# Gmod2 flash utility

# User manual

*An utility to dump and flash Gmod2 cartridges*

*by Daniël Mantione*

*October 2021*



## Introduction

This manual describes the Gmod2 flash utility. The utility has been written to accompany my Gmod2 cartridge design, which is designed for developers and hobbyists. It will also work on the mass produced Gmod2 cartridges by Individual Computers.

The utility allows you to flash and dump Gmod2 cartridges on storage media attached to your Commodore computer.

Because my Gmod2 cartridge design comes both in a Commodore 64 and Commodore 128 version of the cartridge, the flash utility is available for both computers. The Commodore 128 flash utility will only work on the Commodore 128 version of the cartridge, to flash Commodore 64 Gmod2 cartridges, the Commodore 64 version should be used.

The Commodore 64 and Commodore 128 will from now on be designated as C64 and C128 in this manual.

## Programming compatibility

The original mass produced Gmod2 cartridge contains intentionally simple programming logic. Its logic times the WE pin of the flash ROM with the R/W pin from the CPU, with the PLA chip of the computer in between. As a result mass produced Gmod2 cartridges can only be programmed on computers with a really fast PLA chip and because fast PLA chips create problems in the C64 so called "long boards", the only board where flashing is possible is the assy 250469 "short board".

My DYI Gmod2 cartridges contain an improved programming circuit and do not have this problem. Therefore, programming will normally work on all C64 computers. However, the PLA is still in the loop for the C64 version, and thus the write timing depends on the PLA used in the Commodore 64 computer. Original MOS 8700 PLAs (906114-01) work fine and so do my own PLA20V8 replacements.

The C128 has a very slow PLA and timing is right on the border line of what is acceptable. Therefore programming a DYI C64 Gmod2 inside a C128 may or may not work, possibly also differing between different flash ROM types. The C128 version of the Gmod2 cartridge doesn't have the PLA in the loop and can of course be programmed from the C128. You can program a ROM inside the C128 version of the cartridge and then swap the ROM with a C64 version of the cartridge to get a C64 Gmod2 cartridge in case only a C128 is available.

At the moment it isn't possible to program cartridges from an Ultimate64 mainboard. The reason is the Ultimate64's 48MHz mode: The Ultimate64 checks at the start of a cycle if the cartridge port is involved, if no, a fast 48MHz cycle is possible, if yes, a 1MHz cycle to the cartridge port is made for compatibility. The Gmod2 cartridge activates Ultimax mode in the middle of a cycle during a write cycle and at that time the Ultimate64 has already decided that the cycle would go to RAM, it ignores the Ultimax request and thus the write cycle doesn't arrive at the cartridge. The Ultimate64 even works this way if 48 MHz mode is disabled. There are opportunities to make programming possible in the "write"  and "dynamic" bus modes of the Ultimate64, but at the moment even these bus modes don't help.

## Startup and exit

The flash utility can be loaded from disk, or can flash itself into the cartridge. If the tool is flashed into the cartridge it will autostart on reset.

The C64 utility does a small check on the bank switch register to check if a cartridge is present. If the DYI cartridge is in Magic Desk mode through jumper J1, the utility will detect this and adjust its behaviour. Magic Desk mode does not reduce its capabilities. (The utility cannot be used with regular third party Magic Desk cartridges, for starters because they don't have a flash ROM.)

If this check fails, this is not considered an error condition. In case someone inserts a cartridge that is sligtly different, the tool can still be usefull. You might still be able to use the tool succesfully to dump slightly different cartridges, such as 512KB Ocean cartridges.

Because the C128 version has no EXROM line and there is no J1 jumper, this check is not performed on the Commodore 128, the cartridge is assumed to be present with no questions asked.

On exit, the computer returns to BASIC or, if started from cartridge, the computer continues the normal boot process. The tool is a very well behaving application, that doesn't interfere with variables in the computer's memory used by the operating sytem. After exit, the computer will be a state where BASIC is just as functional as before start. The tool also doesn' t mess up its own memory contents: You can exit and run again, no need to reload from disk.

