# V2BIOS

## Getting Started With V2BIOS for CP/M-80

A Free Software and Documentation Project for the CP/M-80 Environment

0 All software and documentation is in the public domain

0 All software and documentation is in the public domain

**Diskette Drives Supported by V2BIOS and V2FORMAT**

The V2BIOS for CP/M-80 and the V2FORMAT disk format software both support the following Drives and Media:

3.5″ DD PC floppy (720K)
3.5″ HD PC floppy (1.44 Mbytes)
5.25″ mini-floppy (many formats and capacities)
5.25″ HD PC floppy ( 1.2 Mbyte, and 360K capable)
8″ Single and Double Sided, Single and Double Density

Please note that support of a drive and media does not entail support of the MSDOS file system in the case of PC type drives. Throughout this document the terms ‘mini-floppy’ and ‘5″’ drive, (or disk) refers to the older 5.25″ floppy drives that made an appearance, starting around 1975, to about 1984. Exceptions would be 5.25″ drives like the Mitsubishi M4854 which was a 5.25″ emulation of the earlier 8″ drives. Mini-floppy drives used recording frequencies of 125 kHz MF and 250 kHz MFM. Formatted capacities could be as low as 70 kilobytes, and up to 800 kilobytes. They came in packages that were designated as ‘full height’ (3-5/16″), “half-height” and some “one-third height”. They had a spindle rotation of 300 RPM and track-to-track seek times as slow as 20 milliseconds. You may come across this type of drive from time to time, and you may wish to read some of the diskettes supported by these drives. V2BIOS can read a number of them, and you can set it up to read many more. V2FORMAT covers a lot of the formats used on those early drives, so you can produce ‘new’ diskettes and duplicate old formats you discover.

The introduction of the 5.25″ HD drive for the IBM AT in 1984 marked the end of the older mini-floppy concept. The HD drives act like 8″ drives. They use recording frequencies of 250 kHz FM and 500 kHz MFM, and rotate at 360 RPM just like 8″ drives. You get up to 1.2 Mbytes formatted capacity, like a double-sided 8″ drive. They can read and write the old 360K IBM PC ‘standard’ mini-floppy diskettes as well.

The introduction of the 3.5″ 720K drive was a regression. With a 300 RPM spindle speed and data frequencies of 125 kHz MF and 250 kHz MFM, it was a mini-floppy made small. The nice feature was that the media came in a hard case you could fit in a shirt pocket.

Finally, the 3.5″ HD drive was introduced, and it seems to have marked the end of the popular floppy disk concept. Most 3.5″ HD drives are actually “three-mode” drives. They handle 720K media at the same RPM and recording frequencies, and they got back to 8″ drive data frequencies of 250 kHz FM and 500 kHz MFM, giving 1.44 Mbytes capacity at 300 RPM. They can actually run at 360 RPM, giving 1.2 Mbytes capacity, but that seems to be little used. While a 2.88 Mbyte 3.5″ drive was later introduced, it is seldom seen in mass marketed systems.

As far as V2BIOS and V2FORMAT are concerned, the drives are classified as follows:

| Designation | Drive Types | R/W Frequency |
|---|---|---|
| mini-floppy | old mini-floppies | 125 kHz FM 250 kHz MFM |
| mini-floppy | 3.5″ 720K drive | 125 kHz FM 250 kHz MFM |
| HD Drive | HD 3.5″ & 5.25″ | 250 kHz FM 500 kHz MFM |
| HD Drive | 8″ | 250 kHz FM 500 kHz MFM |

Thus, 5.25″ HD and 3.5″ HD drives (with HD media) are treated as 8″ drives. As far as the Versafloppy II controller is concerned, both these drives, without any alterations, are 8″ drives. Even though the 3.5″ drive rotates at 300 RPM, by default, the WD179X controller chip on the Versafloppy II is unaware of this fact. Note that a 3.5″ HD drive with 720K media, is seen as a mini-floppy by V2BIOS and V2FORMAT & the WD179X.

About V2BIOS, V2FORMAT and V2LOADR1 & 2

*V2BIOS*

V2BIOS is a CBIOS for use with the Z3S, D3S, D4S and Versafloppy II floppy disk controller cards and the Digital Research CP/M-80 8-bit operating system. It can be quickly adapted to other FDC boards, and details about this will be covered in the section “Implementing Other FDC Boards”. For this release, however, the software and documentation concentrate on the Versafloppy II (VF-II) as the target controller board.

The minimum requirements to get CP/M-80 operational with the V2BIOS are:

1. An S-100 system containing a VF-II board.
   * The VF-II can be a second FDC in the system.
2. A Z80 CPU board with a clock frequency of 2MHz or higher.
3. An I/O device for communicating with a keyboard and screen (a console, or console emulator).
4. 32K bytes of main memory (48K to 64K preferred).
5. At least one disk drive connected to the VF-II, of 8”, or PC HD floppy of the 3.5” or 5.25” type. You can put any mix of these drives on the VF-II.
6. Some version of CP/M-80 (or work-alike) running on the S-100 host system.

7. The V2BIOS distribution files should be transferred to a diskette that is readable in the host system. If you are not going to customize the V2BIOS at the source code level you could start with just the V2BOOTX and CPMXXK system image files.

V2BIOS disk format characteristics.

Data Tracks

The data, or user tracks can be set to a wide range of data density and sector sizes. You select single or double density, single or double sided, with sector sizes of 128, 256, 512 and 1024 bytes. Depending upon the options selected, diskettes can be prepared which range in size from 70 Kbytes to 1.66 Mbytes. V2BIOS will work with any mix installed in any drive.

System Tracks

The system tracks are of a fixed format (for now), and that format was selected to exhibit the following characteristics:

Single Sided --- a single sided drive can read them
Single Density --- a slow CPU can perform reliable data transfers
Two System Tracks --- compatible with the original CP/M standard
512 byte sectors --- with 9 sectors per track, giving 9,216 bytes for a large CBIOS

NOTE:
V2FORMAT allows you an alternative to this system track structure. With V2FORMAT you can format ‘standard’ diskettes, in which all tracks, including the system tracks, contain the exact same layout. Such diskettes are not ‘Sysgenable’ with the present V2BIOS Sysgen utility.

Advantages of the V2BIOS and V2FORMAT Software

V2BIOS and V2FORMAT operate independently of any system ROM. Some CPU cards have onboard ROM routines that are tied to the CBIOS and Format software supplied with the system. In such cases you are restricted to a maximum memory size for the running CP/M system, and a small subset of format types that can be applied to a diskette. In most cases, the ROM and onboard scratch-RAM on these CPU boards can be switched out, so that, with V2BIOS, you can load a CP/M system which uses all available RAM on your memory card. How to do this will be covered shortly.

**Reading Non-V2BIOS Diskette Formats**

When reading other diskette formats, V2BIOS is set to work with only the SSSD or 3740/1 for 8" formats and one 5" mini-floppy format (but you can add others). When doing so, a message is printed on the console to indicate that a foreign format is being accessed:

STD Disk?

If the drive is designated as an HD drive, V2BIOS then sets up for the 3740/1 format and allows CP/M to check the directory and perform normal read/write operations. If the diskette is not 3740/1 format, the directory read will slow down and a disk error, usually a Record Not Found (RNF) message is printed. If the disk is an old 5" mini-floppy type, and the drive is designated as 5" in the Step Speed Table, V2BIOS sets up for a diskette with the following characteristics:

18 sectors per track
128 bytes per sector
about 78 kilobyte capacity

Internally, the V2BIOS uses the STDDDB structure for 8" SDSD disks, and the ATLDDB structure for the 5" mini-floppy disk.

*Setting V2BIOS to Read Other Formats (optional reading)*

One way to easily read other formats, is to replace the ATLDDB structure with one that describes a common format you want to read often. You would assign a phantom drive select value for it, say CP/M disk "E:". When select disk is called, you must have it modified to test for "E:" (a value of 4 in the C register), set the correct flag byte, and then move that DDB into place for a pre-selected disk drive, usually the last physical drive, so you will make an extended ADRTAB set for that drive (see the section "Implementing Other FDC Boards" for details on ADRTAB). This has been done for the core Z3BIOS, and in fact, phantom drives of any number were implemented, so that many formats could be read. A better way would be to write a utility with many drive formats selectable from a menu, the selected format having an assiociated DDB copied to the ALTDDB area of the V2BIOS, and that may be done yet. A sample of DDBs used for Compupro, Lomas Data Products and Andicom disk formats can be found in the ALTDDBS section.

*V2FORMAT*

V2FORMAT is ready to run under any CP/M-80 system. You just need a VF-II controller installed, and it can be the second controller in a system if you boot from some other controller. Just be sure that the I/O space used by the VF-II does not conflict with other hardware in your system. Ports 63H to 67H are used by the VF-II.

*V2LOADR1 and V2LOADR2*

If you change V2BIOS to work with other system track formats, or if you decide to use a substitute for the DRI CCP (Nova DOS, ZCPR etc.), you should be aware of the different system track loader characteristics.

V2LOADR1

V2LOADR1 expects the full 512 bytes of sector 1, track 0 to be loaded into memory. It then picks up the jump page from the standard DRI CCP, located at byte 130 above the base of sector 1. This page value is then used to calculate the true load address of the system. In this way, one loader can be used for all relocated versions of CP/M.

V2LOADR2

V2LOADR2 must be reassembled for every V2BIOS relocation. It is hard coded to move the rest of the CP/M image into the correct memory starting address. When you assemble V2BIOS, you are prompted for the top of memory to assemble the source addresses for. You must then assemble the V2LOADER2 source, which prompts for the size of the CBIOS, then (3 for the present 3K) and then prompts for the top of CP/M memory, so enter a value that matches that of the V2BIOS you have just assembled. During assembly of V2BIOS you will be given the MOVCPM memory argument, and the relocation value for the V2BIOS that matches it. See the section “Building your own CP/M” for instructions on how to combine the generated files for a new system image. The R, or relocation value for both V2LOADER1 and V2LOADER2 will always be 900, unless you customize the loader address to suit some special needs you want to incorporate into your new system.

Switching Out Onboard RAM and ROM

How the onboard RAM and ROM are switched out during boot up depends upon the scheme implemented on each CPU board. If you are unable to determine how to do this at first, just install a V2BIOS CP/M image that fits below the RAM/ROM area. Once you know the scheme, you can implement it in a number of ways, some of which are.

1. Relocate the bootstrap to main memory, pass control to it, and be sure it is coded to switch out the onboard memory.
2. Bootstrap from ROM, and use a re-coded V2LOADR on the system tracks that performs the switch out.

For the SD Systems SB-200 CPU the following code switches out the RAM/ROM

```
MVI   A,2
OUT   7FH
```

If you use an SBC-200 CPU, an alternate loader V2LD200X.HEX is supplied to combine with the system memory images, so that switching out is accomplished at load time. The loaders are V2LD2001.HEX for auto-system relocation, and V2LD2002.HEX for matching CP/M image relocations. Relocate them by 900 under DDT when making a system image .

