Z-100 LifeLine
Since 1989

March 2022

#WEB

This article was first published in issue #99, June 2005

Z-DOS v4 ZFMT207.COM Documentation

by Steven Vagts
Editor, "Z-100 LifeLine"

ZFMT207.COM

John Beyers has been writing a bevy of programs and utilities for the new Z-DOS v4 to take advantage of every conceivable capability that the Z-100 has to offer. As a result, this DOS has numerous changes that make life considerably easier for Z-100 users and the added flexibility for applications is mind boggling.

The Z-207 Floppy Controller Board has the capability of doing "FM" and "MFM" data encoding, has 50-pin and 34-pin connectors, and has 250 KHz and 500 KHz data rates available. It was logical to make a program that could take full advantage of all this capability.

One such program, DSKCOPY4, is a program to save a bootable floppy disk's image as a single file that could then be taken through a reverse process to reconstruct a bootable system disk whenever needed in the future. This will be discussed in another article.

But the DSKCOPY4 program also pointed to a need for a FORMAT program with more flexibility. So, John wrote ZFMT207 to create weird disk formats for experimentation.

The program is not a replacement for FORMAT, which does a superb job formatting standard floppies and hard drives. Rather, it is used for formatting and thoroughly checking diskettes only.

Using ZFMT207, you will also have to run SYS and LABEL separately to complete the disk. However, you can use the program to come up with some pretty bizarre formats, including some with higher disk capacities than the standard ones we use now.

Background

Let's briefly clarify a few terms that will be mentioned in our discussion and at the same time, briefly review the development of the floppy drive. Finally, we also add a warning to those using floppies on both their PC's and Z-100's.

"FM" and **"MFM"** encoding and **"Single Density"** versus **"Double Density"**: When discussing floppy drives, there are two basic means of encoding information for storage on a diskette. The term "double density" arose from the use of the term "single density" to indicate a type of drive that used Frequency Modulation (FM) encoding to store approximately 90 kilobytes on a single sided floppy. This type of obsolete drive was never used in any IBM-compatible systems, but was used in some older systems.

When drive manufacturers changed the drives to use Modified Frequency Modulation (MFM) encoding, they roughly doubled the recording capacity and began coining the term "double density" to indicate the relative gain and advantage of this new encoding. All modern floppy disk drives use MFM encoding. Even our old Z-DOS one-sided, 5.25", 8 Sectors/Track, 160K diskette, used MFM encoding.

"High Density": Another major change in floppy capacity was made when the number of tracks per inch was doubled, changing from 48 tracks per inch (used for single- and double- density drives) to 96 tracks per inch, and at the same time, increasing the rotation speed from 300 RPM to 360 RPM. These 5.25" drives were called "high density" drives.

"Dual Density": Technology quickly combined the capabilites into "dual density" drives that were capable of both double and high density formats.

The new 3.5" drives arrived next with their track density increased to 135 tracks per inch. The "double density" 720K diskettes spun at 300 RPM and only required a 250 KHz data rate from the controllers, using the common standard for 5.25" drives.

The "high density" 1.44M drives that came next, still spun at 300 RPM but used a 500 KHz data rate to improve capacity. For both 5-1/4" and 3-1/2" sizes, drives that were capable of both "double density" and "high density" are considered "dual density" drives.

"Extended Double Density": If all this wasn't confusing enough, one other factor changed to increase floppy capacity. DOS v1.1 and earlier used 8 sectors per track, which created double density 5.25" disks with a standard 320K capacity.

DOS v2 and later used 9 sectors per track, increasing the capacity to the 360K that everyone is familiar with. Many books treated this as an "extended format". Others ignored it entirely.

"Quad Density": Finally, while not useable on the Z-100, we mention 2.88M drives, if only to make this discussion on floppy drives more complete. As with the other 3.5" drives, these spun at 300 RPM, but required new controllers that supported a 1 MHz data transfer rate, beyond the capability of the Z-100's Floppy Controller. These were considered "quad density" drives.

"PC-type boot loader": This loader differs from the standard Zenith one. The boot loader actually contains two parts:

* A table called the BIOS Parameter Block (BPB), that tells DOS how to access the disk. It contains all the disk parameters. This has been modified in Z-DOS v4 to be compatible with what a PC machine expects to see.

* The boot code for the particular computer that this software is running on. This is obviously machine dependent and is what makes the software bootable on a particular computer. This can not be changed.

The **"PC-type boot loader"** enables the disk to be used on a PC without the PC damaging the existing Z-100 boot loader on the disk. But, see the following warning.

DANGER -- "Disk Volume Tracking in Windows"

Related to the information on the boot loader, I must restate a warning here that was discussed in depth in "***Z-100 LifeLine***", issue #84.

The article, entitled **"DANGER! - Understanding Disk Volume Tracking in Windows"**, provided information regarding a unique 8 byte disk ID that Windows now writes to the boot loader on sector 0 of floppy disks, trashing any previous information.

On our Version 4 Z-100 disks, this is only a slight annoyance, as this area now only contains a noncrucial date and time of the last change to the disk.

Unfortunately, on Zenith formatted version 1, 2 and 3 disks, the information that is trashed is crucial to the boot process.

Once trashed, the disk is rendered unusable and must be reformatted!!

The article concluded with a patch to your Windows PC to alleviate the problem.

One more note. There were two modifications made to the Z-207 Floppy Controller Card to permit using 3.5" drives on the Z-100, and now might be a good time to mention them:

* The **Hughes modification** permitted attaching dual-density 3.5" drives on the 34-pin connector by using the circuitry intended for the 50-pin connector. As it required no changes to the MTR-ROM, it should work with any MTR-ROM, at least v2.5 or later.

However, it disabled the use of the 50-pin connector for 8" drives and used the drive letters C and D for the high density counterparts of drives A and B.

The modification was described in great detail in "*Z-100 LifeLine*" issue #13. Z-DOS v4 still recognizes this modification as a selection on the DRIVECFG screen.

* The **Barfield modification** permitted the use of the dual-density 3.5" drives on the same cable as their 5.25" cousins by using a 5" fast step signal, and it retained the use of the 50-pin connector for the 8" drives.

