Manual for Benn Venn's SD Loader / Card reader for the VZ/Laser computer. V2.8

Give your VZ200/VZ300 or equivalent a new lease on life with this SD card loader. Load .VZ snapshots instantly onto your system, supports both Binary and Basic files. Includes 128kbytes of onboard sram to Max out your VZ system.

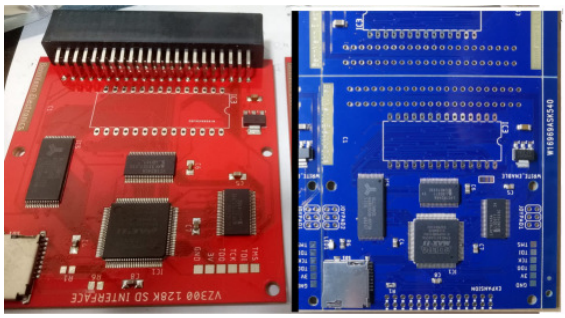
Intro

In Aug 2019, Ben Grimmett, an avid 8-bit enthusiast and entrepreneur of all Gameboy Z80 8-bit things, along with being a previous VZ owner, announced to the Facebook VZ & Laser group that he would create a SD card reader, with its own operating system, for these computers. All of the VZ & Laser community jumped with joy to hear such a thing. Eight weeks later his beta SD card reader was in operation. 50 were initially produced and went up for sale, and, within days just about all of them were sold out.

Moving forward through Covid and world wide lockdowns to around August 2022, Mark II release of the SD card reader was ready for production, and another forty were up for sale. With an updated and bug-fixed operating system, version 2.0 is back better than before.

https://bennvenn.myshopify.com/products/vz300-sd-loader

With it also being designed with an 8-bit I/O port along with a port for Gameboy gamepad joystick things, it comes with 128KB of ram that is accessible via bank switched memory. Great for storing graphics screens for animation.



The old and new versions of the SD Loader.

Benn Venn Electronics SD Loader, encased in a black plastic housing, printed and supplied by Igor Kromin.

Overview of the SD Card reader Operating System

A number of new commands are offered with the SD Card loader/reader.
Loading of BASIC and Binary files, and saving BASIC files are possible.
128 KB of memory is offered via bank switching, split into two pages, mapped from 8000-FFFF.
An I/O port on the reader itself for Gameboy gamepad type joypads with plastic 'knockouts' around the controller ports and the expansion port within the plastic case. They can be snapped out if you wish to install controllers or the expansion header, or left in there for the stock look.

The Laser200, Saloria Fellow, Texet 8000A that came with 2k user RAM with 2k Video RAM have a 2k, or 4k gap within the addressing when used with the SD Loader without modification. As such total memory available will still only be 2k. Your SD Loader for the use on these computers will require a different firmware, available from Benn Venn Electronics. The Loader was originally designed for a 6K user-ram with 2K Video ram VZ/Laser computer, and thusly the address gap will exist with these specific models. The firmware patch for the SD Loader for the use with these computers was released on the Facebook site under the files section on the 19th Nov 2022. Search for the name 'VZ300.POF' and it should be found. Once uploaded in to the Loader, this replaces the SD Loader's ram from 0x8000 to 0xFFFF, assuming only 2k of user ram is fitted above the 2k of video ram. It needs to be programmed with Altera/Intel USB Blaster. Talk directly to Ben should you need more information on this.

Plugging in the SD Loader, and typing the following, should show you your new total amount of RAM available to you in DOS BASIC. Note this does not indicate the extended RAM that is still available through the use of Bank switching. Something which is poorly explained to beginners in most manuals is this : 64k is the top of RAM that BASIC can show. This is absolute everything within the computer. You can read from any of these locations, but half of it is not what you call user friendly accessible RAM.
Take 64k, remove the 16k of ROM, remove the 10k of cartridge area, remove the 2k latch sound/keyboard area, and remove the 2k VIDEO RAM. This all equals 30k, leaving just 34k of RAM.
The SD Loader fills in the gap up towards the top of RAM which is missing if you had a 16k RAM pack installed. The Loader also then adds another 64k which is available for programming usage, however, "Bank Switching" is required, and only chunks of 16kb of this extended 64k are available at any one time. To the beginner this can be quite confusing and "goobledy gook". I could also be wrong with my above dribble – which certainly won't help anyone.