*List Device Operation with V2BIOS*

The distributed software is coded such that the CP/M list device always returns a READY value. The list output is a non-functional routine. In this way you will not suffer system hang-ups should you use the printer device or accidentally toggle the list output (control-p at the command prompt). To add printer support you will have to change the source code at the LIST and LISTST entry points.

**Creating A Working System**

*Quick Overview*

1. Patch a ready-to-Sysgen image to suit your console I/O ports.
2. Format a diskette to V2BIOS standards, using V2FORMAT
4. Execute the patched image with the embedded Sysgen program to Sysgen
   the formatted diskette.
4. Execute the V2BOOTX program to load the system tracks and bring up CP/M.

*Detailed Overview*

The steps involved to get the V2BIOS operational are:

1. Install a Versafloppy II into a working S-100 CP/M system
   * You should boot with  a CP/M system.
   * A copy of the V2BIOS distribution files must be on a diskette placed in the system
2. Patch and save a ready to go V2BIOS CP/M image for your console I/O
   * See the section "Patching The Image Files for Console I/O"
   * If your console is connected to an SD Systems SBC-200 USART you need not patch
   * If you must patch, you need only change six entries in an I/O table
3. WRITE PROTECT all diskettes except the target disk to be formatted.
4. Format the diskette using V2FORMAT with the V2BIOS style format
5. Execute the patched image file, it will ask for a drive to SYSGEN
   * Sysgen the diskette that has been V2FORMATed
6. Run the V2BOOTX bootstrap loader to load the system
   * CP/M should sign on and be ready to use

NOTES

* The present V2BIOS works only with a Z80 or Z80 compatible CPU.
* As long as the CPU has a speed of at least 2 MHz, you can use a Double Density
  format.
* When formatting the diskette, be sure to select Single Sided format if you have a
single sided disk drive as the target.

**Sample Installations**

Here, in more detail, are two ways to prepare a system to use V2BIOS

*Using a system with a working Versafloppy II CP/M*

1. Transfer the V2BIOS *.COM and *.HEX files onto a working diskette
2. With DDT load an image file and patch your console I/O.

See the section "Patching An Image File for Console I/O"

If you have a CPU with onboard RAM/ROM, and you are not able to switch it out, or if it is always available, due to a BIOS that uses routines in the ROM, select the 32K or 48K V2BIOS image to be sure the installed CP/M will not overlay the RAM/ROM area.

3. Save the patched memory image by exiting to the command prompt and typing:

SAVE 41 CPM480.COM - if you used the 48K image file. Add 0 at the end of the name, to make a new file. That way, if you made a bad patch, the original file remains.

4. WRITE PROTECT all but the target diskette

5. Execute V2FORMAT and make a fresh disk in V2BIOS format on the target diskette

6. Now execute CPM48K0 file by typing the saved named at the command line.

You will be prompted for a disk drive, from A to D for the SYSGEN operation. Enter the drive letter, where A is the first drive, and on up to D the fourth drive.

7. Depending upon the drive you selected, run a matching V2BOOTX program, where X is from 1 to 4 for disk drive 1 to 4. A number of 48K system images are available, from CPM48K to CPM48KD, for correct test boots from drive 0 to drive 3.

NOTE:

Each V2BOOTX program delays for a while, and allows you to enter a drive letter from A to D, if needed, so that you can switch to a drive with the formatted and SYSGENed diskette. In fact, you could do all this with a one drive system, by switching diskettes from/to the drive, as each operation waits for a prompt. If you do not change the destination drive, V2BOOTX waits for up to 20 seconds before booting, again, giving you time to swap diskettes.

*Using a CP/M system with some other floppy controller plus a Versafloppy II*

You can install a Versafloppy II into a working CP/M system that uses some other controller if there are no port conflicts. The VF-II uses ports 63H to 67H inclusive. If your present system does not use these ports, you can install a VF-II with one disk drive attached as the first drive and proceed as follows:

1. Transfer the V2BIOS files onto a readable diskette for your present system.

If you have a CPU with onboard RAM/ROM, and you are not able to switch it out, or if it is always available, due to a BIOS that uses routines in the ROM, select the 32K or 48K V2BIOS image to be sure the installed CP/M will not overlay the RAM/ROM area. See the Console I/O Patch Note for details on patching the BIOS for your console device.

2. If you must patch console I/O, load an image file with DDT and patch.

3. Save the patched memory image by exiting to the command prompt and typing:

SAVE 41 CPM48K0.COM - if you used the 48K image file

4. WRITE PROTECT all but the  target diskette

5. Execute V2FORMAT and make a diskette in V2BIOS format on the target diskette

6. Now execute CPM48K0 file by typing the saved named at the command line.

You will be prompted for a disk drive, from A to D for the SYSGEN operation. Enter the drive letter, where A is the first drive, and on up to D the fourth drive. This is referenced to the strapping for the drive in the VF-II, and that would usually be the first, or A drive.

7. Depending upon the drive you selected, run a matching V2BOOTX program, where X is from 1 to 4 for disk drive 1 to 4. A number of 48K system images are available, from CPM48K to CPM48KD, for correct test boots from drive 0 to drive 3.

NOTE:

Each V2BOOTX program delays for a while, and even allows you to enter a drive letter from A to D, if needed, so that you can switch to a drive with the formatted and SYSGENed diskette. In fact, you could do all this with a one drive system, by switching diskettes from/to the drive, as each operation waits for a prompt, except the V2BOOTX, which, if you do not change the destination drive, waits for up to 10 to 20 seconds before booting, giving you time to swap diskettes.

*DRIVE ASSIGNMENT OPTIONS*

The 48K CP/M image comes in 10 types:

CPM48.COM  or (CPMSB48.COM)  for FIRST physical drive (switches out SBC-200 RAM/ROM)
CPM48B.COM or (CPMSB48B.COM) for SECOND physical drive (switches out SBC-200 RAM/ROM)
CPM48C.COM or (CPMSB48C.COM) for THIRD physical drive (switches out SBC-200 RAM/ROM)
CPM48D.COM or (CPMSB48D.COM) for FOURTH physical drive (switches out SBC-200 RAM/ROM)
CPM48M         set to map drive D: to drive B: for reading mini-floppy formats (D:)
CPM48MF.COM    set to sysgen and boot from a mini-floppy as A: drive

A 48K image was used as it gives you reasonable working space, while being sure to fit under any RAM/ROM that may be onboard the CPU card.

The images allow you to apply a SYSGEN to the target drive and boot it up without swapping diskettes. Suppose your target diskette is in the third floppy drive, which could be the "C" drive, under CP/M. Then use the CPM48KC.COM image, once you have patched it for your console I/O, execute the image file and Sysgen the third, or C: drive with it. When you are ready to bootstrap, run V2BOOT3. The third drive boots, and CP/M should sign on and use the third drive as drive "A". In this way warm boots will not incorrectly select the first drive, which in this case may be the host drive you are executing the installation software from. The drive assignments for each image

| System Image | | Drive 0 | Drive 1 | Drive 2 | Drive 3 |
|---|---|---|---|---|---|
| CPM48 | or CPMSB48 | A | B | C | D |
| CPM48B | or CPM48B | B | A | C | D |
| CPM48C | or CPM48C | B | C | A | D |
| CPM48D | or CPM48D | B | C | D | A |
| CPM48M | | A | B | C | D/B D as 5″ mapped to B physical |
| CPM48MF | | A (5″) | B | C | D |

7 All software and documentation is in the public domain

Software

*The V2FORMAT Floppy Diskette Formatter Program*

The diskette formatter program has been written to give you maximum control over the type of format you can apply to any diskette. While the format selection menu is comprehensive, you can bypass it completely after the first time through by selecting the "Save" option. This option allows you to make a new copy of the program for any of your favorite format types, so that when you run it again, all the previously selected options will be remembered, and you need only type the letter of the disk drive containing the media to be formatted. Note however, that even this new version of the program allows you to re-execute the menu options, so that you are not permanently locked into just one format.

Here is the opening screen when V2FORMAT is executed:

V2FORMAT Disk Format

V 1.0.0

for VF-II disk controller

NOTE:

3.5" and 5.25" HD PC drives are supported as 8" drives.

All selectable 5" formats are for older mini-floppy drives.

Will disk be V2BIOS compatible or Standard format ('V','S'):

In order to work with the V2BIOS 'version' of CP/M supplied with the distribution diskettes, you must select the 'V' option. This option ensures that no matter what format the data tracks are to be, the system tracks will always be two single-sided, single-density tracks with nine sectors of 512 bytes each. Details covering this structure were covered in a previous section. By selecting the 'S' option, you will format all tracks, including the system tracks, with the media options presented to you in the menu selections. In this way you should be able to create a format that matches most of the CP/M media types ever created for floppy diskettes. You may find this useful to reconstruct a format type for older diskettes you have, or to assist others with CP/M computers who need software transferred to or from their particular media.

Here are the menu items that follow. Each menu line is displayed in sequence. You must make a selection based on the acceptable options. At any point you can type in a 'Q' option (not shown), to quit the menu an return to the command prompt.

1. format all tracks or format system tracks only (A,S):

2. 5 inch disk or 8 inch disk (5,8):

3. single sided disk or double sided disk (S,D)

4. single density disk or double density disk (S,D)

5. sector size of 128 (1) , 256 (2), 512 (3) , or 1024 (4)  bytes (1,2,3,4):

6. sectors per track 9 (1),10 (2), or 11 (3) - 2 & 3 for 3.5" HD drives (1,2,3)

7. Do a read-verify during format (Y,N)

8 All software and documentation is in the public domain

Type disk letter to format
or X to change options, S to save format or Q to stop (A, B, C, D, X, S, Q):

NOTES

Item (1) can be useful if your system tracks have been damaged. You can safely format them, and leave the data tracks fully intact, thus ensuring access to any data is unhampered. For the V2BIOS, this option places the Disk Descriptor Block (DDB) back in place, so that the V2BIOS can pick up the diskette structure.

For item (2), remember to type in an '8' for high density (HD) PC drives of 5.25" or 3.5" form factor. If you are formatting for old mini-floppies from CP/M systems, or the original low capacity drives used on the IBM PC, select '5'.

Select item (3) as your needs demand.

Same with item (4).

Item (5) helps you match any sector type used for CP/M systems. For the V2BIOS select 1024 byte sectors to get maximum capacity and best disk access times. You are not obligated to do this, since the V2BIOS will work with ANY of the selected options, and for any mix in your floppy drives. That is, you could run V2BIOS with any mix of formats placed all in possible drives on your system, providing they are V2BIOS compatible in format.

Item (6) will only appear if you chose an 8" disk (2) with 1024 byte sectors (5). It just gives you a bit more useable space. Nine sectors gives 1.3 Mbytes, and if you have 3.5" HD PC dives 10 sectors gives 1.5 Mbytes and 11 sectors gives 1.6 Mbytes useable space.

