CoCo SDC

About the cover image:

Darren Atkinson designed the CoCo SDC hardware and software.
http://cocosdc.blogspot.com

Printed circuit board manufactured by Ed Snider.
https://sites.google.com/site/thezippsterzone/

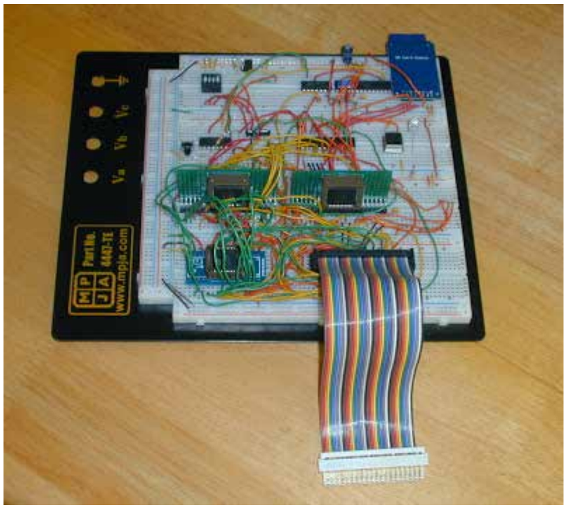
The clear acrylic case designed by Tim Lindner.
https://youtu.be/OdSOUcd60Ok

This manual takes inspiration from Brian Blake's original.

Table of Contents

1 What is the CoCo SDC?

The CoCo SDC is a home-brew project for the TRS-80 Color Computer (CoCo). It has been in various stages of development since 2009. The original plan was to provide floppy controller emulation which worked in conjunction with the Drivewire server. That idea was eventually scrapped in favor of a self-contained system using an SD card.

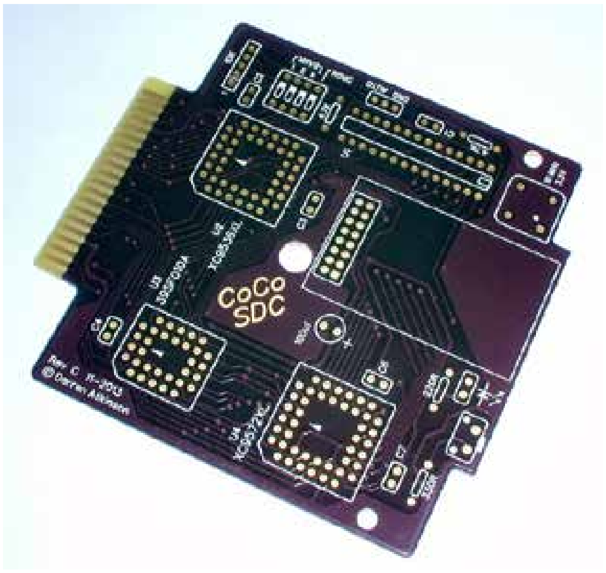

CoCo SDC Prototyping.

A number of high capacity storage solutions have previously been developed for the CoCo, including a MicroSD card interface, a handful of IDE and SCSI interfaces and the very popular Drivewire server.

One drawback of these offerings has been that they aren't compatible with software that was written to interact directly with a floppy disk controller. This isn't so much a problem if you are primarily using the CoCo for BASIC programming or running OS9 software. There are however a number of titles (mostly commercial games) that fail to work with those other systems.

The CoCo SDC aims to solve the compatibility problem by combining the traditional "software hook" approach with a robust emulation of the floppy controller in hardware. This dual mode implementation provides excellent performance for the majority of software which "plays by the rules" while adding a high degree of compatibility with those titles that employ floppy-based copy protection schemes or simply choose to roll their own floppy drivers.

An enhanced LBA access mode has also been incorporated into the firmware, allowing the CoCo SDC to go beyond simply emulating floppy disks and interface with virtual hard disk images as large as 2 gigabytes. Two separate disk images (floppy or hard disk) contained on the same SD card may be "connected" simultaneously.


CoCo SDC Revision 3 Board.

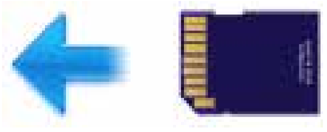
Also on board is 128K of Flash memory which is divided into 8 banks of 16K. These 16K banks are both hardware and software selectable and occupy the cartridge ROM space from $C000 to $FEFF.

One bank of the Flash memory is used to hold the SDC-DOS code which is yet another patched version of Microsoft's Disk Extended Color BASIC 1.1. Included in SDC-DOS are additional commands to mount disk image files on the SD card, program the Flash and execute ROM images contained in the Flash. Drivewire disk support is also included in SDC-DOS.

Features and Specifications

- Atmega 328P AVR micro controller @ 10Mhz
- Custom 512 byte boot-loader allows firmware to be updated by the CoCo
- 128K In-System-Programmable Flash
- Accepts SD/SDHC cards formatted with FAT16 or FAT32 file system
- Emulates a Tandy Floppy Disk Controller
- Emulate Dragon DOS floppy controllers
- LBA access mode for virtual hard disk support
- Extensions to Disk BASIC in SDC-DOS for disk image manipulation
- Drivewire disk protocol with auto-speed configuration for CoCo 1, 2 or 3
- "Disk Switch" button to support multi-disk programs
- PCB can be mounted in a Tandy FD-502 enclosure

Hardware Guide


SD Card Direction.

The SD card socket is a Push-Push type. When removing the card, always push in to release the latching mechanism before sliding the card out. Never use force to pull the card out of the socket. The card must be inserted into the socket upside-down (label facing down, contacts facing up). Use only SD or SDHC cards with the CoCo SDC.

Insert the card into the socket before applying power to the CoCo or Mult-Pak Interface. When power is applied, the LED on the CoCo SDC board should light up momentarily. If the LED does not turn off after a few seconds then this is an indication that the card was not recognized by the hardware. This can happen if the card has not been formatted with a FAT16 or FAT32 file system. It could also indicate that the card was not inserted properly or that there is a problem with the CoCo SDC itself.

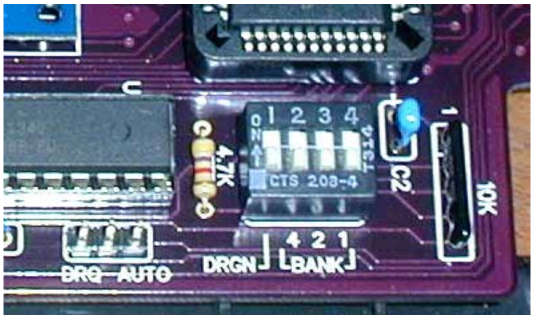
Although SD cards are hot-swappable, the CoCo SDC firmware does not handle that situation very well. It's recommended that you completely shutdown the CoCo and MPI before swapping cards.

