SC126 How-To:
Preparing a MicroSD Card to Transfer Files to/from a Linux System
Revision 1.5.2: 27 Mar 2020 – David Reese

0.0) Acknowledgement: I want to express appreciation to Wayne Warthen, the author of RomWBW, for his assistance in getting me up to speed on this. I'd still be flailing and failing if not for his kind guidance.

1.0) Objective: The purpose of this How-To is to provide a step-by-step guide to getting a microSD card partitioned and formatted to allow transfer of files from a computer running Linux to an SC126 that uses the same card for its native CP/M storage slices. FAT32 (or VFAT) is used because it can be read by both systems after the right tools are installed on the SC126.

NOTE: This task can also be accomplished using a system running Microsoft Windows. Since I no longer run Windows (except at work, where I'm compelled to), these instructions are for users of the Linux operating system. I work in PCLinuxOS, but any modern Linux distribution should work as well.

1.1) Prerequisites: For this work, the user will need:

- A working Linux system that has an SD Card slot.
- A working knowledge of the Linux shell and some of the more common commands for it is useful. We do fully explain all commands used in this task.
- A microSD card. (I know from experience that a cheap 16GB card will work.)
- If the card slot on the Linux system is for a standard sized SD Card, a micro-to-standard SD Card adapter may also be needed (micro SD cards usually include such an adapter).
- A working SC126 Single Board Computer system with a working microSD card interface, and comms established via a serial terminal emulation program.

NOTE: All text displayed by the SC126, all Linux shell commands, and all references to files in the following examples will be shown in **`Courier New Bold.`** All input from the user will be shown in the same typeface, but **`highlighted`**. Highlights for emphasis will be shown in **`this color`**.

The **`[ENTER]`** key must be pressed at the end of all Linux shell and CP/M Console Command Processor (CCP) commands to start processing. This will be assumed as common knowledge going forward.

NOTE ALSO: In all cases where the Linux shell is in use, *the context matters!* Some operations here can be performed as a normal user, while still others ***must be performed as root.*** In all cases, the shell prompt for normal users is **`$`**, and that for root is **`#`**.

{Please proceed to the next page to continue with step (2.0).}

2.0) Procedure to Transfer the Needed Tools from RomWBW:

OVERVIEW: You will need the tool **FAT.COM**. This tool resides in the image file **hd_zsdos.img** that is part of RomWBW. (This image was known as **hd0.img** in earlier versions of RomWBW.) The contents of this image must be written to your microSD card.

2.1) Obtain and extract a copy of the latest RomWBW archive

This archive is available from: https://github.com/wwarthen/RomWBW/releases Extract it to a convenient location on the Linux system being used. It is in .zip format. The shell tool **unzip** can handle this task, or it can be extracted using any of several GUI tools available for Linux such as **ark** or **file-roller.**

(Wayne recommends using *the latest available version* of RomWBW.)

The shell command would look similar to:

```
[wabbit@localhost Downloads]$ unzip RomWBW-v2.9.2-pre.38-Package.zip
```

The command shown above will extract the contents of the .zip file to the Downloads directory of user **wabbit** (i.e., **/home/wabbit/Downloads**) on the Linux system. The shell prompt shows the currently logged directory name just after the login string **wabbit@localhost.**

2.2) Open a shell and change to the directory on the Linux filesystem where the **Binary** subdirectory of **RomWBW** is stored.

Changing directories from the shell prompt is done with **cd**, *for example:*

```
cd /home/wabbit/Downloads/RomWBW/Binary
```

will change from the present directory to the indicated path under the home directory of user **wabbit**. *You should change to the directory where you extracted the RomWBW archive*, and open the **Binary** subdirectory.

2.3) From this Location, *Gain Root Access*. This is done from a shell using **su** or **sudo**.

*2.31) Example Using **su**:* Issue the command:

```
[wabbit@localhost Binary]$ su
```

The system will prompt you as follows:

```
Password: {enter root password here}
```

This prompt is waiting for the **root** password. Once this is supplied, the shell prompt will change from **$** to **#** to indicate root access is granted.

2.0) Procedure to Transfer the Needed Tools from RomWBW, (continued):

2.3) Gain root Access (continued.)

2.32) Example using `sudo`*:*

```
sudo mount -t vfat /dev/mmcblk0p1 /media/2821-0B47
```

combines gaining root access with a command to mount an SD Card named **`/dev/mmcblk0p1`** on mount point **`/media/2821-0B47`** ***if that mount point already exists.*** When the **`Password:`** prompt comes up, enter your user password, *not* the root password. (I don't like sudo, and don't have it on my system. Hey, personal preference.)

2.4) Identify your microSD Card's Device Name:

Insert your microSD card into a reader on your Linux system. Most modern Linux systems will mount the card automatically somewhere under **`/media`**. Once mounted, use **`df -h`** to identify the card's device name. The output of this command looks like:

```
[root@localhost Binary]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        44G   19G   23G  45% /
/dev            1.9G  1.2M  1.9G   1% /dev
none            2.0G   29M  1.9G   2% /dev/shm
/dev/sda2       394G  281G   93G  76% /home
none            2.0G   24K  2.0G   1% /var/run/user/500
/dev/sr0        648M  648M     0 100% /media/LDR_0496_1
/dev/mmcblk0p1   15G  1.6M   15G   1% /media/2821-2AB4
```

2.41) Block Device Names in Linux: Note how Linux designates drives (block devices): **`/dev/type-letter-partition`**. Below are some examples:

`/dev/sda2` is broken down as: SCSI Device A, Partition 2. SATA and PATA (IDE) devices are also managed by the SCSI subsystem, so this device type covers all three types. If you're on a very old Linux system you might also see such identifiers as **`/dev/hda1`** for IDE devices.

`/dev/sr0` breaks down as: Serial Removable Device Zero (an optical drive)

`/dev/mmcblk0p1` goes something like: Multimedia Card Block Device Zero, Partition 1. I'm thinking this is because an SD Card is removable, but I'm not sure. ***You might see your SD Card identified this way, or as a SCSI or IDE device.***

Note: those who wish more in-depth data on this might read:
https://opensource.com/article/16/11/managing-devices-linux

2.0) Procedure to Transfer the Needed Tools from RomWBW, (continued):

2.5) Un-mount the Device

Once the device name is known (**`/dev/mmcblk0p1`** in my case), un-mount it by issuing the command (for example):

`umount /dev/mmcblk0p1` (notice no **`n`** after the first **`u`** in **`umount.`**)

WARNING! If you have data anywhere on your microSD card at this point, the next step will cause you to LOSE IT! If it's dear to you, STOP & back it up FIRST!

2.6) Transfer the Contents of hd_zsdos.img:

This next step will transfer the contents of **`hd_zsdos.img`** to your microSD card. As will be seen later, it transfers four 8 Mb slices to the card, three of which will contain files.

The operation will take several seconds, and the output from the operation will look something like:

```
[root@localhost Binary]# dd if=hd_zsdos.img of=/dev/mmcblk0
66560+0 records in
66560+0 records out
34078720 bytes (34 MB, 32 MiB) copied, 9.03704 s, 3.8 MB/s
```

NOTE: your device name may vary! *Substitute the device name as determined by* ***`df -h`*** *above, omitting any number or* ***`p1`****, or p-whatever, as those are partition references and are not needed for this operation.*

2.7) Remove the SD Card and Insert into the SC126 SD Card Adapter

With the SC126 powered down, once the content of **`hd_zsdos.img`** is successfully transferred, ensure the SD card is not still mounted (see step **2.5**, above). Remove the card from your Linux system, and insert it into the microSD card interface on your SC126 system. Power up the SC126 once the SD card is inserted. The transfer of **`FAT.COM`** to CP/M slice **`G:`** has been accomplished.

2.71) Exiting From a Root Shell: At this point if you used **`su`** to get to a root shell you should also issue the command **`exit`** to return to a normal user's shell for reasons of system security.

At this point, we are ready to do the rest of the work from the SC126 using **`FAT.COM`** and **`FDISK80.COM`**. (In releases after 2.9.2-pre-27, FDISK80.COM should be distributed as part of hd_zsdos.img. As such, it should also be available from **`G:`**.)

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition:

CAUTION: Partitioning operations in **FDISK80.COM** need to be conducted with knowledge of what one is doing, otherwise bad outcomes will result. In most cases, errors are recoverable with some extra work, but it's better to get it right the first time.

Also NOTE: *Do NOT be tempted to use any other tool* from Windows, Linux or any non-CP/M system to create this partition. *Doing so will wipe out any CP/M slices previously created*.

The document **FDisk Manual.pdf** in the **Doc** subfolder of the extracted RomWBW archive is good reading, but it gets rather technical. I have tried to distill that technical stuff down to the bare minimum needed to produce a working FAT 32 partition.

We will see a lot of screenshots from my SC126 to illustrate how to best proceed.