Item (7) should be a 'Y' in most cases, to ensure the diskette is in good condition after each track is formatted. You could select 'N' in cases where you simply want to format a diskette, no matter what it's condition.

Mini-Floppy Formats

The mini-floppy formats are the same as when the core format software was created. If you need more tracks etc, you should look at the source just past DOUSER:. You will see the maximum track value was selected by the diskette size check (76 tracks for 8”, 35 tracks for 5”). The value was saved at MAXTRK for later compare. Note also, if you format 5.25” HD drives for media to insert in older 40 TPI drives, you should call the routine STEPIN twice for each track. You can expand the format query section to include this additional information when a 5” disk size is selected. Use the GETOK routine to implement the user interface on multiple-choice options.

8” and 8” Emulation Formats

8”, 3.5” & 5.25” HD PC drives all have 77 tracks formatted. This avoids using too many formats for 8” emulation, and stops the 8” drive carriage from hitting the ending crash stop if 80 tracks were used.

*The V2SASG Program and V2SASG5*

When you execute a patched image, the Versafloppy Stand Alone Sysgen program (V2SASG) takes over. This Sysgen program does not require the V2BIOS to be operational when it executes. Here is how it looks in operation:

Sign on-

BE SURE A DISK IS ALREADY IN PLACE

AS THE SYSGEN STARTS ONCE THE DRIVE IS TYPED IN

SYSGEN TO DRIVE A, B, C OR D ENTER CHOICE =>

Action ->Type in a drive letter from A to D, corresponding to physical drive 0 to 3.

If the Sysgen is successful, the sign off message is printed -

SYSGEN COMPLETE

If you Sysgen the A, or first drive, V2SASG prints a sign off message that lets you know the processor has been halted so that a reboot is needed. This is done to ensure that a Sysgen made to the boot drive will not be used right away, as it could be located for a different memory size. Thus the system needs a reboot to cover the change of warm-boot and BDOS entry vector.

DRIVE "A" WAS UPDATED - HALTING FOR A REBOOT :

If a physical disk failure is encountered during Sysgen, you will get one of the following messages-

FAILED ON SECTOR READ, QUITING

or

FAILED ON SECTOR WRITE, QUITING

Followed by-

EXITING TO COMMAND PROMPT

These messages indicate a bad sector on a system track. You can run V2FORMAT, select only (S)ystem tracks to format, and restore those tracks. No files will be lost on the target diskette.

If instead of entering a drive from A to D, you enter Q, for Quit, the above message is also printed.

You can integrate V2SASG with new images you make, using DDT. Use no relocation value. If making a drive supporting a mini-floppy boot, use V2SASG5. In that case be sure the drive you Sysgen to is set up as a mini-floppy in the V2BIOS.

*The V2BOOTx Utility*

The V2BOOTx utility has four forms, V2BOOT1 to V2BOOT4. This corresponds to a hard coded boot of physical drive 0 to 3. Once you have a V2BIOS formatted diskette with a V2BIOS CP/M image SYSGENed onto the system tracks, you need a V2BOOTx program to bootstrap the system. A sample position-independent bootstrap program is also supplied, with a listing given in this document. You can use it to incorporate into a system monitor ROM for boot up.

Here are the messages issued when V2BOOTx is executed, in this case V2BOOT1 for the first drive:

```
WAIT FOR TIMED BOOT FROM FIRST DRIVE
OR ENTER 1 OR 2, 3 OR 4 TO SELECT DRIVE
```

If you take no action there is a delay, giving you time to select some other boot drive, and thus over ride the default, or simply wait for the boot. Selecting a number, even the default, skips the delay and the next phase is executed.

Here is the next phase message-

```
LOADER STARTS IN ABOUT 5 TO 20 SECONDS (8mhz to 2mhz system)

OR PRESS ANY KEY WHEN READY TO START BOOT
```

Once again you can force a faster boot by typing any key. At time-out the selected drive is bootstrapped, and CP/M logs to drive A.

If you select Q, instead of 1 to 4, the bootstrap quits to a command prompt with the message-

```
EXITING TO COMMAND PROMPT
```

Just remember, if you want to boot from a drive other than the first, you must have a Sysgened system on that drive that treats it as the system drive. Only the 48K CP/M images have drive variable options. See page 6 for details.

*The V2COPY  Disk Backup Program*

V2COPY will only work with a CP/M booted with the V2BIOS. The V2COPY program makes disk to disk copies of any two diskettes that have the same physical format. Since there are many format options available with the V2FORMAT program, V2COPY will check the Disk Descriptor Block on each diskette to ensure they match. The default situation is a copy from A to B, but you can enter any other combination. Here are the messages during execution:

```
V2COPY  Disk Backup V 1.1

            OPTIONS

1- Press the RETURN key to...
```

Copy ALL information from drive A to the diskette in drive B.

```
2- Press “D” to...
```

Select some other drive to copy from/to

```
3- Type “Q” to...
```

QUIT this program now.

Enter the options you want to work with, and the following prompting message is printed-

```
Insert your LATEST disk in drive A and your OLDEST backup disk in drive B.
OK to continue (Y/N) ?
```

Enter a Y and the copy operation starts.

Each copied track is read back and compared byte for byte with the original data from the source diskette. Any error is printed on the console and you are prompted to try again or quit. Note that before any error warning is printed, the disk operation is retried several times within the V2BIOS. Thus you can be sure that any error reported is quite severe.

**Description of Supplied Files**

*ASM Source Files*

All ASM source files are supplied in TDL Z80 dialect. See the section on Source File Assembly for details.

| | |
|---|---|
| V2BIOS.ASM | The CP/M BIOS |
| V2BOOT1.ASM | bootstrap for drive 0, or A as default |
| V2BOOT2.ASM | bootstrap for drive 1, or B as default |
| V2BOOT3.ASM | bootstrap for drive 2, or C as default |
| V2BOOT4.ASM | bootstrap for drive 3, or D as default |
| V2BOOTX.ASM | minimal bootstrap sample |
| V2BOOTX5.ASM | minimal bootstrap sample for mini-floppy |
| V2COPY.ASM | disk to disk copy, works only with V2BIOS resident |
| V2FORMAT.ASM | diskette formats |
| V2SASG.ASM | stand alone Sysgen for V2BIOS type system tracks |
| V2SASG5.ASM | stand alone Sysgen for V2BIOS type system tracks, mini-floppy |
| V2SYSGEN.ASM | requires V2BIOS resident |
| V2LOADR1.ASM | Image independent system track loader |
| V2LOADR2.ASM | Location dependant system track loader |
| V2LD2001.ASM | SBC-200 use like V2LOADR1, but switches off SBC-200 RAM/ROM |
| V2LD2002.ASM | SBC-200 use like V2LOADR2, but switches off SBC-200 RAM/ROM |

*HEX - BIOS and Sysgen*

| | |
|---|---|
| V2BIOS32.HEX | 32K CP/M V2BIOS |
| V2BIOS48.HEX | 48K CP/M V2BIOS |
| V2BIOS64.HEX | 64K CP/M V2BIOS |
| V2SASG.HEX | stand alone Sysgen |

*COM - Utility*

| | |
|---|---|
| V2BOOT1.COM | Executable bootstrap for drive 0, or A as default |
| V2BOOT2.COM | Executable bootstrap for drive 0, or A as default |
| V2BOOT3.COM | Executable bootstrap for drive 0, or A as default |
| V2BOOT4.COM | Executable bootstrap for drive 0, or A as default |
| V2FORMAT.COM | Diskette Formats |
| V2SYSGEN.COM | Executable BIOS dependant Sysgen |

*COM - CP/M Ready to Patch and Sysgen Image Files*

| | | | |
|---|---|---|---|
| CPM32.COM | or | CPMSB32 | 32K CP/M with drive 0 as A |
| CPM48.COM | or | CPMSB48 | 48K CP/M with drive 0 as A |
| CPM64.COM | or | CPMSB64 | 64K CP/M with drive 0 as A |
| CPM48B.COM | or | CPMSB48B | 48K CP/M with drive 1 as A |
| CPM48C.COM | or | CPMSB48C | 48K CP/M with drive 2 as A |
| CPM48D.COM | or | CPMSB48D | 48K CP/M with drive 3 as A |
| CPM48M.COM | | 48K CP/M | maps drive D: to drive B: for mini-floppy use |
| CPM48MF.COM | | 48K CP/M | Drive A: is a mini-floppy |

All images are set to use the SBC-200 USART for console I/O. The USART must already be initialized for some baud rate used by your console. The “SB” images have loaders that switch off the onboard SBC-200 RAM/ROM. With CPM48M you can use drives A, B & C for 8” use, then connect a mini-floppy as second drive and access it as D:.

*Other*

| | |
|---|---|
| V2XIOS60.ASM | XIOS source for MP/M 1.1 or MP/M 2.1 |
| V2MPMLDR.ASM | Source for MP/M system loader BIOS |
| V2MPMLDR.HEX | HEX for above for insertion in DRI MP/M Loader |

MP/M offers a lot of good features, but the program TPA is reduced to 48K in most cases. However, if you have banked memory, you can run up to seven concurrent tasks under MP/M 2.1, and eight concurrent tasks under MP/M 1.1. In addition, you can connect up to 16 consoles, or with one console, you can run up to the maximum number of tasks supported my your available memory. Under MP/M 2.1 up to 16 printers are supported, but only one is supported under MP/M 1.1.

The source files are included here for interest only. By examining them you can note how CP/M and the BIOS structure evolved into an effective multi-tasking OS, while remaining compatible with CP/M-80. One thing you will need is an interrupt source to effect task switching. You can actually run MP/M without a task switching interrupt, but if a program becomes compute-bound, in a permanent loop, the entire system locks up.

The XIOS (as the BIOS is called under MP/M) has many assembly-time options. Features include:

Standard PIO operation of the VF-II
DMA operation of the VF-II (you have to add a simple Z80-DMA add-on)
FDC interrupts for end-of-disk-operation signaling

No DRI MP/M distribution files are included, as they are available from a number of sources on the Web. You should also pick up the manual sets.

V2BIOS Assembly Notes

Use ZASM to assemble a new BIOS as follows:

ZASM V2BIOS O - letter "O" for no PRN listing
ZASM V2BIOS D - for disk file PRN listing

When executed you will be prompted for one input variable, the size of the CP/M system (Top of memory) to create the V2BIOS.HEX file for.

ZASM V2BIOS O<CR>

TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21

Here is the input prompt-

ENTER TOTAL MEMORY SIZE IN KILOBYTES EG: 48 OR 64 :64 <-- 64 entered

As assembly nears the end you will see a list on the console which gives you all the details you need to know for MOVCPM, V2BIOS.HEX file relocation and notes covering memory reserves and system track resource usage.

