

G I F (tm)

Graphics Interchange Format (tm)

A standard defining a mechanism for the storage and transmission of raster-based graphics information

June 15, 1987

(c) CompuServe Incorporated, 1987

All rights reserved

While this document is copyrighted, the information contained within is made available for use in computer software without royalties, or licensing restrictions.

GIF and 'Graphics Interchange Format' are trademarks of

CompuServe, Incorporated.

an H&R Block Company

5000 Arlington Centre Blvd.

Columbus, Ohio 43220

(614) 457-8600

Page 2

Graphics Interchange Format (GIF) Specification

Table of Contents

INTRODUCTION page 3

GENERAL FILE FORMAT page 3

GIF SIGNATURE page 4

SCREEN DESCRIPTOR page 4

GLOBAL COLOR MAP page 5

IMAGE DESCRIPTOR page 6

LOCAL COLOR MAP page 7

RASTER DATA page 7

GIF TERMINATOR	page 8
GIF EXTENSION BLOCKS	page 8
APPENDIX A - GLOSSARY	page 9
APPENDIX B - INTERACTIVE SEQUENCES	page 10
APPENDIX C - IMAGE PACKAGING & COMPRESSION . .	page 12
APPENDIX D - MULTIPLE IMAGE PROCESSING	page 15

Graphics Interchange Format (GIF)

Page 3

Specification

INTRODUCTION

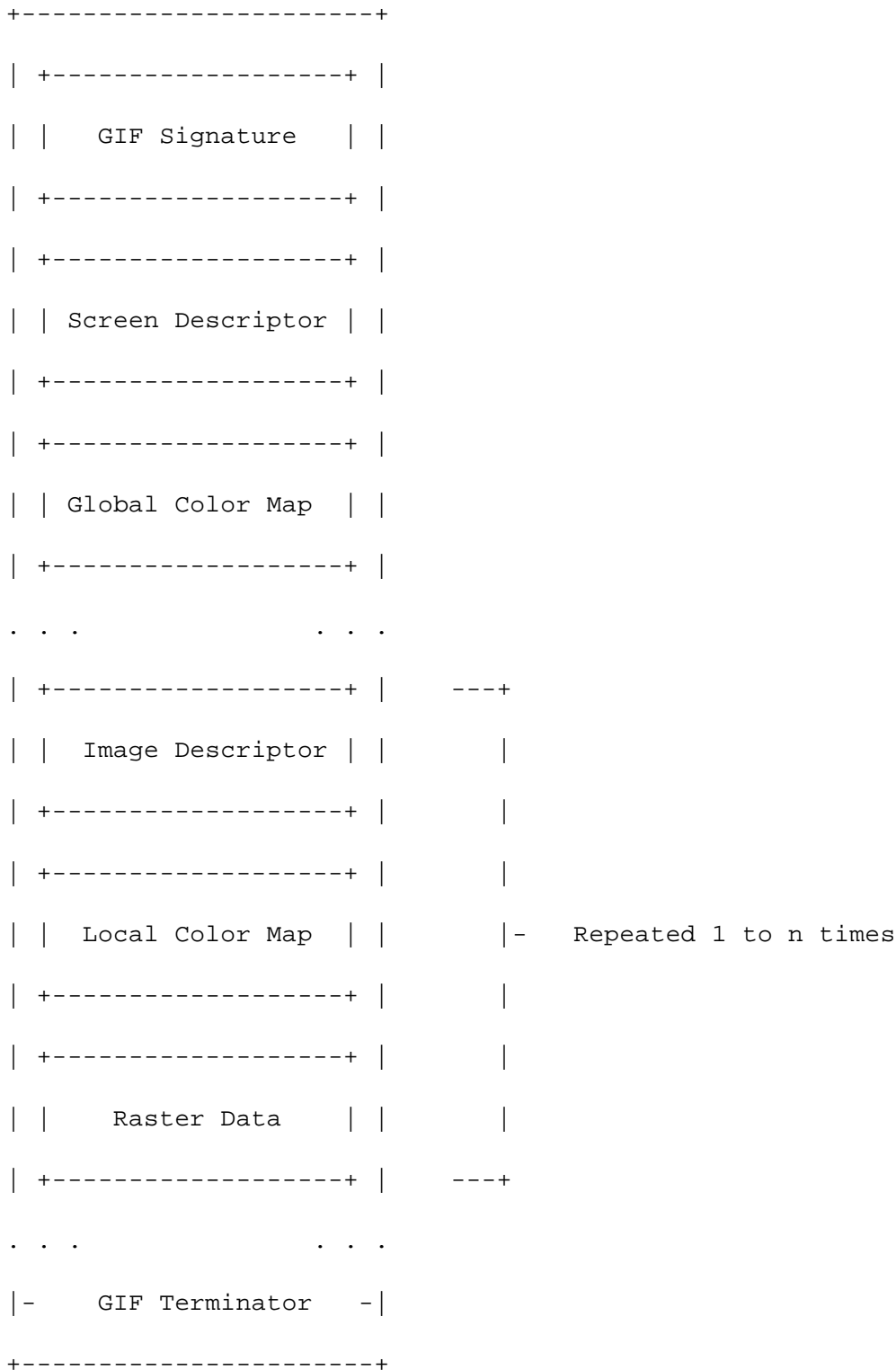
'GIF' (tm) is CompuServe's standard for defining generalized color raster images. This 'Graphics Interchange Format' (tm) allows high-quality, high-resolution graphics to be displayed on a variety of graphics hardware and is intended as an exchange and display mechanism for graphics images. The image format described in this document is designed to support current and future image technology and will in addition serve as a basis for future CompuServe graphics products.