For this reason it is the preferred option, but it requires Z-DOS v4 and a selection on the DRIVECFG screen.

Whether the Hughes or Barfield modification was made to the Z-207 card isn't important to any software except IO.SYS. And this is taken care of by the selection made in DRIVECFG.

The only precaution is to make sure to select the correct DOS drive letter for the disk type when you are using the Hughes modification.

~~~~~~

Z-100 Format Z207 Device Version 4.06

This program is NOT intended as a replacement to the DOS FORMAT command, you will have to run SYS and LABEL to complete the disk. It does allow you the option of creating most types of disks bootable by our MTR ROM that DOS and our Z-207 controller card are capable of using. IO redirection (PRN or NUL) can be used for Help Screen and Bad Sector Display.

Usage: [a:\][pathname\]ZFMT207 [?] [/x][d:][/x]

Where anything in [brackets] is optional and:

```
 a:\path  - Before command is the drive\pathname with transient command file.
 d:       - Is the drive identifying the disk to be formatted.
 /x       - Is any combination of the following switches:
   /A[n]  - n=0 Forces Low Density MEDIA on DUAL DENSITY drives, n>0 is High.
   /B[n]  - Specify BASE physical sector # (Default=0,HP PCDOSv1 compatible).
   /C     - Clear the directory entries and FAT, update to "PC" style loader.
   /D[n]  - n=# of Sectors to use for Root Directory entries (Default=1).
   /F[n]  - n=# of FAT's to create (Default is 1 instead of 2).
   /G     - If the previous FAT is readable, retain the old bad sectors.
   /H[n]  - n=# of Format Retries if Format Verify is active (default=3).
   /I     - Use INTERLEAVE for sector ordering.
   /K[n]  - n=log2 Cluster Factor, defaults to 0. (n>7 uses 1 sector/FAT)
   /L[n]  - Use FM mode. (n<>0 use 0FFh as filler instead of the default 0)
   /M     - Create single-sided disk with double-sided media.
   /N     - Suppress the onscreen prompts.
   /O[n]  - Override SPT default or if n=0 use MAX possible.
   /P     - Updates old disks with new "PC" style Boot Record, data is saved.
   /Q     - Fill each sector with location info Trk/Side/Sec/SecSize.
   /R[n]  - n=# of Reserved Sectors, default is 1 instead of # = 512 bytes.
   /S[n]  - Special CP/M media, FM trk/side 0 (n=/X def), MFM rest of disk.
   /T[n]  - n=# of Tracks, default is 40 tracks on an 80 track device.
   /U     - Do NOT Double Step when # of tracks is less than 1/2 of total.
   /V[n]  - n=#; 0=no verify; 1=VafterFMT; 2=VwithFMT; 4=Add'l Write Verify.
   /X[n]  - Extended format,0=128,1=256,2=512,3=1024 bytes/sector (default=3)
   /Y[n]  - n=0-16, use 0FnH for FAT ID instead of 0F8H.
   /Z[n]  - Track Zero SPT if Special CP/M media (default=26)(n=0 is Max)
```

Figure 1.
ZFMT207 HELP SCREEN

~~~~~~

Invoking the ZFMT207 program's Help Screen.

To invoke the ZFMT207 Help Screen, type:

ZFMT207 ?{CR}

or

ZFMT207 /?{CR}

where {CR} will represent the {RETURN} key (the {CR} is omitted for the remainder of this article).

The ZFMT207 Help Message of Figure 1 is displayed.

The Help Screen and Bad Sector Display can be printed out by using IO redirection (PRN or NUL); for example:

ZFMT207 /? >PRN

The ZFMT207 program is invoked as follows:

Usage: **[a:\][pathname\]ZFMT207 [?] [/x][d:][/x]**

Where anything in [brackets] is optional and:

a:\path - Before command is the drive\pathname where the transient command file is located.

d: - Is the drive identifying the disk to be formatted.

/x - Is any combination of the following switches:

/A[n] - If n=0, it forces the formatting of Low Density MEDIA on DUAL DENSITY drives. If n>0, it forces High Density.

This switch is only necessary when the program can not detect the media type from the pre-existing format. Low Density (n=0) is the default if no parameter is given.

It can also be used to override the media detection process in the case where the pre-existing format is incorrect on the disk.

/B[n] - Specify BASE physical sector number (Default=0, HP PCDOS v1 compatible). Normally the base physical sector number on each track is 1. While this has been the industry standard for many years, early in the development of personal computers, some disks (such as those used by Hewlett-Packard) had a base of 0.

Actually, it is possible to start with any number. Some copy protection schemes start with sector 100. Our new BIOS can deal with any starting sector number, but all "PC's" and our original BIOS's must start with sector 1. So, unless you want to use it as a form of copy protection or need to use the disk on an early computer, do not use this switch.

Without the switch given, the default is 1, but if the switch is used, it is assumed that some other value is intended. If a value for n is not provided, the default therefore becomes 0. **Use this switch carefully.**

/C - Clear the directory entries and FAT and update to "PC" style loader. This is similar to the original FORMAT in that you can not change the physical characteristics of the disk - number of tracks, sector size, sectors per track, or base sector number - because a low level format is not performed.

But you can change the logical configuration, number of reserved sectors, number of FAT's, number of Root Directory entries, cluster factor and the FAT ID. You can also use the /G switch to retain the previously detected bad sectors.

All data is lost, but not actually overwritten unless the size of the FAT's and Root Directory encroach on the previously written data.

/D[n] - Where n sets the Number of Sectors to use for Root Directory entries. The default is 1. Each Root Directory entry uses 32 bytes, so depending on the sector size, a variable amount of entries will fit in each allocated sector.

If you request a standard disk format, the previously defined number of Root Directory entries is used. All "PC's" and our original versions of BIOS must use these defaults. Our new BIOS can deal with any size for the Root Directory.

/F[n] - Where n gives the Number of FAT's to create. The default is now 1 instead of 2.

All standard DOS formats use 2 FAT's. MS-DOS will write to all copies of the first FAT, but may not read from any but the first FAT. You must use some other program, like Norton, to recover a file from a FAT other than the first. Only our new BIOS can pass to DOS other than 2 FAT's.