Jumper Settings

The three-pin jumper strip provides two mutually exclusive options for board configuration; Cartridge Auto-Start and Dragon DRQ Mode. The default setting has neither option enabled (no jumper installed).

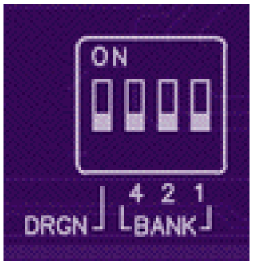
Installing a jumper between the center pin and the AUTO pin connects the Q clock to the CART interrupt pin. This causes the computer to automatically start executing the program in the selected Flash bank at power-up. Do NOT use this option to auto-start SDC-DOS or other Disk BASIC ROMs.

Installing a jumper between the center pin and the DRQ pin is required to support emulation of a Dragon DOS floppy controller. Do NOT install a jumper in this position when using the board with a CoCo.



Jumper & DIP Switches.

DIP Switch Settings



The board includes a 4-position DIP switch that is used to configure which bank of Flash is active at power-up or reset and which addressing scheme is used to communicate with the controller.

CAUTION: Make sure the computer's power is off before making any changes to the DIP switch settings!

Three of the switches specify the Flash bank to activate upon power-up or system reset. The switches are labeled on the board as 4, 2 and 1. The eight Flash banks are numbered 0 to 7. Place only those switches whose sum equals the desired bank number into the ON position. For example, to select bank 5, place the switches labeled 4 and 1 into the ON position and leave the switch labeled 2 in the OFF position. The board is provided with SDC-DOS in bank 0 of the Flash and all three switches in the OFF position.

The DRGN switch selects the address scheme for the controller. In the OFF position the controller will use the CoCo address scheme. In the ON position, the controller will use the Dragon DOS address scheme. The different schemes are summarized in the following table.

Usage	CoCo Address	Dragon Address
Drive Control Latch	FF40	FF48
Flash Data Register	FF42	FF4A
Flash Control Register	FF43	FF4B
Command/Status	FF48	FF40
FDC Track Register I/O Register 1	FF49	FF41
FDC Sector Register I/O Register 2	FF4A	FF42
FDC Data Register I/O Register 3	FF4B	FF43

CoCo & Dragon Address Schemes.

How is the SDC different from competing products?

- No reliance on expensive third-party modules like the 4D systems uDrive.
- Does not use a slow serial interface based on an obsolete part (6551 ACIA).
- True emulation of the floppy controller hardware for maximum compatibility.
- Supports the popular Drivewire protocol for PC-based disk images.
- Eight banks of in-system-programmable Flash instead of an EPROM.
- Ability to "switch disks" for multi-disk programs via a button on the controller.
- SD cards are FAT-formatted and require no special imaging utility for a PC/Mac.

Probably the only drawback of the device is the fact that the SDC does not come with an enclosure.

John Strong has been known to make 3D printed cases available. Here is his website: http://strongware.net/author/johnstrong/

2 Getting Ready For Fun

The Basics

There are a few very important things that must be touched upon before we get into the actual operation of the CoCo SDC:

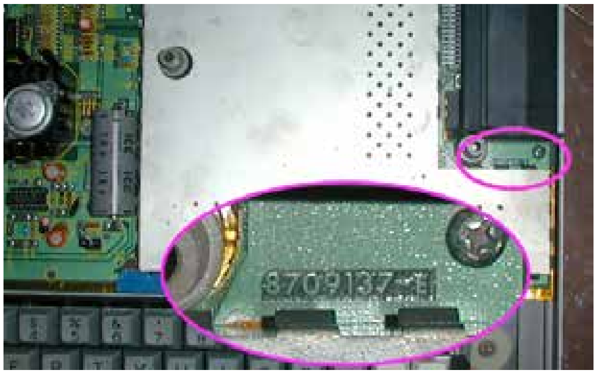
1. NEVER insert or remove the CoCo SDC into a CoCo that is turned on! Just like any other device that uses a Color Computer cartridge port, inserting the CoCo SDC into your Color Computer can damage the Color Computer, CoCo SDC, or both.

2. Although SD cards are hot-swappable, the CoCo SDC firmware does not handle that situation very well. It's recommended that you completely shutdown the CoCo and MPI before swapping cards.

3. The firmware in the CoCo SDC does not currently support long file names. You must ensure that the names of all files and directories which are to be accessible by the CoCo conform to the older 8.3 naming conventions.

D & E Compatibility Issues

The CoCo SDC is compatible with all versions of the Color Computer and Dragon Computer lines. However, after getting the CoCo SDC into the hands of some users, it was discovered that Flash programming does not work correctly on certain CoCo 1 motherboards. The two earliest CoCo 1 boards known as the 'D' and 'E' boards are the culprits.



Board Identifiers.

The Cartridge Select Signal ($\overline{\text{CTS}}$) on these boards exhibits too slow of a rise-time which causes problems for the high-speed Flash chip. This does not affect normal operation of the CoCo SDC in terms of being able to read data or execute code from the Flash. When writing to the Flash however, the slow rise time often results in incorrect data being stored in the chip.

There are a few options to deal with this problem:

1. Do not use a CoCo 1 with one of the aforementioned motherboards to program the Flash. This option is not ideal, especially if you don't have another suitable CoCo in your possession.

2. Use a Multi-Pak Interface when programming the Flash. The signal buffering in the MPI acts as a kind of filter for the $\overline{\text{CTS}}$ line, producing a nice clean transition. This is a good option if you do not wish to modify your CoCo and you happen to own an MPI.

3. Perform a simple modification to the CoCo 1 motherboard to fix the problem (see details below).

Identifying the problem boards

To determine if your CoCo 1 has one of the problematic motherboards you will need to open the case and look inside. The boards in question have a large metal shielded area that encloses all of the main logic chips including the RAM, CPU, SAM, VDG and PIAs. There should be a number printed on the board just below the cartridge port which ends with "-D" or "-E" as seen in the photos below.

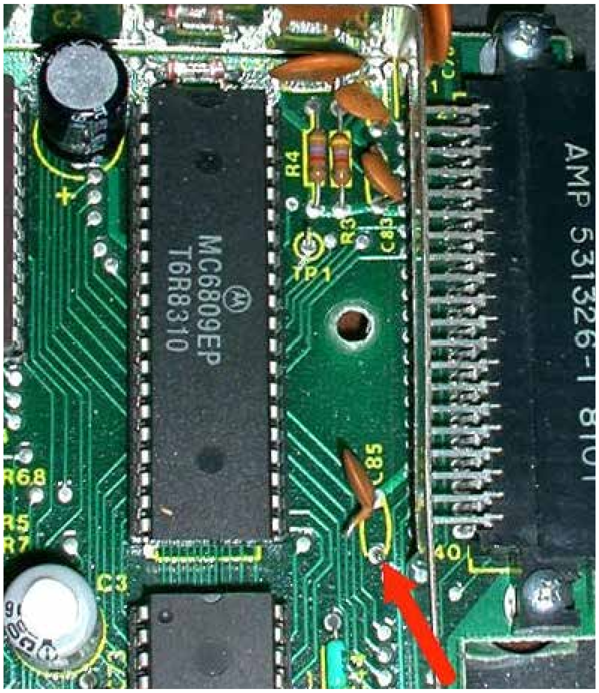
If your board has a smaller RF shield which only covers the SAM and RAM chips, or has a number printed on the board (near the front-left corner) that ends in '285' then this is what is often referred to as the 'F' board. The 'F' board does not exhibit the problem and needs no modification.