The main focus of this document is to provide the technical information necessary for a programmer to implement GIF encoders and decoders. As such, some assumptions are made as to terminology relevant to graphics and programming in general.

The first section of this document describes the GIF data format and its components and applies to all GIF decoders, either as standalone programs or as part of a communications package. Appendix B is a section relevant to decoders that are part of a communications software package and describes the protocol requirements for entering and exiting GIF mode, and responding to host interrogations. A glossary in Appendix A defines some of the terminology used in this document. Appendix C gives a detailed explanation of how the graphics image itself is packaged as a series of data bytes.

Graphics Interchange Format Data Definition

GENERAL FILE FORMAT



Graphics Interchange Format (GIF) Specification

GIF SIGNATURE

The following GIF Signature identifies the data following as a

valid GIF image stream. It consists of the following six characters:

G I F 8 7 a

The last three characters '87a' may be viewed as a version number for this particular GIF definition and will be used in general as a reference in documents regarding GIF that address any version dependencies.

SCREEN DESCRIPTOR

The Screen Descriptor describes the overall parameters for all GIF images following. It defines the overall dimensions of the image space or logical screen required, the existence of color mapping information, background screen color, and color depth information. This information is stored in a series of 8-bit bytes as described below.

```

      bits
      7 6 5 4 3 2 1 0  Byte #
+-----+
|           | 1
+-Screen Width -+      Raster width in pixels (LSB first)
|           | 2
+-----+
|           | 3
+-Screen Height -+    Raster height in pixels (LSB first)
|           | 4
+---+-----+---+-----+      M = 1, Global color map follows Descriptor
|M|  cr |0|pixel| 5      cr+1 = # bits of color resolution
+---+-----+---+-----+      pixel+1 = # bits/pixel in image
|  background  | 6      background=Color index of screen background
+-----+
|0 0 0 0 0 0 0 0| 7      (color is defined from the Global color
                        map or default map if none specified)
+-----+

```

The logical screen width and height can both be larger than the physical display. How images larger than the physical display are handled is implementation dependent and can take advantage of hardware characteristics (e.g. Macintosh scrolling windows). Otherwise images can be clipped to the edges of the display.