With the improved disk reliability, the default has now been set to create only 1, allowing the extra memory to be used for data.

/G - If the previous FAT is readable, retain the old bad clusters. Since some bad sectors only show up once in a while, this is a way to keep all sectors ever detected as bad, marked bad in a newly created FAT.

Remember that if the cluster mapping changes, ALL sectors in the previously bad clusters will be mapped to their new cluster, and some good clusters may end up being marked as bad.

This could happen when the previous cluster factor is greater than 1. Since we do not know which sector caused the cluster to be marked as bad, it is assumed all the sectors in the cluster are bad. If all these sectors do not end up in the same cluster under the new format, then some good clusters might also be marked as bad.

/H[n] - Where n sets the Number of Format Retries if Format Verify is active. The default is 3, if /H is specified without [n]. If /H is not specified, no retries are attempted. Otherwise, retries are performed only in the case when Verify with Format is also specified. If it finds a sector verify error, a complete re-format of the track is performed and then a complete re-verify is done.

/I - Use INTERLEAVE for sector ordering. If the "inter-sector" gap is too small, reading consecutive sectors requires a complete rotation of the disk to get back to the next logical sector.

Interleave permits faster disk access by changing the order of track sectors. In a standard disk with 9 sectors per track, without interleave, the disk may have to make a complete revolution in order to reach the next sector to be accessed.

With interleave set, the order of the sectors is changed so that the next sector that requires access is the next to be located under the head and is nearly instantly accessible. So, if verify is taking way too long, Interleave may help to avoid the extra rotation for every sector read.

/K[n] - Where n=log2 Cluster Factor. The default is 0 and when n>7, 1 sector/FAT is used.

Cluster factor is the number of sectors per cluster. Each cluster has one entry in the FAT.

If the total clusters is greater than 4086, then 16-bit FAT's are used, otherwise a 12-bit FAT entry is used by DOS.

Obviously, if you increase the number of sectors in each cluster, you will reduce the total sectors required for each FAT, thus increasing the available space on the disk for data.

The log2 values are as follows:
0 = 1 sector/cluster
1 = 2 sectors/cluster
2 = 4 sectors/cluster
3 = 8 sectors/cluster
4 = 16 sectors/cluster
5 = 32 sectors/cluster
6 = 64 sectors/cluster
7 = 128 sectors/cluster, the largest cluster factor DOS will use.

Any value for n that is greater than 7 will request a calculation of the minimum cluster factor that will cause only 1 sector for each FAT, maximizing the available data area.

/L[n] - Use FM mode, where n<>0 uses 0FFh as filler instead of the default 0. You must use the /L switch for FM encoding mode, no default formats will use the FM mode.

Normally, a diskette is formatted using MFM encoding. Only 8" single sided disks have previously supported FM mode. When using FM mode, only half as many bytes will fit on each track, so it is uncertain why anyone would choose it for any other reason than testing or just curiosity.

The passed parameter is because our 1797 chip will support two different filler bytes in FM mode, 0 or 0FFh. Again, it is not certain why or what difference the value of the byte makes to the final format.

As a final note and passing thought for further discussion, this switch permits us to change the encoding strategy to all FM, perhaps providing some degree of security, similar to encryption.

/M - Create single-sided disk on double-sided media. This switch is always necessary on 3-1/2" or 5-1/4" media, and for old 5-1/4" media, to permit using a double- sided diskette in an old single-sided drive. It is also required on double sided 8" media when you want to force single sided format.

However, because 8" drives differentiated double-sided media from single-sided media by the location of the diskette's sector holes, this may not work on single-sided drives.

/N - Suppress the onscreen prompts. This affects some of the questions asked and some of the logic used to process the other switches:

Command line drive - If no command line drive is given, you will be prompted for the drive to format and be instructed to place the media to format in the drive, regardless of the condition of /N.

If the command line drive given is NOT the "default" drive, no prompt for media to format is given, regardless of the condition of /N.

If it is the "default" drive and /N is used, the prompt to insert the correct media will be skipped.

If write protected/no media is detected, you will be prompted or re-prompted for media to format regardless of the condition of the /N switch.

Invalid/Marginally desirable Switch values - With some of the switches that support parameters, a check is made for invalid values and marginally desirable values. If the /N switch is detected, the questions to verify the invalid or undesirable values will be skipped, and it is assumed the value to be correct or, if not, then abort the program.

Display of disk characteristics - After processing all command line switches and detection of media type, the physical and logical values that are going to be used to create the newly formatted disk are displayed. If the /N switch is detected, the question requiring the {RETURN} key to continue will be skipped.

/O[n] - Where n is the number of Sectors Per Track desired. Early DOS used 8 sectors per track; later DOS used 9 sectors per track.

Use this switch to override the SPT default or if n=0 use the maximum possible. If no switch is given, the defaults are determined by sector size, FM/MFM mode, transfer rate, and RPM according to the following table:

Xfer/BPS	MFM-300	MFM-360	FM-300	FM-360
250K 128	26	26	13	13
250K 256	16	13	8	6
250K 512	9	7	4	3
250K 1024	5	4	2	2
500K 128	52	52	26	26
500K 256	32	26	16	13
500K 512	18	15	9	7
500K 1024	10	9	5	4

If a value of 0 is given or no value is given, the maximum possible Sectors Per Track is determined, given the drive's specific RPM detected and the minimum gaps required by our 1797 chip on the Z-207 floppy controller card.

Interleave might be needed to avoid the possibility of one rotation for every sector read when using the maximum possible sectors.

CAUTION: If you use the maximum SPT, the drive's RPM becomes a major factor. A slower drive will permit more sectors per track than a faster one. If the disk is later used on a faster drive, it MAY still be readable, but writing MAY make the disk unusable!!

There are a couple of exceptions for the use of this table. If you specify 77 tracks on a high density disk or it is an 8" drive, 8 SPT is forced assuming the old Zenith formats.

/P - Updates old, used disks with the new "PC" style Boot Record without losing the previously written data. No alteration of either the physical or logical characteristics are permitted, as this would cause the data to become invalid.