The flash utility uses the high level KERNAL API for the user interface and the low level KERNAL API for device operations. This means that the tool can take full advantage of custom KERNALs, JiffyDOS, SpeedDOS or DolphinDOS will be no problem. It can also take full advantage of the fast serial capabilities of the Commodore 128.

Because a 1541 drive is slow by default, and a Gmod2 cartridge can contain a lot of data, a KERNAL based fastloader is even recommended, but the tool will of course still work with a stock C64 and 1541, it will just be a bit slow... but you can use its UI- command to speed it up a bit, the tool has special code to support it and will disable the screen during TALK operations.

## DOS command line

After the program has been started, you will end up in a DOS like command line interface. It will display a prompt with the current device number:

```
8>
```

... and you can type Commodore DOS commands that will be sent to the device, i.e.

```
8> n:newdisk,aa
```

... to format a disk.

To switch to a different device, type the device number:

```
8> 10
10>
```

To view a directory, type $:

```
8> $
```


The DOS command line interface has been chosen to allow you to have full control of your storage devices while working with the tool. A menu system would have always been incomplete.

In order to do the actual work, the tool supports several special commands, of which you can get a short description with the "help" command:

## Reading or programming the entire flash memory

Normally you want to dump program all of the flash memory. There is one little catch: A Gmod2 cartridge contains 512KB of data. That is a huge amount of data for an 8-bit computer, and a huge amount for the classic storage media of these computers. For example, a 1541 floppy drive can store only 170KB of data on a floppy disk side. In order to be able to read or program all of the flash memory in one go, you will need a C64 storage medium that can contain files of 512KB in size, or 2065 blocks, in C64 terminology.

These storage media do exist. For example the Commodore 1581 can store 720KB on a 3.5" diskette. The popular SD2IEC also has no problems with 512KB files and the 1541 Ultimate and Ultimate64 support an IEC device that provides direct access to USB sticks that can contain 512KB files as well. Therefore if you have one of these storage media, you can read or program the flash memory in one go.

If you would like to dump a cartridge into the file mydump with type SEQ on device 10, you can do this with the following commands.

Switch to device 10:

```
10
```

Set the file name:

```
file:mydump
```

Dump the cartridge:

```
fdump
```

When the process is completed there will be a file called "mydump.seq" on device 10 and it will be 512KB in size. Note that Commodore storage devices will normally refuse to overwrite files, the file should not yet exist before you start.

If you have a file mycartridge with type SEQ on device 10, and would like to flash it, you can do it with the following commands.

Switch to device 10:

```
10
```

Set the file name:

```
file:mycartridge
```

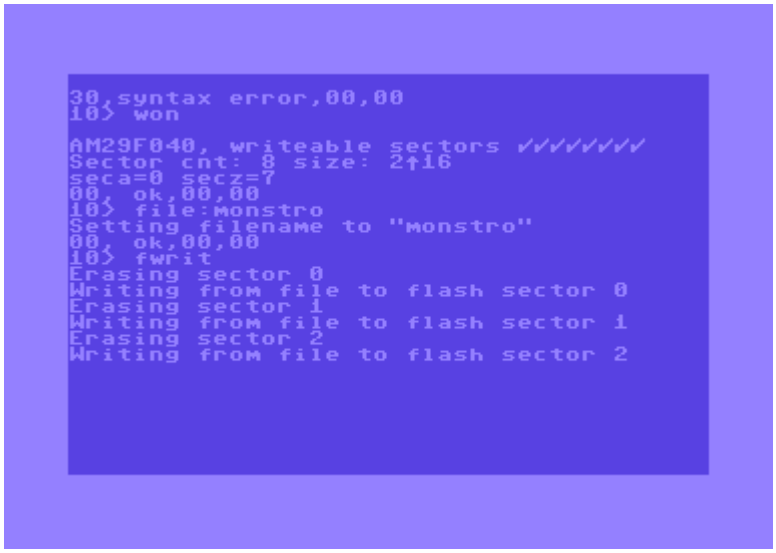
Enable the write commands:

```
won
```

Flash the cartridge:

```
fwrit
```

Programming will start:




After a while the flash memory will have been programmed with your file.

