Z-100 LifeLine

Since 1989

March 2020

#131

A Professional Journal Exclusively for the Heath/Zenith Z-100 Computer

~~~~~~

Z-100 LifeLine Web Site: https://z100lifeline.swvagts.com

~~~~~~

HOWGOZIT

By now you should have noticed that **WE HAVE a new Z-100 LifeLine Website and Email address!** Please make note of these.

The last issue, #130 was my 100th issue of the Z-100 LifeLine as Editor and Publisher!!! And as part of the celebration, I attached a new ZBASIC program, CAKE.BAS. Hopefully, you were curious enough to key it in and run it, but if not, it was a side view of a simple graphic cake with the message "HAPPY BIRTHDAY, Z-100 LifeLine". However, the fun part that you could not make out unless you at least read the remarks in the code was the rockets and fireworks launching from the top of the cake (actually from the top of some of the letters)!

If you did try to key it in, because I could not find a font that did not use proportional spacing, the number of spaces listed in the program were hard to figure out, and most probably caused you some frustration. I am sorry about that, but I hope the result was worth it and that you enjoyed the program. Later, in this issue, I'll describe some of the more interesting aspects of the program.

With the last issue I also introduced you to a source of inexpensive test equipment - from Ebay and China. For about $15-25.00, you can now buy test equipment of all kinds that would make an excellent gift for some child interested in a career in electronics.

The last issue described a pair of interesting Transistor Testers. This issue introduces the DL4YHF2 Frequency Counter and Crystal Tester. I've also added a new page to the Z-100 LifeLine website, **Inexpensive Test Equipment**, that will contain all the equipment that I have reviewed in the next few issues.

One word of caution. I've now constructed several of these kits and you must remember that these are simple kits with amazing capabilities built in. However, you also get what you pay for - that is, these are not designed with a lot of care to part tolerances and have no additional delicate corrective circuitry. So, not all of these will work as they should.

Some of the kits I constructed came with a bad or incorrect part. So always test the parts that you can BEFORE installation.

Some completed kits did not have the full frequency range specified. One Transistor Tester, in spite of all I did and checks that I made, just would not work.

Nevertheless, those that did work are utterly amazing in what they can do in such a small package. My recommendation: At prices this low, buy the equipment in pairs. Odds are that at least one will work. Also, you have spare parts, if you need them, which also helps in troubleshooting a bad unit. At these prices, it is best for a little insurance.

Finally, let me know if you have any difficulty. I may already have a spare part that you can use, or I may have already found a solution to your problem. I would also like your success stories, if you have any. Thanks.

Now let's introduce the next find...

DL4YHF2 Frequency Counter & Crystal Tester

The DL4YHF Frequency Counter & Crystal Tester is sold as a kit and requires some experience in soldering skills. However, I have assembled the kit, created a schematic, and included everything you need in this manual. This manual can also be found on my website, on the Test Equipment page.

This is a simple five-digit frequency counter kit based on a PIC single chip 16F628 micro-controller, with a crystal oscillator measurement function, programmable frequency setting, and LED digital display. It is most commonly used to measure the oscillation frequency of a crystal or crystal oscillator.

Part/Model Number: YS9283, B8O8, and a few others, but it is actually a generic, unbranded circuit board based on a circuit developed by Wolfgang Buscher, DL4YHF. I have also added the ability to test the four-pin crystal oscillator, so we will call the modified device the DL4YHF2 Frequency Counter and Crystal Tester.

All the parts are through hole components, so the kit is easy, if you have basic soldering skills, and it is simple to operate. You will need to locate or purchase additional parts to add my crystal oscillator check function, but these are described later in the manual.

So, check out this issue's insert, **DL4YHF2 Frequency Counter & Crystal Tester Manual.**

ZBASIC CAKE.BAS Program

The ZBASIC CAKE.BAS program was mostly a quick program to make issue #130 a bit more special. It was actually my wife's idea. However, it felt good to get back into ZBASIC and I've always been impressed with the graphics capability of such an early application. Perhaps it is nothing compared to today's standards, but in 1982?

If you took the time to play with it, I hope you found it interesting. If you no longer have ZBASIC capability, I understand, but hope that you will follow along with my explanation and reasoning. I'm sure you can probably find better ways to accomplish the same graphics affects, but that is one of the things that I like about BASIC - there are usually several ways to accomplish something, and usually there are tradeoffs on which way you may choose to go.

So let's discuss some of the more interesting aspects of this program.

First off, I wish everyone would include a Title block in their program. In addition to the Title, it should include the author & perhaps contact info & the date, but most importantly, what computer and version of the programming language is being used. BASIC programs especially seem to lack this important piece of information, and there are so many different versions, most of which have certain programming quirks that a person must figure out. Will it run with BASIC-80, QBASIC, GWBASIC, BASIC-80, etc.?

This CAKE.BAS program uses the H-19 ESCape codes and graphics characters, so it may work on the H-8 or H-88/89/90 using BASIC-80 or something similar. It will NOT work with anything later or some form of the PC-clone BASICs.

Following the Title block, I like to set up the various equates to use the ESCape codes that I'll be using. So, generally starting at line 100, I like to define the ESCape character, E$, and then the ESCape codes to turn ON/OFF Reverse Video, Graphics Mode, sometimes even Colors to be used. I also like to initialize the ZBASIC RANDOMIZE function, in this case with line 130.

Looking in the ZBASIC manual, I could find no mention of RANDOMIZE TIME/DATE. So, I'm not sure where I picked up this statement. The reference manual only talks about the <expression> which is used as a random number seed value, and the examples simply show a number. But if you use the same number each time, the random number generated has the same number every time, and you have to remember to give a different number each time the program is run.

Just using RANDOMIZE TIME will not work; it generally gives the error, "Overflow in 130". For example, when I used the ZBASIC command PRINT TIME, DATE, and TIME/DATE, it gave:

```
  61279     16      3829.938
```

The Random Number Seed is limited to -32768 to +32768, hence the overflow error.

If you have not already done so, I recommend adding a note to the RANDOMIZE Statement of the Reference Guide, page 10.143 of the ZBASIC manual, "For a suitable Random Number Generator, use the statement 'RANDOMIZE TIME/DATE'".

Next, the COLOR statement is obvious; it sets the screen colors. COLOR 1,3 sets Blue on a Cyan background.

But the LINE statement of line 1010, could use some explanation. The LINE statement takes the form:

```
  LINE [(X1,Y1)]-(X2,Y2) [,[attribute]][,b[f]]
     Where:
        X1 and X2 are a column position
        Y1 and Y2 are a row position
        Attribute is a screen color
        B[f] is a fill background color
```

It permits the drawing of lines in absolute and relative locations on the screen.

LINE is the most powerful of the graphics statements. It is so important to my program that I'll repeat the reference guide, page 10.90, here verbatim, with a few of my comments interspersed.

It allows a group of pixels to be controlled with a single statement.

A Pixel is the smallest point that can be plotted on the screen. If you get up close and personal with your Z-100 screen or display, you can make out the distinct dots of light that make up a character. Each of these dots of light is a pixel. The ZBASIC screen can display 25 lines of characters, each 80 characters long. This gives us a screen 640 (80x8) pixels long x 225 (25x9) pixels tall.

The simplest form of LINE is:

```
LINE - (X2,Y2)
```

This will draw from the last point to the point (X2,Y2) in the foreground attribute.

We can include a starting point also:

```
LINE (0,0) - 639,224)
```

This will draw a diagonal line down the screen.

The statement:

```
LINE (0,100)-(639,100)
```

will draw a horizontal bar across the screen, 100 pixels down from the top line of pixels (about mid-screen).

We can append a color argument to draw the line in green, which is color two:

```
LINE (10,10)-(20,20),2
```

If we used a RND Function, we could make the line appear anywhere on the screen in any random color.

The RND Function takes the form RND(X) and returns a random number between 0 and 1. The same sequence of random numbers is generated each time the program is run, unless the random number generator is reseeded using the RANDOMIZE statement discussed above.

However, X<0 always restarts the same sequence for any given X. X=0 repeats the last number generated. X>0 or X omitted generates the next random number in the sequence.

For example, the statement:

```
PRINT INT(RND*100)
```

will print a number between 0 and 100. The INT function is used to restrict us to whole numbers (drops any decimal amount).

So, getting back to displaying our random lines and colors, we can use the program:

```
10 CLS
20 LINE -(RND*639,RND*224),RND*7
30 GOTO 20
```

to draw lines forever on the screen using a random color for each.

The final optional argument to LINE is ",b" for a box, or ",bf" for a filled box. The syntax indicates that we can leave out the attribute argument and include the final argument as follows:

```
LINE (0,0)-(100,100),,b
```

will draw a box in the foreground attribute.

Or:

```
LINE (0,0)-(200,200),2,bf
```

will draw a filled box with color attribute 2 (Green).

The ",b" tells BASIC to draw a rectangle with the points (X1,Y1) and (X2,Y2) as opposite corners. This avoids giving the four separate LINE commands:

```
LINE (X1,Y1)-(X2,Y2)
LINE (X1,Y1)-(X1,Y2)
LINE (X2,Y1)-(X2,Y2)
LINE (X1,Y2)-(X2,Y2)
```

which perform the equivalent function.

The ",bf" means draw the same rectangle as ",b", but also fill in the interior of the box with the selected color attribute.

When out of range coordinates are given in the LINE command, the coordinate which is out of range is given the closest legal value. In other words, negative values become zero, Y values greater than 224 become 224 and X values greater than 639 become 639.

So, in our Cake program, the statement:

```
1010 LINE (30,120)-(620,207),3,BF
```

draws our cake outline and fills in with Cyan.

To draw in our Cake message, we have the 2000 series of statements. The color of choice is set by COLOR 1,3, which is Blue on Cyan background.

The LOCATE statement takes the form:

```
LOCATE [row],[col][,[cursor]]
```

Where:

Row is the screen line number between 1 and 25 (not to be confused with pixel locations used elsewhere)

Col is the screen column number between 1 and 80.

Cursor is set to indicate if it is visible or not. Zero is OFF, non-zero is ON.

The LOCATE statement moves the cursor to the desired screen character position. Subsequent PRINT statements begin placing characters at this position. Optionally, it may be used to turn the cursor ON or OFF.

So, our 2000 series statements place the cursor at the desired location for each character to be printed on the screen. The F$ puts us in Z-100 Graphics mode. The series of graphics letters is used to print our message in characters 3 lines high across the front of the cake.

When we are done, line #2075 is used to locate to Home, turn OFF graphics and reset our screen color to the default white on black background.

Now, it gets more interesting.

We want to fire rockets that explode in the air in the form of fireworks! At a fireworks display, we generally see a rocket fire into the air, it explodes with a POP, and then the remnants drift down until they burn out. The series of statements beginning at line number 2100 sets us up for the first.

