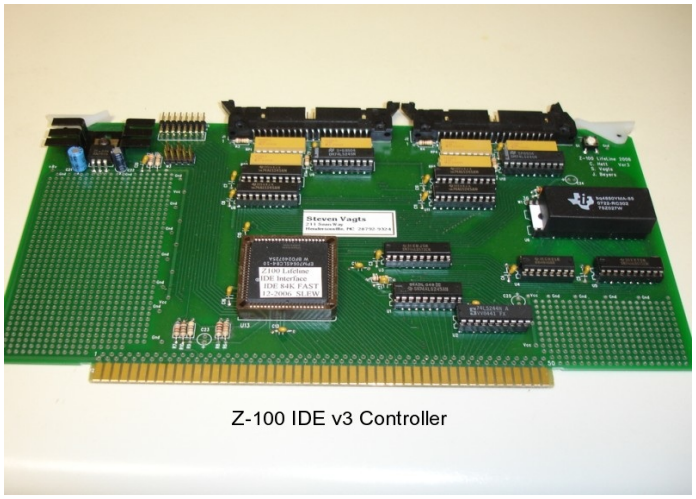
Z-100 LifeLine
Since 1989

March 2022

#WEB

~~~~~~



Z-100 IDE v3 Controller

Z-100 LifeLine
IDE Controller

by Steven W. Vagts
Editor, Z-100 LifeLine

Z-100 LifeLine
IDE Controller

The new Z-100 LifeLine IDE Controller Cards have been in use now for several years without a single complaint. The board appears to work well with nearly everything we can throw at it, including the new Compact Flash (CF) memory cards, but when I tried the Disk-on-Module (DOM) memory units, the one I was testing was a bit flaky - sometimes not recognized correctly. So I can not recommend those.

Unfortunately, I have sold my last spare card, so these are no longer available. I am sorry that you missed out.

This article is intended to document all features of this great board, in case someone comes across one on E-bay or from a friend and has no idea what they have obtained.

And do not miss the article on the First Operational IDE Controller in the Z-100, **IDE_FirstOperational.PDF**, also found on this Website.

IDE Controller Board FEATURES:

* Two IDE drive connectors which can each have a master and a slave
* Compact Flash memory cards or Disk On Module(DOM) plug-in memory devices can replace any or all of the IDE hard drives. These are prepared in the same manner as their hard drive cousins.
* A 512Kb non-volatile sRAM chip provides:
  - FAST boot code
  - Room for other programs for FAST access
  - A Real Time Clock recognized by Z-DOS
* Two NVsRAM LED's and four IDE Drive LED's to show drive activity
* Hardware Breakout Switch
* Gold-plated S-100 bus connector fingers

System Requirements:

The IDE Controller requires a Heath/Zenith H/Z-100 series computer with:

* A minimum of 192Kb motherboard RAM
* A Z-ROM v4.3 monitor ROM
* Z-DOS v4.06 Operating System

Some limitations are also imposed on the IDE hard drives:

* Drives can retain any PC Operating System, which could still use:
  - a 16-bit FAT (e.g., Windows 9x) or
  - 32-bit FAT (e.g., Windows 98SE).
* Each Z-100 accessible partition will be limited to 2 Gbs in size (imposed by the maximum 16-bit FAT that would be usable by the Z-100)
* Maximum drive size that could be used with the Z-100 will be 137 Gbs, as limited by the 28-bit LBA addressing standards before ATA/ATAPI-6.
* We have found that some software (e.g., Peachtext 5000) do NOT recognize partitions having sizes greater than 32 Mb. We recommend creating one or two 30 Mb partitions on each drive for use with this software.
~~~~~~

* The IDE drives will ONLY be recognized by Z-DOS v4.06. No previous versions of DOS, nor other operating systems, will recognize the IDE drives. We hope to modify other operating systems to use the IDE drives in the future.

Note: Newer drives, larger than 137 Gbs, are already on the market that meet the new 48-bit LBA addressing standard of ATA/ATAPI-6. These will NOT be supported by the Z-100. This is not considered a problem because at 2 Gb each, what is a person going to do with 67 partitions?