Motherboard Modification

Please note that any modification to the CoCo is performed at your own risk. Although it is highly unlikely that this modification will cause any problems with other hardware, I can't be held responsible for any damage or loss of functionality that may occur should you choose to go through with it.

The modification is rather simple and involves cutting just one leg of a capacitor. Be sure to disconnect power to the CoCo and discharge any static electricity from your body before touching any of the components inside the CoCo. The affected capacitor is located within the shielded area so you will need to remove the metal cover to gain access. Find the capacitor labeled C85 which is located next to the cartridge port (see photo).

Using an appropriate tool, cut the front leg (the one nearer the keyboard) of the capacitor to sever the connection. That's it! Replace the metal cover, close up the case and you are good to go.



CoCo 1 D & E board mod.

1 Available from: http://cocosdc.blogspot.com

Finding a Suitable Enclosure

Before you plug in the CoCo SDC, you should consider an enclosure for the device. Ideally, an FD-502 enclosure is preferred, as the FD-501 enclosure is slightly different and requires some modifications to work properly. Both enclosures need to be modified to provide easier access to the DIP Switches, while the FD-502 already provides easy access to the SD card slot and the push button by the SD card slot.

The SD card slot and push button ARE accessible, with the FD-501, however, it's advisable to trim some excess material from the housing in order to make it easier to access these features of the CoCo SDC.

Updating SDC-DOS

The SDCSETUP.DSK[1] image contains a utility program that can be used to install the firmware for a CoCo SDC controller. Both the micro controller code and the SDC-DOS (Disk BASIC) ROM image can be installed using this utility. When using a CoCo 1 or 2 a minimum of 32K RAM is required to perform an installation of SDC-DOS and 64K RAM is required to install the MCU firmware.

The disk image may be copied to an SD card or accessed via DriveWire. With the disk image mounted, run the utility by entering:

```
RUN "SETUP"
```

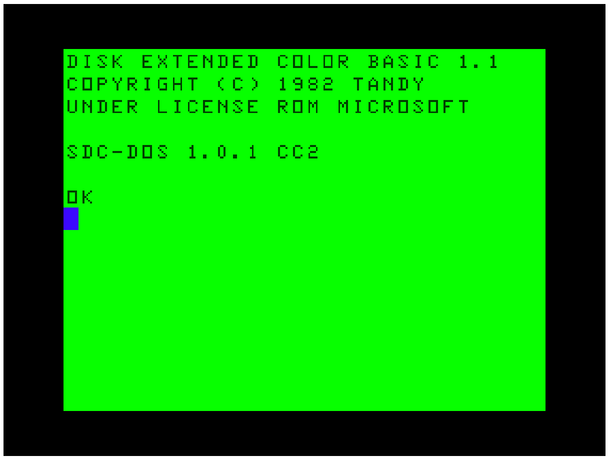
You will be presented with the following menu options:

```
V   DISPLAY INSTALLED VERSIONS
F   INSTALL MCU FIRMWARE
D   INSTALL SDC-DOS
Q   QUIT
```

Press the V key to display the version information of the software currently installed in the CoCo SDC controller. This will display both the MCU firmware version and the SDC-DOS version.

If your firmware is older, press the F key to begin the process of installing the ATmega MCU firmware. This will first load the firmware into memory and perform a checksum validation. The version number of the firmware to be installed is also displayed. Before installation begins you will be asked for confirmation by pressing the Y key. After installation is complete the CoCo will re-boot.

Press the D key to perform an installation of SDC-DOS. This will first load the ROM image into memory and ask which of the 8 Flash banks should be used as the destination. You may install over the version of SDC-DOS that is currently running if so desired. When installation is complete the CoCo will re-boot using the newly installed version (switching banks if necessary).



```
DISK EXTENDED COLOR BASIC 1.1
COPYRIGHT (C) 1982 TANDY
UNDER LICENSE ROM MICROSOFT

SDC-DOS 1.0.1 CC2

OK
```

Boot screen for SDC-DOS.

Rescuing After a Failed Update

This is mostly for folks who have attempted an update on a D or E board CoCo 1, and ended up with a CoCo SDC that will only boot to DECB. The recovery steps should be performed on either a modified D or E board CoCo 1, a CoCo 2, or a CoCo3 – as long as the DIR command hasn't been issued with any arguments prior to attempting the update.

Before we get to the actual steps, it might help to understand a little about the boot process of the CoCo SDC. Mounting a disk image is actually a task performed by the Atmega micro-controller, not SDC-DOS. The commands embedded in SDC-DOS tells the Atmega to mount the disk images at the location you specify. However, if your CoCo SDC crashes after an update attempt, it's likely you will not be able boot to SDC-DOS if you chose to over-write bank 0 of your Flash memory.

The way around this is to use a file the Atmega will automatically mount when the CoCo is first powered on. The STARTUP.CFG file is this file. If you create an ASCII text file named:

```
STARTUP.CFG
```

…and save it to the root of your SD card, the Atmega micro-controller will read that file at boot. The STARTUP.CFG file must contain the following line in order cause the Atmega to automatically mount the SDC101.DSK file:

```
0=SDC101.DSK
```

When this file is read by the Atmega micro-controller at boot, the CoCo SDC is already mounted to the SDC101.DSK.

DO NOT COMPLETE THE FOLLOWING PROCEDURES ON AN UNMODIFIED D OR E BOARD CoCo 1!

Recovery Steps

In order to recover from this condition, use the following steps:

1. Create the STARTUP.CFG file as described earlier
2. Copy STARTUP.CFG to the root of the SD card on a PC or Macintosh
3. Change the CoCo SDC DIP Switches to select bank 1 (for DECB)
4. Place the SD card in the slot on the CoCo SDC.
5. Make sure your CoCo is turned OFF!
6. Insert the CoCo SDC into the cartridge port of your Color Computer.
7. Turn on your CoCo.
8. Enter the DIR command (with no arguments if you're using a CoCo3 – just to be safe).
9. Enter RUN "UPDATE" and follow the prompts.
10. Your CoCo SDC should now be back to the way it was when it was shipped to you.

You can test this by powering down your CoCo, setting the DIP Switches back to bank 0, and restarting your CoCo. It should boot up to the DECB message with the SDC-DOS and version label.