This just writes the bootloader to sector 0 on the disk. No verification is permitted either, as a newly detected bad sector may be assigned to a valid data file.

/Q - Fill each sector with location info Trk/Side/Sec/SecSize. This is only valid if you are doing a low level format.

For all sectors not in the system area, boot-loader, FAT's and Root Directory, ASCII text identifying the physical location of each sector is written by the write track (format) command instead of all 0E5h's.

This can be useful if you are trying to locate specific bad regions on a track by using the RDTRACK utility.

/R[n] - Where n is the number of Reserved Sectors. The default is 1, reserving one sector of 512 bytes.

If no /R switch is given, it is assumed that the disk should be bootable. This requires the 512 byte boot loader.

If 128 byte sectors were used, 4 are reserved.

If 256 byte sectors were used, 2 would be reserved.

If 512 or 1024 byte sectors were used, then 1 would be reserved.

If /R is given without a number, then 1 is assumed.

To eliminate bad sectors from the FAT's and Root Directory, you may give more than is required, but sector 0 must always be good.

/S[n] - Special CP/M media, where FM encoding is used for track 0/side 0 and MFM encoding is used for the rest of disk.

No bootloader is written and no verification is performed. While this may not have any real use, putting FM on track 0 and MFM on the rest of the disk is a valid CP/M format.

You may specify the sector size for track 0/side 0 with the n parameter: 0=128; 1=256; 2=512; 3=1024 bytes/sector.

You can further specify SPT (Sectors Per Track) on track 0/side 0 with the /Z switch.

/T[n] - Where n is the number of Tracks. The default is 40 tracks on an 80 track device.

If no switch is given, 40, 80 and 77 tracks are assumed depending on the device type.

If only /T is given without a number specified, double stepping is assumed (i.e. 360K media in an 80 track device).

Many drives will format some additional tracks - some 80 track drives will format 82 tracks, but be sure that verify does not mark all sectors on the additional tracks as bad.

/U - Do NOT Double Step when the number of tracks is less than 1/2 of total.

In some cases, GEMINI for example, not double stepping will allow the use of 80 track devices with simulated 40 track media on the first half of the disk.

/V[n] - Where n can be 0-7; n=0 gives NO verification.

With the deteriorating condition of our 20 year old media and the fact that we are finding it difficult to buy new media, quite a few VERIFY options have been added so we might have a better chance of identifying bad sectors and still be able to use the old media.

If the /V switch is not specified, "Verify After Format" is assumed. If you really do not want to verify at all, you must specify n=0 (/V0).

The value for n is actually using a binary bit pattern to provide greater flexibility in our verification choices. The bits are defined as:

bit 0 (001)	**Verify After Format**, after the complete disk has been formatted.
bit 1 (010)	**Verify With Format**, after each track has been formatted.
bit 2 (100)	**Verify With add'l Write**, after each track has been written.

The values for n therefore become:

n=0 (000)	**No verification** is done.
n=1 (001)	**Verify After Format**, after the complete disk has been formatted.
n=2 (010)	**Verify With Format**, after each track has been formatted.
n=3 (011)	**Verify With Format** and **Verify After Format**.
n=4 (100)	**Verify With add'l Write**, after each track has been written.
n=5 (101)	**Verify After Format** and **Verify With Write**
n=6 (110)	**Verify With Format** and **Verify With add'l Write**
n=7 (111)	**Do all three verifies.**

Verify Descriptions:

"Verify With Format" is the first available verify. It is not valid with the /C switch, and must be given for the /H switch to be meaning-ful. This verifies each track immediately after it has been formatted.

If an error is found, and you have specified the /H switch, we can attempt to re-format the track and then re-verify.

Not until all attempts have failed will the sector be marked bad in the FAT. If a bad sector is located that is needed for the boot loader, the FAT's, or the Root Directory, ZFMT207 is immediately aborted with a message.

You may still be able to use the disk if you reduce the size of the FAT's, the number of FAT's, or the size of the Root Directory. You may also extend the size of the reserved area, past the bad sector, and still be able to use the disk.

"Verify After Format" is the next available verify. This is the default verify and identical to the original verify in FORMAT. It has an advantage of time, in that it might catch some problems associated with what may be called "short term magnetic fade".

Some disks with bad sectors do not show up bad in the "Verify With Format" test, but do show up bad just a few minutes later in the "Verify After Format" test. The extra time may allow the bad sector's magnetic image to fade just enough so that it is no longer reliable.

This verify is also appropriate for a "longer term magnetic fade" test, but you would have to reformat the disk with the /C switch a day or week later.

"Verify With Write" is the last verify. Because, up to this point, we have not actually done a "write to sector" on the newly formatted disk, we can not be absolutely certain that the 1797 chip will succeed in creating a valid CRC and be able to read it back.

So we write each track and then verify each track before stepping to the next track. If either the write or the verify shows a bad sector, it is marked in the FAT.

For an even more comprehensive verify, please see the Verify Batch File Example discussed at the end of this article.

/X[n] - Extended format options, where each value of n allows setting the sector size: **0=128, 1=256, 2=512, 3=1024** bytes/ sector. The default is n=3, 1024 bytes/ sector.

If the /X switch is not given, 512 bytes/sector is assumed for 3-1/2" and 5-1/4" disks. For 8" disks 1024 bytes is assumed for MFM and 128 bytes/sector for FM.

If only /X is given without the parameter n, 1024 bytes/ sector(3) is assumed.

/Y[n] - Where n can be any number, 0 to 16; to use 0FnH for the FAT ID instead of 0F8H.

The media ID byte or FAT ID is the first byte of the FAT and is used in earlier versions of DOS to identify the format of the disk.

It can be used by our latest BIOS for the same purpose, if the sector 0 becomes unreadable.

This switch is necessary for one of Zenith's old formats that used the incorrect FAT ID - double density, 96 tpi(80 track), 8 SPT double sided disk. Zenith originally used 0FDh, and later changed it to 0FBh.

To create the old Z-DOS one-sided, 8 SPT, 180K disk, use /Y13/O8.