The Z-100 LifeLine Design Team:

Charles Hett is Hardware Guru
John Beyers is Software Guru
Steven Vagts is Z-100 LifeLine Editor and Project GoFer

Project Conception:

The LLIDEHD driver is key to the success of PC /Z-100 interchangeability, enabling us to retain the use of our IDE hard drive in a PC and as a means for transferring data between the PC world and our Z-100's. This can be accomplished by simply placing the information on the drive in a format recognized by both systems.

Further, since we have complete control over all aspects of doing this from a Z-100's perspective, it is possible to leave a primary partition on the IDE drive that would contain the PC's Operating System (Windows) that would be completely undisturbed by operation in the Z-100.

In short, this means that we could use a properly prepared system drive from a PC, place it in the Z-100 to work on or transfer data files, then return with the information for processing on the PC. In broad terms, this involves assigning one or more partitions on the IDE drive for use on the Z-100.

As conceived, there would be three ways of preparing an IDE hard drive for use in the Z-100. The first, easiest, and preferred method is to use the PC in which the drive may be used. We say this because there are so many different PC's and operating systems. The only way to ensure troublefree operation in a PC is to prepare the drive in that machine.

For those less interested in PC interchangeability, there would eventually be two methods of preparing the IDE drive once installed in the Z-100. One would use a Z-100 version of FDISK. The other would be a quick and dirty format similar to that done on a floppy drive. This last version would not provide any means of compatibility with a PC.

Both these last two programs are on hold, waiting for someone with the necessary expertise and time to work on them. In the meantime, we will stay with the PC procedures.

IDE Devices:

The are several choices for IDE devices that can be used with your Z-100 LifeLine IDE Controller. These are discussed in great detail, along with the necessary preparation procedures, in the article **IDE_FDISK_Prep.pdf** on the Website.

The Altera EPM7064SLC84 Chip

The brains behind the IDE Controller is the Altera EPM7064S 84-pin Field Programmable Logic Device (FPLD) Chip. Most of the pins of the Altera chip could be programmed as either inputs or outputs and it eliminated the need for normal LS TTL logic to control the IDE functions of the Controller. It also processed the breakout switch, controlled the LED indicators, and controlled the read/write operations of the NVsRAM chip (memory and Real Time Clock).

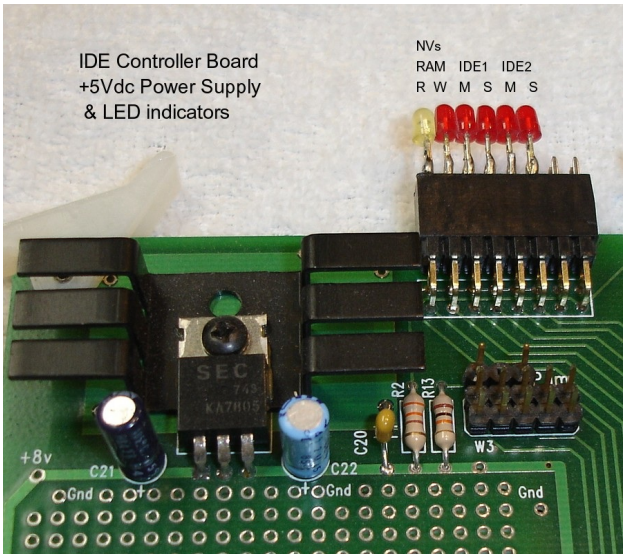
Breakout Switch

A breakout switch is located in the extreme upper right corner of the IDE Controller Board to generate an NMI (non-maskable interrupt) signal for breakout. The switch could be pressed at any time to exit to the monitor-ROM hand prompt.

Then, to return to the application in progress, at the hand prompt, press {G}o and {RETURN} {RETURN}.

The right-most pins on the LED connector can also be connected to another, optional breakout switch on a cable that you may locate anywhere in the Z-100 case.

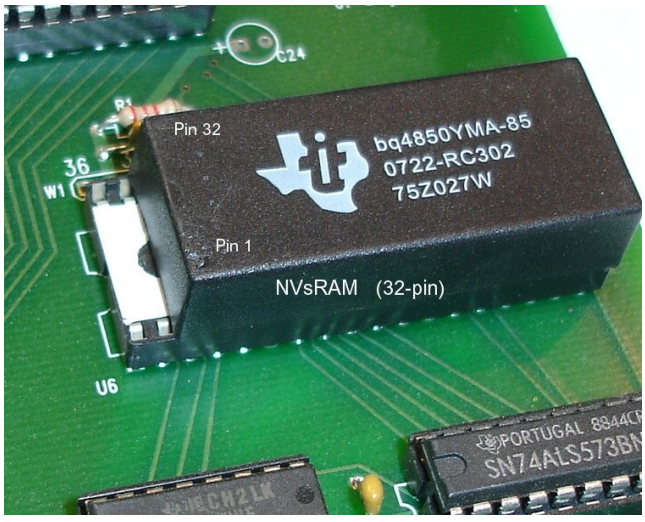
LEDs



The IDE Controller Board came standard with an LED assembly mounted on an LED connector to provide indications for NVsRAM and IDE drive activity. This assembly could be removed and replaced by your own assembly to mount a bank of LED's somewhere on the front panel. These were very handy for software debugging and discerning proper IDE card activity.

From the left in the photo, the yellow LED was the NVsRAM Read indicator. Next was the red NVsRAM Write indicator. The other four are the usual activity LEDs for the left IDE1 connector, master and slave drives, then the right IDE2 connector, master and slave drives.

IDE NVsRAM Preparation:



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.

The NVsRAM chosen for distribution was the Texas Instruments BQ4850YMA, a 512Kb NVsRAM (non-volatile static RAM) model with real time clock. The Dallas DS1647Y model has similar capabilities and is a direct replacement, if necessary. While these models use only 32 pins, the 36-pin socket was installed to allow using the larger 1024Kb models from both companies.

However, if these 1024Kb models include real time clocks, additional hardware and software is required to process this capability. Some clock models include alarms and wakeup capability!

CAUTION: When installing the 32-pin NVsRAMs, ALWAYS leave the four empty pins of the socket to the LEFT! A jumper, located just above the NVsRAM socket, is also required to be modified and jumpered to accommodate the 36-pin units.

Programming the chip is not difficult, but for frequent changes, it can be a pain, so I developed a batch file to make programming somewhat easier. It has been distributed with the other software for the controller boards, but will also be listed later.

First off, memory in the NVsRAM was limited. Depending upon the model, it could 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. So, due to space and circuit requirements, we settled on a 512K NVsRAM with an internal clock to best suit our needs.

Everything fit comfortably on the 512 NVsRAM, with plenty of room for additional files.

Z-100 Boot Process:

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 look 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 labeled 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 shop, I generally assign my drive letters as:

- A: thru D: to the floppy drives to maintain compatibility with the other operating systems
- E: thru H: to the MFM hard drive partitions
- I: is my imaginary drive (map as drive A:)
- J: automatically assigned by IO.SYS to the EEPROM on either the LLSCSI board or the LLIDE board, if either is detected.

The next available drive letters (beginning with K:) are then assigned by IO.SYS to any drive and their partitions found on either LLSCSI or LLIDE board.

This has worked out well for me in the past.

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 such 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 all partitions on each of up to four IDE drives and assign them drive letters, until the last drive letter is used.

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.

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 (programming) 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 this new expanded version.

