Appendix D.

Data File Formats

Overview

This chapter describes the output file data formats of the Text Scrap, Photo Scrap, and geoWrite files. The Photo Scrap and Text Scrap files are designed so that text and graphics data can be shared between applications. This is the format used by the Photo and Text manager desk accessories. Both the Text and Photo Scraps are stored as sequential system files on disk. When the user performs a cut or copy operation from inside an application, a Photo Scrap or Text Scrap file is created on the application disk. The user can then quit the present application, start up a new one and paste the contents of the Scrap file into the new application's document. Scraps can also be collected into Albums using the Photo Manager or Text Manager desk accessories. The geoWrite output format is important for programmers desiring to output geoWrite format from their programs or read geoWrite documents into their documents.

Section two of this paper describes the Photo Scrap. Section three documents the Text Scrap. The final section describes geoWrite's format.

Future Releases

The Photo and Text Scrap formats have been expanded in the past to include new features and may be expanded in the future. To avoid problems, applications should check the version string in the File Header block of the Text or Photo Scrap files before using the data. Checking version strings is described in the File System chapter. Bytes 89 - 92 (decimal) of the File Header contain the ASCII string, V1.1, or a later version of it. Version 1.1 was the first general release format contained in any data file.

If the scrap file is an older format than your application supports, it will have to be converted, something the application will probably want to provide. If the format is newer than the application, then the application should refrain from using it.

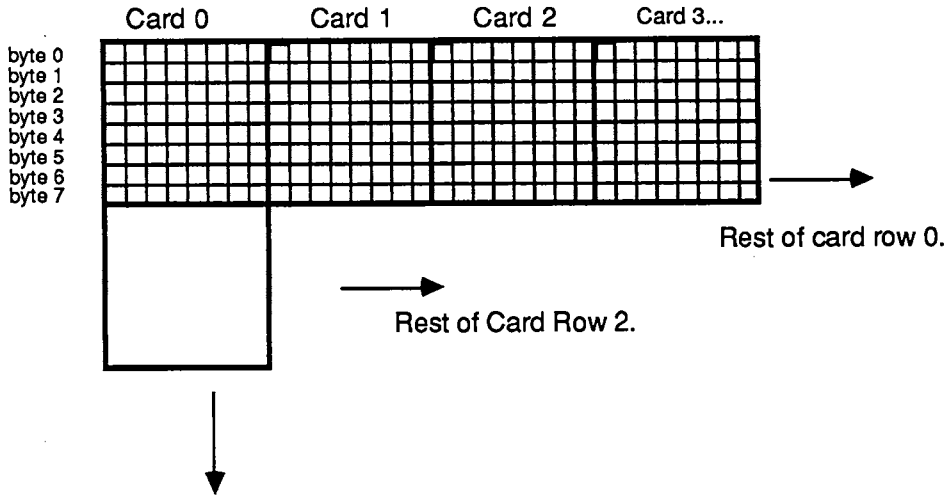
Photo Scrap

The Photo Scrap presently supports a single Bit-Mapped Object. A Bit-Mapped Object is a GEOS object for storing compacted bit-mapped data. Compacted data and Bit-Mapped Objects are described in detail in the Graphics chapter. Photo Scraps consist of a Bit-Mapped Object followed by compacted Color Table for the bit-mapped area described by the Bit-Mapped Object.

In uncompacted form, the Color Table contains one byte of color information for each card generated by uncompacting the Bit-Mapped Object. The Color Table bytes are taken from the one thousand byte color table that normally determines the colors of each of the cards on the the screen in standard high-resolution bit-mapped mode. A card, as referred to here, is the same as a Programmable Character as described in the c64 manual. The reader is referred to the description of bit-mapped graphics, cards, and color bytes starting on page 121 of Commodore 64 Programmer's Reference Guide.

In c64 hi-res bit-mapped mode, a card takes up eight bytes and defines an eight pixel wide by eight pixel high square on the screen. Each card is associated with a byte which determines its color. For example, the first color byte in the Color Table controls the color of the upper left most card on the screen. The second color byte determines the color of the second 8 x 8 card which appears just to the right of the first card and so on.

