

```

*** GEOS VLIR (Variable Length Index Record)
*** Document revision: 1.4
*** Last updated: Nov 27, 2005
*** Compiler/Editor: Peter Schepers
*** Contributors/sources: Paul David Doherty,
                        Andreas Varga,
                        Joe Forster/STA

```

Later on in the life of the C64, the GEOS OS came out. It was a system much like many other windowing OS's (MAC OS, Windows) in that it used icons, windows, a mouse pointer and resource drivers. In order to contain all the information needed for the windowing system (icon, window position, creation time/date), a new filetype called VLIR was needed and directory changes were made. While GEOS files might not be of interest to many of the emulator users, it is likely that these files will be encountered, and knowledge of them would be helpful.

There are actually two types of GEOS files, VLIR and SEQuential. Don't confuse the GEOS SEQuential type with that of the standard D64 SEQ file. They are related, but not the same. VLIR are described in more detail following this paragraph. GEOS SEQuential files are all non-VLIR files, including normal PRG, USR and SEQ types.

GEOS files usually have an entity attached called an INFO block. It contains ICON info, author, file description, load address etc. However, just because an INFO block does not exist for a given file, does not mean that the file is not a GEOS file.

Each GEOS VLIR file or application is comprised of many separate chains (called RECORDS) for different sections of the app/file. Each RECORD can be loaded in separately and overtop of other ones. Below is a dump of the first directory sector of the GEOS 2.0 disk. Note the first entry seems normal enough, but the rest have additional information in the normally unused section of the entry.

```

      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----
00: 12 04 82 13 00 47 45 4F 53 20 56 32 2E 30 20 45  <- Normal entry
10: 4E 47 4C 2E A0 00 00 00 00 00 00 00 00 00 58 00
20: 00 00 83 02 02 44 45 53 4B 20 54 4F 50 A0 A0 A0  <- First GEOS file.
30: A0 A0 A0 A0 A0 02 0F 01 04 58 08 13 0D 23 78 00  <- Note extra info
40: 00 00 83 0B 13 43 4F 4D 4D 20 31 33 35 31 28 61   from offset $15 to
50: 29 A0 A0 A0 A0 08 0F 00 0A 58 05 09 15 24 03 00   $1D.
60: 00 00 83 0F 0D 4D 50 53 2D 38 30 31 A0 A0 A0 A0
70: A0 A0 A0 A0 A0 0F 05 00 09 56 07 19 01 00 04 00
80: 00 00 83 08 0D 43 4F 4E 46 49 47 55 52 45 A0 A0
90: A0 A0 A0 A0 A0 08 05 01 0E 58 05 1F 0B 31 4E 00
A0: 00 00 83 01 10 50 41 49 4E 54 20 44 52 49 56 45
B0: 52 53 A0 A0 A0 01 08 00 06 57 08 0C 0E 00 12 00
C0: 00 00 83 0B 05 70 72 65 66 65 72 65 6E 63 65 20

```

```

D0: 6D 67 72 A0 A0 0B 12 00 05 56 0A 09 13 2D 16 00
E0: 00 00 83 08 11 70 61 64 20 63 6F 6C 6F 72 20 6D
F0: 67 72 A0 A0 A0 08 07 00 05 58 05 19 0C 10 16 00

```

Lets analyze the second entry to see what's all involved with GEOS files. Note, the offset values have been changed to 0 to make referencing easier.

```

00: 00 00 83 02 02 44 45 53 4B 20 54 4F 50 A0 A0 A0
10: A0 A0 A0 A0 A0 02 0F 01 04 58 08 13 0D 23 78 00

```

Byte: \$02: C64 filetype (see the section on D64 for an explanation) REL files are not allowed.

03-04: Starting track/sector (02/02 from above) of C64 file if GEOS filetype is \$00. If GEOS filetype is non-zero, track/sector of single-sector RECORD block

05-14: Filename (in ASCII, padded with \$A0, case varies)

15-16: Track/sector location of info block

17: GEOS file structure

\$00 - Sequential

01 - VLIR file

18: GEOS filetype

\$00 - Non-GEOS (normal C64 file)

01 - BASIC

02 - Assembler

03 - Data file

04 - System File

05 - Desk Accessory

06 - Application

07 - Application Data (user-created documents)

08 - Font File

09 - Printer Driver

0A - Input Driver

0B - Disk Driver (or Disk Device)

0C - System Boot File

0D - Temporary

0E - Auto-Execute File

0F-FF - Undefined

19: Year (1900 + value)

1A: Month (1-12, \$01 to \$0C)

1B: Day (1-31, \$01 to \$1F)

1C: Hour (0-23, \$00 to \$17) in military format

1D: Minute (0-59, \$00 to \$3B)

1E-1F: Filesize, in sectors (low/high byte order)

If the values at byte \$18 is 00 then we have a normal, sequential, C64 file, without an info block. If the value at byte \$18 is anything other than 00, we have a GEOS file, be it VLIR or sequential, with an info block.

One big addition to the directory is the TIME STAMP contained at \$19-\$1D. The year is simply the number 1900 plus whatever is stored at that location so the year 2005 would have a \$69 (105 decimal) stored there. The rest of

the values are limited as described above.