3.1) Startup and Verification:

Start up your SC126. You should see a great deal of startup data pass by, but the important part is in this table:

```
Unit         Device       Type              Capacity/Mode
----------   ----------   ---------------   --------------------
Char 0       ASCI0:       RS-232            38400,8,N,1
Char 1       ASCI1:       RS-232            38400,8,N,1
Disk 0       MD1:         RAM Disk          384KB,LBA
Disk 1       MD0:         ROM Disk          384KB,LBA
Disk 2       IDE0:        Hard Disk         --
Disk 3       SD0:         SD Card           15100MB,LBA
```

The important part of this table is at the very bottom. It shows the Disk Number (aka Unit Number) of the SD Card, which is 3. Check and remember ***what your system displays,*** whatever that might be – it will vary with different revisions of RomWBW and with any customization to the drive table that may have been made previously.

Choose **C** for CP/M at the **Boot Selection?** prompt. As CP/M boots, it will display the following table:

```
Configuring Drives...

   A:=MD1:0
   B:=MD0:0
   C:=IDE0:0
   D:=IDE0:1
   E:=IDE0:2
   F:=IDE0:3
   G:=SD0:0          These four highlighted 8 MB CP/M slices were created by
   H:=SD0:1          the dd operation just performed in the previous procedure.
   I:=SD0:2          Three of them (G:, H: & I:) have files stored on them.
   J:=SD0:3

   1932 Disk Buffer Bytes Free

CP/M-80 v2.2, 54.0K TPA
```

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition: (continued)

3.11) Verification: We need to verify the contents of the CP/M slices that exist on the system. For systems prior to **2.9.2-pre.38** you may need to get a directory of **B:** (which is the default drive for the system). Shown below is a partial directory.

```
B>DIR
B: ASM      COM : CLRDIR   COM : COPY     CFG : COPY     COM
B: DDT      COM : DDTZ     COM : DIF      COM : DUMP     COM
B: ED       COM : FA16     CFG : FDISK80  COM : FILEATTR COM
```

The highlighted file, **FDISK80.COM**, is needed for our next steps, but let's go ahead and verify **G:** next. Log to it and get its directory (again, we only show part of the directory):

```
B>G:
G>DIR
G: ZXD      COM : ASSIGN   COM : FAT      COM : FDU      COM
G: FORMAT   COM : INTTEST  COM : MODE     COM : RTC      COM
G: SURVEY   COM : SYSCOPY  COM : SYSGEN   COM : TALK     COM
G: TIMER    COM : TUNE     COM : XM       COM : ZSYS     SYS
G: CLRDIR   COM : COMPARE  COM : DDTZ     COM : FDISK80  COM
G: FLASH    COM : NULU     COM : UNARC    COM : ZAP      COM
G: ZDE      COM : ZDENST   COM
```

FAT.COM is needed after the FAT32 partition is built to format and access it. If this file is not present, *something went wrong* with the **dd** operation in the previous procedure, and you should stop, go back and start fresh with a clear SD card.

If all is well, we may safely proceed.

{Please proceed to the next page to continue with step (3.2).}

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition: (continued)

3.2) Start FDISK80

From the **G:** slice, the Console Command Processor (CCP) prompt should look like: **G>**. From there, log back to the **B:** slice or, if working with the latest RomWBW, from **G:**, issue the command **FDISK80** (as shown below)

```
G>B:                (If working from G:)  G>FDISK80
B>FDISK80
```

The opening screen of this program should appear as:

```
FDISK80 for RomWBW, UNA, Mini-M68k  ----  Version 1.50-16
created 28-May-2015
                  (Running under RomWBW HBIOS)

HBIOS unit number:
```

Remember that number I told you to remember in step 3.11? (For me, it's 3.) Yeah. Enter it at the **HBIOS unit number:** prompt. Now the screen should look like this:

```
FDISK80 for RomWBW, UNA, Mini-M68k  ----  Version 1.50-16 created 28-May-2015
                (Running under RomWBW HBIOS)

HBIOS unit number: 3
Capacity of disk 3:  ( 15G)  30924800      Geom d7e01010
Nr  ---Type- A --      Start         End   LBA start  LBA count  Size
 1      ??   * e5 997:229:37  997:229:37  3857049061 3857049061    2T
 2      ??   * e5 997:229:37  997:229:37  3857049061 3857049061    2T
 3      ??   * e5 997:229:37  997:229:37  3857049061 3857049061    2T
 4      ??   * e5 997:229:37  997:229:37  3857049061 3857049061    2T
>>
```