The n can be any number from 0-16, where n=0 gives 0F0h ... n=16 gives 0FFh. The standard Media ID bytes are as follows:

0F0h - 3.5" Floppy 2-sided, 18 SPT
0F1h - 0F7h Have no standard meaning
0F8h - Fixed Disk, or non standard floppy format
0F9h - 96tpi 2-sided, 9 SPT (720K)
0FAh - 96tpi 1-sided, 8 SPT (320K)
0FBh - 96tpi 2-sided, 8 SPT (640K)
0FCh - 48tpi 1-sided, 9 SPT (180K)
0FDh - 48tpi 2-sided, 9 SPT (360K)
0FEh - 48tpi 1-sided, 8 SPT (160K)
0FFh - 48tpi 2-sided, 8 SPT (320K)

/Z[n] - Track Zero SPT (Sectors Per Track) if Special CP/M media. The default is 26.

This switch is only valid if /S, Special CP/M media, is also specified. It works like the /O switch, so that you may select how many Sectors Per Track to use on track 0/side 0 when creating special CP/M media.

If n = 0 the maximum sectors per track is used (see the explanation for switch /O[n]).

If n is not given, 26 Sectors Per Track is assumed.

Other Information

Two boot loaders presently exist. One if the total size of the reserved area is less than 512 bytes (not enough space to be bootable), and the second if the reserved area is greater than or equal to 512 bytes.

The second boot loader is suitable for all our versions of Z-DOS and MS-DOS, so it is no longer necessary to format a disk with a specific version of FORMAT.

If you were creating a disk for Z-DOS, or Version 2 or 3 of MS-DOS using the original BIOS, IO.SYS had to be the first entry in the Root Directory and it had to be contiguous. MSDOS.SYS had to be the second entry in the Root Directory and it also had to be contiguous. The SYS command was required to transfer the system files correctly to the new disk.

Z-DOS Verson 4 no longer has any restrictions for any of the system files, so you can just use the COPY command.

It is also no longer required that IO.SYS be the first entry in the Root Directory or for IO.SYS to be contiguous because the new boot loader searches for IO.SYS and now decodes the FAT.

If the total bytes in the reserved area is less than 512 bytes, the "small" boot loader is used. It has enough information for DOS/BIOS to use the disk as a data disk, but it not bootable.

If you try to boot from the disk, "TOO SMALL" is displayed on the screen and you are returned to the Monitor hand prompt.

Media Types Supported

1 or 2 sided 8",
1 or 2 sided 5 1/4" high/low density
1 or 2 sided 3 1/2" high/low density

Default Formats Supported

Any media created can be used by BIOS version 4. The defaults used if no or incomplete switches are given are set up so that the media can also be read on a "PC".

There are a couple of "incompatibilities" with our older versions of Z-DOS and MS-DOS (unique FAT ID, number of Directory Entries and Cluster Factor):

* Z-DOS double density, 96 tpi (80 track), 8 SPT double sided disk (Use /Y13/O8)
* Ver2 & 3 MS-DOS 96tpi double sided, extended 9 SPT (Use /D9/K2)

Otherwise the switches are fairly straight forward for creating standard media types for our Versions 1, 2, 3 and 4 of Z-DOS/MS-DOS.

ZFMT207 Examples

At the end of this article is a long reference table of all the media supported by each DOS version. Each disk type also has the specific ZFMT207 command given with the necessary switches provided.

Let's go through a couple of examples.

NORMAL 5-1/4" DISKETTE:

Assuming drive A is our 5.25" drive, to create a normal, 360K disk, place a diskette in drive A and type the command:

ZFMT207

The computer responds with:

Which Floppy DRIVE to Format (DOS drive letter)?
Place DISK to Format in Drive _

When you press {A}, A is placed at the underline position above.

The computer proceeds to verify the disk is in the drive, and responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
LOW Den 48tpi 5 1/4" Drive, VERIFY=Aftr Fmt;
Using 250K xfer, MFM mode, 40trks,
* 09spt, 512bps, 2sides, 1base*
7 RootDir(112 entries), 2spc, 02 fats,
* 2 spf, 01 reserved, 0FDh MediaID*
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Formatting Track xxx Side x...

Where 'xxx' represents the track number and 'x' blinks 0 and 1 as the format operation progresses. After the format is complete, the computer begins the verification as requested and displays:

Verifying Track(Side) @ Sector xxxx

Where 'xxxx' represents the climbing sector numbers as each is checked. As bad sectors are located, each is listed as:

Verify Bad Sector xxxx, will be marked
BAD in the FAT(s).

And the verification continues until the last message is displayed:

Disk successfully created.
Found x new bad clusters.

And you are returned to the DOS prompt. At this point, ZFMT207 is complete.

Now you can run DIR or ZDIR to see the space available on the disk, or run CHKDSK for a more complete listing of the status. As mentioned earlier, if you want a bootable disk, you must run SYS, to place the system files properly on the disk, by using a command such as:

SYS E: A: /M/C

Which, if you don't remember, copies IO.SYS from drive E: to drive A: as the first file; MS-DOS.SYS as the second, and COMMAND.COM as the third. While Z-DOS version 4 no longer requires the files to be contiguous and in that order, the other versions of DOS do, so it's nice to stay in that habit of placing them there.

Also, if you use labels, run the LABEL utility to place a label on the new disk.

This first example used the most basic of ZFMT207 commands.

Let's do another with a switch.

LOW DENSITY FM 5-1/4" DISKETTE

Assuming drive A is our 5.25" drive, to create a low density, 160K disk, place a diskette in drive A and type the command:

ZFMT207 A:/L

The A: tells the computer that we will be using drive A: for this operation and switch /L will use the archaic FM mode for the operation. The computer responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
LOW Den 48tpi 5 1/4" Drive, VERIFY=Aftr Fmt;
Using 250K xfer, FM mode, 40trks,
04spt, 512bps, 2sides, 1base
4 RootDir(64 entries), 1spc, 02 fats,
1 spf, 01 reserved, 0FEh MediaID
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Formatting Track xxx Side x...

After the format, the computer begins the verification:

Verifying Track(Side) @ Sector xxxx

When complete, the computer displays:

Disk successfully created.
Found x new bad clusters.

