2"  34 Z207p 1
    C: LOW Den.  48tpi 5 1/4"  34 Z207p 2
    D: LOW Den.  48tpi 5 1/4"  34 Z207p 3
    E: Fixed Disk Partition       Z217p
    F: Fixed Disk Partition       Z217p
    G: Fixed Disk Partition       Z217p
    H: Fixed Disk Partition       Z217p
    I: Imaginary Drive mapped to A:
    J: EEPROM on IDE  LifeLine Board  s
    K:  512K EPROMDSK/CLOCK (programming)
    L: ST3290      FAT16    261MB LIDEp 0
```

CONFIG.SYS is controlling all of this and for those of you who are still using the old three or four line CONFIG.SYS, you will have to get used to the new expanded version.

The new CONFIG.SYS has its own article on the “Z100 LifeLine” website. It explains CONFIG.SYS in much greater detail, with examples.

While CONFIG.SYS may appear very complex, it really is not so bad once you are familiar with the outline structure.

The following is a much abbreviated version with all the comment lines still included to help you along. The version you use for your computer can be edited down much further. I recommend not eliminating all the comments, however, to ease in making changes at a later date.

**CONFIG.SYS contains:**

```
Rem   SECTION 1 - Common Commands
; lines always included
Comment=   ;
; Permit comments on line after command
Break=On      ; Check for CTRL-C
Buffers=32,8  ; 32 buffers, 8 sector R/W
Files=30      ; 30 open files allowed
LastDrive=Z   ; Permit 26 drives
;     SECTION 2 - Custom Commands
; Options defined by a letter:
:0   ; Default Selection - No key pressed
#0   ; Include Common Block LLIDEHD
:P   ; NVsRAM Programming
device=EPROMDSK.SYS /256/W0
#0   ; Also include LLIDEHD after EPROMDSK
:R   ; RAM Disk Installed
device=Z205DSK.SYS
:Z   ; Last option ID - don't load anything
; Last option id - don't do anything
::   ; Section 3 - Condensed Commands
; This is where common # blocks are defined:
#0   ; Common block 0
device=LLIDEHD.EXE /F   ; IDE device driver
Install=SHARE.EXE  ; Drive Partition >32Mb
##   ; Section 4 - Secondary Custom Commands
; These are lines common to all options.
^Z   (Crtl-Z marks end of file)
```

**Section Descriptions:**

Section 1 lists the commands that are common to all options. Many of these commands could have been moved to the individual options to accommodate special requirements of applications.

For example: a particular application may require more buffers than usual; so you would list Buffers under each option in Section 2 and specify the number for each application.

Section 2 is where special drivers are loaded for their particular application. If a particular driver is needed under several options, it could be placed in Section 3 as a common block, defined as a pound sign(#) and a number.

For example; the LLIDEHD driver will be required in most options. Rather than listing it individually under each, #0 is a marker placed to define those commands in Common Block #0.

Section 3 lists and defines all the Common Blocks. These blocks, defined with a pound (#) sign, may be used in multiple individual options given in Section 2.

Section 4 is provided for any command that has to be executed after all the drivers are loaded. Finally, finish editing the file with a CTRL-Z as an End-of-File marker, if needed.

Note the INSTALL=SHARE.EXE line in the CONFIG .SYS file. SHARE is necessary and should be installed any time you are operating a system with a drive containing one or more partitions greater than 32Mb.

The deal is that old "File Control Blocks" or "FCB's" cannot hold pointer information in its "Reserved Fields", on files located on disk locations past 32Mb's. FCB's will only work correctly as long as a file is physically located within the first 32Mb's of a partition's start. If part of the file lies past this 32Mb range, the FCB does not complain or generate an error, it just rolls the pointer value over, through zero, and gives DOS a new garbage value as an internal disk pointer. The next disk read gives junk to your program, and the next write corrupts your disk!

Without going into detail, the reason SHARE is the solution is because it was already doing the required fix for a different reason in small partitions. For a more detailed explanation, see the SHARE help screen in the Z-DOS v4 help directory.

Now we have all the parts in place. The only other change to worry about is the Boot command.

**ROM v4.3 Boot Up Sequence**

Unless you have been using Autoboot all your life, you are already familiar with most of the Boot options. Remember, Autoboot can be disabled by setting section 3 of S101 on the motherboard to ON or 0 (toward the rear of the computer).

You can also get to the hand prompt by pressing the {DELETE} key at which time the computer will respond with **"Boot Abort"** and the hand prompt.

**Note:** The version 4.3 Monitor ROM now displays the Help screen before the hand prompt. Otherwise, it will display an error message and the hand prompt as in earlier ROMs.

Unless you have played with the LifeLine SCSI board, you are probably not familiar with all the changes to the Boot options since version 3.0 of the ROM. So, let me briefly cover these Boot options.

The **Boot** command will boot the Disk Operating System from a diskette, MFM hard drive, a SCSI drive from a Z-317 controller card, or a "*Z-100 LifeLine*" SCSI or IDE Controller Card.

The Boot Syntax is:

**Boot [F1-4][Unit#][S][:partname]**

Where:

**F1** specifies a drive attached to the Z-207 34-pin connector and may be either a 5-1/4" or 3-1/2" floppy drive.

**F2** specifies a drive attached to the Z-207 50-pin connector and may be either a 3-1/2" or 8" floppy drive.

**F3** specifies a hard drive; either an MFM drive attached to a Z-217 or Z-317 controller or a SCSI drive attached to a Z-317 controller.

**F4** specifies a bootable EEPROM on the LLSCSI controller board or, when used with the [S], an NVsRAM on the LLIDE controller board.

**Unit#** specifies the drive unit number as set on the drive's DS (drive select) jumpers. This may be 0, 1, 2, or 3.

**S** specifies a secondary floppy or hard drive controller; it also specifies the NVsRAM on the LLIDE Controller Card

**:partname** specifies the partition name on a hard drive to boot from.

To manually boot the computer, press the **{B}** key. With the version 4.3 Monitor ROM, the computer will now display the default boot device as set by the S101 switches on the motherboard.

On my computer, the display is:

*Default Booting Primary Z207 34pin Unit 0*
*Input BOOT string<CR>_*

and the computer will wait for you to type on the keyboard to give it more information.

If you press the **{RETURN}** key, the computer will begin to boot the operating system from the default device, drive zero, as determined by the setting of switch S101 on the motherboard.

This device can be a 34-pin floppy drive unit zero, a 50-pin floppy drive unit zero, the first MFM hard drive, or the bootable EEPROM on the LLSCSI board.

If you press the **{F1}**, **{F2}**, **{F3}**, or **{F4}** key followed by the **{RETURN}** key, the computer will boot from unit zero of a specific device without regard to the settings of S101 on the motherboard.

The monitor ROM can support up to four drives of each type. Therefore, you can boot from any drive by typing its unit number, 0, 1, 2, or 3 after the **{F1}**, **{F2}**, or **{F3}**. If the device is not present or is faulty, after about 30 seconds a "Device Error" message will appear on the screen.

You can boot the computer from any partition on the Z-317 SCSI drive or MFM hard drive if there is an operating system on it. To do this, type:

**{B}{F3}{:partname}{RETURN}**

The screen will display:

*Default Booting Primary Z207 34pin Unit 0*
*Input BOOT string<CR>f3:(partition name)*

The optional **{S}** key is used to boot from the Secondary device (the second Z-207 floppy disk controller card, second hard drive controller card, or from the LLIDE controller card (the LLSCSI controller card is considered the primary card). If the device is not present, or is faulty, after about 30 seconds a "Device Error" message will be displayed.

To Boot from the LLIDE controller card, you must type:

**{B}{F4}{S}**

The computer will display:

*Default Booting Primary Z207 34pin Unit 0*
*Input BOOT string<CR>f4S*

## IO.SYS Boot Screen

The Boot sequence begins by displaying the IO.SYS Boot Screen. This screen looks very similar to the screen displayed during the DRIVECFG configuration process. It lists the MS-DOS Version Number and BIOS Version Number and then all the drives configured by DRIVECFG in a display window (an example is shown earlier). In the bottom of the window, more system information is displayed:

*MTR ROM V4.x, xxxK RAM, xxK COLOR Video, CPU, and x.xxxx MHz CPU Speed.*

Below the window, messages appear indicating the status of the normal boot process. These messages are:

*** Initializing Motherboard Parity ***

This message only appears on a power-up (cold) boot. The parity is not checked on a warm boot (when you press the breakout switch or {CTRL}-{RESET} to return to the hand prompt).

*Hit any key within 3 seconds for alternate CONFIGURATION*

It is at this point where you have three seconds (this time can be adjusted in DRIVECFG) to decide if you want to use the default CONFIG .SYS, some option within it, or a different-named CONFIG.SYS. From the CONFIG.SYS example above, we have a couple of configuration options.

Press any key to interrupt the boot process. The message will then change to:

*Select CONFIG.SYS option (A-Z) ->_*

Then we can choose **{O}** (the default configuration), **{P}** for Programming the NVsRAM, **{R}** for configuring a RAM disk (if a RAM card is installed), or **{Z}** for loading nothing or doing anything (good for troubleshooting purposes).

Of course, if you have other configuration options in your CONFIG.SYS file, here is where to exercise those options.

For this run, we let it time out and the computer will use the default CONFIG.SYS, displaying the message:

*Using default CONFIG.SYS optn.*

After CONFIG.SYS is executed, the AUTOEXEC.BAT file is executed and the computer will display the NVsRAM drive letter:

*J:\>ECHO OFF*
*J:\>_*

## Programming the NVsRAM

As the NVsRAM is read only once it is programmed, we must discuss the procedures for programming the NVsRAM.

When we interrupt the boot process, by pressing any key at the CONFIG.SYS question, the computer will replace the message with:

*Select CONFIG.SYS option (A-Z) ->_*

Then we can press a key to choose a configuration option. To choose to enable programming the NVsRAM, we need to press **{P}**.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**
**Caution:**

When the NVsRAM is prepared for programming, the write protection is removed from the NVsRAM. The checksum thus becomes changed and, unless you run CHKSUMEP before rebooting or shutting down the computer, the NVsRAM will be reported as **corrupt** on the next bootup! You will have to reboot to another device and run CHKSUMEP before booting again to the NVsRAM.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

The computer will load the EPROMDSK.SYS driver and update the drive list in the IO.SYS boot screen to reflect new K: and L: drives:

*K:  512K EPROMDSK/CLOCK (programming)*
*L: ST3290  FAT16  261MB LIDEp 0*

and below the window, the boot session is completed when we see the DOS prompts:

*J:\>ECHO OFF*
*J:\>_*

To program the NVsRAM from the IDE drive L:, we need to change to the \EPROM directory on the L: drive and then run our EPROMPGM.BAT file that we created earlier. The series of commands are:

**J:\>L:**
**L:\>CD EPROM{RETURN}**
**L:\EPROM>EPROMPGM{RETURN}**

To program the NVsRAM from the 3-1/2" floppy drive (preparing this backup disk is explained shortly), we do not need an EPROM directory, instead we have our EPROMPGM.BAT file in the root directory and additional files in the \EPROMPGM directory. Booting to the floppy, using the same procedures given above, the command sequence is simply:

**B:\>EPROMPGM{RETURN}**

In either case, the batch file is run and a new CHECKSUM is generated and also placed on the NVsRAM.

**Note:** The EPROMPGM.BAT does not need to be run for minor changes. If you just wanted to add a file, the NVsRAM is treated as any other drive letter. Copy the new file to the K: drive and, when all the changes are completed, run CHKSUMEP to generate a new Checksum.

**Alternate NVsRAM Programming**

Once the NVsRAM has been programmed using the above procedures, there is a set of 3 programs that are of tremendous assistance in maintaining the status of the NVsRAM chip.

