G64 (raw GCR binary representation of a 1541 diskette)

- **Document revision:** *1.9*
- **Last updated:** *Feb 19, 2008*
- **Compiler/Editor:** *Peter Schepers*
- **Contributors/sources:** *Markus Brenner, Immers/Neufeld: “Inside Commodore DOS”, Wolfgang Moser*
- **Wiki rendition:** *Eek/Retrograde*

Introduction

This format was defined in 1998 as a cooperative effort between several emulator people, mainly Per Hakan Sundell (author of the CCS64 C64 emulator), Andreas Boose (of the VICE CBM emulator team) and Joe Forster/STA (the author of Star Commander). It was the first real cooperative attempt to create a format for the emulator community which removed almost all of the drawbacks of the other existing image formats, primarily D64. The G64 format is not specifically designed to hold only 1541 images, but they are presently the only G64 images in existance and why this document only refers to the 1541 and D64's.

In this wiki rendition, formatting - especially around tables - has been reworked to make the information more easy to consume.

File Format

The intention behind G64 is not to replace the widely used D64 format, as D64 works fine with the vast majority of disks in existence. It is intended for those small percentage of programs which demand to work with the 1541 drive in a non-standard way, such as reading or writing data in a custom format. The best example is with speeder software such as Action Cartridge in “warp save” mode or Vorpal and V-MAX which write track/sector data in another format other than standard GCR. The other obvious example is copy-protected software which looks for some specific data on a track, like the disk ID, which is not stored in a standard D64 image.

One protection method that G64 has trouble emulating is data alignment between tracks. Some protection methods rely on data being in exact positions when the head is stepped from one track to another. Imagine two concentric circles representing the data tracks, with a drive head reading data from one track, stepping over to the other track and expecting to find some specific data where it is now. Unless you can read track data from a 1541 so it is aligned with the previous track, write it into the G64 appropriately, and also read the resulting G64 data with this alignment in mind, the protection check will likely fail. Other methods like weak bits are also hard to emulate.

G64 has a deceptively simple layout for what it is capable of doing. We have a signature, version byte, some predefined size values, and a series of offsets to the track data and speed zones. It is what's contained in the track data areas and speed zones which is really at the heart of this format.

Each track data area is simply the raw stream of GCR data, just what the read head would see when a diskette is rotating past it. How the data gets interpreted is up to the program trying to access the disk. Because the data is stored in such a low-level manner, just about anything can be done. Most tracks will be in the standard format with with SYNC markers, GAP, header, data blocks and checksums. The arrangement of the data when it is in a standard GCR sector layout is covered at the end of this document. It is the tracks that don't follow the standard which are the reason for G64's existance and the hardest to decode.

Below is a dump of the header, broken down into its various parts. Following that is a breakdown of the track offset and speed zone offset areas, as they demand much more explanation.

Now, why are there 84 tracks defined when a normal D64 disk only has 35 tracks? By definition, an image of a 1541 must include all the tracks that a real 1541 can access, which is at most 42 tracks and 42 half tracks. Even though using more than 35 tracks is not typical, it was important to define this format from the start with what the 1541 is capable of doing, and not just what it typically does. Some 1541 drives may have problems reading past track 40, and pushing the head past track 42 might be somewhat hazardous to the health of the drive as the head could get stuck.

The typical value seen for the maximum track size is 7928. This is the value used for 1541 images which use standard GCR encoding. This value is determined by the fastest write speed possible (speed zone 0), coupled with the average rotation speed of the disk (300 rpm), and assuming normal Commodore GCR data formatting. After some math, the answer that actually comes up is 7692 bytes. Allowing for a slower disk rotation of -3%, , which would allow more data to be written, and some rounding, 7928 bytes per track was arrived at.

Even though it might appear so, it is very important to know that this maximum track size value is not a fixed or hard-coded value. This value depends on the what the original disk was and the GCR encoding used. Non-1541 images such as SFD1001 or 8050 will result in different, likely larger, track sizes. Also, disks with non-standard GCR encoding like those using V-MAX can result in tracks exceeding 8000 bytes.

Since it is a flexible format in both track count and track byte size, file sizes can vary greatly. However, given a few constants like 42 tracks with no halftracks, a consistent track size of 7928 bytes and no speed offset entries, the typical file size will be 333744 bytes.

In my investigation using MNIB (a utility by Markus Brenner that allows you to nibble a 1541 diskette to the PC in G64 format) on a cleanly formatted 1541 disk (using the built-in 1541 format command), I saw the following numbers, compared with the defaults that MNIB uses: