



## BackBit C64 Documentation v1.1

© 2019 Evie Salomon

[For the latest version of this document, visit <https://backbit.io>](https://backbit.io)

### Philosophy

When I had access to some Commodore 64's back in 2018, I was searching for the ideal solution to make it easy to load programs. The 1541 disk drive proved to be way to tedious, unreliable, and painfully slow.

I spent some time researching more modern media solutions for the Commodore 64 like the SD2IEC, EasyFlash3, and Ultimate-II. While these solutions did provide some convenience as compared to a 1541 disk drive, they did not perform to my expectations.

SD2IEC: I was initially very excited about this option. It seemed to be the perfect solution, incorporating modern SD media and improving speed. Unfortunately, most programs I tried to load would simply not work, and those that did often loaded slowly, even though the device had the capability to load fast. Furthermore, to make things work I had to identify specific fast-loaders that were compatible with the device and learn commands to change device numbers. I had to master the art of the swap file and unlabeled buttons to deal with multi-disk titles. Needless to say, I got frustrated and kept looking for a better solution!

EasyFlash3: It made sense to me to use the cartridge slot. Why not have instant loading and bypass confusing menus and loading commands? I was also initially excited about this option, but I ran into some roadblocks. I realized that I could not simply load titles onto the cartridge at will. I had to embed them onto the SD2IEC and then manually load them to the cartridge, a process that took several minutes per title. Even though I could put quite a few small programs on one cartridge slot, it was still very tedious to change content and I was limited to only 7MB of programs.

Ultimate-II: While this seemed like a great device on the surface, I was still concerned that it might not fulfill all my needs. Even though it emulates the 1541 perfectly, programs can still take

a long time to load. There are also reports that it is incompatible with some versions of the 64, like the SX-64.

Based on my frustration with the above devices, I decided to make my own! I took inspiration from the best parts of each device I discovered:

- A solution that loads on startup through the cartridge slot;
- A solution that uses common media like SD;
- A solution that eliminates the need to wait for loading or swap disks.

## How It Works

Rather than relying on complicated FPGA circuitry to emulate the signals the C64 expects, BackBit uses the power of software to create an intuitive user experience that is reliable, consistent, and effortless.

There are two signals that are critical for BackBit to operate: The 6502's CPU cycle clock, and the memory request lines exposed to the expansion bus. There is an expectation that memory access has a certain lag time from when the address is requested to when the data should be made available, and a tendency for this to happen in a certain window within the CPU's clock cycle. By balancing these requirements, BackBit is able to trick the C64 into thinking it is a pre-programmed and ever-changing RAM device.

## BackBit Levels

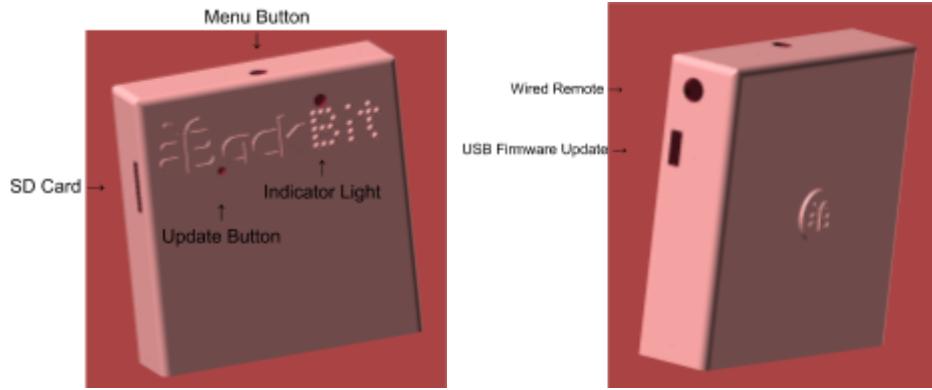
To make it easy to understand the operation of BackBit, think about the 3 levels of BackBit:

Level I: Load single-file programs, such as 8K cartridges, 16K cartridges, and programs up to 50K in size. Many devices can do this already, so this is just the basic functionality you would expect from a storage device.