3 Using the SDC

If you had to go thru the CoCo SDC update process, then you already are familiar with one of the enhanced commands. As stated earlier, SDC-DOS is yet one more version of an extended or patched DECB.

DRIVE - The Status

DRIVE has several uses. It's first and most basic use is to provide a display of the current drive configuration and status.

Typing DRIVE on your CoCo with the SDC plugged into it should give you a display of the drive mappings that resembles this:

```
0: ON   GAMEPAK1.DSK  0
1: OFF  ----          0
2: ON   DW #0         2
3: ON   DW #0         3
```

Each line in the table shows the current mapping information for one of the four logical drive numbers. The first column represents that drive number.

The second column in the table indicates whether or not Disk Image mode is currently on or off. When Disk Image mode is off, the corresponding real floppy drive will be used instead. Assuming a floppy controller is present.

The third column identifies the current Disk Image being mapped to the drive number. There are two possible configurations for this:

1. For an image located on an SD card this will be the name of the image file. (Only the name of the disk image will appear here, not the full directory path)
2. For a Drivewire image this will be "DW #n" where n is a Drivewire server virtual drive number.

The fourth column shows the index of the current virtual floppy disk within the larger 'hard disk' image file (0 - 255).

DRIVE is also used to assign disk images to drives on the CoCo – real drives or virtual drives.

DRIVE – Mounting SD Based Images

The DRIVE command is multi-faceted. As stated above, issuing just the DRIVE command will give you a status on the CoCo SDC's drive mapping. However, the DRIVE command, as displayed in the SDC-DOS updating section, also instructs the Atmega micro-controller to map the disk images stored on the SD card to a virtual drive.

The proper context for the command is:

```
DRIVE n, "path name"
```

Where n is the drive number; path name is the folder that holds your target disk image.

For example, issuing:

```
DRIVE 0, "APPS/TW64/TW64.DSK"
```

Would tell the Atmega to map DRIVE 0 to the TW64.DSK image in the folder listed in the path name.

Another example; let's say you want to play Pitstop II, on the PITSTOP.DSK image, located in the Epyx folder under games. The command you would issue is:

```
DRIVE 0, "GAMES/EPYX/PITSTOP.
DSK
```

Entering the DIR command would display a list of the files on the disk image, just like it would with real floppy hardware. You can LOAD & RUN or LOADM & EXEC just as you would with conventional hardware.

Multiple Disks

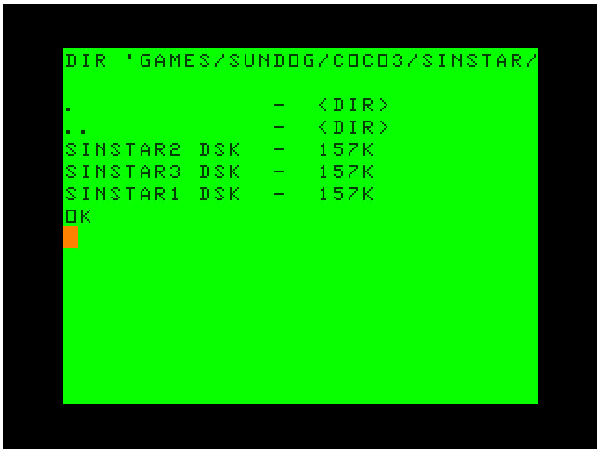
As discussed earlier, one of the features CoCo SDC has over products that preceded it to market is the ability to use software that contains multiple disks, as well as software that uses non-standard DSKCON routines.

In order to use games and applications that utilize multiple disks, the CoCo SDC has a way of knowing when this is necessary. The disk images for multi-disk games and applications need to be located in the same folder, and the last character of the disk title must be a number. An example from early will be used – Sinistaar.

Entering the following command:

```
DIR "GAMES/SUNDOG/COCO3/SIN
STAR/
```

Returns the following:

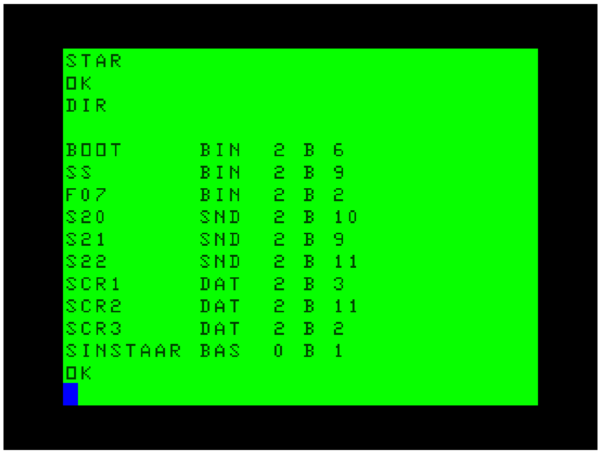


```
DIR 'GAMES/SUNDOG/COCO3/SINSTAR/
.                 -   <DIR>
..                -   <DIR>
SINSTAR2  DSK     -   157K
SINSTAR3  DSK     -   157K
SINSTAR1  DSK     -   157K
OK
```

Now, to mount the first disk image in the folder, simply enter:

```
DRIVE 0, "GAMES/SUNDOG/COCO3/
SINSTAR/"
```

When the CoCo SDC mounts this image, the red LED on the SDC will blink one time – indicating that the lowest numerical disk in the folder has been mounted – in this instance, one. Entering the DIR command now will result in SINSTAR1.DSK's directory:



```
STAR
OK
DIR

BOOT       BIN   2  B  6
SS         BIN   2  B  9
F07        BIN   2  B  2
S20        SND   2  B  10
S21        SND   2  B  9
S22        SND   2  B  11
SCR1       DAT   2  B  3
SCR2       DAT   2  B  11
SCR3       DAT   2  B  2
SINSTAAR   BAS   0  B  1
OK
```

From here, you would RUN "SINSTAAR" to load and run the game. When prompted for disk number two, simply push the button on the CoCo SDC, next to the SD slot, one time. The red LED will blink twice indicating that disk two is now selected. Follow the prompts on screen each time the game or application asks for the next disk, press the CoCo SDC button to select the next numerical disk and continue.

DRIVE – With Wildcards

Wildcard characters (* and ?) can be used in the file portion of the path name but not in the directory portion(s). Assuming there were no other files in the GAMES directory whose name started with the letters CH, the above command could be shortened to:

```
DRIVE 0,"GAMES/CH*.DSK
```

You may also omit the extension from the file name. In this case the system will first try to mount a file with the given name that has no extension. If no such file exists then .* is substituted for the missing extension and the system uses the first wildcard match, if any. This means the above command could be further shortened to:

```
DRIVE 0,"GAMES/CH*
```

Startup Configuration File

Entering a DRIVE command every time you start up the CoCo can be inconvenient, especially if you tend to use a particular disk image file on a regular basis. To alleviate this problem you can add a startup configuration file to the SD card.