**EPRDFILE.COM** - Reads all the data stored in the NVsRAM and stores it in the same directory as EPTEST.DAT.

When finished running, it reports:

*File Successfully Written from EPROM.*

and EPTEST.DAT is written to the current directory. The file is 525Kb so the EPTEST.DAT command cannot be run from the NVsRAM or 5-1/4" floppy and will require a healthy chunk of a 3-1/2" floppy!

For that reason, the set of three files are placed in the \EPROM directory on the IDE drive.

**EPCPFILE.COM** - This command compares the previously saved EPTEST.DAT file with the NVsRAM again to ensure they are the same.

When complete, it will hopefully report:

*End of Compare EPROM and File.*

If there are any differences, each is reported as a separate line. For example:

*Found in File 00h in EPROM A0h for sector 0 at Offset 645*

If the files are completely different, these reports will scroll seemingly forever on the screen. Press **{CTRL}-{C}** to exit.

**EPWRFILE.COM** - This command makes recovery for a corrupt NVsRAM quick and easy. It will copy the data file, EPTEST.DAT, which was automatically generated by the EPROMPGM.BAT batch file, back into the NVsRAM in the event the NVsRAM becomes corrupted for some reason other than internal failure. The NVsRAM does **NOT** require to be in programming mode.

**Note:** There is a write enable jumper located beside the NVsRAM on the IDE Controller Card. The pins are not installed and it is presently enabled by a trace between the two pin holes. If you wish to disable ALL writes to the NVsRAM, including EPWRFILE, cut the trace and install pins to enable writing when necessary.

**Creating a Backup Floppy**

In spite of all the precautions that we have built into the IDE Controller Card and NVsRAM programming, the IDE Card and NVsRAM are critical to the booting process of the Z-100. If for some reason, the NVsRAM is erased, or you have to change it out, it is best to have a bootable backup disk to be able to reprogram the NVsRAM.

Therefore, once the NVsRAM has been programmed and is working to your satisfaction, make a bootable 3-1/2" disk using **FORMAT B:/s/v**, which will install IO.SYS, MSDOS.SYS, and COMMAND.COM on the floppy to make it bootable.

**Caution:** Running **FORMAT /s/v** from an NVsRAM using compressed system files will copy these compressed system files to the floppy drive. These will generate an error if DRIVECFG is subsequently run on the floppy disk.

**Note:** While a 360K 5-1/4" floppy disk will have enough space for the critical files and could be used, it would not be able to contain all the files to fill the 507K free space on the NVsRAM.

Next, create the \EPROMPGM directory and copy all the files to it from the IDE drive's \EPROMPGM directory. Finally, copy the following files to the root directory on the floppy drive:

| | | |
|---|---|---|
| COMMAND.COM | IO.SYS | MSDOS.SYS |
| AUTOEXEC.BAT | EPROMPGM.BAT | CONFIG.SYS |
| EPROMDSK.SYS | LLIDEHD.EXE | CHKSUMEP.COM |
| EPRDFILE.COM | EPCPFILE.COM | EPWRFILE.COM |
| TESTIDEP.COM | | |

Modify PATH in AUTOEXEC.BAT and modify EPROMPGM .BAT to load the files from the floppy drive.

Next, copy any additional files as you may desire into the Root directory on the disk. This is an emergency disk and you might need to do any number of actions while trouble-shooting your situation. So, such files might include, depending upon your computer configuration:

| | | |
|---|---|---|
| ASGNPART.COM | DETECT.COM | DSKCOPY4.COM |
| EDLIN.COM | FLAGS.COM | FORMAT.COM |
| LOOK.COM | PART.COM | PREP.COM |
| SYS.COM | ZDIR.COM | ZFMT207.COM |

**CAUTION:** Remember to leave at least 525Kb of space for the EPTEST.DAT data file generated by EPRDFILE during the NVsRAM programming.

Finally, reboot to the floppy and reprogram the NVsRAM to ensure the floppy disk works as intended. Label the disk as follows:

"Emergency IDE NVsRAM Programming Disk"
"Run EPROMPGM.BAT"
"Z-DOS v4.06 (Bootable)"

and store it in a safe location.

## Setting the Clock

Setting the Clock could not be easier.