And you are returned to the DOS prompt. ZDIR would show 157Kb free at this point. After you ran SYS and loaded the Z-DOS v4 system files, you would be down to about 34Kb free! Isn't progress in technology great!

Ok, enough fun. Let's try some more practical examples.

HIGH DENSITY 3-1/2" DISKETTE

Assuming drive B is our 3.5" drive, to create a standard high density, 1.4Mb disk, place a diskette in drive B, and type the command:

ZFMT207 B:

The computer responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
DUL Den 135tpi 3 1/2" Drive, VERIFY=Aftr Fmt;
Using 500K xfer, MFM mode, 80trks,
18spt, 512bps, 2sides, 1base
14 RootDir(224 entries), 1spc, 02 fats,
9 spf, 01 reserved, 0F0h MediaID
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Formatting Track xxx Side x...
Verifying Track(Side) @ Sector xxxx
Disk successfully created.
Found x new bad clusters.

And you are returned to the DOS prompt. ZDIR would show 1424Kb free at this point.

While this high density disk is in the drive, what would happen if we tried formatting a low density disk? Let's see...

ZFMT207 B:/A0

The computer responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
DUL Den 135tpi 3 1/2" Drive, VERIFY=Aftr Fmt;
Using 250K xfer, MFM mode, 80trks,
09spt, 512bps, 2sides, 1base
7 RootDir(112 entries), 2spc, 02 fats,
3 spf, 01 reserved, 0F9h MediaID
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Verify Bad Sector xxxx, ERROR: in
Bootloader, FAT(s) or Root Directory!

The program figured out that the media was not correct and barfed! The same occurred when I tried the reverse - formatting a double density disk to high density.

Note: When the program is checking Media Type, it is checking to see if the diskette had been previously formatted. If no switches are given and the program can not find the parameters of a previous format, such as when they were destroyed in a manner similar to the above experiment, the program will get as far as, "*Verifying Disk in Drive, Checking Media Type....*" and display the error:

ERROR: Must use /A switch with Unformatted
/Unknown Media in Dual Density Drive

and return you to the DOS prompt.

Affect of Sector Size on Disk Capacity

Here is a question for you. It's fairly common knowledge that small files waste less space if the sector size is small, right?

If you fill a disk having 1024 bytes per sector with a bunch of 100 byte files, you are going to waste a lot of space, because the minimum space the 100 byte file can fill is one sector - or 1Kb of space.

If the disk had 512 bytes per sector, you could theoretically place twice as many 100 byte files on it, because each file is only using 512 bytes. If the disk had 256 byte sectors... well, you get the point.

Why not use a smaller number of bytes per sector than the standard 512? Well, let's try 128 byte sectors on our high density diskette by using the /X0 switch, and we will even save more space by using only one FAT (the /F switch):

ZFMT207 B: /F/X0

The computer responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
DUL Den 135tpi 3 1/2" Drive, VERIFY=Aftr Fmt;
Using 500K xfer, MFM mode, 80trks,
52spt, 128bps, 2sides, 1base
28 RootDir(112 entries), 1spc, 01 fats,
128 spf, 04 reserved, 0F8h MediaID
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Formatting Track xxx Side x...
Verifying Track(Side) @ Sector xxxx
Disk successfully created.
Found x new bad clusters.

And you are returned to the DOS prompt. Any guess what ZDIR would show for the disk's free memory? Remember, for our standard disk, it was 1424Kb free at this point.

Well, if you guessed smaller, you would be right. It shows 1020Kb free!! Why?

Well, each sector has to have a gap between it and the next for separation. Due mostly to drive rotational speed tolerances, head tolerances, and other factors, such as interleave, several bytes separate one sector from the next. This becomes wasted space that becomes a major factor when the sector size is so small!

If we selected 1024 byte sectors per track by using the /X3 switch, the parameters become:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
DUL Den 135tpi 3 1/2" Drive, VERIFY=Aftr Fmt;
Using 500K xfer, MFM mode, 80trks,
10spt, 1024bps, 2sides, 1base
4 RootDir(128 entries), 1spc, 01 fats,
3 spf, 01 reserved, 0F8h MediaID
Hit RETURN to continue, or any key to abort.

And when the formatting was completed, ZDIR would show 1592Kb free at this point. So, you can see why the selection of 512Kb for sector size is a trade-off.

DOUBLE DENSITY 3-1/2" DISKETTE

Assuming drive B is our 3.5" drive, to create a standard double density, 720Kb disk, place the appropriate double density diskette in drive B and type the command:

ZFMT207 B:/A

Remember, the /A0 switch really isn't necessary, as the program will use the previous format parameters from the disk, or it will default to the low density setting if the disk has not been formatted before. The computer responds with:

Verifying Disk in Drive, Checking Media Type....
Will perform LOW LEVEL format on:
DUL Den 135tpi 3 1/2" Drive, VERIFY=Aftr Fmt;
Using 250K xfer, MFM mode, 80trks,
09spt, 512bps, 2sides, 1base
7 RootDir(112 entries), 2spc, 02 fats,
3 spf, 01 reserved, 0F9h MediaID
Hit RETURN to continue, or any key to abort.

When you press {RETURN}, the computer responds with:

Formatting Track xxx Side x...
Verifying Track(Side) @ Sector xxxx
Disk successfully created.
Found x new bad clusters.

And you are returned to the DOS prompt. ZDIR would show 713Kb free at this point.

Media Supported by DOS Version

This is a complete list of the media types supported by our previous DOS versions and the ZFMT207 command, with appropriate switches, required to create each disk.

Double stepping 40 track media in an 80 track device would also require the /T switch.

"x:" is the drive you want to format.