You will need to use a computer with an SD card reader to create a plain-text ASCII file in the root directory of the SD card. The name of the file must be "STARTUP.CFG". The contents of the file may contain lines of text which specify the initial mount points for drives 0 and/or 1 as shown in the example below.

```
0=Nos9Lev2.dsk
1=Utils.dsk
```

You can also specify the path name of the directory to be set as the Current Directory:

```
D=/CoCo/Games
```

Make sure all file and path name components conform to the 8.3 naming conventions.

DIR

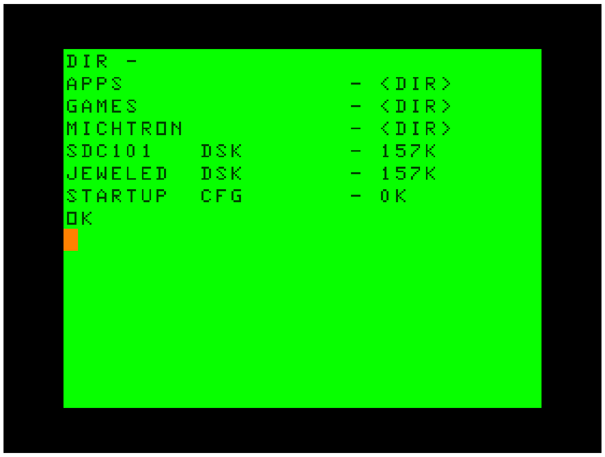
Let's say you have an SD card arranged into folders, and you want to load one of your favorite games, but you cannot remember for sure which folder the image is in. That's where the DIR command will come into play. The DIR command in SDC-DOS works very much like DIR in DECB, in that it gives you a directory listing of the current disk – be it a real floppy disk or a disk image assigned to a drive using the DRIVE command.

Just entering DIR after first powering on the CoCo, with the CoCo SDC inserted will most likely end in an I/O? error since no image was mounted. This may not be the situation if you had to use the STARTUP.CFG file discussed earlier, so we'll start with the assumption of no mounted disks.

When used with an SD/SDHC card, entering:

```
DIR -
```

Will result in a directory listing of the root of the card as shown below, including any files, disk images or folders:



```
DIR -
APPS                   - <DIR>
GAMES                  - <DIR>
MICHTRON               - <DIR>
SDC101    DSK          - 157K
JEWELED   DSK          - 157K
STARTUP   CFG          - 0K
OK
```

DIR results of SD Card.

You can also look into folders to see what disk images reside inside them. For example, I know that Sinistaar by Sundog Software is a child folder to the Sundog folder. Using the following command, you can work down to find out what the disk names are for Sinistaar:

```
DIR "GAMES/"
```

Lists the files and folders in the GAMES folder.

```
DIR "GAMES/SUNDOG/"
```

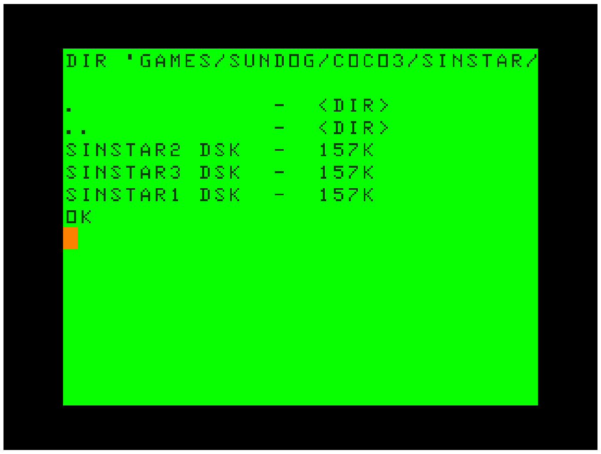
Lists files and folders in the GAMES/SUNDOG folder.

```
DIR "GAMES/SUNDOG/COCO3"
```

You should have the picture by now…

```
DIR "GAMES/SUNDOG/COCO3/
SINSTAR/"
```

That last command will result in the following:



```
DIR "GAMES/SUNDOG/COCO3/SINSTAR/
.                -   <DIR>
..               -   <DIR>
SINSTAR2 DSK     -   157K
SINSTAR3 DSK     -   157K
SINSTAR1 DSK     -   157K
OK
```

DIR results of subdirectory of SD Card.

That is a lot of typing to pull a directory, but, this depends solely on the file structure of your SD card.

DIR can also use wildcard characters (* and ?). For example, if you're looking for disk image and you only remember the first few characters, try:

```
DIR "GAMES/SIN*"
```

Which returns a list of games starting with the characters SIN.

Further, you can also search for specific files extensions.

For example:

```
DIR "MUSIC/*.ORC"
```

Returns a list of files with the ORC file extension. Likewise, entering:

```
DIR "APPS/*.DSK"
```

Returns a list of disk image files in the APPS folder.

Setting Current Directory

You can specify a Current Directory for commands that access the SD card. Once specified, all subsequent commands that refer to files or directories on the SD card are relative to the Current Directory unless the path name begins with a slash (/).

Examples:

```
DIR = "GAMES/ACTION"
DIR = ".."
DIR = "/"
```

Explaining DIR

Information displayed by the DIR command for each item is presented as 4 columns; Name, Extension, Lock Status and Size.

```
CASINO     DSK   -   157K
EGYPT      SDF   -   228K
GAMEPAK1   DSK   L   157K
GAMEPAK2   DSK   -   157K
GR2K       <DIR>
```

When an L appears in the third column instead of a hyphen (-), it indicates that the file is locked. A locked disk image may still be mounted, but you cannot make changes to its contents. Any attempt to use commands such as SAVE or KILL on a locked image will result in a ?WP ERROR.

For files, the fourth column displays the size of the file in kilobytes. For directories, the fourth column simply displays <DIR>.

Locking Disk Images

There are several way to mark a file as read only. Primarily, the CoCo SDC honors the FAT 16/32 read only attribute.

On Mac OS X this is accessible by choosing “Get Info” from the file menu while the file is selected. In the resulting window turn on the “Locked” check box.

On Windows, files can be lock by displaying the Properties and clicking the “Read-only” check box.

In addition, the SDF and VDK file formats include an internal value to mark the image as read only.

Creating New Disk Images

You can create a new, blank disk image file on the SD card by adding the word NEW as a final parameter to the DRIVE command. If the specified file already exists it will not be erased or replaced.

```
DRIVE 0,"SYSTOOLS.DSK",NEW
```

To create a single sided disk image in the SDF format use the NEW+[1] option.

```
DRIVE 0,"SEVENLNK.SDF",NEW+
```

To create a double sided, 40 track, disk image in the SDF format use the NEW++ option.

```
DRIVE 0,”SEVENLNK.SDF”,NEW++
```

A new SDF disk image is like an un-formatted floppy disk. You will need to use the DSKINI command to format the SDF image otherwise all disk operations will result in IO errors.