GOSUB 6000 is just a simple routine to set a random color for the rocket. You may notice that I had to add +1 to the equation. It seems that when INT drops the decimal, the number never gets to show 7 (white), and I did not want to show black on our black background. The simple fix was just to add one.

GOSUB 5000 was just to place a time delay for the rocket trail, the explosion, and the falling embers.

Another consideration was that we could use a random generated number and then an IF...THEN GOTO statement to fire each rocket from a random location on the cake. However, I found that when I was done, there was enough randomness that I didn’t need to make it any more complicated.

So I chose to fire the rockets from any letter that may look like a rocket launcher - ‘Y’, ‘H’, and ‘I’.

Two more comments of note, each rocket must be fired twice, once with the random color, and again at the default background color. This erases the track of the rocket from the screen.

We must also note the location at the end of the rocket’s track. We also need to convert the screen position numbers! The LINE command uses screen PIXEL positions, while the LOCATE command uses screen CHARACTER positions. If you choose the rocket positions correctly (divisible by 9 and 8), the numbers are converted in line number 7010 to give you whole numbers for the explosion LOCATE command. However, you could easily just use INT to drop the decimal part of the location.

The last interesting element of the program is the 7000 series of statements to create the explosions.

As we did with the rocket trails, we need to draw the initial explosion, erase it with the next level of the explosion, erase that with the next level of the explosion, etc., until the last remnants of the explosion are deleted, for 5 levels. I liked the final affect. I’m sure there may be other ways to do this but this worked to my satisfaction. Adjust the time delay loop to adjust the timing if you wish.

That’s about it. I hope you enjoyed the program. I found I enjoyed dusting off ZBASIC for this quick effort. Happy 100th issue to me.

Recent Repair Log

Hard Drive Will NOT Boot

I’ve been suffering with a hard drive on my test bed computer that would not boot when it was cold. I wouldn’t think too much of it, because after 10-15 minutes it would decide to work and I’d have no issues after that. I was always on another mission at the time, so it was usually forgotten until the next cold morning.

Oh, I would do the usual quick troubleshooting, changing the controller board and the data separator card, and even the cables, to no avail. So, I just figured it was the drive, or perhaps the motherboard.

Well, that drive finally died, with a terrible grinding sound, but then I found that the replacement drive was suffering from the same issue! What were the odds?

Now seriously doubting the cold drive theory, I finally set some time aside to troubleshoot the issue.

Sure enough, changing the motherboard solved the problem, but this was one of my favorite motherboards, with 256K RAM chips for 768k, 24MHz crystal, no other issues. I’d been using it for years. I could not stand the thought of retiring it.

Interestingly, the computer would still boot to the floppy drive and IDE controller. So I figured that it had to be something S-100 Buss related, that didn’t work for booting to a hard drive, but would still do everything else. How many signals worked only for the hard drive?

Well, it turns out, quite a few. The following list shows the signals used by the hard drive controller, but not the floppy controller:

S-100

Pin#	**Signal:**	**ICs Used:**
3	XRDY	U177, U206, U205
14	DMA3*	U???
15	A18	U163, U213
16	A16	U163, U213
17	A17	U163, U213
18	SDSB*	U182, U227
19	CSDB*	U182, U180
22	ADSB*	U182, U196, U197, U213
23	DODSB*	U182, U198
26	pHLDA	U180
32	A15	U162, U196
33	A12	U162, U196
34	A9	U162, U196
37	A10	U162, U196
44	sM1	U227
47	sMEMR	U227, U214, U195
48	sHLTA	U227
55	DMA0*	U???
56	DMA1*	U???
57	DMA2*	U???
58	sXTRQ*	U227
59	A19	U163, U213
60	SIXTN*	U182
61	A20	U163, U213
62	A21	U163, U213
63	A22	U163, U213
64	A23	U163, U213, U215
73	INT*	U177, U202, U208, U158
74	HOLD*	U185, U186
84	A8	U162, U196
85	A13	U162, U196
86	A14	U162, U196
87	A11	U162, U196
96	sINTA	U227
97	sWO*	U227, U214

By the way, this logic does not work for the floppy controller, as all the lines used for the floppy controller are also used by the hard drive controller - except the DC power lines. The hard drive controller and data separator card have their own power lines from the power supply.

I set about swapping out chips down the list from another good board, and quickly found that U163 was causing another problem.

The new chip in U163 was causing only a hyphen to be displayed near home, and not a hand prompt. I thought it strange, but kept changing it out until I found one that worked. When I got the hand prompt, the hard drive also sometimes booted, sometimes not. It was nothing conclusive, but it was strange. I completed the entire list with no other problems, then went back to U163. I checked all the pins on the IC and in the socket and they were soldered fine on the solder side of the board.

Well, just a short time later, the drive would no longer Boot at all. I finally decided to try another hard drive, and it worked flawlessly! Apparently, it was my hard drive failing all along.

With all the work I did trying to troubleshoot this problem, I created another troubleshooting aid - the S-100 Buss Pin Definitions sheet attached to this issue. It provides a list of all the S-100 Buss pins, their purpose, what ICs that they are attached to in their respective schematic sheet, and the page number where that signal is discussed in the Technical Manual.

Computer Will Not Display Hand Prompt

Actually, the computer gave a continuous tone from the speaker and no video at all.

You may recall that our Z-100 has a set routine to power up, and one of the first steps is to check the first 64K of motherboard RAM to ensure there is enough RAM to test circuits and display status messages. If it can't get far enough on its power up checks, instead of video, it must rely on beeps or tones to notify the user of a problem. Such was the case here.