**Note:** Since the original publishing of this article, we have updated the setting of the NVsRAM clock from the IDECLOCK utility to a set of four new ICLKxxx utilities.

**ICLKSET.COM**

To set the NVsRAM Clock, run DOS' DATE and TIME commands as normal. Next, run **ICLKSET** without anything else on the command line. ICLKSET will then auto Calibrate the clock and set the Date and Time from the DOS setting. If successful, the computer will respond with:

*IDE EPROM Clock successfully set from*
*MS-DOS date and time.*
*Present date/time is (press any key to EXIT):*
Saturday March 26, 2022 13:53:36

If anything else is placed on the command line, ICLKSET will display a help message and the present programmed date/time.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

*ICLKSET Version 4.05*

*This utility will set the Dallas DS1647 or the TI bq4850Y clock on the LifeLine IDE card in accordance with the parameters given.*

*There are 4 valid command line parameters:*
*ICLKSET /? Will display this help screen.*
*ICLKSET{RETURN} will set the NVsRAM clock date/time from MS-DOS.*
*ICLKSET{SPACE} will display the current date/time.*
*ICLKSET off will turn OFF the clock in the NVsRAM. This is useful if the clock will not be used for a long period of time (years) and sets the OSC bit in the seconds register to 1.*
*Anything else on the command line, will display this message, then the programmed date/time from the clock.*

*Present date/time is (press any key to EXIT):*
*Saturday March 26, 2022 13:56:38*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Pressing any key will exit the routine.

**Notes:**

* This utility is meant for use with the Dallas DS1647 or Texas Instruments BQ4850Y NVsRAM clock on the "*Z-100 LifeLine*" IDE Controller Card and sets the Date/Time from the current MS-DOS Date/Time, if no command line parameter is given.

* If the command line parameter is 'off' or 'OFF', the NVsRAM oscillator bit is turned off to conserve battery power, in the event the NVsRAM will not be used for a very long period of time (years), by setting the OSC bit of the seconds register to 1.

* Of particular interest is the Read and Write bits of the Clock Control Register. The Read bit, when set to one (40h), prevents updating the registers from the internal clock, so updates do not disturb the reading. The Write bit, when set to one (80h), prevents setting the internal clock from the registers, until the registers are set to do so.

* The NVsRAM clocks are very accurate, averaging +/- 1 minute per month.

* The TI BQ4850 NVsRAM clock uses a calibration procedure to fine-tune the accuracy of their clock. The ICLKCAL.COM utility can be used to adjust this calibration, if you feel it is needed.

**CAUTION:** Do NOT use ICLKCAL on Dallas clocks. They do not use Calibration and this utility stalls while waiting for the clock to respond.

**ICLKTST.COM**

To check the clock date and time, run **ICLKTST.**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

*ICLKTST Version 4.05*

*This utility will display to the screen the current calibration setting and the date/time from the Dallas DS1647 or Texas Instruments bq4850Y clock on the LifeLine IDE Controller Card.*
*As the Dallas clock does NOT use calibration, It will display 00.*
*Anything on the command line will display this message.*

*Please press any key to exit.*

*Current Calibration Setting = + 04*

*Saturday March 26. 2022 14:08:22*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Note:** This utility will display to the screen the programmed time from the Dallas DS1647 or TI bq4850Y clock on ONLY the LifeLine IDE Controller Card.

**ICLKCAL.COM**

To calibrate the clock, run **ICLKCAL.**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

*ICLKCAL Version 4.05*

*This program reports the current calibration on a Texas Instruments bq4850 NVsRAM clock and the current deviation in timer tics, then resets the clock date/time from DOS.*

*CAUTION: Do NOT use on Dallas clocks. They do not use calibration and this utility stalls while waiting for the clock to respond.*

*There are 250,000 timer tics per second; 2.6 million seconds/month. The reported deviation is measured over a two second period.*
*The file source code has more information on the calculations.*

*The Current Calculation setting was + 04.*
*Resetting calculation to zero and testing...*
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
   Total timer tics deviation = + 07

     Each calibration value is equivalent to 2
     positive and 1 negative time tic in deviation,
     So the new calibration setting is + 04.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The computer will respond with the help message and then provide the Current Calibration Setting being used. On the last line, the Day of the Week, Date, and Time are displayed, with the time incrementing. For example:

```
    Current Calibration Setting = + 04.