Ejecting a Disk Image

In most cases it is not necessary to eject disk images under SDC-DOS. To switch disks you can simply mount a new image in place of an existing one. One situation where the need to eject does arise is when you want to move an image to a different drive number.

For example, if you try to mount an image in drive 1 that is already mounted in drive 0, the system will produce an ?AO ERROR (already open). To accomplish this you must first eject the image from drive 0 by using the UNLOAD argument in the DRIVE command:

```
DRIVE 0,UNLOAD
```

Using the CoCo SDC with Drivewire

Having the functionality of storing and accessing your disk images right from an SD card is great. The possibilities are almost endless. However, the CoCo SDC is not a one trick pony.

1 In SDC-DOS 1.2 the system was changed to create 40 track SDF images instead of 35 Track.

Years ago Drivewire was released to act as a disk image file server for Color Computer users. With the Drivewire Server software running on a PC or Mac as a server application, a user could connect to their Color Computer to the server machine bit banger <-> COM port and viola, you could change disk images at will on the server, while being able to run almost all of your favorite software.

The CoCo SDC has the Drivewire protocol already built in, and will communicate with a PC or Mac running the Drivewire server software.

Connecting via the Color Computer

To access disk images on the Drivewire server, you use the DRIVE command as previously explained; instead of a string argument identifying an image file on the SD card, you provide a Drivewire virtual drive number (prefixed with #) in the range of 0 to 63:

```
DRIVE 2,#0
```

If you have a virtual 'hard disk' image containing an array of up to 256 floppy images, you can specify the index of the desired floppy image as a third argument to the DRIVE command:

```
DRIVE 2,#0,125
```

What this does is assign whatever disk image you have pre-loaded in Drivewire to virtual floppy #2 in SDC-DOS. Thats all. You cannot actually change the disk image from SDC-DOS.

Of course in Drivewire, you need to have disk images assigned to a floppy disk. Once the CoCo SDC has been assigned to a Drivewire disk position, you can switch disks at will with the Drivewire GUI on the host computer.

It should be stated here that Drivewire was created for use with HDB-DOS, a product developed and sold by Cloud 9.

There is no support in SDC-DOS or the CoCo SDC for sending a signal over the cable asking Drivewire to switch disk images. Of course if you have the server running on a nearby PC you should be able to use the server's UI to do this, allowing you to use games and applications that reside on multiple disks.

It should also be pointed out that Drivewire support in SDC-DOS was provided more as a convenience feature for transferring files between a PC and the SD card. You don't get quite the same feature-set that HDB-DOS provides.

An example of this would be if you wanted copy a disk image – let's say Flight Simulator II – from your PC to the CoCo SDC. The steps you would take are:

Mount the FSII disk image to Drive 0 in Drivewire.

Create a new floppy image on the CoCo SDC with the following command:

```
DRIVE 0, "GAMES/FSII.DSK",NEW
```

Assign the CoCo SDC drive 1 to the Drivewire server with the following command:

```
DRIVE 1, #0
```

Finish the process by entering the following command:

```
BACKUP 1 TO 0
```

Finally, the floppy controller emulation features of the CoCo SDC are only available to images located on the SD card. The Drivewire support is affected by the same compatibility issues that apply to HDB-DOS or CoCoNet since it is implemented completely in software by SDC-DOS.

If you run a program which has its own floppy I/O routines from an image on the Drivewire server, it will run until the point where those routines are first executed. At that time the CoCo SDC will detect that the floppy hardware is being accessed and try to translate it to the corresponding SD card image (if any). This would be indicated by the red LED on the CoCo SDC turning on and staying on when the disk access would normally occur, causing your Color Computer to enter into a locked state.

Accessing Real Floppy Disks

In addition to disk images located on SD cards and the Drivewire server, SDC-DOS will also provide access to real floppy disks if you have a Mult-Pak Interface and a separate floppy controller. When powering-up the system, the switch on the MPI must be set to the slot number containing the CoCo SDC board.

SDC-DOS will examine the hardware plugged into the MPI looking for the highest numbered slot containing a floppy controller. If found, the floppy controller will be used for any drive number in which Disk Image mode has been turned off.

To turn off Disk Image mode for a particular drive number and thereby utilize the floppy controller, specify OFF as the second argument in the DRIVE command:

```
DRIVE 1,OFF
```

Turning off Disk Image mode disables any SD card image or Drivewire image currently mounted under the specified drive number, but does not eject (unload) the image. You can reestablish access to the underlying image by simply turning Disk Image mode back on for that drive:

```
DRIVE 1,ON
```

Automatic Program Execution

When SDC-DOS starts following power-on or cold reset, it will search the RS-DOS formatted, mounted disk images (as specified in the STARTUP.CFG file) for a BASIC program file named “AUTOEXEC.BAS”. If such a file is found, it will be automatically loaded and run. You can hold down the SHIFT key to bypass this feature.

EXP

The EXP command has been added to provide quick access to a program for browsing the contents of the SD card (Explorer utility). Entering the EXP command will cause the system to search the root directory of the SD card for a disk image named SDCEXP.DSK. If found, the disk image will be automatically mounted in drive 1. If the disk image contains a BASIC program named AUTOEXEC.BAS then it will be automatically loaded and run.

You can use this command to start The SDC Explorer program described here: http://cocosdc.blogspot.com/p/sdc-explorer.html

DEF DW = n

You can now change the DriveWire speed configuration by using the DEF DW=n command. Specify the CoCo platform number (1, 2 or 3) as the value for n to select the desired speed:

n	Speed	Comment
1	38,400 bps	
2	57,600 bps	Faster op-amp required to work correctly on a CoCo 1
3	115,200 bps	CoCo 3 only

When running on a CoCo 1 or 2 you cannot specify 3 as the parameter since those machines are not capable of running at true double-speed.

4 Using the Flash

An issue involving Flash programming on a CoCo 1 has been discovered.

Please see Chapter 2, section "D & E Compatibility Issues" for details.

The CoCo SDC contains 128K of Flash memory which is divided into eight banks of 16K. All eight banks of the Flash are user-programmable. The board is provided with SDC-DOS pre-programmed into bank 0 and stock Disk BASIC 1.1 in bank 1. SDC-DOS adds extensions to the Disk BASIC commands which make it easy to take advantage of the Flash memory.

Care should be taken when using these commands to avoid accidental destruction of data. You should always keep copies of the Flashed data elsewhere so it can be re-programmed if necessary.

Running a Cartridge Image

The Flash memory will typically be used to hold images of cartridge-based software (Program Paks).

The RUN command in SDC-DOS has been extended to facilitate the execution of a cartridge image from any of the 8 Flash banks. Simply pass the bank number, prefixed with the @ character, as an argument to the RUN command.

```
RUN @2
```

This has the effect of activating the specified bank and performing a cold re-boot of the CoCo. If the first two bytes of the cartridge image are "DK" then the normal start-up process occurs, allowing Extended BASIC to transfer control to the cartridge image at C002 during initialization. If anything else appears in the first two bytes then control is transferred to C000 immediately after the hardware is initialized.