Level II: Emulate up to 8 disk images at the same time, spanning from device 8 to 15. Rather than relying on the IEC port, this emulation happens through overriding KERNAL vectors. If a program does not implement copy protection, chances are it will work perfectly from the original disk image. Games that are a great example of this include Lode Runner, Jumpman, Ultimate Wizard, and many games designed to work with IDE64. While only a single 1541/71/81 disk image can be used by default, you can create a BBT file with the BackBitTool to use all 8 slots.

Level III: Access up to 4GB of extended data. Is this really a limit? Not really, since it would take the C64 about a day just to process that much information. To access this wild amount of data, you must use the BackBitTool to package a BBT file and then use the BackBit API to access the data. More details are explained in that section of the documentation.

# Anatomy



The menu button up top allows you to return to the menu once a program is running.

The indicator light lets you know when requests are being made to the cartridge.

The SD card slot accepts any Micro SD card formatted to FAT32 or FAT16. The slot is designed so that cards will not “fall in” to the cartridge and will be easy to remove.

Connect a Micro-USB cable to upgrade firmware. Once software is set up correctly, press the Update button to perform the update. It is not possible to “BRICK” the cartridge. In case of a failed update, you can easily restart the upgrade process.

The wired remote allows you to connect a BackBit Button (available for purchase separately), which simply breaks out the indicator light and menu button, so you don’t have to reach behind the computer, which can sometimes cause unacceptable static electricity interference.

## Summary of Features

- Load programs from Micro-SD card formatted with FAT16/32 (32GB limit);
- Firmware is upgradeable via USB cable;
- “Rip & Burn” – built-in disk copier to facilitate archiving/reproducing real disks;
- Real time clock to facilitate accurate versioning of files (w/ replaceable CR2032 battery);
- Innovative BBT file format to assemble and run multi-disk programs w/o disk swapping;
- Extended data access provides up to 4GB of data per program;
- Physical button to return to menu from active program;
- Auto-Start mode: Automatically start a specific program w/o user interaction;
- Presentation mode for exhibiting programs non-destructively;
- Optional wired remote control (available for purchase separately).

# File Browser

The BackBit file browser allows you to navigate the entire contents of your SD card. There are a few restrictions in how your card must be arranged:

- Must be formatted in FAT32 (or FAT16);
- Must have a folder entitled **BACKBIT** (case-insensitive) in the root folder of your card;
- No single folder can have over 1,000 files.

## Key Conventions:

↑ Key functions the same as SHIFT + ↑CRSR↓.

← Key functions the same as SHIFT + ←CRSR⇒.

🕹 In many cases either Joystick #1 or Joystick #2 can be used instead of cursor keys.

Press any function key to see a reference of function commands on the status bar.

## Browser Commands:

🕹 Up / ↑ / SHIFT + ↑CRSR↓ - Navigate to last files or item

🕹 Down / ↑CRSR↓ - Navigate to next file or item

🕹 Left / ← / SHIFT + ←CRSR⇒ - Navigate to last page

🕹 Right / ←CRSR⇒ - Navigate to next page

🕹 Fire / RETURN - Choose selected item

↶+ RETURN - Open selected disk without automatically starting

A - Z / 0 - 9 / Space Bar - Search for title beginning with selected characters

CLR HOME - Reset search / Go to top of listing

INST DEL - Remove last search character / Close popup / Navigate to parent directory

RUN STOP + RESTORE - Open BASIC session

RESTORE - Refresh directory (useful after swapping SD Card or encountering errorL)

F1 - Show UTILitieS menu (not specific to any item), or hide if already visible.

F3 - Show PROPertieS for current item, or hide if already visible.

F5 - Show SAVES for current item, or hide if already visible.

F7 - Show TOOLS for current item, or hide if already visible.

SHIFT LOCK - Select on bootup to show technical information

# Loading Programs

## These file formats are supported:

- PRG/P00 (single program up to 50K);
- T64 (only supports loading first program from tape image);
- CRT (8K or 16K standard cartridges, emulation provided for UltiMAX cartridges);
- D64/D71/D81 (1541/71/81 disk; emulated using KERNAL vectors);
- BBT (BackBit format; see BackBitTool section).