--Z-DOS--

96 tpi, double density, double side <0FDh>

```
DR5D8 LABEL  BYTE
      DW     512     ; Sector size
      DB     4       ; Cluster factor
      DW     1       ; Reserved sectors
      DB     2       ; # of FAT's
      DW     144     ; # of dir entries
      DW     80*8*2  ; # of physical
                     sectors (1280)
```

ZFMT207 x: /08/Y13

48tpi, double density, single side <0FEh>

```
DR5S4 LABEL  BYTE
      DW     512     ; Sector size
      DB     1       ; Cluster factor
      DW     1       ; Reserved sectors
      DB     2       ; # of FAT's
      DW     64      ; # of directory entries
      DW     8*40    ; # of physical
                     sectors (320)
```

ZFMT207 x: /08/M

48tpi, double density, double side <0FFh>

```
DR5D4 LABEL  BYTE
      DW     512     ; Sector size
      DB     2       ; Cluster factor
      DW     1       ; Reserved sectors
      DB     2       ; # of FAT's
      DW     112     ; # of directory entries
      DW     8*40*2
      ; # of physical
                     sectors (640)
```

ZFMT207 x: /08

8" double density, double side

```
DR8D2  LABEL BYTE
       DW    1024    ; Sector size
       DB    1       ; Cluster factor
       DW    1       ; Reserved sectors
       DB    2       ; # of FAT's
       DW    192     ; # of directory entries
       DW    77*8*2  ; # of physical
                     sectors (1232)
```

ZFMT207 x:

8" single density, single side

```
DR8S1  LABEL BYTE
       DW    128     ; Sector size
       DB    4       ; Cluster factor
       DW    4       ; Reserved sectors
       DB    2       ; # of FAT's
       DW    104     ; # of directory entries
       DW    77*26   ; # of physical
                     sectors (2002)
```

ZFMT207 x: /L (Single Sided Media)
ZFMT207 x: /L/M (Double Sided Media)

--VER2 & VER3--

48tpi double density, single side (160K)

```
BPB0   LABEL NEAR
       DW    512     ; BPB_SECSZ
       DB    1       ; BPB_SPAU
       DW    1       ; BPB_RES
       DB    2       ; BPB_NFATS
       DW    64      ; BPB_DIRENTS
       DW    8*40    ; BPB_SECS(320)
       DB    0FEh    ; BPB_MBYTE
       DW    1       ; BPB_FATSECS
```

ZFMT207 x: /O8/M

48tpi extended density, single side (180K)

```
BPB0E  LABEL NEAR
       DW    512     ; BPB_SECSZ
       DB    1       ; BPB_SPAU
       DW    1       ; BPB_RES
       DB    2       ; BPB_NFATS
       DW    64      ; BPB_DIRENTS
       DW    9*40    ; BPB_SECS(360)
       DB    0FCh    ; BPB_MBYTE
       DW    2       ; BPB_FATSECS
```

ZFMT207 x: /M

48tpi double density, double side (320K)

```
BPB1   LABEL NEAR
       DW    512     ; BPB_SECSZ
       DB    2       ; BPB_SPAU
       DW    1       ; BPB_RES
       DB    2       ; BPB_NFATS
       DW    112     ; BPB_DIRENTS
       DW    8*40*2  ; BPB_SECS(640)
       DB    0FFh    ; BPB_MBYTE
       DW    1       ; BPB_FATSECS
```

ZFMT207 x: /O8

48tpi extended density, double side (360K)

```
BPB1E  LABEL  NEAR
       DW     512    ; BPB_SECSZ
       DB     2      ; BPB_SPAU
       DW     1      ; BPB_RES
       DB     2      ; BPB_NFATS
       DW     112    ; BPB_DIRENTS
       DW     9*40*2 ; BPB_SECS(720)
       DB     0FDh   ; BPB_MBYTE
       DW     2      ; BPB_FATSECS
```

ZFMT207 x:

96tpi double density, double side (640K)

```
BPB3   LABEL  NEAR
       DW     512    ; BPB_SECSZ
       DB     4      ; BPB_SPAU
       DW     1      ; BPB_RES
       DB     2      ; BPB_NFATS
       DW     144    ; BPB_DIRENTS
       DW     8*80*2 ; BPB_SECS(1280)
       DB     0FBh   ; BPB_MBYTE
       DW     1      ; BPB_FATSECS
```

ZFMT207 x: /O8

96tpi extended density, double side (720K)

```
BPB3E  LABEL  NEAR
       DW     512    ; BPB_SECSZ
       DB     4      ; BPB_SPAU
       DW     1      ; BPB_RES
       DB     2      ; BPB_NFATS
       DW     144    ; BPB_DIRENTS
       DW     9*80*2 ; BPB_SECS(1440)
       DB     0F9h   ; BPB_MBYTE
       DW     2      ; BPB_FATSECS
```

ZFMT207 x: /D9/K2

8" double density, double side

```
BPB2   LABEL  NEAR
       DW     1024   ; BPB_SECSZ
       DB     1      ; BPB_SPAU
       DW     1      ; BPB_RES
       DB     2      ; BPB_NFATS
       DW     192    ; BPB_DIRENTS
       DW     77*8*2 ; BPB_SECS(1232)
       DB     0FEH   ; BPB_MBYTE
       DW     2      ; BPB_FATSECS
```

ZFMT207 x:

8" single density, single side, 4 reserved

```
BPB4   LABEL  NEAR
       DW     128    ; BPB_SECSZ
       DB     4      ; BPB_SPAU
       DW     4      ; BPB_RES
       DB     2      ; BPB_NFATS
       DW     104    ; BPB_DIRENTS
       DW     77*26  ; BPB_SECS(2002)
       DB     0FDH   ; BPB_MBYTE
       DW     6      ; BPB_FATSECS
```

ZFMT207 x: /L (Single Side)
ZFMT207 x: /L/M (Double Side)

8" single density, single side, 1 reserved

```
BPB4C  LABEL NEAR
       DW    128
       ; BPB_SECSZ
       DB    4
       ; BPB_SPAU
       DW    1
       ; BPB_RES
       DB    2
       ; BPB_NFATS
       DW    68
       ; BPB_DIRENTS
       DW    77*26
       ; BPB_SECS(2002)
       DB    0F8H
       ; *BPB_MBYTE*
       DW    6
       ; BPB_FATSECS
```

ZFMT207 x: /L/R/Y (Single Side)
ZFMT207 x: /L/R/Y/M (Double Side)

--VER4--

The following formats are created by default (no switches) depending on the media type detected (1 or 2 sided, high or low density) and the drive type (40/80 track or 8"). MFM is always assumed. 512 bytes per sector are assumed on 3.5" and 5.25", 1024 bytes per sector on 8" drives.