A diagram of the organization of bytes in the bit-mapped mode screen is:



Byte Organization in Bit-Map Screen

Photo Scraps are not limited to the size of the screen. While most applications create scraps which are smaller than the full screen, there will eventually be those which will construct a Scrap from an object larger than the screen size. The Color Table and bit-mapped data may be greater or less than full screen size for hi-res bit-mapped mode.

Consequently, three bytes containing the dimensions of the Bit-Mapped Object appear before the first COUNT/Bit-map pair. The first byte contains the width of the bitmap in bytes and is followed by two bytes containing the height in scanlines. Multiplying the two together gives the total number of graphic bytes to be generated by the following COUNT/Bit-map pairs. The height must always be divisible by 8 as only complete card rows are cut or copied to the Photo Scrap. The width of the scrap is always in complete cards. These restrictions are necessary because each color byte represents the color of a complete 8 byte card.

The color table is compacted using the same compaction schemes used to compact the Bitmap Object into COUNT/Bit-map pairs. Thus even the color information appears in the PhotoScrap as a series of COUNT/Bit-mapped pairs. The Color Table COUNT/Bit-map

starts just after the last graphics COUNT/Bit-map. After the proper number of graphics data bytes have been uncompacted, the next COUNT/Bit-map pair begins the compacted Color Table. The number of data bytes divided by 8 gives you the number of ColorTable bytes to be uncompacted. The Figure below shows the structure of the Photo Scrap.

Photo Scrap Data Format

Byte Number	Contents	Purpose
0	Width	The width in bytes of the bitmap picture
1-2	Height	The height in scanline of the bitmap picture
3	Count	Three modes for storing bitmap data depending on Count: 0-127: use next byte COUNT times (repeat count) 128-220: use next (COUNT-128) bytes once each (straight bitmap) 221-255: use next byte as BIGCOUNT (a repeat count), repeat the following (COUNT -220) bytes BIGCOUNT times
4-end of bitmap	Bitmap Data	The bitmap data in one of the three COUNT modes
--	Count	New mode byte
-	Bitmap Data	The bitmap data in one of the three COUNT modes
-		More Count/Bitmap Pairs
-	Color Table	Color Table stored compacted. One color byte generated for each uncompacted card.

To summarize, the Photo Scrap is made up of three-dimension bytes, followed by one large compacted Bit-Mapped Object, followed by a Color Table. Both the Bit-Mapped Object and the Color Table are a collection of COUNT/Bit-map pairs in different compaction formats. A COUNT/Bit-map pair consists of a format byte followed by a series of data bytes in the indicated compaction format. As described in the graphics chapter in this manual, uncompacted Bit-Mapped Object data must be reordered from scanlines to cards. The Color

Table contains, in compacted form the Bit-Mapped Mode color bytes for each 8 by 8 card defined by the uncompacted Bit-Mapped Object.

Text Scrap V1.2

This section describes the V1.2 Text Scrap. The V2.0 Text Scrap is a superset of the V1.2 Text Scrap. The only addition to Text Scraps for V2.0 is a ruler escape that contains positioning information. The ruler escape and is described in the next section.

The Text Scrap is an ASCII string with embedded escape characters. The escape characters are requisitioned from the nonprintable ASCII chars, sometimes called control chars*. There are two escape chars found in Text Scraps. First is TAB (char $9). It is up to the application to support or not to support tabs as it wishes. The second escape character is given the constant name NEWCARDSET ($23). It signals the beginning of a 4 byte font/style escape string. The first two bytes after NEWCARDSET are the font ID of the font to be used to display the following text. The final byte in the string indicates the style of the following text: plain, bold, italic underline and/or outline. Each style is controlled by a bit in the style byte. Setting the bold bit, for example turns on bold face. The significance of each bit is shown below.

A complete NEWCARDSET escape string will appear whenever there is a change in either font or style. The Text Manager desk accessory will not display tabs, font and style changes but they are stored within the Text Scrap nonetheless. Applications must expect these special characters, in addition to regular ASCII characters within the Text Scrap file. The structure of the Text Scrap is shown immediately below.