Normally, pressing the RESET button on the CoCo will re-activate the Flash bank set by the DIP switches on the CoCo SDC board. You can force the system to retain the bank selection of the RUN command by suffixing the command with ,R:

```
RUN @2,R
```

After using this option you will need to fully power-down the CoCo (and Multi-Pak Interface) in order to restore the normal Reset behavior.

Erasing Banks and Sectors

The active Flash bank appears in the CoCo memory map from C000 to FEFF. Each of the 16K banks is further divided into four sectors of 4K:

Sector	Address Range
0	C000 - CFFF
1	D000 - DFFF
2	E000 - EFFF
3	F000 - FEFF

On a CoCo 3, the last 256 bytes (FE00 to FEFF) are not accessible using SDC-DOS 1.2 or earlier.

The Flash chip uses the sector divisions for erase operations. Before programming data into the Flash, the sectors to be programmed should first be erased. Erasing the flash has the effect of setting all bits to '1' resulting in byte values of FF. The KILL MEM command can be used to erase a single sector or an entire bank.

Erase all four sectors of bank 3:

```
KILL MEM @3
```

Erase only sector 2 of bank 6:

```
KILL MEM @6,&HE000
```

When erasing a single sector you can specify any address from C000 to FEFF. The entire sector containing that address will be erased. You are not allowed to erase any part of the active bank (the one from which SDC-DOS is running).

Writing to the Flash

Writing data to the Flash involves a process where one or more bits in a byte are cleared to '0'. Once a bit has been cleared it can only be changed to a '1' through an Erase operation. The WRITE MEM command is used to program one or more bytes.

```
WRITE MEM @bank, source, desti
nation, count
```

@bank	Bank number in which the data will be written (0-7)
source	Starting address of the source data
destination	Address in the Flash where the data will be written (C000-FEFF)
count	Number of bytes to write

The count argument will be clipped if necessary to prevent writing past the end of the Flash address space.

Copying a Block of Memory

When creating a utility program to manage the Flash, one feature that is often needed is the ability to quickly move a block of memory. Consider the situation where you wanted to copy the contents of one Flash bank to another. This would require that data from the source bank first be copied to a temporary buffer in RAM before writing it to the destination bank. The COPY MEM command has been provided for this purpose.

```
COPY MEM [@bank,] source, desti
nation, count [USING slot]
```

@bank	The Flash bank number to activate during the copy (0-7)
source	Starting address of the source block
destination	Address where the block will be copied to
count	Number of bytes in the block
USING slot	The Multi-Pak Interface slot to activate during the copy (1-4)

The bank and slot arguments are both optional and mutually exclusive. You can provide one or the other, but not both. The bank argument can be provided when you are copying data from a specific Flash bank on the CoCo SDC board. The slot argument allows you to copy data from the ROM of another cartridge when using a Multi-Pak Interface.

Be careful when using the COPY MEM command as it can easily crash the CoCo if a block is copied to a location used by the system.

The BASIC program listing below shows how to copy the contents of a ROM cartridge into one of the Flash banks of the CoCo SDC.
The example assumes an MPI is attached and the CoCo has at least 32K RAM. It does not perform any validation of the input parameters.

```
10 CLEAR 200,&H3FFF ' RESERVE A
16K RAM BUFFER AT $4000
20 INPUT "COPY FROM MPI
SLOT";SL
30 COPY MEM &HC000,&H4000,16384
-256 USING SL
40 INPUT "DESTINATION BANK";BK
50 KILL MEM @BK ' ERASE BEFORE
WRITE
60 WRITE MEM @BK,&H4000,&HC000,
16384-256
```

5 About File Formats

The CoCo SDC supports four different disk image formats. The primary format is referred to as the DSK format and can be used for the imaging of both floppy disks and hard disks. The secondary format, known as SDF, was created specifically for the CoCo SDC and is used for imaging floppy disks only. The third is the JVC format. The fourth is VDK, popular for use with Dragon emulators.

No special name or extension need be assigned to an image file for the purpose of format determination. When a disk image is mounted the firmware detects which format the image uses by examining the file to see if it contains an SDF format signature. Nevertheless it is recommended that an extension which is indicative of the image format be used for identification by humans.

DSK Images

The DSK image format is named for the extension most commonly appearing on such files. Images in this format consist of a simple sector array with each sector being 256 bytes in length. This is the most common format used in the CoCo world.

In order to be recognized as a valid DSK image, the file size must be an exact multiple of 256 bytes. The minimum file size is 82,944 bytes which is equal to 324 sectors or 18 tracks of a single-sided CoCo disk (enough to accommodate the Disk BASIC directory track).

The disk geometry associated with a DSK image is determined by the file size. For floppy images the number of sectors per track is always 18. There are either one or two tracks per cylinder (equal to the number of sides) and a maximum of 80 cylinders. The largest file size for a floppy image is 737,280 bytes or 2880 sectors (double-sided 80 cylinders).

Disk Geometry Table for DSK Images

File Size in Bytes	Sectors	Disk Type	Cylinders	Sides
Less than 82,944	<324	Invalid		
184,320 or less	≤ 720	FD	40	1
368,640 or less	≤ 1440	FD	40	2
737,280 or less	≤ 2880	FD	80	2
737,536 or more	> 2880	HD	80*	1*

* only when accessed through the floppy interface mode

An image with more than 2880 sectors is considered to be a hard disk. If a hard disk image is accessed using the floppy interface mode, only the first 1440 sectors can be used. In this situation those sectors are accessible as a single-sided 80 track floppy disk. The controller's LBA interface mode must be used to access sectors beyond the first 1440 in a hard disk image.

JVC Images

This disk image format is an array of sectors with a 1 to 4 byte header prepended to the front. The header bytes are described in the following table:

Byte Offset	Length	Description
0	1	Sectors Per Track. Default is 18.
1	1	Side Count. Default is 1.
2	1	Sector Size Code. Default is 1 (256 bytes per sector).
3	1	First Sector ID. Default is 1.

All of the bytes are optional, but are interpreted in the order listed in the table. If omitted their default values are assumed. The CoCo SDC requires the *Sectors Per Track* to be 18, and the *Sector Size Code* to be 1. It will honor a *Side Count* of 1 or 2.

VDK Images

This is also an array of sectors prepended by a header.

Byte Offset	Length	Description
0	2	ASCII ‘d’ and ‘k’.
2	2	Header size (little-endian).
4	1	Version of VDK format.
5	1	Backwards compatibility version.
6	1	Identity of file source.
7	1	Version of file source.
8	1	Number of tracks.
9	1	Number of sides.
10	1	Flags:

Bit	Meaning	Bit	Meaning
0	Write Protect	1	Advisory lock
2	Mandatory Lock	3	Disk Set
4	Unused	5	Unused
6	Unused	7	Unused

Byte Offset	Length	Description
11	1	Compression flags and name length.