Remove the SD Loader, and type this:
PRINT PEEK(30897)+256*PEEK(30898)
Record the number. Then plug the SD Loader back in, and type it again. The number should have changed, indicating that you now have more user RAM memory.

The card reader BIOS O/S firmware was designed around a SDHC class memmory card being present. This means 2gb or greater. Ths O/S uses the FAT32 file system which allows for up to a 32 gig patition which is a lot of VZ/Laser software in anyones book.
Be aware when formatting your SD Card that the SD Loader project is expecting a MBR FAT32 format, and not a GUID FAT32 format. The difference in wording is subtle, but the physical format of the records, sectors, etc are vastly different. You may require a hex editor to wipe out the first few sectors on the SD Card in order to format the card successfully with a MBR FAT32 format.

The saving of a BASIC file is limited to eight (8) characters within the filename.
The directory listing has support for long filenames (greater than 8 characters), being displayed as '>'
Sub directories are shown in the VZ's inverse colour.
A small HELP screen is availabe from BASIC.

The SD new operating system commands are :

HELP	- shows a simple help page of commands.
DIR	- lists the files on the memory card.
CD"filename"	- Moves into the sub-directory named "filename"
LOAD"filename"	- Loads the program named "filename".
SAVE"filename"	- Saves the current program in memory as "filename"
JOY	- Enables the loaders joysticks I/O port. Disables normal VZ joysticks.
NOJOY	- Disables the loaders joysticks I/O port. Enables normal VZ joysticks.

CLI - Command Line interface

During the boot process, pressing <SPACE> will allow you into the command line interface. A few extra commands are added to within the CLI, listed below, which are mostly for memory manipulation - moving and writing commands. These are a simple way to modify or enter data into memory. Type HELP for a list of commands.

Note, careful with the keyboard inputs within the CLI, as key bouncing may occur.

Commands within the CLI are :

HEXVIEW address(16)	: View memory locations.
WRITE destination:datdatdat…	: Write directly to a memory location.
COPY src:dest:count	: Copy from scr to destination an amount of memory.
FILL destination:val:count	: Fill an amount of memory with a constant.
JUMP address(16)	: Jump to an address in memory.
IN address(8)	: Read a byte value in from a port I/O address.
OUT address(8):data	: Write a byte value to a port I/O address.
BASIC	: Boot to BASIC
DIR	: Perform a directory listing.
LOAD filename	: Load in a file named filename.
RUN	: RUN a program in memory.
CLS	: Clear the screen.

Formatting the memory card

Take a 2 to 64 gig micro SD card, (Tested: 64GB SDXC) Plug it into an adapter, plug the adapter into your favourite operating system on your home computer, and format away the card. MBR FAT32.
Preference for formatting is to use a Windows operating system to format the memory card.
Both Linux and OSX FAT32 formats can be a little off, (clearly for this to be a known factor, these formats are not 100% compatible with Windows). Perhaps the newer versions of these Operating Systems use the newer GUID properties rather than old-school MBR. Unexpected results can potentially happen when reading/writing to the memory card, and corruption to both data and filenames, and even data loss is quite possible. For 100% confidence in things always working, always format the SD Card within a known opeerating system that can guarantee a MBR FAT32 format. Eg, 200% confidence is absolute guaranteed when formatting is done by a Windows 7 system !.

A note from Ben mentions that cluster size is important, even though it can handle clusters from 512bytes to 64kbytes. If the Cluster is larger than the file size, the O/S doesn't need to link chains.

I personally have found that the cluster size of 16KB has worked fine, having stored well over 200 .VZ snapshot files in tens of sub-directories. Always nicely eject the SD card, otherwise data loss is quite possible if files in cache are still yet to be written. An ejection forces the writing and the closure of any files still in cache memory.