Text Scrap

The Text Scrap file, as it appears in memory, begins with two bytes which contain the total number of bytes to follow. (Note that these bytes don't count themselves in the total.) After these two count bytes follows a mandatory NEWCARDSET escape string.

* In ASCII the normal printable character set starts with the character '0' which has a number $20. The first $20ASCII characters ($0 - $1F), are unprintable as they don't correspond to any letter or number like 'a' or '0'. These characters are often used to embed command strings in text.

The escape string is four bytes long and begins with NEWCARDSET. The next two bytes are the font ID number. The low 6 bits of this word contain the point size of the font. The upper 10 bits contain a unique number for the font. The font word is followed by a style byte in which each bit signifies a style, as shown in the table above. Setting a bit in the style byte will turn its associated function on. Clearing the bit turns the function off. All style bits reset to 0 indicates plain text printing.

Text Scrap File Format

Byte number	Contents	Purpose		
0-1	Length	Number of bytes to follow in file		
2	NEWCARDSET	NEWCARDSET (= $17). Start of Font/Style command string.		
3-4	Font ID	The low 6 bits of font ID is the point size of the font. The upper 10 bits is the unique number of the font in which the following text should appear.		
5	Style Byte	Constant	Value	Function
		SET_UNDERLINE	10000000	Bit 7=1: turn on underlining unde rlined
		SET_BOLD	01000000	Bit 6:=1: turn on bold face
		SET_REVERSE	00100000	Bit 5=1: turn on reverse video
		SET_ITALIC	00010000	Bit 4=1: turn on italics
		SET_OUTLINE	00001000	Bit 3=1: turn on outline
		SET_ SUPERSCRIPT	00000100	Bit 3=1: turn on outline
		SET_ SUBSCRIPT	00000010	Bit 3=1: turn on outline
		SET_ PLAINTEXT	00000000	All bits=0, indicates plain text
6 to end	Text String	The ascii text with embedded tabs, font/style, and if V2.0, ruler escapes.		

GEOS Fonts

Font Name	Number Top 10 bits	Point Sizes Last 6 Bits	ID Word
BSW	0	9	$00 09
University	1	6 10 12 14 18 24	$00 46 $00 4A $00 4C $00 4E $00 52 $00 58
California	2	10 12 13 14 18	$00 8A $00 8C $00 8D $00 8E $00 92
Roma	3	9 12 18 24	$00 C9 $00 CC $00 D2 $00 D6
Dwinelle	4	18	$01 12
Cory	5	12 13	$01 4C $01 4D

geoLaser Fonts

Font Name	Number Top 10 bits	Point Sizes Last 6 Bits	ID Word
Commodore	26	10	$06 8A
LW_Roma	27	9 10 12 14 18 24	$06 C9 $06 CA $06 CC $06 CE $06 D2 $06 D8
LW_Cal	28	9 10 12 14 18 24	$07 09 $07 0A $07 0C $07 0E $07 12 $07 18
LW_Greek	29	9 10 12 14 18 24	$07 49 $07 4A $07 4C $07 4E $07 52 $07 58
LW_Barrows	30	9 10 12 14 18 24	$07 89 $07 8A $07 8C $07 8E $07 92 $07 98

Fonts 5 - 25 are on GEOS Font Pack 1

Font Name	Number Top 10 bits	Point Sizes Last 6 Bits	ID Word
Tolman	6	12 24	$01 8C $01 98
Bubble	7	24	$01 D8
FontKnox	8	24	$02 18
Harmon	9	10 20	$02 4A $02 54
Mykonos	10	12 24	$02 8c $02 98
Boalt	11	12 24	$02 CC $02 D8
Stadium	12	12	$02 30
Tilden	13	12 24	$03 0C $03 4C
Evans	14	18	$03 92
Durrant	15	10 12 18 24	$03 CA $03 CC $03 D2 $03 D8
Telegraph	16	18	$04 12
Superb	17	24	$04 58
Bowditch	18	12 24	$04 8C $04 98
Ormand	19	12 24	$04 CC $04 D8
Elmwood	20	18 36	$05 12 $05 24
Hearst	21	10 12 18 24	$05 4A $05 4C $05 52 $05 58
Brennens	22	18	$05 92
Channing	23	14 16 24	$05 CE $05 D0 $05 D8
Putnam	24	12 24	$06 0C $06 18
LeConte	25	12 18	$06 4C $06 52