The CoCo SDC will honor a *Number Of Sides* value of 1 or 2. It will also honor the *Write Protect* bit.

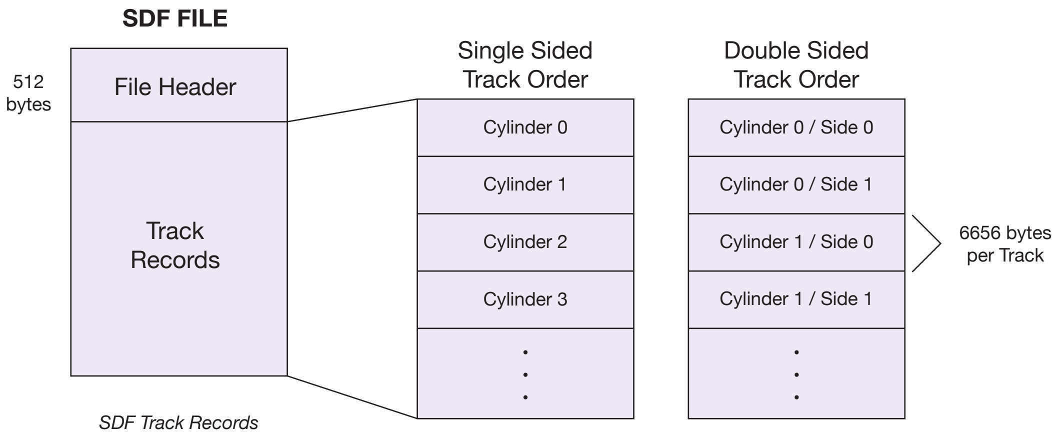
SDF Images

The SDF image format is used to represent floppy disks that have a non-standard layout which is anything other than 18 sectors per track and 256 bytes per sector using standard numbering of the tracks and sectors. The SDF format is similar to the DMK format supported by most CoCo emulators. It has been augmented to provide better performance within the limited resources of the Atmega328 micro controller.

The dmk2sdf program has been created for converting a DMK image to the SDF format. A Win32 command line executable along with the ANSI C source code can be downloaded using the link: https://goo.gl/q61D6s.

SDF File Format

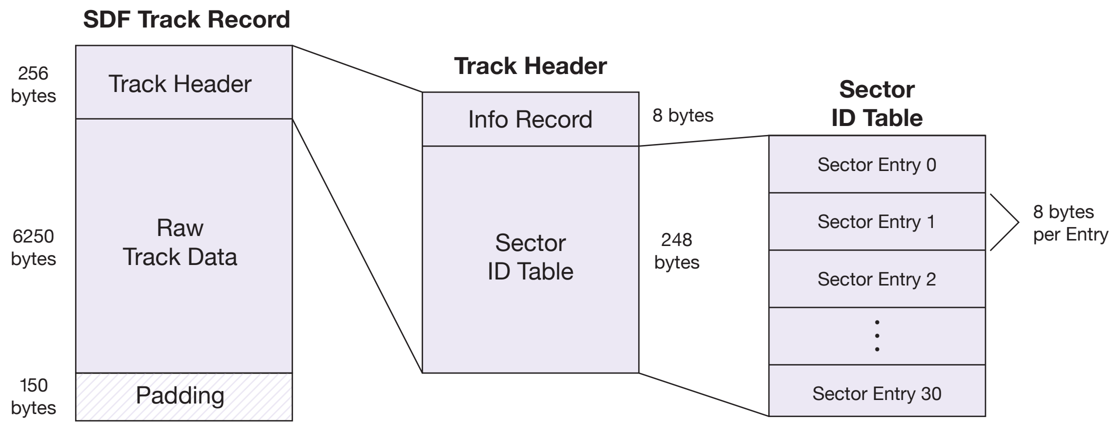
An SDF file consists of a header followed by a variably-sized array of track records. The track records are arranged in ascending order corresponding to their physical position on the disk (cylinder and side).



Contents of the SDF 512 byte File Header

Byte Offset	Length in Bytes	Description
0	4	Format signature and version string. The ASCII characters ‘SDF1’ appear at the beginning of the file to identify it as a version 1 SDF image. The numeric character may be incremented in future versions.
4	1	Number of cylinders (80 max).
5	1	Number of sides (1 or 2).
6	1	Write permission: 0x00 = Read/Write ; 0xFF = Read-Only.
7	1	Nested sectors flag: 0x00 = NO ; 0x01 = YES. This byte is set to 0x01 if the disk is known to use a copy-protection scheme in which the ID field for one sector is contained within the Data field of another.
8	504	Reserved. All remaining bytes in the header should be set to zero.

Following the File Header is the array of Track Records. Each track record begins with a 256 byte header and is then followed by 6250 bytes of raw track data. There are 150 bytes of unused padding at the end of a track record which are included to align every track on a 512 byte boundary within the file.



The fixed track size of 6250 bytes can accommodate either a single-density (125 kbps) or double-density (250 kbps) track at 300 rpm. The SDF format does NOT support 8 inch floppy disks or high density (500 kbps) images.

Any part of a track which is recorded in single-density has each byte written twice in succession. This preserves the correct spacing of data on mixed-density tracks.

Contents of the SDF 256 byte Track Header

Byte Offset	Length in Bytes	Description
0	1	Number of used entries in the Sector ID Table.
1	7	Reserved. All seven of the remaining Info Record bytes should be set to zero.
8	248	Sector ID Table.

Each entry in the Sector ID Table is 8 bytes in length and contains information about one sector recorded on the track. The total size of the table is 248 bytes and can accommodate a maximum of 31 sector entries for a single track. All used entries must appear sequentially from the beginning of the table. The unused entries must be filled with zeroes and placed at the end of the table.

Byte Offset	Length in Bytes	Description
0	2	The 14 low-order bits of this 16 bit field contain the offset from the beginning of the Track Record to the first byte of the sector's ID field within the raw track data. The two high-order bits (14 and 15) are used as flags. Bit 14 is set for a sector recorded in single-density. Bit 15 is set if the ID field has an incorrect CRC. This 16 bit integer field is stored in little-endian order (LSB first).
2	2	The 14 low-order bits of this 16 bit field contain the offset from the beginning of the Track Record to the first byte of the sector's Data field within the raw track data. The two high-order bits (14 and 15) are used as flags. Bit 14 is set if the sector's Data field uses a Deleted Data Mark. Bit 15 is set if the Data field has an incorrect CRC. This 16 bit integer field is stored in little-endian order (LSB first).
4	1	The Track Number byte copied from the sector's ID field.
5	1	The Side Number byte copied from the sector's ID field.
6	1	The Sector Number byte copied from the sector's ID field.
7	1	The Size Code byte copied from the sector's ID field.

Index