The value of 'pixel' also defines the maximum number of colors within an image. The range of values for 'pixel' is 0 to 7 which represents 1 to 8 bits. This translates to a range of 2 (B & W) to 256 colors. Bit 3 of word 5 is reserved for future definition and must be zero.

Graphics Interchange Format (GIF)

Page 5

Specification

GLOBAL COLOR MAP

The Global Color Map is optional but recommended for images where accurate color rendition is desired. The existence of this color map is indicated in the 'M' field of byte 5 of the Screen Descriptor. A color map can also be associated with each image in a GIF file as described later. However this global map will normally be used because of hardware restrictions in equipment available today. In the individual Image Descriptors the 'M' flag will normally be zero. If the Global Color Map is present, its definition immediately follows the Screen Descriptor. The number of color map entries following a Screen Descriptor is equal to $2^{(\# \text{ bits per pixel})}$, where each entry consists of three byte values representing the relative intensities of red, green and blue respectively. The structure of the Color Map block is:

```

      bits
      7 6 5 4 3 2 1 0  Byte #
+-----+
| red intensity | 1   Red value for color index 0

```

```

+-----+
|green intensity| 2   Green value for color index 0
+-----+
| blue intensity| 3   Blue value for color index 0
+-----+
| red intensity | 4   Red value for color index 1
+-----+
|green intensity| 5   Green value for color index 1
+-----+
| blue intensity| 6   Blue value for color index 1
+-----+
:           :   (Continues for remaining colors)

```

Each image pixel value received will be displayed according to its closest match with an available color of the display based on this color map. The color components represent a fractional intensity value from none (0) to full (255). White would be represented as (255,255,255), black as (0,0,0) and medium yellow as (180,180,0). For display, if the device supports fewer than 8 bits per color component, the higher order bits of each component are used. In the creation of a GIF color map entry with hardware supporting fewer than 8 bits per component, the component values for the hardware should be converted to the 8-bit format with the following calculation:

$$\langle \text{map_value} \rangle = \langle \text{component_value} \rangle * 255 / (2^{*\langle \text{nbits} \rangle} - 1)$$

This assures accurate translation of colors for all displays. In the cases of creating GIF images from hardware without color palette capability, a fixed palette should be created based on the available display colors for that hardware. If no Global Color Map is indicated, a default color map is generated internally which maps each possible incoming color index to the same hardware color index modulo $\langle n \rangle$ where

<n> is the number of available hardware colors.

Specification

IMAGE DESCRIPTOR

The Image Descriptor defines the actual placement and extents of the following image within the space defined in the Screen Descriptor. Also defined are flags to indicate the presence of a local color lookup map, and to define the pixel display sequence. Each Image Descriptor is introduced by an image separator character. The role of the Image Separator is simply to provide a synchronization character to introduce an Image Descriptor. This is desirable if a GIF file happens to contain more than one image. This character is defined as 0x2C hex or ',' (comma). When this character is encountered between images, the Image Descriptor will follow immediately.

Any characters encountered between the end of a previous image and the image separator character are to be ignored. This allows future GIF enhancements to be present in newer image formats and yet ignored safely by older software decoders.

```

      bits
      7 6 5 4 3 2 1 0  Byte #
+-----+
|0 0 1 0 1 1 0 0|  1  ',' - Image separator character
+-----+
|                |  2  Start of image in pixels from the
+- Image Left  +-   left side of the screen (LSB first)
|                |  3
+-----+
|                |  4
+- Image Top   +-   Start of image in pixels from the

```

```

|          | 5   top of the screen (LSB first)
+-----+
|          | 6
+- Image Width -+      Width of the image in pixels (LSB first)
|          | 7
+-----+
|          | 8
+- Image Height -+    Height of the image in pixels (LSB first)
|          | 9
+---+---+---+---+-----+      M=0 - Use global color map, ignore 'pixel'
|M|I|0|0|0|pixel| 10      M=1 - Local color map follows, use 'pixel'
+---+---+---+---+-----+      I=0 - Image formatted in Sequential order
                                I=1 - Image formatted in Interlaced order
                                pixel+1 - # bits per pixel for this image

```

The specifications for the image position and size must be confined to the dimensions defined by the Screen Descriptor. On the other hand it is not necessary that the image fill the entire screen defined.