The new CONFIG.SYS is fully explained in great detail in the article, **CONFIG4.pdf**, on our Website. It also includes some 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)
```

CONFIG.SYS Section Descriptions:

Section 1 lists the commands that are common to all options. Many of these commands may be moved to the individual options to accommodate special requirements of certain 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 32 Mb.

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 32 Mbs. FCB's will only work correctly as long as a file is physically located within the first 32 Mbs of a partition's start. If part of the file lies past this 32 Mb 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 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 a 3-1/2" or 8" floppy drive.

F3 specifies a hard drive; either an MFM drive attached to a Z-217 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 To boot from on a hard drive.

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 in 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 mother-board.

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 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 of my screen was 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:\>_

EPROMPGM.BAT Batch File

The 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.

For my case, I created an \EPROMPGM directory on my MFM hard drive and copied all the files that I wanted on the NvsRAM to that directory. These could also be copied to a floppy drive, just change the batch file to use the correct source location.

The files that I wanted were:

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

Next, copy any additional files as you may desire into the Root directory on the disk.

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: If you are using a large capacity floppy, remember to leave at least 525Kb of space for the EPTEST.DAT data file generated by EPRDFILE during the NVsRAM programming.

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

The EPROMPGM.BAT file is already loaded on any IDE device that I prepared, usually in the EPROM0 (without MFM hard drive configuration) or EPROM1 (with MFM hard drive configuration) directories. It is also located on the Z-DOS v4.06 CD-ROM.

A simplified version is also listed here.

EPROMPGM.BAT

```
ECHO.
ECHO Also Remember: Any time that the EPROM
     is placed in Program
ECHO Mode, the present checksum is destroyed
     and the EPROM is no
ECHO longer bootable, whether the EPROM was
     programmed or not.
ECHO.
ECHO       Once again, ensure the EPROM is
          drive  p %1  q.
ECHO Type {CTRL}-{C} to exit or
PAUSE
ECHO Copying IO.SYS, MSDOS.SYS, & COMMAND.COM
COPY IO.SYS %1/v
COPY MSDOS.SYS %1/v
COPY COMMAND.COM %1/v
ECHO Copying CONFIG.SYS, EPROMDSK.SYS, and
     LLIDEHD.EXE
COPY CONFIG.SYS %1/v
COPY EPROMDSK.SYS %1/v
COPY LLIDEHD.EXE %1/v
ECHO Copying AUTOEXEC.BAT, SYS.COM, &
     DRIVECFG.COM
COPY AUTOEXEC.BAT %1/v
COPY SYS.COM %1/v
COPY DRIVECFG.COM %1/v
```