1055 BYTES OF SYSTEM TRACKS LEFT
8161 BYTES TOTAL BOOT TRACKS SPACE NEEDED
16   OF 18, 512 BYTE SECTORS USED ON SYSTEM TRACKS
32   BYTES OF MEMORY LEFT BY BIOS
D800 IS START OF CCP
E006 IS START OF BDOS
EE00 IS START OF BIOS
F678 IS LAST BIOS RUNTIME CODE OR STATIC DATA LOCATION
F6E1 IS END OF DISCARDED INITIALIZATION CODE
261C IS CIOTB START IN SYSGENABLE MEMORY IMAGE
2407 BYTES IS WORKSPACE USED BY BIOS
FFDF IS LAST LOCATION USED
FFE0 IS 1st FREE ADDRESS AFTER BIOS
3180 IS BIOS OFFSET      <-- use this value when relocating the BIOS image
61   IS MOVCPM VALUE     <-- use this with MOVCPM to set the image page

NOTE

If you modify the V2BIOS and consume additional memory, beyond the memory-size argument, you will see the following message during assembly:

!! ** BIOS EXCEEDS MEMORY SIZE BY XX BYTES, SET 'BIOSIZ' LARGER ** !!

Where XX will be the amount of ram beyond the assembly size (64K in this example). Re-edit V2BIOS and set BIOSIZ to the next larger size. Increase it by 1 for every additional kilobyte, or part thereof, that it exceeds the original 3K size. Note, however, that the present size leaves only 1055 bytes of system tack space free, so you are limited as to how much you can increase the size. This refers only to code and static data, which is placed on the system tracks. You can add as much additional work space as you wish. As the 1K increments are added, the V2BIOS starts lower in memory and thus TPA is reduced 1K for each increment.

Looking at Assembly Time Errors

Generate a .PRN file and look through it for all “?” that sit on any source line with an error.

**Building Your Own CP/M**

Suppose you modify the V2BIOS source to add features, or just need it relocated to provide an upper memory buffer space for some special application or to fit under some RAM or ROM on a board plugged into your system. You now must combine a few files to make a new image ready for Sysgen. Here is a sample case:

1. Assemble the V2BIOS

```
ZASM  V2BIOS 0

TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21

ENTER TOTAL MEMORY SIZE IN KILOBYTES EG: 48 OR 64 :62 <-- set for 62K top RAM

1055 BYTES OF SYSTEM TRACKS LEFT
8161 BYTES TOTAL BOOT TRACKS SPACE NEEDED
16   OF 18, 512 BYTE SECTORS USED ON SYSTEM TRACKS
32   BYTES OF MEMORY LEFT BY BIOS
D000 IS START OF CCP
D806 IS START OF BDOS
E600 IS START OF BIOS
EE78 IS LAST BIOS RUNTIME CODE OR STATIC DATA LOCATION
EEE1 IS END OF DISCARDED INITIALIZATION CODE
261C IS CIOTB START IN SYSGENABLE MEMORY IMAGE
2407 BYTES IS WORKSPACE USED BY BIOS
F7DF IS LAST LOCATION USED
F7E0 IS 1st FREE ADDRESS AFTER BIOS
3980 IS BIOS OFFSET    <--- write down the new BIOS offset
59    IS MOVCPM VALUE  <--- write down the MOVCPM value
```

2. You should rename the V2BIOS.HEX file. Under CP/M type-

```
REN V2BIOS62.HEX=V2BIOS.HEX<CR>
```

3. Now run MOVCPM to make a relocated image, using the value 59, supplied by the assembler-

```
MOVCPM 59 *<CR>  <-- remember to include an '*' as the second argument

MOVCPM reports as follows-

CONSTRUCTING 59K CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 34 CPM59.COM"
```

At the command prompt type -

4. SAVE 34 CPM59.COM<CR>

5. Now load the image under DDT by typing-

```
DDT CPM59.COM<CR>
```

DDT responds with-

```
DDT VER 2.2
NEXT PC 2300 0100
```

6. Next read in the new V2BIOS.HEX with the correct offset, type two commands-

```
IV2BIOS59.HEX<CR>   I = Insert, followed by file name, no space
R3980<CR>           R = relocation factor, followed by value, no space
```

DDT responds with-

```
NEXT PC
2807 0000  <--- Note that 2807 can change, it depends upon the actual BIOS size
```

7. Now read in the system track loader as follows-

```
IV2LOADR1.HEX<CR>
R900<CR>  <--- relocate to 900 (a hex value, the H not needed)
```

DDT responds with

```
NETX PC
2807 0000 <--- the previous values  should not change.
If they do, a relocation error has occurred.
You must exit and try again from step (5), making sure
you type in the correct values.
```

8. Now read in the V2SASG.HEX file for later Sysgen-

```
IV2SASG.HEX<CR>
R<R>      <--- No relocation needed
```

9. Now return to the command prompt by entering a <CONTROL-C> or typing-

```
G0<CR>    <--- The go to 0 (a ZERO), warm boot command
```

10. Now you must save the combined image to include all modules.

In this example the 2807 indicates that page 28H is the number of  pages (256 byte chunks) of image to save. While the combined modules exceed page 28H by 07 bytes, the offset is already 256 bytes, as everything is loaded at 0100H. A simple hex conversion gives 28H as 40 decimal, so type the following-

```
SAVE 40 CPMSYS59.COM<CR>
```

You can now erase the CPM59.COM file if you like.

You are now ready to Sysgen to a prepared diskette. Simply type the saved system file name at the prompt, and the V2SASG program executes and prompts you for the drive to be written to.

At the command prompt type -

```
CPMSYS59<CR>
```

And you can Sysgen your selected diskette.

**Minimal Bootstrap**

Here is a listing for a minimal bootstrap loader. It uses Z80 relative addressing to ensure location independence. You should place it in ROM or NVRAM if you can.

Restrictions

1. It must start from 200H or higher, to avoid being over written by the boot sector.

2. If you use it to turn off onboard RAM/ROM (see optional code, commented out), then you must relocate it to off-board memory and transfer control to it. Otherwise change the system track loader to disable onboard RAM/ROM.

3. The one memory dependency is the location needed to start loading of the boot sector. This is shown as BOOT, and it is location 0000H.

```
                        ;       MVI     A,2     ;SBC-200 RAM/ROM BIT
                        ;       OUT     7FH     ;TURN OFF ONBOARD RAM/ROM

   0200    3E7E                 MVI     A,7EH   ;SELECT DRIVE 0, LOWER HEAD, HD disk
                                                ;SINGLE DENSITY, WAIT ENABLED
                        ;       MVI     A,1EH   ;CONTROL BYTE FOR MINI-FLOPPY

   0202    D363                 OUT     DCMD
   0204                RETRY:
   0204    21 0000              LXI     H,BOOT
   0207    3E0B                 MVI     A,0BH   ;RESTORE DRIVE
   0209    D364                 OUT     WCMD    ;SEND TO CONTROLLER
   020B    06C8                 MVI     B,200   ;DELAY TILL VALID STATUS
   020D    10FE        ..WAIT:  DJNZ    ..WAIT
   020F    DB64        ..NB:    IN      WSTAT   ;GET STATUS
   0211    CB47                 BIT     0,A     ;BUSY (y/n)
   0213    20FA                 JRNZ    ..NB    ;YES,KEEP WAITING
                       SECTOR:
   0215    3E01                 MVI     A,1     ;START WITH SECTOR 1
   0217    D366                 OUT     WSECT   ;SET SECTOR REG
   0219    3E88                 MVI     A,88H   ;ISSUE READ SECTOR COMMAND
   021B    D364                 OUT     WCMD
   021D    01 0067              LXI     B,0067H ;SET UP BC FOR PIO READ NEXT
   0220    EDB2                 INIR            ;DO PIO SECTOR READ
   0222    EDB2                 INIR
   0224    06C8                 MVI     B,200   ;DELAY TILL VALID STATUS
   0226    10FE        ..WAIT:  DJNZ    ..WAIT
   0228    DB64        ..NB:    IN      WSTAT   ;GET STATUS
   022A    CB47                 BIT     0,A     ;BUSY (y/n)
   022C    20FA                 JRNZ    ..NB    ;YES,KEEP WAITING
   022E    E69D                 ANI     9DH     ;WAS READ SUCCESSFUL(y/n)
   0230    CA 0000              JZ      BOOT    ;GET SYSTEM
   0233    18CF                 JMPR    RETRY
```

NOTE:

If the boot sector read fails, it will retry . . . . forever.

**Patching The Image Files for Console I/O**

Under DDT patch the console I/O patch area, starting from 261CH, (see the dump on the following page). You should only have to patch in six entries that provide port and status mask information for a console device.

NOTE

* If your console device uses positive logic for the status mask, that is, some bit, in a logical 1, or high mode, then you just patch the six bytes.
* If your console device uses negative logic masks (a logical 0 or low set bit) then you will patch in a CMA (compliment accumulator) byte as well in the NOP patch space as shown.

1 - Load the selected CP/M image file under DDT and Dump from 2610
* You should see a patch area that looks like the area listed further on.
* Immediately following the ASCII string CIOTB are the six patch bytes.

2 - In the first byte enter the hex port value of your console status port
3 - In the next byte enter the hex value of your keyboard status port
* In many cases these ports are the same, but the CIOTB has entries for both in case you have a scheme that uses different hardware for keyboard and console status, and data ports. Thus you must make valid entries for each port, even if they are exactly the same.
4 - In the third and fourth byte enter the hex value for the keyboard and console data I/O ports.
5 - In the fifth byte enter the console ready status bit. In this case, for an SBC-200 USART, bit 0 is set, thus 01 is seen.
6 - In the sixth byte place the keyboard data available status bit. In this case bit 1 is used, so a 02 is seen.

7 - If one or both of the status bits are negative logic, that is, the bit indicated in byte 5 or 6, goes to a 0, when the device is ready, you will patch in the CMA byte into a NOP at the points shown. For a console using negative logic, put a 2FH at location 265BH. For a keyboard using negative logic, put a 2FH byte at location 262FH.

Suppose you have an SD Systems VDB-8024 video board for main console. Here are the I/O assignments to consider

- common status port = 0
- common data port = 1
- receive data available mask bit (keyboard) = 1 = 02H
- xmit buffer ready mask bit = 2 = 04H
- positive logic for status bits

Here is the operation with DDT to patch the table for this device:

```
DDT CPM48A.COM<CR>  <- DDT will load the image file CPM48A.COM
                    This is a CP/M image for 48K that uses drive A
                    as the system drive.

DDT VERS 2.2
NEXT PC
2A00 0100
```

Now use the S command to fill in values, starting at 261C
You will fill in the status port values, the data port values
and then the xmit and rcv mask values

S261C<CR>
7D 00<CR> output status port value
7D 00<CR> input (keyboard) status port value
7C 01<CR> output data port value
7C 01<CR> input data port value
01 04<CR> transmit ready mask
02 02<CR> received character ready mask
3A .<CR> <- type a period "." and <CR> to stop entries

Since positive logic is used, you do not have to patch in the CMA op-codes.