    Current Wait States = 0.
    Please hit ANY KEY to exit.

    Wednesday, February 14, 2007  08:53:16
```

Pressing any key will exit the routine.

**Note:** This program reports the current calibration on a Texas Instruments BQ4850 NVsRAM clock and the current deviation in timer tics, then resets the clock date/time from DOS.

**CAUTION:** Do NOT use on Dallas clocks. They do not use Calibration and this utility stalls while waiting for the clock to respond.

**ICLKREG.COM**

To view the clock registers, run **ICLKREG**.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   ICLKREG Version 4.05

   This utility will display the top 16 bytes of the
   NVsRAM, the area in NVsRAM memory containing the
   clock registers.
   If the clock is running, the right most 8 bytes
   contain the clock data registers, with byte 9
   incrementing each second.
   The clock data is: 8/  9/  A/  B/  C/  D/  E/  F/
                      Ctl/ ss/ mm/ hr/ wd/ dy/ mo/ yr/

   NOTE: To update the internal clock, first note
   the value of register 8. Next, set register 8 to
   80h to disable writes to the internal clock.
   Change the clock data registers as desired, then
   reset register 8 back to the original value noted
   earlier to write the registers to the clock.
   Writing a number greater than 3F to the control
   register, byte 8, may disable reading or writing
   to the clock!!!

   Input Location to Write (0-F); press {RETURN} to
   exit.

    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
   00 00 00 00 00 00 00 00 24 58 17 14 07 26 03 22

   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

**Notes:**

* This utility will display the top 16 bytes of the NVsRAM, the area in memory containing the clock registers. If the clock is running, the rightmost 8 bytes contain the clock data, with byte 9 incrementing each second.

* The clock data is:
  8/ 9/ A/ B/ C/ D/ E/ F
  ctl/ss/mm/hr/wd/dy/mo/yr

Where:

- Location 8 is the control register and should not be changed before exiting.
- Bit 7 (80-FF hex) is the write bit - set to 1, it halts clock updates from the registers.
- Bit 6 (40-7F hex) is the read bit - set to 1, it prevents clock updates to the registers.
- On the Texas Instruments' NVsRAMs, bits 0 thru 4 contain the calibration settings. On the Dallas NVsRAMs these are not used.

**NVsRAM Tests**

To check the NVsRAM memory locations, run **NVsRAMck**.

**Notes:**

* Running the NVsRAMck utility will test all RAM locations in the NVsRAM device by writing a value of the user's choice to all locations in the NVsRAM. It then goes back and looks for differences.

* If the clock is running, the clock register locations are not changed. However, if the clock is NOT running, all locations are overwritten, including the clock.

**WARNING:** Writing any value greater than 40h, will make the clock unreadable and it will have to be turned ON and set by ICLKSET!

**Monitor ROM NVsRAM Test**

ZROM v4.3 is able to test each boot device by using the **{F8}** function key. This comes in handy if you suspect a problem with the NVsRAM on the LLIDE controller card.

It will perform a checksum and will check it against the one stored in the NVsRAM when it was last programmed. If it is different, it will report an EEPROM Error. If correct, the test will repeat until you press the {DELETE} key. To initiate this test, type:

**{F8}{F4}{S}{RETURN}**

Upon pressing the **{F8}** key, the computer screen will blank and then display the following:

```
    Default Booting Primary Z207 34pin Unit 0
    Input BOOT string<CR>_

                TEST COUNT =

    TYPE <DELETE KEY> TO ABORT
```

As you press the **{F4}{S}** keys, they are inserted at the cursor. When you press the **{RETURN}** key, a third line is added to the screen:

*Booting Secondary EEPROM Unit 0*

and the COUNT begins to increment or an error message is displayed.