While there may be other problems, the first thing to do is swap planes of RAM, if you have more than one plane. Lacking that, you need to find spare RAM to substitute RAM chips until you can find the chip that has gone bad.

Depending upon the bad chip, other symptoms may include no beeps, one beep, occasionally even a second, but still no video.

In my case, I found six bad chips on the motherboard, all Texas Instrument (TI) chips, five TMS4164-15NL, with the lot number P8214 and one TMS4164-Z20NL, with the lot number P8218.

After much research looking for more information, I also found a blurb in the installation of PC-Emulator boards that I had written back in 1996, that if you cannot get a short beep on power up, you may also try removing all S-100 boards and checking the following ICs on the motherboard:

U164 - 74LS240
U211 - 8088 or 8088-2
U219 - 74LS74
U221 - 74LS32

Completely Dead Motherboard

This motherboard was saved until last. It was completely dead, no beeps, nor video.

As with the last board, I tried changing the chips listed above. No affect. The only recourse was to begin swapping out chips. I began with the big chips; ZROM, CPU chips, etc., but finally began changing out whole groups of 4-5 chips at a time, beginning from the center of the motherboard, all around the video board connectors, progressing to the front of the board, then working clockwise, through the RAM, toward the back on the left edge, then down the right edge from the front.

As luck would have it, I found U241, a 74LS244, bad! Ah, Vagts luck at work. So, note to self and to all of you - add the U241 chip to change, if no beeps or video.

Well, while I was getting sound, it was a short beep, but still no video - still had issues. But now it sounded more like a RAM problem. So, while we are here, let's discuss this a bit next...

Power Up Beeps and Clicks or Life Before the Second Beep

One of the least understood circumstances, especially for the new Z-100 owner, is the 'no video' complaint. Let's try to make some sense of this, all too common problem.

The Z-100 has a lot of diagnostics built into the machine, both in hardware and software. The software is quite detailed, in the form of disk-based diagnostics, but in order to use it, you must be able to boot up to a disk.

The built-in ROM-based diagnostics are more critical, and from the start, Heath/Zenith tried to have you covered.

For a full explanation of what the ROM is doing during power up, please refer to Paul Herman's "*Z-100 LifeLine*", issue #2, "**Exploring the ROM**".

For our purposes, I'll only touch on the high points.

The following is a listing of the initial steps made within the monitor ROMs version 2.x. I don't know if changes were made in version 3.x.

* Upon a Reset or Power Up, the 8085 CPU begins to execute instructions and provides the first Beep.

* The computer then swaps processors to the 8088 CPU which begins executing its instructions and jumps to the monitor ROM entry point. The ROM code is mapped to the top of the first megabyte of RAM memory.

* If the 8088 passes its internal tests, the program checks to see if the MTR-100 data segment has already been initialized by looking for a ROM version number. If the number is found, this was a Reset and the next steps are skipped:

- A check is made of the monitor ROM checksum.
- The parity logic is checked.
- The first 64k RAM is checked.

Note: This is the RAM located in the first column, U101 through U109, with U101 being the parity chip. The second bank is U117 to U125, with U117 the parity chip. The third bank is U147 to U155, with U147 the parity chip.

- A quick check is also made of the second 64k RAM (if found).

* If any errors were found in the above steps, they are addressed now:

- If an error was found in the **first** bank of 64k RAM, there is no RAM with which to process and display the error. Processing simply ceases after the first Beep, and **no video is displayed.**
- If the first bank of 64k RAM passed the checks, the errors are displayed as follows:

```
"ERROR IN CPU. CHIP U211"
"ERROR IN ROM. CHIP U190"
"ERROR IN PARITY. CHIP U153"
"ERROR IN FIRST BANK PARITY RAM. CHIP U101"
"SECOND BANK RAM ERROR. CHIP Uxxx"
```

* The program jumps to the command processor, which Beeps success.

In summary, the first beep was generated by the 8085 processor right at the start - we are off and running. This second beep indicates that all the start-up diagnostics and initialization were done successfully.

As I understand it, the amount of testing done by the ROM chip was essentially the same for all ZROMs earlier than ZROM v4.0, it tested only the first **column** of RAM on the motherboard (more on this in a second), whether it contained 64k RAM chips or 256k RAM chips and did a quick check of the second column of RAM, if found, on the motherboard. To test the other RAM chips, you either had to boot up and use the disk-based diagnostics, or you had to physically swap the RAM chips between the first column and the third column and restart.

The reason for testing the first bank of 64k RAM, or the first 3 banks (192k) of 256k RAM, was to be able to notify the user of a RAM problem before boot up as this minimal RAM would be needed to display messages relating to boot problems, and the limited ROM size meant to keep things minimal. ROM size increased with ZROM 3.x to a 256k ROM, but as I recall, no change was made to the RAM test.

Well, my recent testing actually shows that with any ROM v2.x, it is actually the first TWO (2) columns of RAM tested on the motherboard at power up! Any bad RAM in the first column would cause a short beep and no video. However, if there were any bad RAM in the second bank (middle column), the bad RAM chip was actually displayed on the screen - by chip number!

I don't recall ever having read that in the manual. So, for those who are still using ROM v2.x, to troubleshoot a beep, but no video, just try swapping out the RAM chips between the first and third column on the motherboard.

I consider ZROM v4.x one of John Beyers' greatest accomplishments - a major enhancement to the Z-100's programming ability. One benefit of the larger ROM, was that John could now check all the RAM on power up, not the extensive testing of the disk-based diagnostics, but enough for basic needs.