Upon power up the SD card will be initialised, FAT32 will be initialised. If VZDOS.VZ is found on the root directory, it will be copied to $4000 and executed. This allows the latest firmwares to be launched without reflashing the EPROM. BASIC is loaded by default. This is for both Mk I and Mk II Loaders.
From the Help : "IF VZDOS.VZ is found on the SD Card this new version of the BIOS will be loaded in placed of the default BIOS."

Mark I SD Card reader loaders, find, download, and copy over the v2.0 operating system file 'VZDOS.VZ' on to your memory card to upgrade this particular SD Card reader to the latest version.
Mark II of the SD card readers were supplied with v2.0 of the firmware BIOS operating system. At the time of editing, v2.2 of 'VZDOS.VZ' is now available and should be copied to the root of your SD memory card.

On the Loader PCB there is a jumper which enables EEPROM re-writing which allows you to update the firmware version via CLOADing, and this will directly to the EEPROM. Once finished, the jumper can be removed to disable re-writing.

VZDOS.VZ : BIOS firmware for SD memory card.
VZDOS.BIN : BIOS firmware for EPROM/EEPROM programming.
VZDOS.OBJ : BIOS firmware for EPROM/EEPROM programming. Exact same file as above .BIN
VZDOS.WAV : BIOS firmware for cassette tape CLOAD'ing to the EEPROM once the jumper is set.
These files can be found within the files section of the VZ/ Laser Fans Facebook group.

For early Mark I released SD card readers, with an early version of the firmware/O/S, that refuse to read memory cards correctly, try this fix.

Format Ultra 16Gb micro sd card.
Use hex program "HxD".
Load sd card physical drive
Go to top / offset 0.
zero out the entire Sector 0 of the physical memory card disk to wipe out the partitions.
Save drive.
Format normally. FAT32. select Default sector allocation.
Should work fine using SD loader DOS v1.8

An interesting feature to experiment with is the further use of the auto-boot 'VZDOS.VZ' file as an autostart program, which could be a selection menu, a game, or an auto boot graphics and music demonstration program.

Operating System Commands

HELP
From version 2.0, typing in 'HELP' will give you a one/two page screen showing some hints and tips.

Directory Listing
Type DIR to show the contents on the memory card. This will show 16 files and then stop advising you to press space to continue or press break to end the listing.
The directory listing will show long file names as > .
This is when a file is saved from Windows with a filename greater than eight characters.

Loading Files
On your home computer, scrounge the internet and download all of the .VZ snapshot files that you can find. Copy these over on to your SD memory card. Take the card and plug it into your VZ Card reader.
Perform a directory listing to see your files.
To load a particular basic or binary .VZ file, simply type :
LOAD "VZ FILE"
Note that the FAT32 system only allows for eight characters for the filename.
The operating system will detect if the file is basic (F0) or binary (F1). If it is a binary file, it will attempt to automatically execute the file. If it is a basic file, the file will load, but will leave you at DOS BASIC, allowing you to LIST or RUN the program, or CSAVE to tape as an example.

CD – change directory
The command CD will allow you to change from one directory to another.
CD"directory name" : will change from the current directory into the directory named "directory name"
CD".." : will change from the current directory, up one directory level.

Saving Files
SAVE"filename" will allow you to save BASIC programs in memory to the SD memory card.
Note that the file system only allows for a maximum of eight characters within the filename.
If you attempt to use nine or greater, a syntax error will occur.
If you attempt to save a file with the same filename that already exists on the memory card, the operating system will ask / confirm if you wish to overwrite. Nice touch Ben.

JOY
The JOY command will enable the SD card reader/loader joystick interface that is built into the reader. This will disable the normal VZ/Laser joysticks and its interface.

NOJOY
The NOJOY command will disable the SD loader joystick interface to allow the use of normal VZ joysticks that are attached to their own VZ/Laser interface.