The file formats are unsupported at this time, but support may be added for them in the future:

- G64/NIB files: BackBit does not physically connect to a disk drive, so it cannot generate these advanced formats. Use NIBTools to convert these files to D64;
- TAP files: BackBit does not emulate tape loading. Convert these to a format BackBit can recognize, like PRG or D64;
- T64 files with more than 1 program: Try converting these to individual PRG/D64 files;
- Disk-based programs that access the serial port directly at \$DDXX, incl. fastloaders: Since BackBit does not physically connect to the C64's serial port, it cannot handle these requests. Fortunately, with some disassembly skills, it is possible to make any program run on BackBit;
- CRT files that utilize UltiMAX mode: BackBit supports one RAM access per CPU cycle, whereas UltiMAX requires two RAM accesses per CPU cycle (one from the CPU itself and one from the VIC-II chip). As a workaround, there is an UltiMAX emulator built-in which supports most of the original ULTIMAX titles in unmodified CRT format;
- CRT files over 16K: Cartridges over 16K utilize banking since the expansion port only exposes 16K of data at a time. There are many banking schemes specific to particular games, as well as a large selection of EasyFlash CRT files that contain up to 1MB of data. BackBit cannot support these formats natively. It is possible to create BBT versions of games with equal or greater functionality than bank-swapping cartridges.

## Presentation Mode

Accessible via the Utilities menu (F1), this puts the cartridge in Presentation Mode, identifiable with a blue border. This allows the cartridge to be used in an exhibit or party environment.

Presentation Mode operates similarly to the File Browser with these exceptions:

- Function Key menus and directory refresh are unavailable;
- The last directory navigated in the File Browser is the only one accessible;
- All contents of the current directory are recursively combined into one list;
- Hold menu button while physically resetting computer to disable Presentation Mode.

## Autostart Mode

Accessible via the Tools menu (F7), this puts the cartridge in Autostart Mode. In this mode, the selected program is started automatically upon bootup.

While in Autostart Mode, the following restrictions apply:

- The File Browser is inaccessible;
- The menu button resets the program;
- Hold menu button while physically resetting computer to disable Autostart Mode.

## Rip/Burn

With BackBit, it is possible to read and write disks with a real 1541 disk drive, simply by plugging the drive into your C64 at the same time the Cartridge is present.

To “Rip” a disk, select this option from the Utilities menu (F1). Then choose a device to read and press return. The disk will be saved as “New Disk” (or as a version inside that disk if one is already present).

To “Burn” a disk, select this option from the tools menu (F7). Only D64 images can be written to a 1541 drive. Then choose a device to write to and press return.

## Clock

BackBit comes with a Real Time Clock. This means that even when the power is shut off, your cartridge will remember the time of day. To set the clock for the first time, choose Set Clock from the Utilities menu. Use the cursor keys to navigate the fields and modify values. Be sure to set your time zone, indicated as an offset from UTC. Press return to save changes or menu button to abort.

## Properties

Access properties by pressing F3. This will give you detailed information about a file and allow you to rename it by pressing enter on the name. More options will be provided in future updates, as specifications are ironed out (such as the ability to configure menu-based access to documentation, graphics, title images and music).

## Saves

Access saves by pressing F5. Then choose an item with RETURN and it will show a popup menu with actions. Normally when you open a disk image, it loads the most recent modifications that have been made, such as high scores, user progress or levels. By selecting the original version here, you can load the original unmodified version of the program. You can also do many things with modified versions of the program:

- Open a specific version, i.e. an adventure game with your user’s progress;
- Rename the version;
- Duplicate, i.e. fork the version;
- Erase the version;
- Export to a regular disk image, so that the version doesn’t have to be associated with the original disk anymore.

# Destructive Operations

Preservation is a key goal in the BackBit user experience. By default, no original disk image will be modified. It's as though every disk in BackBit has a backup of the original stashed away. However, there are a few ways you can potentially destroy content, and they should be used with caution:

- Using the Erase option from the Tools menu (F7). There will be a popup that asks you if you are sure;
- In the Saves menu (F5), you can erase a specific version;
- Write errors may occur depending on the quality of the SD card you use, static electricity, or poor connections—so it is recommended that you periodically backup your data to another medium.