The first column of RAM on the motherboard was still special, and John developed a means to report bad chips by using a long beep and a number of clicks, repeated indefinitely until the computer was turned off. So, if an issue was found in the first column RAM, there would be a long beep, and then a number of clicks (1-8) to indicate which RAM chip was bad; one click for chip U109, to eight clicks for U102 (from the rear to the front).

Note: In addition to the above memory issues, a long beep and a click could indicate an issue with resistor pack RP101 (a flat back DIP chip that looks like an IC), or with a PAL IC at U110, part #444-126 on 5 MHz 64k RAM (192k total) on the motherboard, part #444-367 on 8MHz, 256k RAM (768k total) on the motherboard, **OR**: U111, U126, U127, U128, U132, U133, U146, U151, U153, U155, or their respective sockets.

Once the first column was complete (1 bank for 64k chips, 3 banks for 256k chips), then the memory was there to display messages for the other two columns and a message would be displayed for any other bad chips, for example, 'SECOND BANK RAM ERROR. CHIP U145'.

Keep in mind that for these purposes, each column of RAM on the motherboard, is one bank for this memory test, not to be confused with the 12 banks of RAM tested in the disk-based diagnostics, if 256k chips are installed.

I hope you find this helpful. Two more notes before closing this memory topic.

One of the first errors that you may get while performing the disk-based diagnostics RAM tests is "Memory Parity Error", and a list of suspect chips. The list does NOT include any suspect RAM chips. After noting the suspect chips, always press the space bar to continue the tests. The full memory test may actually reveal the bad RAM chip(s) by number, and replacement may also fix the parity error.

When running the disk-based diagnostics, ALWAYS check that the configuration is correct before running the RAM diagnostics. If it is not correct, you will get all kinds of errors.

What banks to check?

If using 64k RAM chips, the test should be configured to test banks 0, 1, and 2 (192k RAM on the motherboard) and any additional banks required for the Z205 RAM cards.

If using 256k RAM chips, the test should be configured to test banks 0 through 11 (768k RAM on the motherboard).

The video RAM are in banks 12, 13, and 14, if you are using all the video RAM.

Hard Drive "Track 0 Contains Bad Sectors"

I had gotten a donated Atasi model 3046, 39Mb MFM hard drive from a friend and just could not wait to see if it had survived all these years...

I checked that it had the terminal resistor, but was not sure of the jumper settings for drive unit 0:. I checked the jumpers from the 34-pin connector, but it had three jumpers installed - for DS1, DS2, and DS3!

Note: If you can see the traces on the circuit board or have access to an ohmmeter, you can locate the Drive Select jumpers. One side of each of the Drive Select jumpers will be connected to the 34-pin connector as:

```
  Pin 32 to DS3
  Pin 30 to DS2
  Pin 28 to DS1
  Pin 26 to DS0
```

I removed all the jumpers and set the drive for what should have been DS0. After booting to a floppy with Z-DOS v3, I ran ASGNPART 0:.

Note: This drive has two drive LEDs. The right LED came on whenever the drive was ready (on speed and ready for a read/write command). The left LED apparently is the normal drive LED.

The drive was recognized as unit 0: because the left drive LED lighted, but I received the error "Read error on drive" and no partitions were shown.

OK, this drive appeared to need a PREP. I was running Z-DOS v3 at the time, so I tried that first. The drive specifications for the Atasi 3046 were:

```
  7 heads; 39 Megabytes
  285h cylinders
  286h reduced write cylinder
  140h pre-comp cylinder
  1 step rate
  28Ah parking cylinder
```

To use a drive larger than 32 Mb, you need to use PREP with the /k switch, which changes the normal 512-byte sector size to a 1024-byte sector size, and would then allow you to use up to 64Mb.

However, I did not want to use the /k switch, for a couple of reasons:

- I did not want the extra size because I have no use for it. A 30Mb drive is more than adequate for my use, even with multiple operating systems.
- I was not certain that the Gemini PC-emulator system would care for the 1024-byte sector size, though I think that it does.
- I was even more doubtful that CP/M operating systems would like the 1024-byte sector size.

Anyway, I would only use 5 heads, which would provide less than 30Mb. You could also reduce the number of cylinders to accomplish the same thing.

Running PREP from the floppy drive, the software reported "initializing the disk", but then promptly also reported "Track 0 contains bad sectors!"

I should have run DETECT at this time and probably would have been spared the following long waste of time, but truthfully, I did not think of it.

Anyway, I remembered that Z-DOS v4 PREP is not limited by the Track zero problems of the earlier versions because PREP will work around the bad sectors until it can find a clean, working string of sectors needed to act as Track 0.

Sure enough, PREP /Q/T1 (I did not want to wait around for 7 test passes) seemed to work great:

"Formatting Drive..." completed normally
"Media Test in Progress, pass 1, writing cylinder xxxx", where xxxx is the cylinder number.

Without stretching this all out word for word, all appeared normal. I got two bad sectors; 5680 of cylinder 63, and 11380 of cylinder 126, and the testing continued... at a crawl...

Keep in mind that 285h cylinders equates to 645 cylinders in decimal and while PREP is programmed with hexadecimal numbers, its progress report is in decimal...

Reached 110 cylinders in two hours... OK.
Reached 248 in 8 hours; 512 in 28 hours; 600 in 36 hours... What the hey?

I woke up the morning of the second day to find the test pass must have completed, but the final format shutdown because the maximum bad sector count was exceeded!!