Note: Sub directories need to be created on to the memory card from within the Windows operating system, (ie CMD → MD FILENAME). Or within Linux /OSX if this is working for you. (not recommended)

SD Card reader I/O Ports

55 RAM map. Set to 0 for VZ200 compatibility (RAM mapped from $9000-$FFFF
Set to 1 for VZ300 compatibility (RAM mapped from $B800-$FFFF
56 SPI port CFG for SD interface.
57 SPI port Data for SD interface.
58 Bit0 = EPROM(0) or SRAM(1) mapped to DOS ROM AREA (Note, SRAM write is enabled even when EPROM is selected. This means you can code a loop to read and write from the DOS ROM area to copy itself into ram before switching to SRAM)
Bit1 = SRAM BankLow - DOS ROM SRAM bank 0 or 1 ($4000-$67FF)
Bit2 = SRAM BankHigh - SRAM Bank 0 or 1 (Expanded RAM up to $FFFF)
Bit3 = on board LED if fitted. (If you mod your VZ's ROM /CS line, you can use this signal to disable the original ROM and map in SRAM allowing you to load BASIC rom's from SD card. This has not been implemented yet.
74 Control the SD Loader's Joypad SNES controlling interface. Enabled by default.
OUT74,0 : Disable the SNES interface. Enables the use of VZ joysticks on the OEM interface.
OUT74,1 : Enable the SNES controller interface. Disables the OEM VZ Joystick interface.

SNES Joypads

SNES pad IO addresses - The original VZ addresses are supported as well as the expanded joypad address space which gives you access to all 12 buttons on the SNES pad.

SNES pad I/O addresses. (All 12 buttons on the SNED pad are supported)

```
constant Cont1a          : integer := 46;
constant Cont1b          : integer := 45;
constant Cont2a          : integer := 43;
constant Cont2b          : integer := 39;
constant ContBa          : integer := 42;
constant ContBb          : integer := 37;
constant Joy1a           : integer := 64;
constant Joy1b           : integer := 65;
constant Joy2a           : integer := 66;
constant Joy2b           : integer := 67;
constant DDR0add         : integer := 68;
constant DDR1add         : integer := 69;
constant DDR2add         : integer := 70;
constant IO0add          : integer := 71;
constant IO1add          : integer := 72;
constant IO2add          : integer := 73;
constant JoyDisablePort  : integer := 74;
```

-The Cont ports are as the VZ hardware engineers intended, port values are decimal.
-JoyXY ports are 2 bytes each controller to map the full 12 bits of the SNES pad or 8 bits of the NES pad.
-DDRx is the IO port direction register, 0=output, 1=input for each pin.
-IOx is the Input/output port for each 8 bit IO port. You can read back the port even if it is set to an output to check the current state of the pin.

To check if the VZ/Laser joysticks or SNES pads are working, this simply program will show the input figures.

```
10 PRINT INP(42);INP(43);INP(37)
20 GOTO 10
```

VZDOS routines, video and music fiddlings.

$5F00 - OpenFile
Description : opens selected file
In : reg HL = pointer to filename. Reg A = filehandle number. Starting typically at 1.
Call : call 5f00h
Jump table : 5ff0h

$5F03 - Select Handle
Description : Selects Handle. Loads first frame into the ram buffer.
In : a = filehandle number. Starting typically at 1.
Call : call 5f03h

$5F06 - DrawFrame
Description : Draw frame from buffer to video ram.
Call : call 5f06h
Jump table : 5FF6h

$5F09 - FAT_NextSector
Description : Performs a sector read from the SD Card, and then performs a fast copy of 512 bytes
: from 'ram buffer' to a destination memory location that is configurable ($FB80)
In : Word location at $FB80 is a designated "destination memory" location for the copy.
: Example : Setting $FB80 to $7000, copies a sector of 512 bytes over from the rambuffer
: to the video screen (which is at $7000).
: This will copy an entire MODE(0) screen, or 512 bytes (one quarter) of MODE(1) screen
: Second example, setting $FB80 to $B000, then calling $5F09, will copy 512 bytes from
: the RAM buffer, which all has its own internal pointers, to $B000, where you are then
: free to manipulate or further read from this location.
: This feature is used in the few continuous music examples that have been released.
Call : Call $5F09