The INFO BLOCK stores items like the ICON, size, load address, file types, description, etc, and is always only 1 sector long. Since there is a fixed space to store information, the ICON height, width and bitmap length must be the same. Here is a sample info block, and layout...

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
	-----																-----
00:	00	FF	03	15	BF	3F	83	F8	40	44	04	8E	38	E2	B5	93	····ø?É~@D' é8,µì
10:	59	AA	92	A9	B5	93	59	AA	92	A9	BF	93	F9	B1	93	19	Y™í©µìY™í©øì~±ì
20:	8E	10	E1	BB	93	B9	B5	93	59	AA	92	A9	B5	93	59	AE	é'·ªìµµìY™í©µìYÆ
30:	92	E9	B1	93	19	80	10	01	BF	93	F9	FF	D7	FD	80	38	íÊ±ì'Ä'·øì~`◇"Ä8
40:	03	FF	FF	FF	83	05	00	00	2E	D9	54	00	2E	50	68	6F	····É····.ÿT'.Pho
50:	74	6F	20	4D	67	72	20	20	20	56	32	2E	31	00	00	00	to'Mgr'·V2.1···
60:	00	43	68	72	69	73	20	48	61	77	6C	65	79	00	00	00	'Chris'Hawley'···
70:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	··········
80:	00	00	00	00	00	00	00	00	00	53	61	76	65	20	70	68	········Save'ph
90:	6F	74	6F	20	69	6D	61	67	65	73	20	69	6E	20	61	20	oto'images'in'a'
A0:	53	61	76	65	20	70	68	6F	74	6F	20	69	6D	61	67	65	Save'photo'image
B0:	73	20	69	6E	20	61	20	70	68	6F	74	6F	20	61	6C	62	s'in'a'photo'alb
C0:	75	6D	20	66	6F	72	20	6C	61	74	65	72	20	75	73	65	um'for'later'use
D0:	20	69	6E	20	67	65	6F	57	72	69	74	65	20	6F	72	20	'in'geoWrite'or'
E0:	67	65	6F	50	61	69	6E	74	2E	00	02	85	06	85	07	85	geoPaint.·Ö·Ö·Ö
F0:	08	85	09	85	16	A9	04	85	11	A9	00	85	10	4C	2F	C2	'Ö·Ö·©·Ö·©·Ö·L/-

Byte: \$00-01: Contains \$00/\$FF since its only 1 sector long

02-04: Information sector ID bytes (03 15 BF). The "03" is likely the bitmap width, and the "15" is likely the bitmap height, but rare exceptions do exist to this!

05-43: Icon bitmap (sprite format, 63 bytes)

44: C64 filetype (same as that from the directory entry)

45: GEOS filetype (same as that from the directory entry)

46: GEOS file structure (same as that from the dir entry)

47-48: Program load address

49-4A: Program end address (only with accessories)

4B-4C: Program start address

4D-60: Class text (terminated with a \$00)

61-74: Author (with application data: name of application disk, terminated with a \$00. This string may not necessarily be set, or it may contain invalid data)

The following GEOS files have authors:

- 1 - BASIC
- 2 - Assembler
- 5 - Desk Accessory
- 6 - Application
- 9 - Printer Driver
- 10 - Input Driver

75-88: If a document, the name of the application that created it.

89-9F: Available for applications, unreserved.

A0-FF: Description (terminated with a \$00)

Note: all the text strings above are in ASCII, not PETASCII.

If the file is a VLIR, then the RECORD block is of interest. This single sector is made up of up to 127 track/sector pointers, each of which point to program sections (called RECORDS). VLIR files are comprised of loadable RECORDS (overlays, if you wish to use PC terminology). The first RECORD is what is always loaded first when you run that application. After that, the OS loads whatever RECORD it needs. Here is a partial sample of the RECORD sector...

```
00: 00 FF 08 00 09 04 09 03 0A 0A 0B 11 0F 11 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Byte: \$00-01: Contains 00/FF since its only 1 sector long
 02-03: Starting track/sector (8,0) for the first RECORD
 04-05: Starting track/sector (9,4) for the second RECORD
 06-07: Starting track/sector (9,3) for the third RECORD
 08-09: fourth RECORD (10/10)
 0A-0B: fifth RECORD (11/17)
 0C-0D: sixth RECORD (15/17)
 0E-0F: seventh RECORD (0/0)

When a T/S link of \$00/\$00 is encountered, we are at the end of the RECORD block. If the T/S link is a \$00/\$FF, then the record is not available.

Note that if you add up the sectors so far, we have only used two, one for the INFO sector and one for the RECORD sector. Obviously there are more used, and they are contained in the sector chains from the RECORD sector. Each t/s link in the RECORD sector points to a chain of sectors, the length of which is included in the sector count for the GEOS file. Doing a VALIDATE on a GEOS disk when not in GEOS would de-allocate all these sector chains (and the RECORD sector as well), which would not be good!

Below is the dump of a BAM sector from a GEOS-formatted D64 image and there are some changes for GEOS. We have all the typical data (forward t/s pointer to 18/1, DOS type, disk name/id) but we also have the GEOS-format signature from \$AB-BC. From observation, any GEOS-formatted disk will have the GEOS signature written in the same place (offset \$AB) in the sector that contains the disk name/ID.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00:	12	01	41	00	13	FE	DF	1F	0A	69	29	0D	15	FF	FF	1F	· · A · · , fl · · i) · · · · ·
10:	0C	96	D6	16	01	00	00	10	00	00	00	00	11	FD	BD	1D	· ñ ÷ · · · · · " Ω ·
20:	15	FF	FF	1F	0C	96	D6	16	00	00	00	00	00	00	00	00	· · · · · ñ ÷ · · · · ·
30:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	· · · · ·