If you have to add the CMA op-codes you would use DDT as follows-

S262f<cr> <- go here
00 2F<CR> <- patch in 2FH
00 .<CR>  <- stop patching = type period “.”, then <CR>

S265B<CR> <- now go here
00 2F<CR> <- patch 2FH
00 .<CR>  <- stop patching

Be sure to note if both input and output status bits are negative logic. You may have an input status bit of negative logic, and an output status bit of positive logic. Just patch the needed routine.

Now return to the command prompt by one of two ways:

G0<CR> <- Go command, location 0 or
^C     <- type a <control-c>

Now save the image, using a new name by typing a P in the name field to represent a Patched version.

SAVE 41 CPM48AP.COM<CR>

Now execute the CPM48AP file when you have a diskette ready to Sysgen the image to. Remember to make a newly formatted diskette that is V2BIOS compatible.

Following is a DDT memory dump of the patch area, followed by a .PRN listing, with addresses adjusted to reflect the actual image position under DDT.

NOTE

If you make changes to the V2BIOS.ASM source, the location of CIOTB may shift around. This is not important if you rewrite the source to fit your console I/O, but if you don't, or if others use your new image files, make a note of where to find CIOTB in case they have to patch the image. The image location is printed for you at assembly.

Here is the dump showing the original SBC-200 USART I/O support.

*Memory Image Viewed Under DDT*

```
2610 00 EE B7 F2 8C F4 C9 43 49 4F 54 42 7D 7D 7C 7C . . . . . . . C I O T B } } | |
2620 01 02 43 49 S3 3A 9D F4 4F 3A AO F4 SF ED 78 00 . . K I S . . . . . . . . . . .
2630 00 A3 00 00 C8 00 00 3E FF C9 43 49 CD AS F4 CA . . . . . . . . . . K I . . . .
2640 BC F4 3A 9F F4 4F ED 78 00 00 E6 7F C9 43 4F 53 . . . . . . . . . . . . C O S
2650 CS 3A 9C F4 4F 3A Al F4 SF ED 78 00 00 A3 Cl C8 . . . . . . . . . . . . . . .
2660 00 00 3E FF C9 43 4F CD DO F4 CA E7 F4 59 3A 9E . . . . . C O . . . . . . . .
2670 F4 4F 7B 00 00 ED 79 C9 DB 04 E6 01 C8 3E FF C9 . . . . . . . . . . . . . . .
```

*Source Code Listing of Patch Area ajustted for memory image addresses*

```
2616    43494F5442          .ASCII  \CIOTB\
261C    7D          CSTAT:  .BYTE   0
261D    7D          KBSTAT: .BYTE   0
261E    7C          CIO:    .BYTE   1
261F    7C          KBIO:   .BYTE   1
2620    01          COUT:   .BYTE   4
2621    02          KIN:    .BYTE   2
2622    434953              .ASCII  \KIS\
2625                TTYIS:
2625    3A F49D             LDA     KBSTAT
2628    4F                  MOV     C,A
2629    3A F4A0             LDA     CIN
262C    5F                  MOV     E,A
262D    ED78                INP     A
                ;     ADD CMA FOR NEGATIVE LOGIC IN NEXT BYE

262F    00                  NOP
2630    00                  NOP
2631    A3                  ANA     E
2632    00                  NOP
2633    00                  NOP
2634    C8                  RZ              ;RETURN FALSE IF UNAVAILABLE
2635    00                  NOP
2636    00                  NOP
2637    3EFF                MVI     A,0FFH
2639    C9                  RET             ;RETURN TRUE
263A    4349                .ASCII  \KI\
263C    CD F4A5     TTYIN:  CALL    TTYIS   ;GET STATUS
263F    CA F4BC             JZ      TTYIN   ;WAIT UNTIL DATA AVAILABLE
2642    3A F49F             LDA     KBIO
2645    4F                  MOV     C,A
2646    ED78                INP     A
2648    00                  NOP
2649    00                  NOP
264A    E67F                ANI     07FH    ;DROP PARITY
264C    C9                  RET
264D    434F53              .ASCII  \COS\
2650                TTYOS:
2650    C5                  PUSH    B
2651    3A F49C             LDA     CSTAT
2654    4F                  MOV     C,A
2655    3A F4A1             LDA     COUT
2658    5F                  MOV     E,A
2659    ED78                INP     A
                    ;  ADD CMA FOR NEGATIVE LOGIC IN NEXT BYTE
265B    00                  NOP
265C    00                  NOP
265D    A3                  ANA     E
265E    C1                  POP     B
265F    C8                  RZ              ;RETURN FALSE IF NOT EMPTY
2660    00                  NOP
2661    00                  NOP
2662    3EFF                MVI     A,0FFH
2664    C9                  RET             ;RETURN TRUE
2665    434F                .ASCII  \CO\
2667                TTYOUT:
2667    CD F4D0             CALL    TTYOS     ;GET STATUS
266A    CA F4E7             JZ      TTYOUT    ;WAIT UNTIL BUFFER EMPTY
```

```
266D    59                  MOV     E,C
266E    3A F49E             LDA     CIO
2671    4F                  MOV     C,A
2672    7B                  MOV     A,E
2673    00                  NOP
2674    00                  NOP
2675    ED79                OUTP    A         ;OUTPUT THE DATA
2677    C9                  RET
```

**Implementing Other FDC Boards**

*V2BIOS*

Examine the V2BIOS source and look at the control tables as follows:

OPRTAB:

This table shows the upper four bits of the control byte used to control the following-

- diskette side select
- diskette size (8″ or 5″), but PC 5.25″ and 3.5″ HD drives are treated as 8″ drives
- recording density (single or double)
- wait state enable bit – to synchronize the CPU with FDC byte transfers

ADRTAB:

This table shows the lower four bits of the control latch byte, used to select a disk drive.

When modifying for a different FDC board you have a few options.

If your controller has considerably different bit location for control, keep the present tables and logic, since they are nibble-oriented, and then call a translate routing to shift any bits around.

If your controller is a close match, nibble-wise, just modify the tables to give the needed effect. Note that the V2BIOS checks to see if a VF-II or Z3S/Z4S controller is in place, and then sets up the OPRTAB to match. The ADRTAB is the same for these FDCs.

SPDTAB:

This table shows the bit settings used to control the disk drive step rate. There are four entries for SPDTAB, in order to cover four physical drives of various step-rate characteristics. By default he table is all 0’s, allowing the fastest step rate for 8″ and PC 5.25″ or 3.5″ HD drives. For an old mini-floppy at some physical drive location, set the value to 83H.

*V2FORMAT*

If you are using another controller with a Western Digital WD179X series part, see the section “PIO FLOPPY DISK EQUATES” and enter correct control bytes to match your hardware.

**Checking Your Versafloppy II if Read Errors Persist**

The VCO on the Versafloppy II goes a long way to keep double density data read operations trouble free. But it can drift out of phase with age. If you notice read errors creeping in, try another diskette first to see if it is a media problem. If errors persist, adjust the VCO center frequency as follows:

REMOVE ALL DRIVE CABLES FROM J1 & J2. On the top left of the board is a potentiometer, R19 used to set the center frequency of the VCO. Beside it is TP-2. You should have a square wave pulse here of about 0.5 micro-seconds, or a 2 MHz square wave. If it is not within range, adjust R19. Rock it back and forth to see the pulse width change, and set it around 0.5 micro-seconds. Without floppy cables the pulse should be steady.

I have found a somewhat better spot to set from. Check at U5-7, for a 1 micro-second pulse (1 MHz). As above, adjust R19 to rock back and forth for pulses above and below the range, and stop at about 1 micro-second. Again, without floppy cables the pulse should be steady.

Live Testing

Connect the drive cables.

Boot the system and check as follows:

When adjusting the VCO with R19, you can read DD data tracks under CP/M. Just keep typing the DIR command, or run some software that reads the directory or data tracks.

If you cannot even boot correctly, just keep resetting the system and adjust R19 when it tries to read the directory tracks, or run V2BOOTX if you work with a VF-II in a host system.

Another thing to try is replacing U13, the VCO chip. You can substitute as well.

| IC U13 | C33 |
|---|---|
| 74LS124 | 47 pf - original parts |
| 74S124 | 150 pf – higher speed part with compensating capacitor value |

**Floppy Drive and Media Notes**

Be sure to read a following section “The Strange World of 3.5” Media”

*8” Drives*

Connect to J1. Strap each drive on the drive PCB for a unique address. Look for the DS, or Drive Select header area, and place a jumper for the necessary drive position. You will usually see DS1 to DS4, or DS0 to DS3, allowing for up to four drives. Some drives also support a Binary Select option. In that case four jumper pairs are used to select drive 0 to 15. V2BIOS is written for the original DS selection. If, when you boot up, or read the drive, you get hung up or have excessive read or write errors, check your Floppy Disk Drive User Guide and be sure that options have been set such that the heads load and contact the media whenever a read, write or seek command is executed. Remember to use a terminator as needed, on the last drive. For maximum storage capacity you will need double density, double sided media. The so-called single density diskettes seem to work fine at double density. In any case you can even convert them to double sided. With a single-hole paper punch, make holes in the dust jacket, front and rear, where the double-density index holes are located. Then use an opaque sticker to cover over the original index hole for the single-sided media. Note that some single-sided 8” media was not ‘finished’ on the second side, so that a format would not ‘take’.

*5-1/4” Drives, Old Style Mini-Floppy*

This includes 180K or 360K drives from old IBM PC’s and clones. Connect to J2. Now just use the same notes for the 8” drives in the above paragraph. You might notice that only three drive select jumper options are available. In any case, J2 only supports three drives. This is a common feature of 5-1/4” drive connectors. You can format diskettes in these drives at single or double density, but use the 5” drive option in the V2FORMAT program to do so.

*3.5” and 5.25” IBM PC and Clone High Density (HD) Drives.*

This covers so called 1.44 Mbytes and 1.2 Mbytes drives.

*HD Drive Rules In Brief – So you can skip reading the detailed notes*

3.5” & 5.25” HD drives as 8” Emulators

Media to Use – High Density (HD) for both 8” SD and DD formats
Special Mods – None (unless using PC cables, see next section)
Reliability – Excellent

3.5” and 5.25” HD drives as mini-floppy emulators

Media to Use – 720K or 360K for both mini-floppy SD and DD formats
Special Mods – None (unless using PC cables, see next section)
Reliability - Excellent
Compatibility – can’t place 5.25” media, as formatted, into “real” mini-floppy drives

The 1.44 Mbytes drives are still available, and I use them pretty much exclusively, unless I have to read an actual 8” diskette. I will cover a bit more detail here, since I moved to these drives long ago and have had no problems with them. In fact, with the 3.5” drives, I have noted the most reliable operation of any drives. I also like the plastic case of the 3.5” diskettes and the small amount of storage space they consume when not in use.