The remainder of the string is composed of text with embedded tabs and possibly more NEWCARDSET escape strings. There is no special character appearing as the last character in the scrap so the application must compare the number of bytes read with a total as computed from the first two bytes of the file.

The table on the next page contains the presently supported GEOS fonts. The geoLaser fonts are designed to look as closely as possible to the fonts inside an Apple LaserWriter.®®® When preparing documents to be laser writed, these fonts should be used.

To summarize, the Text Scrap begins with a length word, followed by a mandatory Font/Style change command string, and followed by ASCII chars, tabs, and possibly more Font/Style change strings. This is the V1.2 text scrap.

Version 2.0 Ruler Escape

A ruler escape was added to the V2.0 Text Scrap to maintain compatibility with geoWrite files when justification and multiple "rulers" (formatting changes) within the page were added. A ruler escape need not appear anywhere in the text scrap, but if it appears, it will appear at the beginning of the file, or at the beginning of a paragraph. Paragraphs are defined as ending with a CR, so a ruler escape will always be preceeded by a CR. Ruler escapes are 27 bytes long. They contain information about the document's margins, paragraph justification, and color, if supported.

Tabs are not displayed in the Text Manager even though they appear in the ruler data in the file. In applications that use tabs, the tab character causes spacing to the position of the next tab is set. A wrap to the beginning of the next line is done if no tab is defined in the currently active ruler to the right of the position of the embedded tab character. The format of the V2.0 ruler escape is shown below.

Format of Ruler Escape		
Byte	**Content**	**Description**
1	**ESC_RULER**	ESC_RULER constant
2 - 3	**Left Margin**	Left Margin in pixel positions. Range 0 - 319
4 - 5	Right Margin	Left Margin in pixel positions. Range: Left Margin > Right Margin <=319
6 - 21	Tabs	Each tab is one word long Bit 15 1 for decimal tab, decimal points aligned 0 for normal text tab. bits 14 - 0 bits 0 - 14 are for the tab position, < Right Margin
22 - 23	**Paragraph Margin**	How far to indent paragraphs. Range is 0 - 319
24	**Justification**	Bits for justification and line spacing Bits 0 - 1 0 for left justified text 1 for centered text 2 for right justified text 3 for left and right (fully) justified text Bits 3 - 2 0 for single spaced text 1 for one and a half spaced text 2 for double spaced text
25	**Text Color**	The color of the text. Currently no GEOS application use this byte
26 - 27	**Reserved**	Reserved for future use

geoWrite Output File Formats

Like the Text Scrap, there is a V1.1 and a V2.0 geoWrite output format. The version numbers are different for the output file formats and the program releases. You will find geoWrite with version strings of V1.2, V1.3, and V2.0 for the Writer's Workshop, while the output file formats are either V1.1 or V2.0.

In both formats, documents are stored in VLIR files. In general, each record in the VLIR file stores one page of text. Some records are used to store pictures and, in the case of V2.0 files, header and footer information. This arrangement is show below.

VLIR Format for geoWrite Files		
Record #	**V1.1 Format Files**	**V2.0 Format Files**
0-60	Text Pages	Text Pages
61	Text Page	Header, empty for none
62	Text Page	Footer, empty for none
63	Text Page	Researved
64-127	Pictures in BitmapUp format	Pictures in BitmapUp format

The major difference between the V1.1 and V2.0 formats is that the Writer's Workshop V2.0 version supports headers and footers. Pages 61–63 may be used to store text pages with the earlier releases of geoWrite, but these will not be carried when editing with the geoWrite V2.0. This is probably not a problem since no one has ever gotten close to actually being able to store a 64 page document on a 1541 disk. When double sided support for the 1571 becomes available this may become possible.