**Note:** As this test is only performing a CRC check, it will only report the **"EEPROM is Corrupt"** message. See the Error Messages listed below. Reboot in NVsRAM programming mode using an alternate boot device and run CHKSUMEP.COM, EPWRFILE.COM, or EPROMPGM.BAT to recover.

**TESTIDEP.COM**

This utility can be found on the Z-DOS v4.06 CD-ROM in the directory:

X:Z-DOSv4.06 > LLIDE > NVsRAM > TestPgms

TESTIDEP can be run at any time from the DOS prompt. Like the test above, it simply calculates a Checksum and compares it with the stored value on the NVsRAM. If it was successful, it reports:

*IDE EEPROM IS OK*

If not, it reports:

*IDE EEPROM IS CORRUPT*

Run **EPWRFILE.COM** as discussed above to recover, or reboot in NVsRAM programming mode and run **CHKSUMEP.COM** or **EPROMPGM.BAT.**

**Known Software Issues**

As with any new hardware or software changes, some incompatibilities have to be expected and this implementation did not disappoint. The first problem has already been reported.

It seems that Peachtext 5000 does not recognize the new IDE drive partitions that are detected and assigned drive letters during the boot up. It does recognize the NVsRAM just fine.

Further investigation has shown that it is a problem with the Peachtext software not being able to recognize partitions greater in size than 32 Mb! Peachtext, and probably other software of the period, never expected dealing with memory storage greater than 32 Mbs so their addressing capability is very limited.

The fix is simple enough. While Peachtext and other applications of the period may be run from the larger partitions, those partitions that you wish to use as data disks must be kept smaller than 32 Mb.

If you find other program applications that have a similar problem, or if you find another fix, please contact me at:
z100lifeline@swvagts.com.

## LLIDE Controller Error Messages

Possible errors that may appear during the Boot test or during the Boot process include:

| Errors: | Reasons: |
|---|---|
| *EEPROM is Corrupt Device Error* | Checksums do not match. |
| *NO IO.SYS* | IO.SYS missing or corrupt. |
| *Could not find DOS, disk not a system disk* | MSDOS.SYS missing or Corrupt. |
| *Bad or missing Command Interpreter* | COMMAND.COM missing or corrupt. |
| *ERROR: No LifeLine IDE Drives Detected!* | Drive not detected! Check cables & power. |
| ****Driver NOT installed due to prior ERROR**** | Drive incompatible with LLIDE Controller Card. Try another drive. Check Drive DS/CS jumpers. |
| Drive not listed on IO Boot Screen or Info Wrong. | Drive incompatible with LLIDE Controller Card. Try another drive. Check Drive DS/CS jumpers. |

For any error messages involving the NVsRAM, all errors except mechanical or internal failure can generally be corrected by running **CHKSUMEP.COM**, **EPWRFILE.COM**, or **EPROMPGM.BAT**. See the section on programming the NVsRAM above for the procedures.

## Advantages of an IDE controller

Besides the obvious about adding a high capacity hard drive to the Z-100 and the declining reliability and availability of MFM hard drives, some persons have older Z-100's that never had the power cables to upgrade to a hard drive system. This upgrade requires NO additional power connectors. It DOES require upgrade to Z-DOS v4.06 and MTR-ROM v4.3, available at nominal additional cost with the IDE Controller Board.

A major advantage of the IDE upgrade, is the greatly reduced power requirements. If you are replacing an MFM hard drive setup with an IDE drive setup, instead of powering a data separator card and a massive Z-217 controller card with their hefty power requirements, the new IDE controller requires a small 5 volt regulator that barely gets warm, and that's the warmest part on the board, even at 10 MHz!

If the power supply fan noise had been driving you nuts in the past, there may be a possibility of finally being able to replace the fan with a slower or slightly smaller unit, though it is still not recommended!

## Closing

I think I've shown that even the lowly dual-floppy Z-100 can be easily converted into a hard working, proud machine with a little labor of love.

I hope this exercise has helped you to make an informed decision as to whether you want to upgrade your Z-100 to this exciting new capability.

If you have any questions or comments, please email me at:

z100lifeline@swvagts.com

Cheers,

Steven W. Vagts