This time I remembered to try DETECT - and bad sectors were scrolling off the screen, skipping every 1 to even as far as 18 good sectors in between. Using another jumper on the drive, which I think switched the drive from physical to logical sectors, DETECT showed bad sectors reported under all heads.

Figuring that the attached drive controller was bad, I scrapped the drive. Sure enough, the platters inside were spotless. What a shame.

Let's talk about MFM drives a bit:

Z-100 MFM Hard Drives

Since the first 5, 10, or 12Mb hard drives (called Winchester Drives by Zenith) were placed in the Z-100 in the mid-80s, there have been several more advanced types developed - MFM, RLL, SCSI, IDE and others. The drives usually found in the Z-100 were the first type - MFM - and the subject of this article.

Another drive type - SCSI - with considerably larger capacities, became available for the Z-100 in the late 80s with a SCSI Controller marketed by C.D.R. Paul Herman, editor and publisher of the "*Z-100 LifeLine*" at the time tried to get a special order of boards from C.D.R. adapted specifically for the Z-100. However, it soon became evident to Paul, and several volunteers working as his staff, that they needed to develop their own controller, and the new Z-100 LifeLine SCSI/EEPROM board was created.

As the MFM and newer SCSI systems became more scarce, attention turned to the newer IDE technology and another group of volunteers, John Beyers, Charles Hett, and I researched and developed the new Z-100 LifeLine IDE NvsRAM board, which finally shipped in late 2008.

You can find additional information on these newer systems elsewhere on the website. This article will concentrate on the use of the initial Winchester hard drive.

The Heathkit/Zenith MFM hard drive installation was comprised of a Z-217 Winchester Controller Card in the card cage, a separate, unique Data Separator Board that was normally mounted near or over the hard drive and the MFM hard drive itself.

Note: For full procedures on installing an MFM Hard Drive in the Z-100 series computer, please see Issue #59, September - October 1998 of the "*Z-100 LifeLine*" or the Z-100 MFM Hard Drive article on the Repair and Modification page of the website.

Preparing the Hard Drive

IMPORTANT: Early MFM hard drives are fragile and can be damaged easily. In all hard drives, while the drive's platter is spinning, the read/write heads float on a very thin layer of air, separating the heads from the platter's surface. However, the read/write heads on these early drives came to rest on the surface of the disk platter when rotation stopped. Any bumping, knocking, or dropping may cause the heads to bang against the surface of the platter. A severe bump, especially while the platter is spinning, could actually damage or gouge out a small area in the platter and cause a "crash", where an important portion of a program is unreadable and lost because the disk surface was damaged. Further, the read/write head may also be damaged.

For early hard drives, it is CRITICAL to run a disk utility that parks the heads in an unused portion of the disk - a storage or parking area - before the heads come to rest. Such a utility is SHIP, an external command packaged in CP/M and MS-DOS operating systems.

Later MFM drives had an auto-park feature that placed the heads down after the last usable sector of the drive, in an unused area. But even then, the heads could be damaged from a sudden drop of the drive.

MFM drives are recognized by their two ribbon cable card edge connectors, one with 34 conductors and the other with 20 conductors.

RLL drives also have these but the drive model number includes an R. For example, an ST-138 is an MFM drive, while an ST-138R is an RLL drive, with different formatting, capacities, and controller boards.

ESDI drives also have similar cable connections, but cannot be used.

MFM drives are becoming available from Ebay and the used market, sometimes at ridiculous prices and there is no guarantee that any of these will work. But let's assume that you find one with possibilities and want to try it. What is involved?

CAUTION: You cannot just slap an MFM drive from another computer into your Z-100 and expect it to work, without completely reformatting the drive. It will require low level formatting using the Z-100 PREP command, partitioning using the PART command, and a high level formatting of each partition using the FORMAT command. These commands are unique to the Heath/Zenith CP/M and MS-DOS (now referred to as Z-DOS) operating systems.

There are numerous manufacturers of MFM drives, each with different sets of programming plugs, jumpers, and terminating resistors. If you have a specific brand that you can't figure out, try emailing me at the "*Z-100 LifeLine*". I may have the info needed, or at least some suggestions.

As I mentioned earlier, please refer to Issue #59 for the full procedures for installing a hard drive in the Z-100. However, there are a few specific reminders:

* Try to install the new drive alone and boot to a floppy to run the Winchester Disk Utilities. It can be run from another hard drive, but you would hate to accidently PREP the wrong drive!

* If the new drive is installed alone, insure the terminating resistor pack is installed. If it is the second drive, only install the terminator resistor on the hard drive installed **last** on the 34-pin connector ribbon.

* Double check that the ribbon cables are installed per the directions in Issue #59. Insure all connectors are fully seated and that pin one of each ribbon connector (the ribbon cable may also have a red edge) is at the correct end of the connectors on the drive, controller, and data separator.

* Before running PREP on a hard drive, you must install the Format Enable Jumper on the Z-217 Controller Card.

* Check for a programming plug on the new drive before installation and make a note of the position of any jumpers. Try this setting first and if unsuccessful, try the other positions before giving up.

Hint: If you can see the traces on the circuit board or have access to an ohmmeter, you can locate the Drive Select jumpers. One side of each of the Drive Select jumpers will be connected to the 34-pin connector. The one we are interested in is DS0, connected to pin 26 on the 34-pin connector.

Hint: Before changing any positions on the programming plug, install the hard drive temporarily, boot up the computer, and run ASGNPART 0:. If the drive is already setup as drive 0: the drive LED should light. If it does not, check ASGNPART 1: and even 2: and 3: before giving up. If the LED will not light in any position, check the cables for an improper connection and finally try a different position on the programming plug.