```

40: 00 00 00 00 00 00 00 00 0F FC FD 05 00 00 00 00 ..... ".....
50: 00 00 00 00 06 05 05 05 01 00 40 00 00 00 00 ..... @.....
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
80: 00 00 00 00 00 00 00 00 01 00 02 00 11 FF FF 01 .....
90: 41 70 70 6C 69 63 61 74 69 6F 6E 73 A0 A0 A0 A0 Applications++++
A0: A0 A0 4A 44 A0 32 41 A0 A0 A0 A0 13 08 47 45 4F ++JD+2A++++`GEO
B0: 53 20 66 6F 72 6D 61 74 20 56 31 2E 30 50 BD 1E S'format`V1.0PQ'
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
D0: 00 00 00 00 00 00 00 00 00 00 00 00 18 2D 00 00 ..... _...
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Bytes:\$00-01: Track/Sector location of the first directory sector (should be 18,1, but it doesn't seem to matter)

02: Disk DOS version type
\$41=1541

03: \$2A, DOS version.

04-8F: BAM entries for each track, in groups of four bytes per track, starting on track 1

90-9F: Disk Name (padded with \$A0)

A0-A1: Filled with \$A0

A2-A3: Disk ID

A4: Usually \$A0

A5-A6: DOS type, usually "2A"

A7-AA: Filled with \$A0

AB-AC: Border sector trk/sec (see below)

AD-BC: GEOS ID string ("GEOS format V1.x", in ASCII)

BD-DC: Unused (usually \$00)

DD-FF: Free sector info for tracks 36-70, used on double-sided 1571 disks only!

The first two bytes at \$AB/\$AC (\$13 \$08) are the track/sector location of the BORDER SECTOR, which is one sector long, has the same layout as a normal directory sector, and is used for copying files from one disk to another. When you need to copy a file, you drag the icon to the bottom of the screen and the file is moved from its normal location in the directory to the border sector. Since it is only 1 sector, it can only hold 8 entries.

If you want to check to see if a disk is formatted for GEOS, check the string in the BAM sector starting at \$AD (offset 173) for the string "GEOS format". If it does not match, the disk is not in GEOS format. This is the way that GEOS itself verifies if a disk is GEOS formatted.

On a D64/1541, here is how GEOS allocates blocks as it either saves files or needs blocks for other purposes:

Saving files: When there's still at least one free block in the current

track, then the next block is searched for starting at the sector which is away from the current one by at least the soft interleave. The exceptions are track 25 and above where the interleave changes to the original interleave _minus one_.

When stepping to the next track, the sector where the next block of the files is saved to is computed as following:

$$\text{New sector} = (\text{Next track} - \text{Previous Track}) * 2 + 4 + \text{Soft interleave}$$

Getting the Border Sector: When allocating the GEOS border sector, the search starts upwards from sector 19/0 for a free sector and then from sector 1/0 if no free one has been found yet.

Getting the first sector for saving a file: Start searching from sector 1/0 and, on the first track that has at least one free sector, you start searching at the sector computed by the form above.

Seeing as this is not an emulator format, I will not comment on its relative merits. It is simply another C64 file type.

How to detect if a file is GEOS

-
1. Check the bottom 3 bits of the D64 file type (byte position \$02 of the directory entry). If it is not 0, 1 or 2 (but 3 or higher, REL and above), the file cannot be GEOS.
 2. Check the FILE STRUCTURE and FILE TYPE bytes (byte position \$23 and \$24 respectively). If they are both \$00, then the file is not GEOS, but is a normal D64 filetype (no INFO block either).
 3. Check the FILE STRUCTURE byte (position \$23). If it is anything other than \$00 or \$01, then the file is not GEOS (as those are the only legal values).
 4. If you've reached this point, and everything looks ok, then the file is GEOS. Now, if the FILE STRUCTURE byte (position \$23) is a \$01, then the file is a GEOS VLIR, otherwise it is a GEOS SEQ.
 5. Check the FILE TYPE byte (position \$24). If it is non-zero, then there is likely an INFO block attached. Check the track/sector pointer at position \$21/22 to see if they are valid numbers (track within the specs of the disk and sector in range for the track value). If they look good, then the info block exists.

From:
<https://wiki.retrograde.dk/> - **RetroWiki**

Permanent link:
<https://wiki.retrograde.dk/doc:cbm:disk:geos>

Last update: **2020/10/26 16:38**