```
ECHO Copying ASGNPART, DATETIME, DEBUG,
     EDLIN, FC, & FLAGS
COPY ASGNPART.COM %1/v
COPY DATETIME.COM %1/v
COPY DEBUG.COM %1/v
COPY EDLIN.COM %1/v
COPY FC.EXE %1/v
COPY FLAGS.COM %1/v
ECHO Copying FORMAT, LOOK, SETLPS, XCOPY, & ZDIR
COPY FORMAT.COM %1/v
COPY LOOK.COM %1/v
COPY SETLPS.COM %1/v
COPY XCOPY.EXE %1/v
COPY ZDIR.COM %1/v
ECHO Copying CHKSUMEP and EPROMPGM.BAT
COPY CHKSUMEP.COM %1/v
COPY EPROMPGM.BAT %1/v
ECHO Copying SHARE and ICLK??? utilities
COPY SHARE.EXE %1/v
COPY ICLK????.COM %1/v
ECHO Copying EPRDFILE, EPCPFILE and EPWRFILE
COPY EP??FILE.COM %1/v
CHKSUMEP
ECHO  EPROM Programming Completed.
GOTO Y
:X
ECHO This Batch File requires a drive letter
     filespec be added
ECHO for the EPROM...  For example: EPROMPGM K:
ECHO.
ECHO               p EPROM programming NOT
     completed.  q
:Y
ECHO.
ECHO Remember to run CHKSUMEP after making
     any other EPROM changes.
```

Description:
EPROMPGM must be invoked with a drive letter (colon not needed) for the NVsRAM. If the drive letter is not included, you will receive the error at "Y".

WARNING: Ensure the drive letter is correct, or you may erase the wrong drive!!

The **COPY** commands copy the system files to the NVsRAM, followed by all the files from the \EPROMPGM directory on the IDE drive. Adjust the COPY files as necessary for your needs.

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.

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 last to save a data file EPTEST.DAT on the default drive (if it has sufficient space) to save the programming in a backup data file. Upon rebooting, the NVsRAM is ready for use.

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. Put the NVsRAM in programming mode, then 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 utilities 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 utility 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 utility makes recovery for a corrupt NVsRAM quick and easy. It will copy the data file, EPTEST.DAT, 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: For any of these commands, you can change the name of the NVsRAM image file by adding the new file name as an argument to the command. For example, using the command:

EPRDFILE EPTEST2.DAT{RETURN}

will create an image file, EPTEST2.DAT. And the command:

EPWRFILE EPTEST2.DAT{RETURN}

will program the NVsRAM from that file.

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, create a bootable, high capacity (the standard 360Kb, 48 tpi, 5" drive will not work) backup disk.

This floppy disk would contain the files necessary to reprogram the NVsRAM using either method described above (i.e., using EPROMPGM.BAT or using the image file, EPTEST.DAT).

CAUTION: Whatever disk you use, remember to leave at least 525Kb of space for the EPTEST.DAT data file generated by EPRDFILE after the NVsRAM programming. Obviously, it will not fit on the 360K, 48 tpi normal floppy.

Note: A 360K 5-1/4" floppy disk will have enough space for the critical files and could be used to reprogram the NVsRAM using the EPROMPGM.BAT batch file method.

CAUTION: When reprogramming the NVsRAM using EPTEST.DAT, the file on an IDE Device (CF card or IDE drive) partition can **NOT** be used. You must use another boot device, such as an MFM or SCSI hard drive, or floppy drive to run EPWRFILE.

If you attempt to run EPWRFILE from an IDE partition, the file will appear to reprogram the NVsRAM, and will even report "*EPROM successfully written from image file*". However, if you check the programming using EPCPFILE, you will find that the programming was NOT changed. AND if you attempt to boot from it, the NVsRAM will still report being corrupt!

Note: Even booting to a Z-DOS v4 floppy and changing to an IDE partition to run EPWRFILE will not work.

This places those with only a 5" 48 tpi floppy drive with no easy way to reprogram the NVsRAM, because the EPTEST.DAT file is too large to fit! They could still use the Backup Floppy (but without the EPTEST.DAT file) and would need to use the manual reprogramming method (using EPROMPGM.BAT) as described above.

Only those who have installed a 96 tpi 5" drive, an 8" drive, or a 3.5" drive would have the room necessary to copy the NVsRAM utilities and the 524 Kbyte EPTEST.DAT file on a floppy disk.

If you have large capacity floppy capability, I highly recommend that you create an emergency backup bootable floppy disk that contains the NVsRAM utilities and the EPTEST.DAT file for NVsRAM reprogramming purposes. I recommend creating this Work Disk, even if you are using another hard drive system, such as the normal MFM Winchester.

Every purchase of the LLIDE Controller board included the bootable 5" floppy disk, "*Z-DOS 4.06 IDE Controller Utilities*" or "*Z-DOS v4.06 For IDE Setup*". This disk included all the files necessary for most NVsRAM functions, except the EPTEST.DAT files. These were placed on any IDE device that I may have programmed as a service to new purchasers of the IDE Controller.

If you did not receive any IDE device from me, you can still create your own EEPROM/NVsRAM Working Disk, as follows:

Create a spare bootable disk for whatever floppy disk system that you will be using; 8", 96 tpi 5", or 3". Run the command; **FORMAT B:/s/v**, which will install IO.SYS, MSDOS.SYS, and COMMAND.COM on the floppy to make it bootable.

Locate the bootable 5" floppy disk, "*Z-DOS 4.06 IDE Controller Utilities*" (or similar) and copy all the following files to your new IDE Work Disk:

AUTOEXEC.BAT	CHKSUMEP.COM
COMMAND.COM	CONFIG.SYS
DRIVECFG.COM	EDLIN.COM
EPCPFILE.COM	EPRDFILE.COM
EPWRFILE.COM	EPROMPGM.BAT
FC.EXE	FLAGS.COM

```
FORMAT.COM      LOOK.COM
ICLKCAL.COM     ICLKREG.COM
ICLKSET.COM     ICLKTST.COM
LLIDEHD.EXE     SETLPS.COM
IO.SYS          MSDOS.SYS
README0.DAT     SHARE.EXE
XCOPY.EXE       ZDIR.COM
```

Locate the directories, EEPROM0 (for NO MFM drives) or EEPROM1 (with MFM hard drives) that I included on any IDE device that I programmed for you and copy the appropriate EPTEST.DAT file to the new Work Disk.

Note: If you do not have these files, or cannot locate them, boot to the new Work Disk, or change the default drive to the drive your Work Disk is in, and run the command EPRDFILE. A new EPTEST.DAT image file will be created from the current NVsRAM programming and saved to your Work Disk for future use.

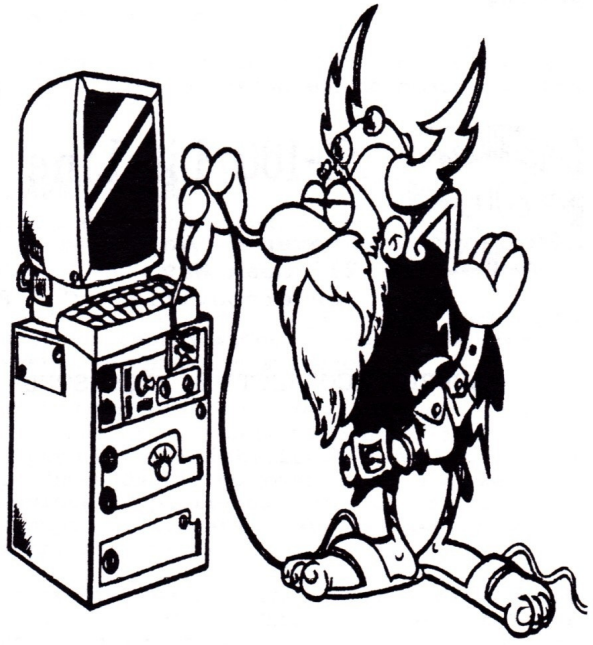
To check the programming of the NVsRAM, just run the command **EPCPFILE EPTEST.DAT.**

To reprogram the NVsRAM, simply run the command **EPWRFILE EPTEST.DAT.**

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

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

and store it in a safe location.



Forty years old and still ticking!

Setting the Clock

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 displays the programmed time from the Dallas DS1647 or TI bq4850Y clock.

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. Pressing any key will exit the routine.

CAUTION: This program reports the current calibration on a Texas Instruments BQ4850 NVsRAM clock and the current deviation in timer tics. 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:

* The rightmost 8 bytes of the clock registers are defined as:

```
      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.**

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.

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!

Project Status:

The IDE Controller boards have now all been sold. While everything we have been throwing at the new boards, including the new Compact Flash (CF) memory cards and Disk-on-Module (DOM) memory units, have been working, we have found that the DOM units require that U9 and U12 be changed to 74HCT245 chips.

Several problems have already been recognized and some still need to be addressed:

Software Not Recognizing IDE Partitions - Without going too deep into details, we have found that PeachText 5000, and probably other early software, do not know how to deal with partitions larger than 32 Mbs in size, probably due to memory addressing limitations.

When we created some smaller partitions on our IDE drives, they were recognized and used by Peachtext 5000 just fine. So, the easiest work-around is to ensure that there is at least one 32 Mb partition on the new IDE drives for use by those applications that will not recognize anything larger.

With all the new drive partitions being placed on the IDE devices, and acerbated by the requirement to have one or more smaller (<32Mb) partitions, it has become very evident that a Partition Listing utility is required to perform the same function that ZDIR does to directories and files.

It appears that the Editor in Peachtext 5000 has scrolling problems with the latest Z-ROM chip. As you know, the standard number of lines per screen is 24 on the Z-100, with a status line on line 25. This problem has been caused by the Z-DOS v4 lines per screen default of 25 lines with line 26 being the status line.