Note: If partition info is displayed after running ASGNPART, do not proceed with PREP until you have tried other options. For example:

- Try running ASGNPART X:(Partition name) E:, where X: is the drive unit number. Then do a directory listing on E:.

- Try running DETECT or VERIFY to see how many bad sectors are found.

- Try reformatting the partition with FORMAT to isolate those bad sectors.

The PREP and PART utilities were available on a special disk entitled 'Winchester Utilities Disk', distributed separately from the earlier MS-DOS versions. The disk and instructions are available from the Z-100 LifeLine Library.

CAUTION: Using PREP is the last resort. It will destroy all the files that may exist on the hard drive. If the disk is from another Z-100, you may need to use PREP only if you consistently encounter an unreasonable number of disk access errors. Do NOT use PREP until you have backed up all important files you wish to keep to floppy disks.

PREP has been updated by John Beyers in the Z-100's Z-DOS v4 that allows it to be much more flexible in its operation - another reason to upgrade.

All versions of the PREP utility enable you to:

- Initialize the surface of the hard disk.
- Test the data retention capabilities of the hard disk.
- Isolate questionable disk sectors.
- Divide the surface of the hard disk into 2 partitions (Z-DOS and CP/M).

PREP takes a long time to run. For small capacity drives, expect it to take about 2 hours for every 10 megabytes in hard drive size. For larger drives, it may take many times that. It runs seven surface passes. However, with the version 4 PREP, you can set the number of passes to make, and it provides a status line to measure progress.

If the hard disk does not contain initialization information (from a prior PREP operation), PREP will prompt you to enter characteristics (in hex) in order to identify the type of hard drive that is being installed in the computer.

While Issue #59 has a list of common drives that were used in the Z-100, I have updated this information for many more manufacturers. Please see the Z-100 MFM Hard Drive article in the list of Repair articles on the Repair and Modification Page of the '*Z-100 LifeLine*' website.

Note: Several of the drives are too large for normal use in the Z-100. Early hard drives and Z-DOS versions in the Z-100 were limited to 32Mb. Later, with the addition of the PREP /k switch (which uses 1024 byte sectors rather than 512 byte sectors), the limit was extended to 64Mb. As I understand it, Z-DOS version 4 can go higher, though I don't recall the limit. Just remember, larger than 64Mb will ONLY work with Z-DOS v4. The fix is easy - just reduce the number of heads or the number of cylinders being used by PREP until the number of megabytes is where you want it.

Once PREP has completed, if you run ASGNPART 0: you will see the two partitions created: Z-DOS and CP/M. If you are satisfied with these two partitions, you will not need to repartition the disk with PART. However, if you wish to change this partition information, you must run the PART utility.

The PART utility is self explanatory. Just follow the procedures as given to change the partition names and sizes as necessary, then choose a default boot partition and save the configuration to the hard drive. When complete, you may need to reboot the computer to the floppy drive again.

Next run ASGNPART 0: to confirm the partitions are as you required.

Before we can use the new partitions, you need to assign drive letters to them and then run FORMAT to do a high level format of each new partition.

Run ASGNPART 0:(partition name) E: to assign the drive letter E: to the first partition. Likewise, assign succeeding drive letters (F:, G:, H:) to the remaining new partitions (up to four at a time).

Run FORMAT X:/s/v to format and load the system files on each new partition, where X: can be E:, F:, G:, or H:.

If successful, you are now in business. Email me if you have any difficulty. I hope this helps clarify the use of Z-100 MFM hard drives.

Closing

Once more, the Z-100 LifeLine website is at:

z100lifeline.swvagts.com or
swvagts.com/z100lifeline

Check out the What's New page. It has access to a new Z-100 Index that includes all three major publications, *Z-100 LifeLine*, *Sextant*, and *REMark*.

The What's New page also has a link to a **For Sale** page that you may wish to check out. I've got quite a collection of parts, software, and manuals to get rid of. This page will grow as I take inventory of my excess stock.

Remember, I have now gone paperless. This news-letter is now shown on the website as a PDF document that you may read and/or print at your leisure. I will send an email to you when each future issue is ready.

Check out the new Website often. I think you will like it.

'Til next time,
happy computing!

Cheers!!!