LOCAL COLOR MAP

Graphics Interchange Format (GIF)

Page 7

Specification

A Local Color Map is optional and defined here for future use. If the 'M' bit of byte 10 of the Image Descriptor is set, then a color map follows the Image Descriptor that applies only to the following image. At the end of the image, the color map will revert to that defined after the Screen Descriptor. Note that the 'pixel' field of byte 10 of the Image Descriptor is used only if a Local Color Map is indicated. This defines the parameters not only for the image pixel size, but determines the number of color map entries that follow. The bits per pixel value will also revert to the value specified in the Screen Descriptor when

processing of the image is complete.

RASTER DATA

The format of the actual image is defined as the series of pixel color index values that make up the image. The pixels are stored left to right sequentially for an image row. By default each image row is written sequentially, top to bottom. In the case that the Interlace or 'I' bit is set in byte 10 of the Image Descriptor then the row order of the image display follows a four-pass process in which the image is filled in by widely spaced rows. The first pass writes every 8th row, starting with the top row of the image window. The second pass writes every 8th row starting at the fifth row from the top. The third pass writes every 4th row starting at the third row from the top. The fourth pass completes the image, writing every other row, starting at the second row from the top. A graphic description of this process follows:

Image

Row	Pass 1	Pass 2	Pass 3	Pass 4	Result
0	**1a**				**1a**
1				**4a**	**4a**
2			**3a**		**3a**
3				**4b**	**4b**
4		**2a**			**2a**
5				**4c**	**4c**
6			**3b**		**3b**
7				**4d**	**4d**
8	**1b**				**1b**
9				**4e**	**4e**
10			**3c**		**3c**
11				**4f**	**4f**

12

2b

2b

. . .

The image pixel values are processed as a series of color indices which map into the existing color map. The resulting color value from the map is what is actually displayed. This series of pixel indices, the number of which is equal to image-width*image-height pixels, are passed to the GIF image data stream one value per pixel, compressed and packaged according to a version of the LZW compression algorithm as defined in Appendix C.

Graphics Interchange Format (GIF)

Page 8

Specification

GIF TERMINATOR

In order to provide a synchronization for the termination of a GIF image file, a GIF decoder will process the end of GIF mode when the character 0x3B hex or ';' is found after an image has been processed. By convention the decoding software will pause and wait for an action indicating that the user is ready to continue. This may be a carriage return entered at the keyboard or a mouse click. For interactive applications this user action must be passed on to the host as a carriage return character so that the host application can continue. The decoding software will then typically leave graphics mode and resume any previous process.

GIF EXTENSION BLOCKS

To provide for orderly extension of the GIF definition, a mechanism for defining the packaging of extensions within a GIF data stream is necessary. Specific GIF extensions are to be defined and documented by CompuServe in order to provide a controlled enhancement path.

GIF Extension Blocks are packaged in a manner similar to that used by the raster data though not compressed. The basic structure is:

```

 7 6 5 4 3 2 1 0 Byte #
+-----+
|0 0 1 0 0 0 0 1| 1      '!' - GIF Extension Block Introducer
+-----+
| function code | 2      Extension function code (0 to 255)
+-----+      ----+
|  byte count   |      |
+-----+      |
:                :      +-- Repeated as many times as necessary
|func data bytes|      |
:                :      |
+-----+      ----+
. . . . .
+-----+
|0 0 0 0 0 0 0 0|      zero byte count (terminates block)
+-----+

```

A GIF Extension Block may immediately precede any Image Descriptor or occur before the GIF Terminator.

All GIF decoders must be able to recognize the existence of GIF Extension Blocks and read past them if unable to process the function code. This ensures that older decoders will be able to process extended GIF image files in the future, though without the additional functionality.