The fix is to run the new command:

SETLPS 24

DIAG giving false errors - The Heath/Zenith Disk-based Diagnostics Program, DIAG, is giving false errors when running the system memory tests from an IDE device, including the NVsRAM. The symptoms vary, but may include freezing the computer, giving 'Wild Interrupts' or 'Divide by Zero' errors, scram-bling info on the screen, false error reports, inconsistent operation, and other unexplained symptoms.

After days of tracking this problem down, we think that it appears whenever SHARE.EXE is loaded. SHARE is required for using disk partitions greater than 32Mb in size.

Until this is figured out, always run DIAG from a floppy drive and do NOT load SHARE.

Compact Flash Cards Not Recognized - There appears to be a problem with some PC clones not recognizing Compact Flash (CF) cards when placed on the same IDE connector as other drives. This requires further testing on both the PC and Z-100 machines.

CAUTION: Do NOT use WINDOWS to prepare and program your new IDE drives. While very tempting to just copy files and directories at the touch of a mouse, Windows places file volume names on all media during file transfers. These are seen on the Z-100 as additional volume labels and are NOT erasable at the present time. This occurs on all media - including floppies.

While they apparently cause no harm, they are listed with all the other file names and litter the start of the directory. Use a DOS startup disk and the DOS XCOPY utility to do your file transfers. The Z-DOS version of XCOPY works just great.

WARNING: Do NOT place a Z-100 bootable floppy disk in a PC for file transfers. The PC Windows system now writes a unique 8 byte disk ID to the boot loader on sector 0 of floppy disks, trash-ing any previous information!

For more information read the article **"DANGER! - Understanding Disk Volume Tracking in Windows"** in issue #84 of the "Z-100 LifeLine".

John Beyers is no longer available. So work on software issues has slowed to a crawl. I am looking for others with experience in assembly language to assist me in deciphering much of John's work so that we may continue to develop the remaining software to make maximum use of our new capabilities.

We are presently relying on the PC's Free FDISK to prepare the IDE drives on a PC. It is unknown when we will get to work on the FDISK and PCFORMAT programs to prepare the IDE drives while installed on the Z-100 computer. The next section discusses the FDISK problems in detail.

FDISK Problem:

MS-DOS FDISK issues:

While troubleshooting the Peachtext 5000 problem mentioned above, I tried a 64 Mb Compact Flash 'CF' Card on a PC. I made two 32 Mb partitions on it to try with PT5000 and ran SCANDISK on each partition - yet, every time I booted with it in the Z-100, the second partition was always listed as NO NAME.

Nevertheless, COPY and FC (File Compare) operations were all successful, or so I thought. I also tried two Caviar 1210 drives with similarly sized partitions with the same results. The COPY and FC operations were also fine.

It seems that while all partitions and partition sizes are being recognized, their assigned labels are not. The bootup screen shows the first partition name correctly, but the rest are all reported as "NO NAME".

Large partitions (those >32 Mb) work great, even if listed as "NO NAME". ZDIR recognizes the directory and new files are added correctly to the FAT.

Small partitions (those <32 Mb) are not working at all. ZDIR shows the directory as garbage or no files listed at all. New files are copied to the FAT correctly and File Compares are fine.

However, once the drive is returned to a PC, the newly added files are NOT listed in the FAT. SCANDISK reports a difference in the two FATs and, feeling that FAT1 is correct, wants to replace FAT2 with FAT1. This results in the new files added by the Z-100 as being lost and is reported by SCANDISK as lost data! Further, a 512 Kb file (my EPTEST.x file) is reported as 2 Mb of lost data!

Free FDISK and Bug:

Note: This version of FDISK.EXE is from FreeDos version 1.2.1 and is meant to run on a PC-clone. John Beyers has reported that it has been run successfully on the Z100 with MFM hard drives, but to run on the Z-100, it needs ZPC installed and you must be in PC Mode. The .INI files must be placed in the same directory as FDISK.EXE. Free FDISK is available from:

http://www.23cc.com/free-fdisk/

Charles Hett, who liked using Free FDISK instead of the Windows FDISK that I was using, was having different symptoms. His disks were all working properly, but showed size discrepancies on the boot screen.