*3.5″ and 5.25″ HD Drives as 8″ Emulators*

Special Considerations for 8″ Emulation – None except use of HD media
Modifications Needed to FDC Card – None – see option note on using PC cables
Reliability – Excellent, and better than 8″ drives

For both drives, insert the correct HD media. HD 3.5″ media has two square holes at the handling end. Be sure the write-protect tab is moved to cover it's hole so that you can format or write to it. Run V2FORMAT and set the drive size as 8″. You can now select any other option and you will get perfect results. If you format the 3.5″ drive, containing HD media, as Single Sided, Single Density, 128 byte sectors, and to "S" or Standard format, you will produce a diskette which is 3740 compatible with the original IBM 3740 SSSD or DRI software distribution standard.

*3.5″ Drive – Bonus Storage Option.*

If you format a 3.5″ drive with HD media and select 1024 byte sectors, you can specify 9, 10 or 11 sectors per track. In that order you will get 1.3, 1.55 and 1.66 Mbytes formatted capacity with these sector options. Because the 3.5″ drive rotates at 300 RPM, and not the 360 RPM of the 5.25″ and 8″ drive, there is space (in time) for one or two extra sectors beyond the nine maximum for 5.25″ or 8″ drives.

*3.5″ and 5.25″ HD Drives as mini-floppy Emulators*

Both the 3.5″ and 5.25″ HD drives can be used to format diskettes to old mini-floppy SD and DD standards. You may want to do this to experience the restrictions of formats that provided space as small as 70K per diskette! Note that V2FORMAT does not at present double-step on mini-floppy formats, to produce track spacing needed to be used for a real mini-floppy. It assumes you have connected an actual mini-floppy drive. In any case, insert 720K media into a 3.5″ HD drive, and 360K media into a 5.25″ HD drive. Now run V2FORMAT and select the 5″ drive option and select the rest of the options to suit your needs.

Remember to see the following section on cables and connectors for using PC floppy, or other cables on the 3.5″ and 5.25″ drives.

**Other Drives**

You can connect older mini-floppy drives to the VF-II. You just have to insert the correct media. In addition, you will have to set the SPDTAB in the BIOS to a value of 83H for the actual drive position (drive 0 to 3) Note that only 3, not 4, drive select lines are available on the mini-floppy connector, J2. This was normal, but there is no reason you could not wire in a fourth line. The IBM "AT Technical Manual" shows line 6 for DS3.

**The Strange World of 3.5” Media**

“Prepare your minds for a new scale of physical scientific values, gentlemen”
Dr. Morbius to Commander J.J. Adams and Lt. Ostraw in the movie “Forbidden Planet”

All the following works with 1.44 Mbyte floppy drives only. Because these drives are so versatile, it is not necessary to use 720K drives.

*720K Floppy Diskette Tricks*

Is it 720K or 1.44 Mbytes?

You can make it format for both, just be ready to drill or punch a hole.

The 720K media pretty well emulates the media used on old mini-floppies. You can use 720K media in a 1.44 Mbyte drive, and select 5” diskette sizes from V2FORMAT and get useable mini-floppy type disks in that 3.5” drive. But, 720K media likely will be of interest only to try out mini-floppy formats, should you wish to see how they work (slow!), or if you want to modify the V2FORMAT program for newer versions of the mini-floppy formats to meet the needs of some mini-floppy drives not supported with the default selections. In this way you can change V2FORMAT, format, test and verify, before committing to the actual mini-floppy drive.

If you now try an 8” diskette format on the 720K media it will always FAIL on the read-verify routine during format. It will also be unreadable under CP/M.

Now carefully drill a hole in the 720K diskette casing, where the second hole is found on a 1.44 Mbyte diskette. Reformat using a 8” option and you will have a working diskette. Try to format it with a 5” option and it will FAIL. Cover the second hole on the bottom side of the diskette case, so the micro-switch in the floppy drive cannot move up into the hole. It will now pass 5” formats and fail 8” formats.

*1.44 Mbyte Diskette Tricks*

The 1.44 Mbyte diskettes work with all the 8” options. Formatting with a 5” option results in a verify failure. Cover the second hole, as above, it will pass on 5” format options, and fail on 8” format options. Remove the temporary cover and you can pass on 8” format options again.

*Conclusions*

Examination of the specification sheets for a number of 3.5” micro-floppy drives shows that since the introduction of the 1.44 Mbyte diskette drive, most are of the “Three-Mode” class. In the 720K and 1.44 Mbyte mode they rotate at 300 RPM. There is a 1.6 Mbyte mode, with 360 RPM spindle speed as well, but I have no use for this, since even at 300 RPM they perform flawlessly as 8” emulators, and give a good amount of additional formatted storage to boot.

Suffice to note, electronics within the 1.44 Mbyte disk drive determine, from the presence or absence of the media flagging hole, what writing bias to use for formatting a diskette. Likely an embedded DSP circuit takes care of the reading and writing, and the data is conditioned and filtered according to the presence or absence of the media-type hole.

**Cables and Connectors for 3.5” and 5.25” HD PC Drives**

*5.25” Drives*

When you examine the drive PC board you will often notice silk screening for drive selections from 0 to 3 or 1 to 4. For the 0 – 3, drive 1 is always selected. For 1 – 4 it is drive 2. The twist in the PC floppy cable ensures that the drive on the last cable connection is drive 0, and the next one is drive 1. Sometimes the drive selections are just soldered or they can even fixed copper traces, and sometimes they are jumpers. Also, many drives have jumpers only for positions 0 and 1 (or 1 & 2), with the PCB 2 & 3 positions not having headers in place. V2BIOS supports up to four drives. You will have to decide how to cable up the number of drives you want. Here are some options:

*Use straight-thru cable (no PC twist)*

You will set each drive jumper for a unique position. If you use more than two drives, be prepared for the possibility of having to solder a jumper wire across pads that do not have a header in place.

| Floppy Jumper | CP/M Drive | |
|---|---|---|
| DS0 or DS1 | A | Note that, with V2BIOS you can re-assign |
| DS1 or DS2 | B | logical drives to any physical drive. |
| DS2 or DS3 | C | The CPM48B, CPM48C and CPM48D |
| DS3 or DS4 | D | do that. |

*Use PC cable*

Have each drive set as the second drive, that is DS1 or DS2, depending upon the designation scheme seen on the drive PCB. This is the default setting for PC drives. Now refer to the cable diagram section for information on minor cable modifications needed to allow for operation of an A: and B: drive.

OR -Modify the 5” (J2) connector on the Versafloppy II as follows-

If you just want to keep the cable intact, you can modify a trace on the rear of the VF-II board, for J2. Here I will deal with connector positions, each position consisting of a vertical pair of pins, so you don’t have to check actual pin numbers. On the rear, count over 7 positions from pin 1 on J2. Cut the trace going from the upper pin, midway between the J2 connector pad and the plate-thru above it. Connect a short jumper from the J2 pad to the upper pad at position J2-5 (two places to the right of position 7). This gives you CP/M drives A and B on the J2 connector. You have just joined pin 10 to pin 14. One the last connectors of an IBM cable, DS1 and MOTOR ON are now activated. On a straight-thru cable this shorts DS0 to DS2.

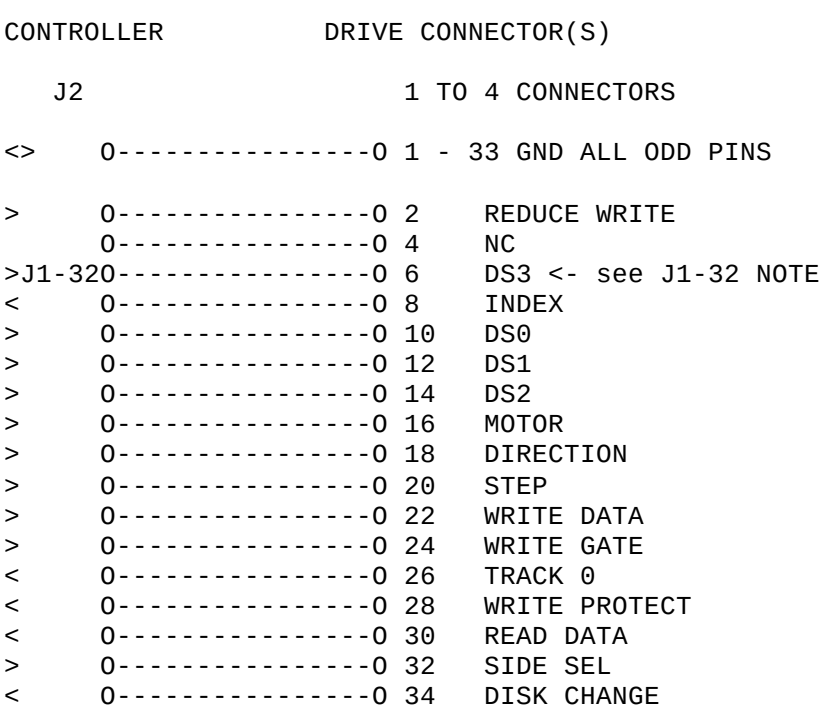
*3.5” drives*

Some older 3.5” drives have user-settable jumpers, but most are hard wired as the second drive, and that hard wiring is not accessible. If you have jumpers, you can set up with a straight-thru cable as with the 5.25” drives. If not just use a PC cable and modify it, or make the J2 cut and jump as above, on pins 10 & 14.

I use PC cables, and put my third and forth drives on the J1, or 8” drive cable. They must be strapped as drive 3 & 4 if I use a pair of 3.5” drives on J2. If I want an 8” drive as #1, I strap it for DS0 or DS1 (that scheme again) and remove the 3.5” ‘A’ drive on J2 to stop double selecting.

**3.5" Floppy Drive Signals and Cable Options**

*STRAIGHT TRU CABLE FOR DRIVES WITH USEABLE "DS" JUMPERS*




-= J1-32 NOTE, - ALL CABLES =-
YOU CAN SOLDER A JUMPER FROM J1-32 TO J2-6 TO ENABLE SELECTION
OF A FOURTH DRIVE ON THE MINI-FLOPPY CONNECTOR.

On the rear of the Versafloppy II, count over 16 places from pin 1 on J1. The upper J1 solder pad at that point is J1-32. Solder a wire to it. On J2 count over 3 places from pin 1. Solder the other end of the wire to the upper pad of J2 at that location.