GLOSSARY

Pixel - The smallest picture element of a graphics image. This usually corresponds to a single dot on a graphics screen. Image resolution is typically given in units of pixels. For example a fairly standard

graphics screen format is one 320 pixels across and 200 pixels high. Each pixel can appear as one of several colors depending on the capabilities of the graphics hardware.

Raster - A horizontal row of pixels representing one line of an image. A typical method of working with images since most hardware is oriented to work most efficiently in this manner.

LSB - Least Significant Byte. Refers to a convention for two byte numeric values in which the less significant byte of the value precedes the more significant byte. This convention is typical on many microcomputers.

Color Map - The list of definitions of each color used in a GIF image. These desired colors are converted to available colors through a table which is derived by assigning an incoming color index (from the image) to an output color index (of the hardware). While the color map definitions are specified in a GIF image, the output pixel colors will vary based on the hardware used and its ability to match the defined color.

Interlace - The method of displaying a GIF image in which multiple passes are made, outputting raster lines spaced apart to provide a way of visualizing the general content of an entire image before all of the data has been processed.

B Protocol - A CompuServe-developed error-correcting file transfer protocol available in the public domain and implemented in CompuServe VIDTEX products. This error checking mechanism will be used in transfers of GIF images for interactive applications.

LZW - A sophisticated data compression algorithm based on work done by Lempel-Ziv & Welch which has the feature of very efficient one-pass encoding and decoding. This allows the image to be decompressed and displayed at the same time. The original article from which this technique was adapted is:

Terry A. Welch, "A Technique for High Performance Data Compression", IEEE Computer, vol 17 no 6 (June 1984)

This basic algorithm is also used in the public domain ARC file compression utilities. The CompuServe adaptation of LZW for GIF is described in Appendix C.

Graphics Interchange Format (GIF)

Page 10

Appendix B - Interactive Sequences

GIF Sequence Exchanges for an Interactive Environment

The following sequences are defined for use in mediating control between a GIF sender and GIF receiver over an interactive communications line. These sequences do not apply to applications that involve downloading of static GIF files and are not considered part of a GIF file.

GIF CAPABILITIES ENQUIRY

The GCE sequence is issued from a host and requests an interactive GIF decoder to return a response message that defines the graphics parameters for the decoder. This involves returning information about available screen sizes, number of bits/color supported and the amount of color detail supported. The escape sequence for the GCE is defined as:

```
ESC [ > 0 g      (g is lower case, spaces inserted for clarity)
                (0x1B 0x5B 0x3E 0x30 0x67)
```

GIF CAPABILITIES RESPONSE

The GIF Capabilities Response message is returned by an interactive GIF decoder and defines the decoder's display capabilities for all graphics modes that are supported by the software. Note that this can also include graphics printers as well as a monitor screen. The general format of this message is:

```
#version;protocol{;dev, width, height, color-bits, color-res}... <CR>
'#'          - GCR identifier character (Number Sign)
```

version - GIF format version number; initially '87a'

protocol='0' - No end-to-end protocol supported by decoder
Transfer as direct 8-bit data stream.

protocol='1' - Can use an error correction protocol to transfer GIF data
interactively from the host directly to the display.

dev = '0' - Screen parameter set follows

dev = '1' - Printer parameter set follows

width - Maximum supported display width in pixels

height - Maximum supported display height in pixels

color-bits - Number of bits per pixel supported. The number of
supported colors is therefore 2**color-bits.

color-res - Number of bits per color component supported in the
hardware color palette. If color-res is '0' then no
hardware palette table is available.

Note that all values in the GCR are returned as ASCII decimal
numbers and the message is terminated by a Carriage Return character.