Files on Commodore storage media are sequential-only, they must be read from start to finish and you only know the exact size of the file once you have read all of it. Because of this, there is no way for the tool to know upfront if the file you are trying to flash has the correct size and therefore, the tool is not able to check for these kinds of errors. If the file isn't exactly 512KB in size, flashing will start normally, but if the tool receives an end-of-file, flashing will fail with an end-of-file eror and part of the flash memory written. If the file is larger than 512KB, flashing will complete succesfully and the first 512KB of the file will be written to flash.

## Reading or programming the flash memory partially

In case you don't have a CBM storage medium available that can carry 512 KB files, you can still program the flash memory from your Commodore 64/128, but you have to do it in multiple passes. The flash program allows you to flash only part of the memory of the Gmod2 cartridge, you can flash invidual sectors of the flash ROM. The sector size can differ for different types of flash ROM. For example, the AM29F040 has 64KB sectors, while the SST39SF040 has only 4KB sectors. This means that if your cartridge contains an AM29F040, you can program multiples of 64KB while with an SST39SF040, you can program multiples of 4KB.

The flash program identifies sectors with their number. So on an AM29F040 the sectors are numbered 0..7, while on an SST39F040, they are numbered 0..127.

You are completely free to decide how many sectors your would like to program in one go. For example, if you have a 1541 floppy drive, you could decide to split the 512KB ROM image into 4 files of 128KB in size. Each of these files will with on a 1541 floppy disk side, and you can go four 4 programming operations. On the other hand, if you have a 1571 floppy drive at your disposal, you can decide to split the 512KB ROM image into 2 files of 256KB in size and fit it on two double sided floppies. The choice is yours.

(A good way to split files on a PC is the "split" command that is available on any Linux system.)

The manual will now explain the procedure how to flash using the first example, that is, you have split the 512KB ROM image into 4 128KB files stored onto 4 1541 floppy disk sides. The 1541 is connected to device 8. The files on the floppies have the filenames "part1", "part2", "part3" and "part4".

Our cartridge is assumed to contain an AM29F040.

Switch to device 8:

```
8
```

Insert the first floppy disk. Set the file name:

```
file:part1
```

Enable the write commands:

```
won
```

The first sector to flash is sector 0:

```
seca:0
```

The last sector to flash is sector 1:

```
secz:1
```

Start the flash operation.

```
fwrit
```

Insert the next floppy disk (or flip it). The first sector to flash is sector 2:

```
seca:2
```

The last sector to flash is sector 3:

```
secz:3
```

Start the flash operation again.

```
fwrit
```

Insert the next floppy disk (or flip it). The first sector to flash is sector 4:

```
seca:4
```

The last sector to flash is sector 5:

```
secz:5
```

Start the flash operation again.

```
fwrit
```

Insert the last floppy disk (or flip it). The first sector to flash is sector 6:

```
seca:6
```

The last sector to flash is sector 6:

```
secz:7
```

Start the flash operation for the last time.

```
fwrit
```

After completion, your cartridge is ready.

Just like when writing the flash memory in one go, there is no checking beforehand on the file size. It is your own responsibility that the files have the correct length.

## Writing the flash tool to cartridge

The flash utility can flash itself to cartridge. This can be done with the commands:

```
won
twrit
```

After rebooting the computer, it will start with the flash tool from cartridge.

## Writing the flash tool to disk

If you have received a cartridge with the flash tool pre-installed and would like to write the tool to disk, you can do this as follows. Select the correct device:

```
10
```

Set the filename:

```
file:gmod2flasher
```

Write the tool to disk:

```
tdump
```

# Command reference

## *$ - Directory*

This command displays the directory on the active device. Wildcards are supported, so for example "$a" displays all files starting with an A.

## *file - Set filename*

With this command you tell the flash tool which file you would like to read or write from/to. All dump and flash write commands use this filename. For example:

```
file:mycartridge.bin
```

... tells the tool to read/write from/to "mycartridge.bin". Wildcards are supported and immedeately expanded, for example "file:m*".

## *seca - Set starting sector*

If you wish to program part of the flash ROM, you can set the first sector of the flash ROM to be programmed with this command. For example:

```
seca:4
```

... to start flashing at sector 4.