3.5 Inch

Formatted Capacity	720K	1.44MB
# of Heads (Sides)	2	2
# of Cyls (Tracks)	80	80
# of Sectors/Track	9	18
Total # of Sectors	1440	2880
# Sectors/Cluster	2	1
# Sectors/FAT	3	9
# of FAT Copies	2	2
# of Root Dir Sectors	7	14
# Reserved Sectors	1	1
# of Bytes/Sector	512	512
# Root Dir Entries	112	224
Media Descriptor	F9	F0
Recorded Density MFM	250K	500K

5.25 Inch

Formatted Capacity	360K	720K	1.2MB
# of Heads (Sides)	2	2	2
# of Cyls (Tracks)	40	80	80
# of Sectors/Track	9	9	15
Total # of Sectors	720	1440	2400
# Sectors/Cluster	2	2	1
# of Sectors/FAT	2	3	7
# of FAT Copies	2	2	2
# of Root Dir Sectors	7	7	14
# Reserved Sectors	1	1	1
# of Bytes/Sector	512	512	512
# Root Dir Entries	112	112	224
Media Descriptor	FD	F9	F9
Recorded Density MFM	250K	250K	500K

8 Inch

Formatted Capacity	616K	1.232MB
# of Heads (Sides)	1	2
# of Cyls (Tracks)	77	77
# of Sectors/Track	8	8
Total # of Sectors	616	1232
# of Bytes/Sector	1024	1024
# Sectors/Cluster	1	1
# Reserved Sectors	1	1
# Sectors/FAT	1	2
# of FAT Copies	2	2
# Root Dir Entries	128	192
Media Descriptor	F8	FE
Recorded Density MFM	500K	500K

ERRORLEVEL Exit Codes

To facilitate the use of Batch files to format disks, the following DOS ErrorLevel exit codes are set when ZFMT207 terminates:

255	ErrorLevel Shows aborted by user (^C or not hitting RETURN key)
254	ErrorLevel Shows bad bios version exit
253	ErrorLevel Shows bad drive specified (non Z-207 drive)
252	ErrorLevel Shows Bad or inappropriate media for switches
251	ErrorLevel Shows Bad low level format
250	ErrorLevel Shows Bad sector in bootloader, FAT, or Root Directory
240-249	Reserved for future use
1-239	ErrorLevel Shows # of new bad clusters detected, > 239 = 239.
0	ErrorLevel Shows no new bad clusters were detected.

The following is an example of how to use a batch file and the errorlevel exit variable to keep formatting a disk until no new bad sectors are found.

We use the /N switch to suppress the request for the RETURN key, the /G to use the existing bad clusters in the FAT, and I/O redirection to the NUL device to not display each of the bad sectors found.

Verify Batch File Example

```
@ECHO OFF
ZFMT207 A:/N/V7 >NUL
IF ERRORLEVEL 240 GOTO END
IF NOT ERRORLEVEL 1 GOTO END
:REFORMAT
ZFMT207 A:/C/N/V5/G >NUL
IF ERRORLEVEL 240 GOTO END
IF ERRORLEVEL 1 GOTO REFORMAT
:END
ECHO.
```

As you can see, this batch file does extensive testing on any suspect floppy disk by performing all three verification checks (/V7 switch) as it formats the diskette the first time. Then if one or more bad sector(s) are found, they are added to the existing Bad Sector Table and the program performs the format again.

During subsequent format passes, the program will use **Verify After Format** and **Verify With Write** (/V5 switch) until a format operation is completed with no additional bad sectors found.

Final Note

While testing has been tried on as many media types and switches as drives were available, there are some odd combinations that were missed. So if you find some switch combination that behaves strangely, please report them to the "*Z-100 LifeLine*".

Messages

The following messages have been taken from the source code and are listed here for reference purposes only.

* Z-100 Format Z207 aborted by user.

* Z-100 Format Z207 only works with BIOS version 4.

* Will create CP/M style data disk (FM track 0, MFM rest of disk) on:

* Will perform HIGH LEVEL format (Clear Directory & FAT, Update "PC" Loader) on:

* Will UPDATE "PC" style loader (data is retained) on:

* Which Floppy DRIVE to Format (DOS drive letter)?
  Place DISK to Format in Drive

* Verifing Disk in Drive, Checking Media Type....

* Disk successfully created.
  Found xxxxx new bad clusters.

* Will perform LOW LEVEL format on:

```
LOW Den  48tpi 5 1/4" Drive
LOW Den  96tpi 5 1/4" Drive
LOW Den 135tpi 3 1/2" Drive
HGH Den 135tpi 3 1/2" Drive
DUL Den  96tpi 5 1/4" Drive
DUL Den 135tpi 3 1/2" Drive
HGH Den 135tpi 3 1/2" Drive
HGH Den  96tpi 5 1/4" Drive
     48tpi Eight inch Drive
```

* ERROR: Formatting track, check drive RPM!

* ERROR: Sector #'s 245(F5h)-254(FEh) are invalid, Do you want to continue?

* ERROR: Drive shows Not Ready.

* ERROR: Disk is Write Protected.

* ERROR: Cannot use Unknown Media for Update or Clear.

* ERROR: Media incompatible with switches.

* ERROR: No 96tpi Media in 48tpi drive for Update or Clear.

* ERROR: No Special CP/M Media for Update or Clear.

* ERROR: Must use /A switch with Unformatted/Unknown Media in Dual Density Drive

* Verifying Track(Side) @ Sector xxxxx

* Bad Sector xxxxx,

* skipping special CP/M

* skipping in reserved area

* CLUSTER already marked BAD.

* will be marked BAD in the FAT(s).

* ERROR: Bad Sector Limit exceeded.

* ERROR: in Bootloader, FAT(s) or Root Directory!

* Found xxxxx bad clusters in previous FAT, marking xxxxx new bad clusters.

I hope this article helps explain this complex utility. If you have any questions or comments, please email me at:
z100lifeline@swvagts.com

Cheers,

Steven W. Vagts