The old version of **FDISK80**'s output is shown above, and it's completely ***BOGUS***, but *the latest version of **FDISK80** has been revised to properly show no partitions defined. Here's what that would look like (using a 2GB SD card):*

```
FDISK80 for RomWBW, UNA, Mini-M68k  ----  Version 1.50-16 created 28-May-2015
                (Running under RomWBW HBIOS)

HBIOS unit number: 3
Capacity of disk 3:  (  2G)  3911680      Geom 3bb01010
Nr  ---Type- A --      Start         End   LBA start  LBA count  Size
 1             00       *** empty ***
 2             00       *** empty ***
 3             00       *** empty ***
 4             00       *** empty ***
>>
```

3.0) Procedure for Using `FDISK80.COM` to add a FAT32 Partition: (continued)

3.2) Start `FDISK80` (continued)

The information concerning **`Capacity of Disk 3:`** is valid in either case. 15 GB is close enough for government work, and the number just after that (30,924,800) has an interesting property. Divide it by 256, and we find *my SD Card* has 120,800 cylinders, because there are 256 sectors per cylinder at the default disk geometry. The 2 GB SD Card has 3,911,680 sectors, and 15,280 cylinders.

3.3) Initialize the Partition Table: this step clears the existing partition table to make room for legitimate partitions. Go ahead and do this, and we will show in the following steps how to prevent loss of the CP/M slices already created.

NOTE: If using RomWBW archives at version **2.9.2-pre-27 or later**, ***this step, (3.3), is not necessary,*** but it won't hurt anything if you do perform the steps as shown below. ***If you decide to omit this, skip ahead to step (3.4), please.***

At the **>>** prompt, press the letter **`I`** (or **`i`**, as **`FDISK80`** is case-insensitive) followed by the **`[ENTER]`** key. The input vanishes, and another **>>** prompt is displayed. Looks like nothing's happened, right? Let's see what's happened.

At the new **>>** press the letter **`P`** and **`[ENTER]`** to Print The Partition Table. You should see something very similar to:

```
>>P
Nr  ---Type- A --       Start          End    LBA start  LBA count  Size
 1              00        *** empty ***
 2              00        *** empty ***
 3              00        *** empty ***
 4              00        *** empty ***
>>
```

At this point, the partition table is initialized, but it's not ready to create a new partition just yet. On to the next step!

3.4) Reserve Space for CP/M Slices: At the **>>** prompt, press the letter **`R`** to reserve space for the CP/M slices that already exist on this microSD card.

```
>>R
Reserve how many CP/M slices (8 max.) [8]:
```

Let's talk a bit about CP/M slices. These are 8MB disks that CP/M uses to store data. The **`dd`** operation done earlier created four such slices. If you want to be able to add more slices to the disk later, you can choose to max out the reservation to eight slices. This is the path I chose.

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition: (continued)

3.4) Reserve Space for CP/M Slices: (continued)

My output (with my input) looks like:

```
Reserve how many CP/M slices (8 max.) [8]: 8
8 CP/M slices have been reserved
>>
```

Here's a data point: Each CP/M slice occupies 65 virtual cylinders on the SD Card. Therefore, 8 slices takes up 8 X 65 = 520 cylinders. If you opted for reserving only 4 slices, 4 X 65 = 260 cylinders. These numbers will be important in the next few steps.

3.5) Create the New Partition: Now we are ready, with space reserved (and recalling our data point on the previous page) to create the new partition. Here we go. At the **>>** prompt, key in **N1** followed by **[ENTER]**. You should see something like:

```
>>N1
Starting Cylinder (default 520):
```

if you are reserving 8 cylinders. If only reserving 4, you should see:

```
>>N1
Starting Cylinder (default 260):
```

I next see:

```
>>N1
Starting Cylinder (default 520): 520
Ending Cylinder (or Size= "+nnn"): 120800
```

I have entered an ending cylinder instead of a size because I did the arithmetic back in step (3.2) to find the number of cylinders on my SD Card. *If you don't know* the numbers for your SD Card, use the **Size=** option by entering **+XXG** where **XX** is the GBs this partition will occupy. (If it's too big, **FDISK80** will truncate for you automatically.)

After entering the ending cylinder value and pressing **[ENTER]**, I printed the partition table again to check it by pressing **P** at the >> prompt. Here's what I see:

```
>>p
Nr  ---Type- A --      Start         End    LBA start  LBA count  Size
 1     FAT16   06    520:0:1  1023:15:16      133120   30791680   15G
 2             00       *** empty ***
 3             00       *** empty ***
 4             00       *** empty ***
   Reserved 8 x 8Mb CP/M slices
>>
```

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition: (continued)

3.5) Create the New Partition: (continued)

NOTE: The value highlighted in the above table shows an ending cylinder of 1023. This is the maximum that will ever be displayed due to constraints of the partition table format. The Logical Block Addressing (LBA) count will show the correct amount of space available on the SD card.