Steven

My new Z-100 LifeLine email address is:
z100lifeline@swvagts.com

S-100 Buss Pin Definitions

Pin:	**Signal:**	**Schematic Sheet: MBoard ICs:**	**Technical Manual Page:**
1	+8 Vdc (B)		
2	+16 Vdc (B)	S2:U228	
3	XRDY (S)	S1:U177,U206,U205,U236	S-100 Ready Line, 2.44
4	VI0* (S)	S1:U164	Slave 8259A INTs, 2.15, 2.21
5	VI1* (S)	S1:U164	"
6	VI2* (S)	S1:U164	"
7	VI3* (S)	S1:U164	"
8	VI4* (S)	S1:U164	"
9	VI5* (S)	S1:U164	"
10	VI6* (S)	S1:U164	"
11	VI7* (S)	S1:U164	"
12	NMI* (S)	S1:U182,U156,U189,U210,U211	Non-mask INT, 2.73
13	PWRFAIL* (B)	S1:U177	Power Fail, 2.73
14	TMA3* (M)		DMA Device Control, 2.66
15	A18 (M)	S1:U213 S2:U163	
16	A16 (M)	S1:U213 S2:U163	
17	A17 (M)	S1:U213 S2:U163	
18	SDSB* (M)	S1:U182,U227	Status Line Disable, 2.41
19	CDSB* (M)	S1:U180	Control Disable, 2.47
20	GND (0 Vdc)		
21	NDEF	S1:U215,U188	Assert if 8088 active, 2.33, 2.37
22	ADSB* (M)	S1:U182,U196,U197,U213	External Processor, 2.66, 2.67
23	DODSB* (M)	S1:U182,U198	External Processor, 2.67
24	Φ (B)	S1:U225,U203,U188 S2:U195	System Clock, 2.81
25	pSTVAL* (M)	S1:U180 S2:U214 S3:U130,U167	Status Valid, 2.45
26	pHLDA (M)	S1:U180	Hold Acknowledge, 2.34, 2.46
27	RFU		
28	RFU		
29	A5 (M)	S1:U197 S2:U181	
30	A4 (M)	S1:U197 S2:U181	
31	A3 (M)	S1:U197 S2:U181	
32	A15 (M)	S1:U196 S2:U162	
33	A12 (M)	S1:U196 S2:U162	
34	A9 (M)	S1:U196 S2:U162	
35	DO1(M)/ED1(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
36	DO0(M)/ED0(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
37	A10 (M)	S1:U196	
38	DO4(M)/ED4(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
39	DO5(M)/ED5(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
40	DO6(M)/ED6(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
41	DI2(S)/OD2(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
42	DI3(S)/OD3(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
43	DI7(S)/OD7(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
44	sM1 (M)	S1:U227	OpCode Fetch, 2.42, 2.134
45	sOUT (M)	S1:U227 S2:U214	Write to Port, 2.42, 2.90, 2.134
46	sINP (M)	S1:U227 S2:U214	Read Port, 2.42, 2.90, 2.134
47	sMEM (M)	S1:U227 S2:U214	Read Memory, 2.42, 2.134
48	sHLTA (M)	S1:U227	Halt Acknowledge, 2.42, 2.134
49	CLOCK (B)	S1:U216,U192	
50	GND (0 Vdc)		

S-100 Buss Pin Definitions

Pin:	Signal:	Schematic Sheet: MBoard ICs:	Technical Manual Page:
51	+8 Vdc (B)		
52	-16 Vdc (B)	S2:U229	
53	GND (0 Vdc)		
54	SLAVE CLR* (B)	S1:U201,U207	Unused, 2.39
55	TMA0* (M)		DMA Device Control, 2.66
56	TMA1* (M)		"
57	TMA2* (M)		"
58	sXTRQ* (M)	S1:U227	Unused 16-bit Req Status, 2.41
59	A19 (M)	S1:U213 S2:U163	
60	SIXTN*	S1:U182	Able to Use 16-bits, 2.41
61	A20 (M)	S1:U213 S2:U163	
62	A21 (M)	S1:U213 S2:U163	
63	A22 (M)	S1:U213 S2:U163	
64	A23 (M)	S1:U213 S2:U163	
65	NDEF		
66	NDEF		
67	PHANTOM* (M/S)	S2:U195,U194,U161,U190	RAM Swap, 2.52, 2.54, 2.64
68	MWRT (B)	S1:U216,U215 S2:U214	Memory Write, 2.4, 2.47
69	RFU		
70	GND (0 Vdc)		
71	RFU		
72	RDY (S)	S1:U177,U206,U205,U236	
		S2:U194 S3:U158	S100 Ready Line, 2.44
73	INT* (S)	S1:U177,U202,U158,U208	INT Request, 2.70
74	HOLD* (M)	S1:U185,U186	CPU Hold Request, 2.34, 2.38
75	RESET* (B)	S1:U201,U207	Reset Signal, 2.39
76	pSYNC (M)	S1:U180 S2:U195	Synchronization, 2.45, 2.55, 2.58
77	pWR* (M)	S1:U180 S2:U214	Valid Write Data, 2.47, 2.48
78	pDBIN (M)	S1:U180 S2:U214	Data Bus IN, 2.40, 2.46, 2.54, 2.63
79	A0 (M)	S1:U197 S2:U181	
80	A1 (M)	S1:U197 S2:U181	
81	A2 (M)	S1:U197 S2:U181	
82	A6 (M)	S1:U197 S2:U181	
83	A7 (M)	S1:U197 S2:U181	
84	A8 (M)	S1:U196 S2:U162	
85	A13 (M)	S1:U196 S2:U162	
86	A14 (M)	S1:U196 S2:U162	
87	A11 (M)	S1:U196 S2:U162	
88	DO2(M)/ED2(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
89	DO3(M)/ED3(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
90	DO7(M)/ED7(M/S)	S1:U198 S2:U178 S3:U132 S4:U244	
91	DI4(S)/OD4(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
92	DI5(S)/OD5(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
93	DI6(S)/OD6(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
94	DI1(S)/OD1(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
95	DI0(S)/OD0(M/S)	S1:U217 S2:U223 S3:U133 S4:U241	
96	sINTA (M)	S1:U227	INT Acknowledge, 2.42, 2.134
97	sWO* (M)	S1:U227, S2:U214	Memory Write, 2.42, 2.83, 2.134
98	ERROR* (S)	S1:U177,U208 S3:U158	RAM Error, 2.15, 2.60
99	POC* (B)	S1:U201,U207 S2:U215	Power-On Clear, 2.39, 2.73, 2.142
100	GND (0 Vdc)		

Note: TMA (Temporary Master Access) was previously named DMA (Direct Memory Access), used by the hard drive controller, but not the floppy controller.

SW Vagts, Editor, Z-100 LifeLine, February 2020