In geoWrite, each document is broken up into separate pages and each page stored in its own VLIR record. A page consists of ruler information followed by text. For a V1.1 geoWrite file the ruler data consists of right and left margin and tab data.

Format of geoWrite V1.1 page		
Byte	**Content**	**Description**
0-1	Left Margin	Pixel position of left margin.
2-3	Right Margin	Pixel position of right margin.
4-19	Tabs	Array of 8 words storing pixel positions of tabs .
20-23	NEWCARDSET	Font and Style of page.
24-?	Text and Graphics	Text of document, may contain ruler, font/style, graphics, or page break escapes. PAGE_BREAK = 1, one byte. Causes geoWrite to begin a new page. ESC_GRAPHICS = 16, includes a picture.
Last Byte	EOF = 1	End of file marker.

The text that follows is stored as ASCII. Escape strings are used for font/style changes and for including pictures. The data for each picture is stored in a separate record. All nonempty pages must start with a font/style escape. A font/style escape cannot be followed immediately by another font/style escape. geoWrite files may also include pictures with an ESC_GRAPHICS. The data for the picture is stored in its own record as a Bit-Mapped Object. See the graphics section for the format of a Bit-Mapped Object.

Graphics Escape String

Byte	Function	Description
0	ESC_GRAPHICS	The escape to graphics control cha r = 16
1	Width	Picture's width in cards
2 - 3	Height	Picture's height in scanlines
5	Record Number	Number of the record containing th e picture data The picture data is a photo scrap.

geoWrite V2.0

Version 2.0 is similar to V1.2 but includes a more extensive ruler escape. This is the same format as found in Text Scrap files. The file format for V2.0 data files is as follows.

geoWrite V2.0 Page Format

Byte	Description
0 - 27	Ruler escape string
28 - 31	NEWCARDSET = 23 font/style escape
32 - . . .	Text of document, may contain rule r, font/style, graphics, or page break escapes. PAGE_BREAK = 1, one byte. Causes geoWrite to begin a new page. ESC_GRAPHICS = 16, includes a pict ure
Last byte	EOF = 0 appears as last byte of do cument.

Further information is also stored in the File Header of V2.0 files. This information includes the height of the footer and header, the page height the document was formatted with (different depending on the selected printer driver), and flags for NLQ and title page modes.

geoWrite V2.0 File Header Informat ion		
Byte	**Contents**	**Description**
137-138	Page Number	Page number to print on first page of this file, need not be 1.
139	Title and NLQ	Bit 7 set = make title page (no header, footer on first page. Bit 6 set means turn NLQ fixed width spacing on.
140-141	Header Height	Theheight in pixels researved on each page for the header.
142-143	Footer Height	Theheight in pixels researved on each page for the footer
144-145	Page Height	Different printers support different vertical resolutions. If the height of the page as stored here does not match what the printer is capable of, then geoWrite 2.0 reformats the file to match the printer.

geoWrite Summary

geoWrite files are devided into pages stored in different records of a VLIR file. These records may also contain bitmap data for pictures included in the document. In addition the V2.0 format includes header, footer, and page height as well as justification, NLQ and title page flags. In V1.1 files, there is only one small ruler – at the top of the page. A different ruler may control each paragraph in V2.0 files.

The above information should be sufficient to enable programmers to read and to create files in any of the formats. It is important to note that each of the earlier versions of output file formats are subsets of the later versions. Thus the V1.1 Text Scrap is a subset of the V2.0 and can be read by the later version Text Manager. The only possible imcompatibility between formats is the ability of V1.1 geoWrite to store text pages in the header, footer, and reserved records. As mentioned above, it is unlikely that a 64 page document will fit on one disk.

Text Scraps and geoWrite files differ in that Text Scraps are meant to be only one page or less. The Text Scrap is designed to be a more generic object, enabling a common ground between word processors.