*STANDARD PC TWISTED CABLE*

```
CONTROLLER                    DRIVE CONNECTORS
   J2                    S.C.                T.C.

<>    O----------------O 1 - 33 GND ALL ODD PINS

>     O----------------O 2  REDUCE WRITE   2    REDUCE WRITE
      O----------------O 4  NC ----------- 4    NC
> NC  O----------------O 6  DS3 ---------  6    DS3
<     O----------------O 8  INDEX -------  8    INDEX
>     O----------------O 10 DS0 ---------  16   MOTOR
>     O----------------O 12 DS1 ---------  14   DS2
>     O----------------O 14 DS2 ---------  12   DS1
>     O----------------O 16 MOTOR -------  10   DS0
>     O----------------O 18 DIRECTION ---  18   DIRECTION
>     O----------------O 20 STEP --------  20   STEP
>     O----------------O 22 WRITE DATA --  22   WRITE DATA
>     O----------------O 24 WRITE GATE --  24   WRITE GATE
<     O----------------O 26 TRACK 0 -----  26   TRACK 0
<     O----------------O 28 WRITE PROTECT- 28   WRITE PROTECT
<     O----------------O 30 READ DATA ---- 30   READ DATA
>     O----------------O 32 SIDE SEL ----- 32   SIDE SEL
<     O----------------O 34 DISK CHANGE -- 34   DISK CHANGE
```

S.C. = STRAIGHT THRU CABLE SECTION
T.C. = TWISTED CABLE SECTION

A look at the schematics for the disk adapter in the IBM publication “AT Technical Manual” shows that pin 10 is actually connected to a second MOT EN 1 – (Motor Enable 1) circuit, and not to a DS0 line. Further, pin 14 on the controller is designated DS1, while pin 12 is designated DS2. While IBM designates pin 6 as DS3, it is not connected to any circuit on their controller. Thus, in order to support the use of all drives strapped as DS1, IBM designed a twisted cable, and implemented a non-standard FDC interface.

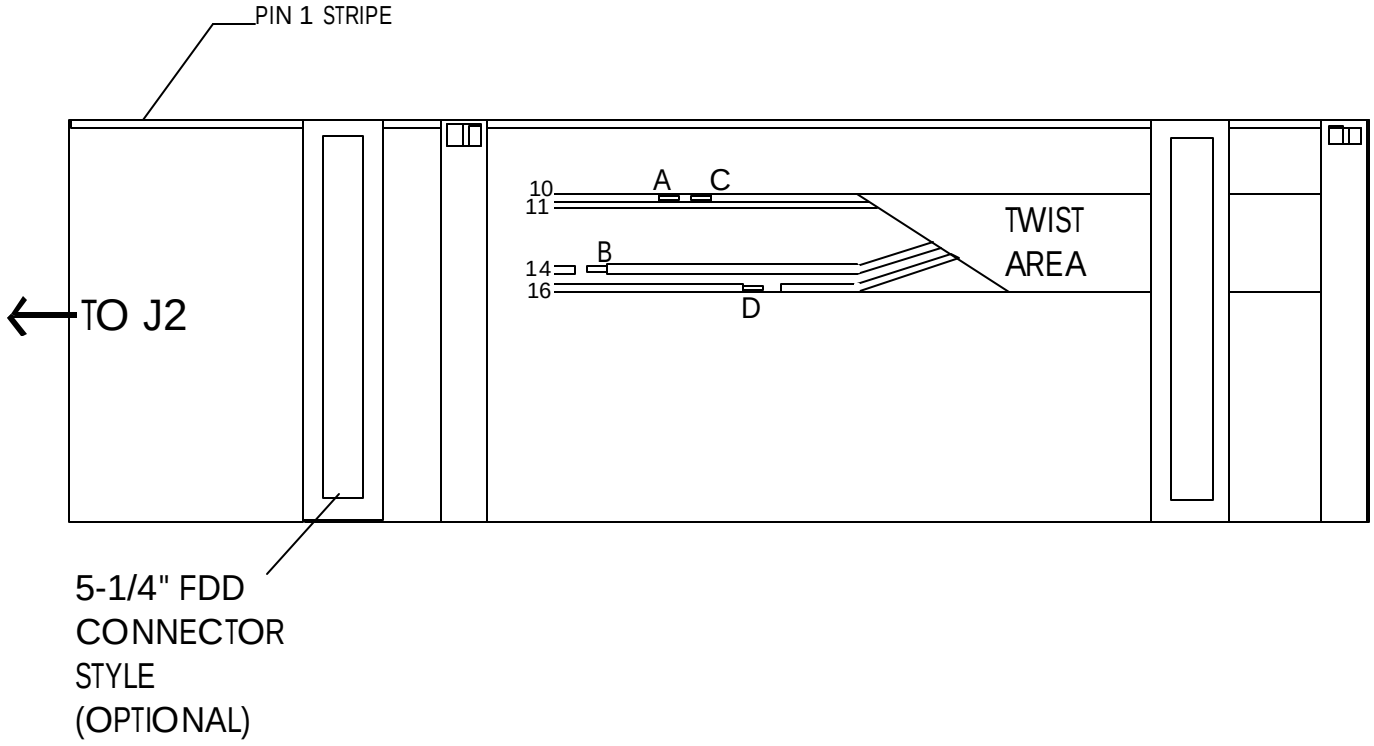
*MODIFIED PC CABLE FOR STANDARD PC HD DRIVES (EACH DRIVE SET TO DS1)*

ALLOWS S.C. SECTION TO BE "B:" DRIVE AND T.C. SECTION
TO BE "A:" DRIVE WHEN CONNECTED TO THE VERSAFLOPPY II

```
CONTROLLER                  DRIVE CONNECTORS
    J2                S.C.              T.C.

<>    O---------------O 1 - 33 GND ALL ODD PINS

>     O---------------O 2  REDUCE WRITE   2    REDUCE WRITE
      O---------------O 4  NC ----------  4    NC
>J1-32O---------------O 6  DS3 ---------  6    DS3
<     O---------------O 8  INDEX -------  8    INDEX
>     O---------------O 10 DS0—A       C- 16   MOTOR
>     O---------------O 12 DS1 ---------- 14   DS2
>     O---------------O 14 DS2 -       B- 12   DS1
>     O---------------O 16 MOTOR —D       10   DS0
>     O---------------O 18 DIRECTION ---- 18   DIRECTION
>     O---------------O 20 STEP --------- 20   STEP
>     O---------------O 22 WRITE DATA --- 22   WRITE DATA
>     O---------------O 24 WRITE GATE --- 24   WRITE GATE
<     O---------------O 26 TRACK 0 ------ 26   TRACK 0
<     O---------------O 28 WRITE PROTECT- 28   WRITE PROTECT
<     O---------------O 30 READ DATA ---- 30   READ DATA
>     O---------------O 32 SIDE SEL ----- 32   SIDE SEL
<     O---------------O 34 DISK CHANGE -- 34   DISK CHANGE
```




Cut wire 10 at A & C, wire 14 at B and wire 16 at D. Strip ends as shown, then join A to B and C to D. See next page for detailed cable modification notes.

*How To Cut and Join The Cable*

Some PC cables have pairs of 3.5” and 5.25” connectors, newer ones usually have just 3.5” connectors, so I will refer to any connector section as “connector(s)”. Look for the colored orientation stripe along one edge of the cable. This is the “Pin 1” flag. Examine the cable and you will note that the connector(s) at one end have a twisted section of cable running through them. This is the PC “A:” drive. The farther end is connected to the PC floppy controller header. In our case that will be J2 on the Versafloppy. It is easy to do all the modifications on the straight section of the inter-connector twist. With an Xacto, or similar knife, cut the webbing between wire 10 and wire 11. Next cut the webbing between wire 15 and wire 16. Now cut the webbing on both sides of wire 14. Make your cuts about 3” long if possible. Lift up wire 10 and cut at point A & C. Lift wire 14 and cut at point B. Lift wire 16 and cut at point D. Strip a small section of the insulation off the ends as shown at A, C, B and D. Now join section A to section B, and finally join section C to section D. Solder these two new connections if you can. Cover up the exposed wire with a small square of electrical tape, folded around the wire and adhering to it’s own sticky side. The cable now supplies a DS0 signal to the DS1 line on the end floppy connector, and a Motor Enable signal to the correct pin as well.

**Customizing The V2FORMAT Program for Other Diskette Formats**

The rules for making a diskette format table are covered now. The sample case is a diskette formatted in Double Density, with 15 sectors per track of 512 bytes each. Both double and single sided entries are always included. The format table is found on the following pages. Here is the breakdown, line by line.

1. This line contains the primary user selected options picked up during the format query routine. It is used here as a tag so that, once the query is complete, the track building routine can traverse all the format tables until a match is found. The ASCII entries are ordered as follows:

- Diskette Size – 8 or 5
- Recording Density  S or D for Single or Double
- Sector length  1 = 128, 2 = 256, 3 = 512, 4 = 1024

2. Length of this format table. Used to index to the next table if no match.
3. The prototype control byte written to the FDC control port to set up disk size, density and side. See EQUATES section in V2FORMAT source.
4. This is the post-index field. Data is picked up in pairs, starting with the first byte, 80 as a counter, the next byte, 4EH, is written 80 times, then 12 is picked up as a counter, to write 0H 12 times, and so on.
5. This is the number of data records, or sectors, per track. This is the primary counter for the next group of fields to be generated for the track image.
6. This pattern precedes each data field, and contains the ID address mark, track number, side number, sector number, sector length, CRC any syncing field. As each track is written the side and track values are updated on the fly.
7. Bytes in sector
8. E5H data pattern for sector, then 84 4EH bytes post sector field.
9. Size of filler field, to be written after all sectors have been formatted in place.
10. 4EH, the data pattern in the filler, or trailing field. Written to disk until interrupted by the next index pulse.
11. Length of the ATL8D3 Disk Descriptor Block, or DDB
12. Length of DDB including sector interleave table
13. Length from TB8D3 to DDB8D3
14. Header for DDB. Four bytes used to identify a VFORMAT DDB.
15. Six bytes to fill in for FDC control use when diskette is selected and set up by the V2bIOS.

16. Flag Byte for disk operation – see following description
17. Number of system tracks for V2BIOS use
18. System track sector size, number of CP/M sectors, data track sector size and number  of CP/M sectors
19. CP/M sectors per track
20. CP/M BSH, BLM and EXM
21. CP/M DSM, or total storage in allocation blocks
22. CP/M max directory entries
23. CP/M ALO0, ALO1, reserved directory blocks
24. CP/M CKS
25. CP/M OFF, reserved tracks at beginning of disk

The following DDB8D3 is the same as items 14 to 25, except it represents the values used for a single sided diskette, thus about half the storage capacity.
TRA8D3 is the sector translate table used by CP/M. It is moved into place at the end of ALT8D3 or DDB8D3, depending on which one is used (based on number of sides for the diskette). The combined structure is placed at the end of sector number 8 on the first system track, starting at location 384, since each system track sector is 512 bytes in length. Thus no DDB can exceed 128 bytes.

FLAG Byte

System track bits

Bit 4 - 1 = 5″ (mini-floppy) 0 = 8″ (or 3.5″ & 5.25″ PC HD disk)

Data track bits

Bit 3 – 1 = double density, 0 = single density
Bit 2 – 1 = 5″ (mini-floppy) 0 = 8″ (or 3.5″ & 5.25″ PC HD disk)
Bit 1 – 1 = double sided, 0 = single sided
Bit 0 – 1 = DDB has been established, 0 = DDB not processed yet

For more information about WD179X format standards, see the manual:

FD179X and TMS279X_DataSheet.pdf

Getting more tracks for old mini-floppy formats.

Again, keep the following in mind:

The mini-floppy formats are the same as when the core format software was created. If you need more tracks etc, you should look at the source just past DOUSER:. You will see the maximum track value was selected by the diskette size check (76 tracks for 8″, 35 tracks for 5″). The value was saved at MAXTRK for later compare. Note also, if you format 5.25″ HD drives for media to insert in older 40 TPI drives, you should call the routine STEPIN twice for each track. You can expand the format query section to include this additional information when a 5″ disk size is selected. Use the GETOK routine to implement the user interface on multiple-choice options.

Examination of the many format tables will assist you in creating a new format type. In many cases you can likely use an existing format table and just add to the number of tracks to be formatted for higher capacity mini-floppy drives. You will then have to increase the DSM value. The BSH, BLM and EXM can stay the same in many instances. If you need more directory entries, increase the CP/M max directory count, and set more bits in the ALO0 & ALO1 bytes. See the DRI CP/M Alteration Guide for this, or just scan over settings used in higher capacity formats. A simple rule for the DSM is just increase the size in proportion to the number of tracks you add to the maximum allowed for the old mini-floppy count. If you go to 80 tracks and remember the 5″ formats were set up for 35 track drives, then the new DSM is 80/35X(DSM) or about 2.2X the present DSM. For the 8″ format shown, 192 directory entries are supported by setting 3 ALO0 bits (11100000B)this covers 64 entries per bit, so you set one additional bit for every 64 additional entries. Bits move from left to right in the ALO fields, so 256 dir entries sets ALO0 to 11110000B. The expressed directory count is always 1 less than actual. The one thing to watch for is when the DSM exceeds 255, say when you add more tracks to a 5″ format that had a DSM below 256. You must change your EXM, or extent mask, and if the drive was using 1024 byte blocks, move up to 2048 byte blocks. In that case the DSM value drops by 50%, since it is a count of (Total Space divided by allocation block size). When the block size changes, you must change the BSH and BLM as well. In all cases, check the CP/M Alteration Guide for more detail.

The sample format table follows . . .

```
;8" DOUBLE DENSITY, 512 BYTE SECTORS – use as template for your own formats
TB8D3:
1.    .BYTE '8','D','3'
2.    .BYTE LEN8D3
3.    .BYTE DCMD8D
4.    .BYTE 80,04EH,12,0H,3,0F6H,1,0FCH,50,04EH
5.    .BYTE 15
6.    .BYTE 12,0H,3,0F5H,1,2H,22,04EH,12,0H,3,0F5H
7.    .WORD 512
8.    .BYTE 0E5H,84,04EH
9.    .WORD 400+400
10.   .BYTE 04EH
11.   .BYTE ALE8D3
12.   .BYTE DLE8D3
13.   .BYTE DDB8D3-TB8D3
ALT8D3:  === double-sided DDB
14.   .BYTE 0DDH,0FDH,0DDH,0FDH <-- ID field for V2BIOS disk format
15.   .BYTE 0H,0H,0H,0H,0H,0H <-- work space for control bytes
16.   .BYTE 00001011B  <-- FLAG byte
17.   .WORD 2         <-- V2BIOS system track count
18.   .BYTE 2,36,2,60 <-- V2BIOS System & Data track info sector mask & CP/M SPT
19.   .WORD 60      <-- CP/M sectors per track
20.   .BYTE 4,15,0  <-- BSH, BLM and EXM
21.   .WORD 561     <-- Change this to match how many allocation blocks are available
22.   .WORD 191     <-- 192 directory entries
23.   .BYTE 11100000B,0 <-- blocks used for directory space
24.   .WORD 48          <-- size of checksum vector
25.   .WORD 2           <-- number of system tracks
ALE8D3      ==    .-ALT8D3
DDB8D3: === single-sided DDB
.BYTE 0DDH,0FDH,0DDH,0FDH
.BYTE 0H,0H,0H,0H,0H,0H
.BYTE 00001001B
.WORD 2
.BYTE 2,36,2,60
.WORD 60
.BYTE 4,15,0
.WORD 280    <-- Note how single-sided = half the blocks of double sided
.WORD 191
.BYTE 11100000B,0
.WORD 48
.WORD 2
TRA8D3: == sector translate table, four CP/M records per 512 byte sector
      .BYTE 1,2,3,4      <- shown for an interleave of 4
      .BYTE 17,18,19,20
      .BYTE 33,34,35,36
      .BYTE 49,50,51,52
      .BYTE 5,6,7,8
      .BYTE 21,22,23,24
      .BYTE 37,38,39,40
      .BYTE 53,54,55,56
      .BYTE 9,10,11,12
      .BYTE 25,26,27,28
      .BYTE 41,42,43,44
      .BYTE 57,58,59,60
      .BYTE 13,14,15,16
      .BYTE 29,30,31,32
      .BYTE 45,46,47,48

DLE8D3      ==    .-DDB8D3
LEN8D3      ==    .-TB8D3

35 All software and documentation is in the public domain
```

**ATLDDBS**

When making a DDB for some ‘foreign’ format keep the following in mind:

STAT DSK:

Using STAT DSK: on a host system that at present supports the foreign format would be very helpful. Information useful in setting up the DDB is as follows:

```
    Drive Characteristics
1 - 10738: 128 byte Record Capacity      <- total storage used by CP/M
2 -  1342: Kilobyte Drive Capacity
3 -   256: 32 Byte Directory Entries     <- number of directory entries
4 -   256: Checked Directory Entries     <- checksum vector is used
5 -   128: Records / Extent              <- size of extent = 128 * 128 (16,384 bytes)
6 -    16: Records / Block               <- size of block = 16 *128 (2048 bytes)
7 -    72: Sectors / Track               <- CP/M Sectors per track
8 -     2: Reserved Track                <- reserved tracks
```

NOTES:

The DSM, or number of storage blocks = item 1 divided by item 6 (10738/16) = 671
Item 3 should be totaled to get the directory size in bytes (256 X 32) = 8,192
    Now calculate the block size from line 6 (16 X 128) = 2048
    Divide the total directory size by the block size (8192/2048) = 4
    The value 4 represents the number of reserved blocks assigned for directory entries. Four bits will thus be set in ALO0.
From these basic values consult the CP/M Alteration Guide to generate the other necessary values for BSH, BLM, EXM and DRM.

You now have to determine the sector size of the foreign diskette. Once that is known you must generate an interleave table. If you know the correct interleave factor of the physical sectors, you next generate the interleave table for CP/M sectors, based on the number of CP/M sectors that fit within a physical sector.

| Physical Sector | CP/M Sectors |
|---|---|
| 128 | 1 |
| 256 | 2 |
| 512 | 4 |
| 1024 | 8 |

Thus, the first entries for a diskette with 1024 byte sectors would be 1,2,3,4,5,6,7,8. In the sample DDBS you will note that Andicom used an interleave of 15, Compupro used an interleave of 3, Lomas Data Products used no interleave. The Lomas diskette was likely read in a track at a time, as this particular diskette was used for the distribution of their CP/M-86 product. You might have to experiment with the interleave. In doing so you will know it is working when the directory looks OK.

```
;ANDICOM - 256 BYTE SECTOR
;DOUBLE-DENSITY, DOUBLE-SIDED

DDB256:

      .BYTE 00001011B   ;FLAG
OFF256:
      .WORD 1           ;OFF <- SEE OFF-NOTE
      .BYTE 1           ;SSLEN
      .BYTE 36          ;SLRPS
      .BYTE 1           ;USLN
      .BYTE 52          ;ULRPS


      .WORD 52
      .BYTE 4,15
EXM256:
      .BYTE 0
      .WORD 486
      .WORD 127
      .BYTE 11000000B,0
      .WORD 32
OFD256:
      .WORD 1


      .BYTE 1,2
      .BYTE 15,16
      .BYTE 1DH,1EH
      .BYTE 2BH,2CH
      .BYTE 5,6
      .BYTE 13H,14H
      .BYTE 21H,22H
      .BYTE 2FH,30H
      .BYTE 9,10
      .BYTE 17H,18H
      .BYTE 25H,26H
      .BYTE 33H,34H
      .BYTE 0DH,0EH
      .BYTE 1BH,1CH
      .BYTE 29H,2AH
      .BYTE 3,4
      .BYTE 11H,12H
      .BYTE 1FH,20H
      .BYTE 2DH,2EH
      .BYTE 7,8
      .BYTE 15H,16H
      .BYTE 23H,24H
      .BYTE 31H,32H
      .BYTE 0BH,0CH
      .BYTE 19H,1AH
      .BYTE 27H,28H

;OFF-NOTE
;BECAUSE THE SECOND SYSTEM TRACK IS ON
;SIDE TWO, IT HAS AN EFFECTIVE OFFSET OF 1
```

```
;COMPUPRO - 1024 BYTE SECTOR
;DOUBLE-DENSITY, DOUBLE-SIDED

DDBCPR:
      .BYTE 00001011B   ;FLAG
      .WORD 2           ;OFF
      .BYTE 3           ;SSLEN
      .BYTE 36          ;SLRPS
      .BYTE 3           ;USLN
      .BYTE 64          ;ULRPS


      .WORD 64
      .BYTE 4,15,0
      .WORD 600
      .WORD 255
      .BYTE 11110000B,0
      .WORD 64
      .WORD 2


      .BYTE 1,2,3,4,5,6,7,8
      .BYTE 25,26,27,28,29,30,31,32
      .BYTE 49,50,51,52,53,54,55,56
      .BYTE 9,10,11,12,13,14,15,16
      .BYTE 33,34,35,36,37,38,39,40
      .BYTE 57,58,59,60,61,62,63,64
      .BYTE 17,18,19,20,21,22,23,24
      .BYTE 41,42,43,44,45,46,47,48

;LOMAS DATA PRODUCTS - 1024 BYTE SECTOR
;DOUBLE-DENSITY, SINGLE-SIDED

DDBLDP:
      .BYTE 00001001B   ;FLAG
      .WORD 2           ;OFF
      .BYTE 3           ;SSLEN
      .BYTE 36          ;SLRPS
      .BYTE 3           ;USLN
      .BYTE 64          ;ULRPS


      .WORD 64
      .BYTE 4,15,0
      .WORD 300
      .WORD 127
      .BYTE 11000000B,0
      .WORD 64
      .WORD 2


      .BYTE 1,2,3,4,5,6,7,8
      .BYTE 9,10,11,12,13,14,15,16
      .BYTE 17,18,19,20,21,22,23,24
      .BYTE 25,26,27,28,29,30,31,32
      .BYTE 33,34,35,36,37,38,39,40
      .BYTE 41,42,43,44,45,46,47,48
      .BYTE 49,50,51,52,53,54,55,56
      .BYTE 57,58,59,60,61,62,63,64
```

39 All software and documentation is in the public domain

40 All software and documentation is in the public domain