Graphics Interchange Format (GIF)

Page 11

Appendix B - Interactive Sequences

The following GCR message describes three standard EGA
configurations with no printer; the GIF data stream can be processed
within an error correcting protocol:

```
#87a;1 ;0,320,200,4,0 ;0,640,200,2,2 ;0,640,350,4,2<CR>
```

ENTER GIF GRAPHICS MODE

Two sequences are currently defined to invoke an interactive GIF
decoder into action. The only difference between them is that different
output media are selected. These sequences are:

```
ESC [ > 1 g   Display GIF image on screen
              (0x1B 0x5B 0x3E 0x31 0x67)
```

```
ESC [ > 2 g   Display image directly to an attached graphics printer.
```

The image may optionally be displayed on the screen as well.

(0x1B 0x5B 0x3E 0x32 0x67)

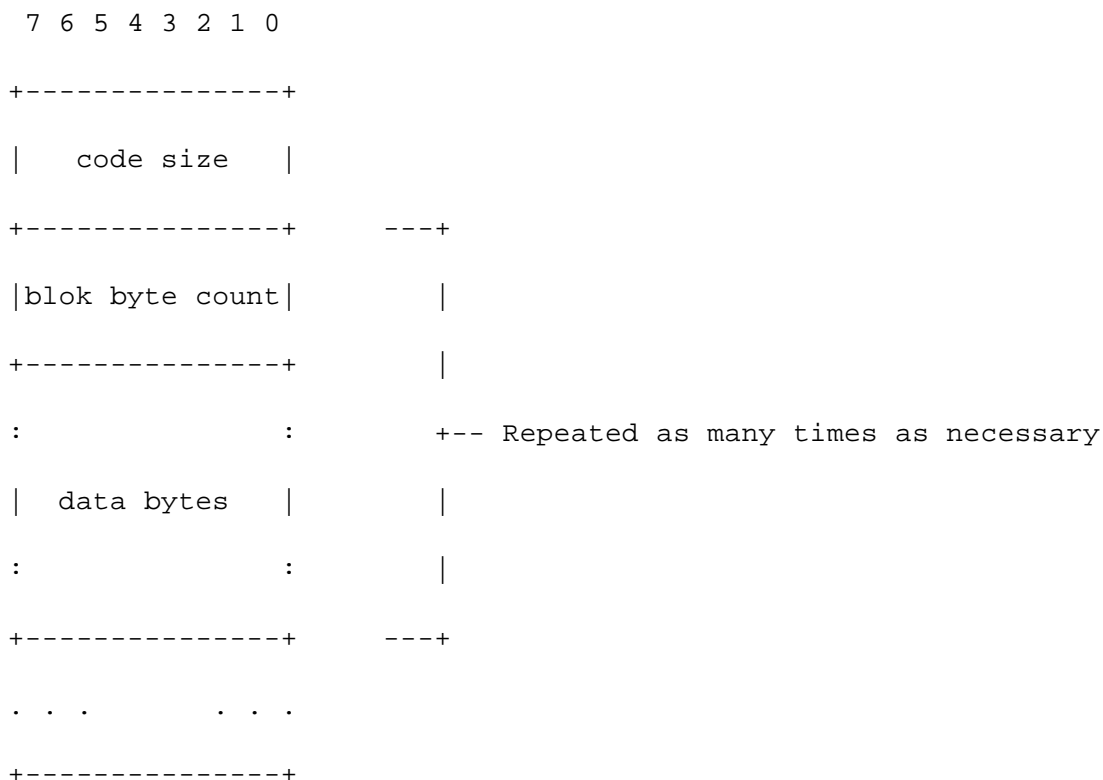
Note that the 'g' character terminating each sequence is in lower case.

INTERACTIVE ENVIRONMENT

The assumed environment for the transmission of GIF image data from an interactive application is a full 8-bit data stream from host to micro. All 256 character codes must be transferrable. The establishing of an 8-bit data path for communications will normally be taken care of by the host application programs. It is however up to the receiving communications programs supporting GIF to be able to receive and pass on all 256 8-bit codes to the GIF decoder software.