# Firmware Upgrade

It is possible to upgrade the firmware via a USB cable. You will need a Micro-USB cable and a paperclip to complete this process. Make sure the cable is a data cable, not merely a charging cable.

The latest firmware can be downloaded at <https://backbit.io>.

The first time you upgrade, you will have to download some software:

[https://www.pjrc.com/teensy/td\\_download.html](https://www.pjrc.com/teensy/td_download.html)

Install the Teensyduino (and Arduino) software using the instructions on the download page. Once the software is installed, find the .HEX firmware file you downloaded from the BackBit website.

Plug in the USB cable—one end to your computer and the other end to the USB port on the BackBit cartridge. It doesn't matter if the cartridge is plugged into the Commodore 64 or not, or whether or not the Commodore 64 is turned on.

Open the Arduino program. Click the Verify (✓) button the first time you run this program. You may have to select Board → Teensy 3.5 from the Tools menu. If the application does not recognize the USB device (it should show up in Tools → Port), you may have to try rebooting your computer or using another cable.

Click Upload (→) in the Arduino program. You will see a Teensy window pop up. Now go to the File menu and click Open HEX File. Select the .HEX firmware file you downloaded from BackBit. Now use the paperclip and gently press the program button on the BackBit cartridge. (in

between the “B” and “a” letters in “Back”. You should see a progress bar indicating the firmware is transferring to the cartridge.

Once this process is complete, unplug the USB cable and reboot your Commodore 64. To verify the firmware version on the cartridge hold shift or activate shift-lock while the computer is booting up. The version is in the top-right corner.

## Production Status

BackBit is actively in production! Currently each cartridge has to be assembled by hand. I will make every effort to fulfill demand for the cartridge, including redesigning the hardware as demand requires to make it easier to mass produce.

## BackBitTool

The BackBit Tool is a cross-platform application available for Linux (32-bit and 64-bit), Mac (64-bit) and Windows (32-bit and 64-bit). Using the tool, you can assemble BBT files which are composed of 3 parts:

1. An auto-start program, i.e. PRG file that will be executed on startup (this is optional, and if not present, a disk image must be present).
2. Up to 8 disk images (D64/D71/D81), assigned automatically to devices 8 through 15. If there is no auto-start program specified, the application will start with LOAD "\*",8,1
3. A single binary file, no larger than 4GB that will be available through the API for extended data access.

This application is free to download and available at <https://backbit.io>.

# BackBit API

The API is made of up “peeks” and “pokes” to the I/O area in memory related to the expansion port, i.e. memory locations \$DE00-\$DFFF in hex. You must have a knowledge of machine language programming to effectively implement features of this API, although it is also possible to use the BASIC programming language to work with most of these features. In that case, you will have to convert these hex numbers to decimal.

## Peek Commands

This section describes what is obtained when peeking bytes from the I/O area at \$DE00-\$DFFF.

CODE	
\$DE00 - \$DE02	<b>Identifier.</b> These bytes always return the sequence \$BA \$CB \$17 while the cartridge is active.
\$DEFF	<b>Device Number Filter.</b> This is the maximum device # that the KERNAL emulation will bypass, so it will typically be one less than the first device # emulated, i.e. <u>\$07</u> if you are using device <u>8</u> . If KERNAL emulation is not active, this will be <u>\$FF</u> .

DATA	
\$DF00 - \$DFFF	<b>Requested data.</b> Data appears here in 256-byte blocks when requested from Extended Data. Poke to \$DF## to configure which 256-byte block out of up to 64K can be accessible here, i.e. \$DF00 to get the first block and \$DFFF to get the last.

## Poke Commands

The cartridge interprets poke commands based on the address written to, **WHILE IGNORING WHAT DATA IS WRITTEN**. This means that you can interchangeably use STA, STX, or STY instructions when writing to the cartridge at \$DE00-DFFF, and can safely ignore the contents of the A/X/Y registers while doing so. The lower 8-bits of the address are interpreted as a command on the I/O1 channel (\$DE##) and as data on the I/O2 channel (\$DF##). For commands that require parameters, poke each parameter byte to \$DF##, then execute the command at \$DE##.