Later, when Charles tried using a Windows 98 version of FDISK, he had problems partitioning a Samsung 1.5 Gb drive into two 32 Mb partitions and one 1.44 Gb partition. FDisk seemed to work fine preparing the drive, and the partitions checked OK with CHKDISK and Norton Disk Doctor. But when the drive was placed on the Z-100, the partitions were recognized but were reported as using an unrecognized format that could not be read from or written to.

He then repeated the entire process using Free FDisk and everything worked fine.

Well, when I tried using Free FDISK, I found that Charles was correct - that drives partitioned with Free FDISK work correctly, with one exception that I have found so far.

Version 1.2.1 that Charles sent me seems to have a bug in judging the size of the last partition. If I create the last partition by using either the default - when the program asks if I wish to use all remaining space and making it the active partition - or if I answer NO and set the amount of memory myself using the amount the program says is left, it seems to be 1 Mb too large and causes a problem.

For example, using the amount of memory for each partition that I had been using with WIN98's FDISK, there should be 28 Mb remaining on the Caviar 1210 drive that I had been using. The Free FDISK program reported that there was 29 Mb left, but if I use the default and answer YES to the question, the memory changes to 28 Mb when it is accepted anyway. And, if I place the 29 Mb in myself, the program will still replace the 29 with 28 Mb.

But, it does not stop there. With the 28 Mb size for this last partition - whether automatic or if I place the 28 Mb there myself, SCANDISK has a problem with this last partition. Every time I create the last partition and then FORMAT (OK), and copy files to it (OK), when I run SCANDISK on that last partition, it reports:

"SCANDISK cannot read from the last cluster on Drive x. This cluster is either damaged, or your system is not configured properly. Drive x may need to have Logical Block Addressing (LBA) enabled to work properly, or its disk partition may be incorrectly marked as a non-LBA partition. Data loss can occur if your LBA setting or disk partition type for this drive is misconfigured."

In actuality, when SCANDISK does the surface scan, it starts calling ALL of the last 100 or so clusters as bad. As each cluster is examined, it takes about 3 minutes, so after an hour of examining each of these and declaring it bad, I quit only about half way through. Instead, I had to go back into FDISK and make that last partition 1 Mb smaller and then everything was fine.

Note: On the 512 Mb CF card, for some reason, I had to drop 3 Mbs because the Free FDISK program would not accept anything less.

On the off chance that Win98 FDISK was having difficulty with the size of this last partition and causing my problems, I repartitioned the drive again with Win98 FDISK and reduced the last partition by 1 Mb, but there was no change in the symptoms listed above.

By the way, I also ran the Win95 version of FDISK on the Caviar 1210 drive before going to Free FDISK. It gave ALL the same symptoms as those partitions created with Win98 FDISK, EXCEPT that when using ZDIR, there were no garbage lines in the directory. It would just say that there were NO FILES!

So when I copied EPTEST.DAT, that file would be the first one listed! Of course, when the drive was placed back in the PC, all the other files would be listed again, less the new one!

End result: With the last partition sizes corrected, I have had NO problems using Free FDISK, as Charles has suggested. I have continued testing with IDE drives and many Compact Flash (CF) cards over the years and the results are consistent. If we cannot locate the reason for the Win9x FDISK problems (cluster size or other difficulty) we will have to use the Free FDISK version to ensure compatibility. Hopefully, by now both programs have been updated, but I continue to use the utilities that work (with the limitations as discussed).

The LifeLine staff feels confident that the IDE Controller Card is now functioning as it should. None of this sounded like a hardware issue with the new boards, but rather, because of the differences of Free FDisk and Win9x FDisk, which should be treating the Format sessions the same, we felt it must be a software issue.

Closing:

This was an exciting, one-time project, especially since our MFM hard drives have become increasingly difficult to find, and repair is virtually impossible.

I hope you find this article helpful, and encourage you to read the other IDE articles on this Website.

If you have any questions or comments, please email me at:
z100lifeline@swvagts.com

Cheers,

Steven W. Vagts