Appendix C - Image Packaging & Compression

The Raster Data stream that represents the actual output image can be represented as:



|0 0 0 0 0 0 0 0| zero byte count (terminates data stream)

+-----+

The conversion of the image from a series of pixel values to a transmitted or stored character stream involves several steps. In brief these steps are:

1. Establish the Code Size - Define the number of bits needed to represent the actual data.
2. Compress the Data - Compress the series of image pixels to a series of compression codes.
3. Build a Series of Bytes - Take the set of compression codes and convert to a string of 8-bit bytes.
4. Package the Bytes - Package sets of bytes into blocks preceded by character counts and output.

ESTABLISH CODE SIZE

The first byte of the GIF Raster Data stream is a value indicating the minimum number of bits required to represent the set of actual pixel values. Normally this will be the same as the number of color bits. Because of some algorithmic constraints however, black & white images which have one color bit must be indicated as having a code size of 2. This code size value also implies that the compression codes must start out one bit longer.

COMPRESSION

The LZW algorithm converts a series of data values into a series of codes which may be raw values or a code designating a series of values. Using text characters as an analogy, the output code consists of a character or a code representing a string of characters.

The LZW algorithm used in GIF matches algorithmically with the

standard LZW algorithm with the following differences:

1. A special Clear code is defined which resets all compression/decompression parameters and tables to a start-up state. The value of this code is $2^{**}<code\ size>$. For example if the code size indicated was 4 (image was 4 bits/pixel) the Clear code value would be 16 (10000 binary). The Clear code can appear at any point in the image data stream and therefore requires the LZW algorithm to process succeeding codes as if a new data stream was starting. Encoders should output a Clear code as the first code of each image data stream.
2. An End of Information code is defined that explicitly indicates the end of the image data stream. LZW processing terminates when this code is encountered. It must be the last code output by the encoder for an image. The value of this code is $<Clear\ code>+1$.
3. The first available compression code value is $<Clear\ code>+2$.
4. The output codes are of variable length, starting at $<code\ size>+1$ bits per code, up to 12 bits per code. This defines a maximum code value of 4095 (hex FFF). Whenever the LZW code value would exceed the current code length, the code length is increased by one. The packing/unpacking of these codes must then be altered to reflect the new code length.

BUILD 8-BIT BYTES

Because the LZW compression used for GIF creates a series of variable length codes, of between 3 and 12 bits each, these codes must be reformed into a series of 8-bit bytes that will be the characters actually stored or transmitted. This provides additional compression of the image. The codes are formed into a stream of bits as if they were packed right to left and then picked off 8 bits at a time to be output. Assuming a character array of 8 bits per character and using 5 bit codes

to be packed, an example layout would be similar to:

```

byte n      byte 5   byte 4   byte 3   byte 2   byte 1
+-.....-+-----+-----+-----+-----+-----+
| and so on |hhhhhggg|ggffffe|eeeedddd|dccccbb|bbbaaaa|
+-.....-+-----+-----+-----+-----+-----+

```

Note that the physical packing arrangement will change as the number of bits per compression code change but the concept remains the same.

PACKAGE THE BYTES

Once the bytes have been created, they are grouped into blocks for output by preceding each block of 0 to 255 bytes with a character count byte. A block with a zero byte count terminates the Raster Data stream for a given image. These blocks are what are actually output for the

Graphics Interchange Format (GIF)

Page 14

Appendix C - Image Packaging & Compression

GIF image. This block format has the side effect of allowing a decoding program the ability to read past the actual image data if necessary by reading block counts and then skipping over the data.

Graphics Interchange Format (GIF)

Page 15

Appendix D - Multiple Image Processing

Since a GIF data stream can contain multiple images, it is necessary to describe processing and display of such a file. Because the image descriptor allows for placement of the image within the logical screen, it is possible to define a sequence of images that may each be a partial screen, but in total fill the entire screen. The guidelines for handling the multiple image situation are:

1. There is no pause between images. Each is processed immediately as seen by the decoder.
2. Each image explicitly overwrites any image already on the screen

inside of its window. The only screen clears are at the beginning and end of the GIF image process. See discussion on the GIF terminator.