$5FF0 - Jump table
$5FF0 : OpenFile
$5FF3 : SD2VRAM
$5FF6 : DrawFrame
$5FF9 : Fat32_Root

Example, taken from Ben's "AHAFAST" video.
Due to the auto loading of the first sector before the loop, we can start at $7200 instead of $7000.
Otherwise everything is 'out-of-whack' by 512bytes, and the video file needs to be further edited.

```
        SDbufferAddress equ 0FB80h
        FAT_NextSector  equ 05F09h

        org     7FE8h
        db      20h, 20h, 00h, 00h, 56h, 5Ah, 44h, 4Fh ; generic header
        db      53h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        db      00h, 00h, 00h, 00h, 00h, 0F1h,00h, 80h

        org     8000h

        ld      a,08H
        ld      (6800h),a               ;set hires mode
        ld      hl, FILENAME
        ld      a, 1
        call    $5f00                   ; OpenFile
        ld      a, 1
        call    $5f03                   ; SelectHandle - This loads the first frame
                                        ; into RAMbuffer, not VRAM! Add an extra sector
                                        ; to the DAT file, or copy from buffer to vram

infloop:ld      HL, 7200h
        ld      (SDbufferAddress), HL
        call    FAT_NextSector          ;512 bytes to vram
        ld      HL, 7400h
        ld      (SDbufferAddress), HL
        call    FAT_NextSector          ;512 bytes to vram
        ld      HL, 7600h
        ld      (SDbufferAddress), HL
        call    FAT_NextSector          ;512 bytes to vram
        ld      HL, 7000h
        ld      (SDbufferAddress), HL
        call    FAT_NextSector          ;512 bytes to vram
        jr      infloop
FILENAME:
        db      22h,"AHA",22h           ; filename to load
end
```

About : The .VZ snapshot file

The .VZ format was put together rather very quickly one night in 1999 by Brian Murray to get "something / anything" to work with Guy Thomason's upcoming new emulator to be able to load and save files. The format is a memory "snapshot" and not necessarily technically correct by any means. It is a legitimate true VZ enthusiast hacked-together format, that became the standard as it was the first slapped-together data format outside of the actual physical hardware. Purists hate this format.
The VZ cassette file format (.CAS) came later which, we may as say, noone in the VZ community uses.
Gavin Turner utilised a .CVZ for cassette images within his DSVZ200 and WINVZ300 emulators.

The ".VZ" file format consists of a 24 byte header, and then followed by the program data.

```
typedef struct vzFile
{       byte vzmagic[4];
        byte filename[17];
        byte ftype;
        byte start_addrl;
        byte start_addrh;
} VZFILE;
```

Header in Assembly:

```
defs    'VZF1'                  ; vzmagic       [04].
defs    'FILENAME12345678'      ; filename      [17].
defb    $F1                     ; Filetype      [01].
defw    $8000                   ; Start address [02].
```

F0 being for a BASIC program with a starting address typically at 7AE9.
F1 being a binary, typically being an autostart game or utility for example.

Resources/References:

https://www.facebook.com/groups/4609469943	VZ/Laser Fans Facebook Group.
https://bennvenn.myshopify.com/products/vz300-sd-loader	Benn Venn loader shop listing.

v1.7 Overwrite support added, break and reuse cluster chains. Added new CLI commands (FILL, COPY). Partitioned SD's now supported. Directories Supported

v1.8 Directory listing 15/16 lines issue bug fix.

v2.0 Further small issues fixed, along with SNES port commands (JOY/NOJOY) for Mk II Loader.

v2.1 fixes.

v2.2 Optimised SD functions (Can dump a sector direct to VRAM)
Included is test code to show how to use it, and a 24 frame per second AHA dat file
At 24fps the video runs fast so we're probably closer to around 28-30 frames per second.
If used in GRAFX mode, that'll be divided by 3, but 10fps is not too bad!
It should at least remove the visible screen drawing.