## *secz - Set end sector*

If you wish to program part of the flash ROM, you can set the last sector of the flash ROM to be programmed with this command. For example:

```
secz:5
```

... to stop after flashing sector 5.

## *edump - Dump EEPROM to file*

This command dumps the EEPROM contents to the file set by the "file" command.

## *ewrit - Write EEPROM from file*

This command writes of contents of the file set by the "file" command to the EEPROM.

## *exit - Exit to BASIC*

Exit the flash tool. If you did start the tool from BASIC you will return to BASIC. If you did boot the tool from cartridge, the boot process of the computer will continue.

### *fdump - Dump flash to file*

This command dumps the EEPROM contents to the file set by the "file" command. Dumping will start at the flash ROM sector specified with the "seca" command and stop after the flash ROM sector specified with the "secz" command.

### *fwrit - Write flash from file*

This command writes of contents of the file set by the "file" command to the flash ROM. Flashing will start at the flash ROM sector specified with the "seca" command and stop after the flash ROM sector specified with the "secz" command.

### *tdump - Dump flash tool to file*

This command will write the flash tool itself into the file specified with the "file" command. This is useful if you have the flash tool in the cartridge ROM, are going to overwrite it and want to keep the tool.

### *twrit - Write flash tool to flash*

This command will write the flash tool itself into the flash ROM. After completion, the computer will autostart the flash tool on reset.

### *won - Writing on*

This command enables the commands that write to flash memory and eeprom. When you enter the command, the tool tries to detect the flash memory chip. Further, it tests wether repeated write commands arrive at the flash ROM reliably. Writing to the flash ROM might for example not be reliable because a mass produced Gmod2 cartridge is used in a long board.

Only if a supported chip is detected and the flash ROM receives commands reliably, the write functionality is enabled. The reason why you must explicitely enable the write commands is that this way, the tool is still usefull for dumping cartridges in case of write incompatibilities.

## File formats

The flash utility works with binary ROM images, the .crt file format is not supported. You can convert between file formats on the PC with the cartconv utility. For example:

```
~> cartconv -i cartridge.crt -o cartridge.bin
~> cartconv -i cartridge.bin -t gmod2 -o cartridge.crt
```

When stored on a C64 storage medium, the file type should be SEQ.

The 1541 Ultimate2(+) and Ultimate64 work with .crt files that also store the EEPROM contents inside the .crt file. This results in 514KB .bin files, 512KB for the flash, 2KB for the EEPROM. If you end up with 514KB .bin files you can split it in to flash and EEPROM manually, buy there also exists a special gmod2crt utility that can handle this type of .crt file. You can download it on CSDB:

https://csdb.dk/release/?id=207012

The command:

```
gmod2crt -c cartridge.crt
```

... will generate separate flash.bin and an eeprom.bin files.

## 1541 UI- mode

The 1541 and many 1541 compatible devices have a UI- command that will make the device a bit faster, but it will then no longer work correctly with the screen enabled. If you type the UI- command, the tool will recognize this and disable the screen during TALK operations with the device, so if you don't have a KERNAL speedloader, you can still benefit from slightly faster disk I/O.

In order to avoid that the tool needs to keep track which device is in UI- mode and which device isn't, you will no longer be able to switch to a different device if you activate UI-. Enter UI+ before switching to a different device. Due to code simplicity, there will be no error messages, instead requests to switch to different devices are no longer recognized as commands.

UI- is still quite slow and to be used when nothing better is available: A KERNAL speedloader is the proper way to get more performance out of the CBM serial bus.

## Supported flash ROMs

At this time the tool supports the following flash ROMs:

AM29F040 (including clones AS29F040 and TMS29F040)
A29040B (including clone AS29CF040)
EN29F040
HY29F040
M29F040
MBM29F040
MX29F040
SST39SF040
SST39VF040
W39L040

Note that some of these chips are 3.3V parts and may need voltage conversion hardware, which doesn't exist on the current cartridges. This isn't relevant for the flash tool, it is algorithmically able to program those chips.

Flash ROMs that have sector protect functionality must be unprotected. The tool will detect protected sectors and refuse to program those, but cannot unprotect these sectors itself.