To prevent conflicts with programs that send non-BackBit requests to the \$DE## memory area, all commands, with the exception of disabling the cartridge memory, have a suffix of \$BB.

When a command is executed, the cartridge will temporarily become unavailable while it is performing the command. This does not interrupt the Commodore's CPU at all, but it makes any

cartridge **memory**, **requests** & **KERNEL usage** inaccessible until the command completes. To determine if the operation has completed, check for the 3-byte identifier (BA CB 17) at \$DE00.

PARAMETERS	
\$DF00 - \$DFFF	<b>Send parameter data.</b> This will add the low byte of this address to the parameter buffer, i.e. \$DF00 will send \$00 and \$DF34 will send \$34. The parameter buffer is limited to 256 bytes, after which it resets and wraps around. In loader or extended data mode, the most recent parameter byte written is used to specify the high address byte.

CONTROL	
These requests control basic interaction with the cartridge.	
\$DE00	<b>Disable cartridge memory.</b> Disables access to cartridge memory at \$8000-\$AFFF. Facilitates hiding the cartridge from a loaded program. Does not affect the I/O area at \$DE00-\$DFFF.
\$DE01 [BB]	<b>Enable cartridge memory.</b> Re-enables access to cartridge memory.
\$DE02 [BB]	<b>Return to browser.</b> Informs the cartridge that the program is done and should go back to the browser.
\$DE03 [BB]	<b>Turn on LED until next request.</b> LED remains <u>on</u> until next request.
\$DE04 [Y# M# D# H# M# S# Z# BB]	<b>Adjust real time clock.</b> Modifies the time to Year (+2K) / Month (1-12) / Day (1-31) Hour:Min:Second in Zone (signed, +/- UTC). Time is retained by the cartridge even after powering off the machine.

EXTENDED DATA	
This feature allows BackBit to access up to 4GB of data from a single program. While this command can take a fraction of a second to run, it sets up a full 64K of data for instant access, 256 bytes at a time. Requested data will be available at \$DFXX with bank switching triggered by writes to the same area. A single poke to \$DFXX switches between 256 byte blocks instantly. Use the BackBitTool to attach data to a BBT file which uses this feature.	
\$DED0 [A# BB]	<b>Access data.</b> Specify 8-bit 64K bank, i.e. \$00 <u>A</u> #0000
\$DED1 [A# B# BB]	<b>Access data.</b> Specify 16-bit 64K bank, i.e. \$ <u>B</u> # <u>A</u> #0000
\$DED2 [A# B# C# BB]	<b>Access data.</b> Specify 24-bit offset, i.e. \$00 <u>C</u> # <u>B</u> # <u>A</u> #
\$DED3 [A# B# C# D# BB]	<b>Access data.</b> Specify 32-bit offset, i.e. \$ <u>D</u> # <u>C</u> # <u>B</u> # <u>A</u> #

Example BASIC program to turn on cartridge LED:

```
10 POKE 57275,0 This accesses location $DFBB, sending the $BB parameter  
20 POKE 56835,0 This accesses location $DE03, sending the LED command
```

Example machine language program to access extended data at address \$HHMMLL:

```
STA $DFHH; access specified 64K data bank  
STA $DFBB; send BB command suffix  
STA $DE00; extended data access command  
JSR WaitForAlive; make sure memory at $DE00...$DE02 returns $BA,$CB,$17  
;note that while waiting for process to complete, KERNAL methods are inaccessible  
STA $DFMM; access 256-byte block of requested data bank  
LDA $DFLL; access single byte of specified 256-byte block
```

Note that you can make unlimited requests to data inside a 64K bank once the bank is requested, and you can make unlimited requests for a byte within a 256-byte block once it is requested. So in summary, these are the access times for any given data:

- 4-6 CPU cycles for the LDA/X/Y to access single byte (w/ optional increment and offset);
- 4-7 cycles for STA/X/Y (w/ optional increment/offset) to get next 256-byte block;
- Fraction of a second to queue up another 64K of data using command request and polling to determine when cartridge comes back online. During this time the C64 can perform other processing.

## User Feedback

This cartridge is being actively supported, so if you have any trouble whatsoever, please send an email to [backbitio@gmail.com](mailto:backbitio@gmail.com).