3.51) Selecting the Correct Partition Type:

So my new partition is in the table. Now I have to edit this partition to select a new type, *since FAT16 is not the desired format.*

(NOTE: The steps that follow below are not really needful, but I prefer to continue to include them so that I know what I selected. The ***FAT FORMAT*** *operation that will be performed a little later on will automatically update the partition type when it does the actual formatting.* ***If desired, skip ahead to step (3.6).)***

At the >> press **T1** and **[ENTER]**. What I see looks like:

```
>>T1
New type (in hex), "L" lists types: L
 00 empty         06 FAT16         0e FAT16 lba     81 Minix
 01 FAT12         07 NTFS/HPFS     0f W95 ext'd     83 Linux
 04 FAT16<32M     0b FAT32         32 UNA slice     8e Linux LVM
 05 DOS ext'd     0c FAT32 lba     52 CP/M
New type (in hex), "L" lists types: 0C
>>P
Nr  ---Type- A --       Start          End   LBA start  LBA count  Size
 1 FAT32 lba   0c    520:0:1  1023:15:16      133120   30791680   15G
 2             00       *** empty ***
 3             00       *** empty ***
 4             00       *** empty ***
   Reserved 8 x 8Mb CP/M slices
>>
```

As you can see above, I've chosen to list the common partition types so I can choose 0c (FAT32 LBA), and I then printed the partition table to check it.

{Please continue to the next page for step (3.6)}

3.0) Procedure for Using FDISK80.COM to add a FAT32 Partition: (continued)

3.6) Writing the Partition Table to the SD Card:

This next step is the knuckle-biter. Now we actually have to write the new partition table to the card. Yep, this is for all the marbles. If we press **Q** right now, we can back out of **FDISK80** and no harm; no foul (but we also accomplish nothing). Press the **W** key at the **>>** and we get:

```
>>W
Do you really want to write to disk? [N/y]: Y
Okay
FDISK exit.

B>
```

When I answered the question, "**Do you really want to write to the disk? [N/y]:**" with **Y**, quick as a flash FDISK80 said **Okay** and printed the message **FDISK exit** and we're back at the **B>** prompt. The deed is done! We've created a Fat32 Partition. Now we need to use **FAT.COM** from G: to format and access it.

{Please proceed to the next page.}

4.0) Procedure for Using **FAT.COM** to Format the New Partition.

4.1 Overview:

FAT.COM was created by Wayne Warthen to support FAT partitions on micro SD cards. This enables transfer of files from a non-CP/M system to the SC126's CP/M slices on that same micro SD card.

FAT.COM has a number of options that can be listed by simply typing in **FAT** at the **G>** prompt, so let's get there.

```
B>g:
G>FAT

CP/M FAT Utility v0.9.7 (beta), 11-Oct-2019 [RomWBW HBIOS]
Copyright (C) 2019, Wayne Warthen, GNU GPL v3

Usage: FAT <cmd> <parms>
  FAT DIR <path>
  FAT COPY <src> <dst>
  FAT REN <from> <to>
  FAT DEL <path>[<file>|<dir>]
  FAT MD <path>
  FAT FORMAT <drv>

CP/M filespec: <d>:FILENAME.EXT (<d> is CP/M drive letter A-P)
FAT filespec:  <u>:/DIR/FILENAME.EXT (<u> is disk unit #)

G>
```

The option we will now use is **FAT FORMAT**. Notice the highlighted line, where it says, in effect, that FAT filesystems are accessed by the **FAT.COM** command by *unit number*, NOT drive letter.

Remember what we found this unit number to be? For me, it's 3. Therefore, when I am addressing my SD Card with **FAT.COM**, **3:** is its unit number. You should use the unit number *you found earlier in step 3.11.* So here I go:

```
G>FAT FORMAT 3:
```

and I see:

```
About to format FAT Filesystem on Disk Unit #3.
All existing FAT partition data will be destroyed!!!

Continue (y/n)?
```

Answering the **Continue (y/n)?** prompt with **y** will start the process of formatting. One of the status LEDs on your SC126 will start blinking, indicating that writing is taking place. The word **Formatting**... appears while this happens (about 42 seconds for my 16 GB drive) followed by **Formatting**... **Done** and that's it. Your SD Card is now ready for use.