Summary from the Encyclopedia of Graphics File Formats

Also Known As:

Truevision Graphics Adapter, Targa Graphics Adapter Image File, VST, VDA, ICB, TPIC

Type	Bitmap
Colors	8-bit, 16-bit, 24-bit, 32-bit
Compression	RLE, uncompressed
Maximum Image Size	None
Multiple Images Per File	No
Numerical Format	Little-endian
Originator	Truevision, Inc.
Platform	MS-DOS, Windows, UNIX, Atari, Amiga, others
Supporting Applications	Too numerous to list
See Also	Lumena Paint, RIX

Usage
Used for the storage and interchange of deep-pixel images, paint, and image manipulation programs.

Comments
A well-defined, well-documented format in wide use, which is quick and easy to read and decompress. It lacks, however, a superior compression scheme.

Vendor specifications are available for this format.

Code fragments are available for this format.

Sample images are available for this format.

The TGA (Truevision Graphics Adapter) format is used widely in paint, graphics, and imaging applications that require the storage of image data containing up to 32 bits per pixel. TGA is associated with the Truevision product line of Targa, Vista, NuVista, and Targa 2000 graphics adapters for the PC and Macintosh, all of which can capture NTSC and/or PAL video image signals and store them in a digital frame buffer. For this reason, TGA has also become popular in the world

of still-video editing.

Contents:

Early work on the the TGA file format was performed at the EPICenter division of AT&T. EPICenter (Electronic Photography and Imaging Center), established in 1984 to manufacture graphics boards, was purchased by EPICenter employees from AT&T in 1987 and renamed Truevision.

The first product produced by EPICenter was called the VDA (Video Display Adapter), which had a resolution of 256x200 and a 24-bit palette providing 16 milllion colors. At the time, it competed with the CGA from IBM. EPICenter's second product was the ICB (Image Capture Board), which launched both EPICenter and AT&T into the realm of video graphics (that is, video capture, manipulation, and output).

At this time, EPICenter purchased a paint package, written by Island Graphics, that later came to be known as TIPS (Truevision Image Paint System). TIPS gave VDA and ICB (and later Targa and TrueVista) users the ability to capture live video images, to create and overlay graphics, and to perform a variety of image-processing functions on bitmap data.

Although there was only one original TGA file format, applications using it created many different filename extensions--one for every graphics display board that EPICenter, and later Truevision, produced. Therefore, .VDA, .ICB, .TGA, and .VST image files created by Truevision applications all are actually in TGA format. Today, the only filename extensions supported are TGA and TPIC on the Macintosh and .TGA on the PC and other platforms.

In 1989, the TGA format was revised, and Truevision has chosen to designate the old and new formats as original TGA format and new TGA format, respectively. The original TGA format is very simple in design and quite easy to implement in code. This makes it an appealing format for developers to work with. As the available hardware technology has become more complex, however, additional file format features, such as the storage of gamma and color correction information and pixel aspect ratio data, have become necessary. The new TGA format was created as a wrapper around the original TGA format to add functionality without sacrificing backwards compatibility with older applications.

Today the TGA format is used on many different platforms world-wide for a variety of image storage, processing, and analysis needs. The Truevision solutions source book lists more than 200 software applications that support the TGA format.

The TGA format became popular primarily because it was the first 24-bit, truecolor bitmap format generally available to the PC community, even predating 24-bit support in TIFF. Truevision also gave developers access to the file format specification and provided support for developers when necessary, including working code and sample images.

TGA is device-dependent in that the structure of the format is designed to fit the requirements of certain display hardware manufactured by Truevision. In practice, this is not a severe limitation, with one minor caveat: TGA does not support the storage of image data as planes of color information.

TGA comes in several flavors, the most common of which are usually referred to as the Targa 16, Targa 24, and Targa 32 formats. These designations identify the type of Truevision hardware that created the file, and the numbers indicate the depth (number of bits per pixel) of the image data the files contain. Less commonly found variants include VDA, ICB, and Targa M8.

All of the Truevision adapters were originally designed to interface with the ISA bus found in the IBM PC platform. For this reason, all data in TGA format files, including the image data, is stored in little-endian format. This includes TGA files created by the NuVista card residing in the otherwise big-endian Macintosh, and the PCI-bus versions of the Targa 2000 series of boards, for which both PC and Macintosh versions currently exist.

As with many popular and useful formats, TGA variants have come into being, designed by third parties to incorporate proprietary extensions. Versions of the popular freeware program Fractint at one time created unreadable TGA files. A campaign by one of the authors of this book to get the Stone Soup Group, the creators of Fractint, to change the filename suffix (from TGA to something else) seems to have been successful.

File Organization

The original TGA format (v1.0) is structured as follows:

- Header containing information on the image data and palette
- Optional image identification field
- Optional color map
- Bitmap data

The new TGA format (v2.0) contains all of the structures included in the original TGA format and also appends several data structures onto the end of the original TGA format. The following structures may follow the bitmap data:

- Optional developer directory, which may contain a variable number of tags pointing to pieces of information stored in the TGA file
- Optional developer area
- Optional extension area, which contains information typically found in the header of a bitmap file
- Optional color-correction table
- Optional postage-stamp image
- Optional scan-line table
- Footer, which points to the developer and extension areas and identifies the TGA file as a new TGA format file

As you can see, both the new and original TGA format files are identical in structure from the header to the image data area. For this reason, applications that

read only original TGA format image files should have no problem reading new TGA format images. All information occurring after the image data may be ignored.

The TGA format specification available from Truevision is detailed and well-written. It is, in fact, one of the best written format specifications that was reviewed for this book, and we heartily congratulate Truevision on their effort. The TGA format is complex, but the clarity of the description in Truevision's specification makes it easy to read and understand. Truevision also distributes on floppy disk the Truevision TGA Utilities, which is a collection of utilities and C source code used to manipulate both TGA-format files and Targa videographics display adapters.

File Details

This section describes the various components of a TGA file in greater detail.

Header

The TGA header is eighteen bytes in length and is identical in both versions of the TGA file format. The structure of the TGA header is as follows:

```
typedef struct _TgaHeader
{
  BYTE IDLength;        /* 00h  Size of Image ID field */
  BYTE ColorMapType;    /* 01h  Color map type */
  BYTE ImageType;       /* 02h  Image type code */
  WORD CMapStart;       /* 03h  Color map origin */
  WORD CMapLength;      /* 05h  Color map length */
  BYTE CMapDepth;       /* 07h  Depth of color map entries */
  WORD XOffset;         /* 08h  X origin of image */
  WORD YOffset;         /* 0Ah  Y origin of image */
  WORD Width;           /* 0Ch  Width of image */
  WORD Height;          /* 0Eh  Height of image */
  BYTE PixelDepth;      /* 10h  Image pixel size */
  BYTE ImageDescriptor; /* 11h  Image descriptor byte */
} TGAHEAD;
```

IDLength is the number of significant bytes in the image identification field starting at byte 12h (following the header) and may be in the range of 0 to 255. The IDLength set to 0 indicates that there is no image identification field in the TGA file.

ColorMapType indicates whether the TGA file includes a palette. A value of 1 indicates the presence of a palette, while a value of 0 indicates that no palette is included. If the value of this field is not 0 or 1, then it is probably a value specific to the program or developer that created the TGA file. Truevision reserves the ColorMapType values 0 to 127 for its own use and allots values 128 to 255 for use by developers.

ImageType indicates the type of image stored in the TGA file. There are currently seven TGA image types. Colormapped images (pseudocolor) use a palette. Truecolor images do not use a palette and store their pixel data directly in the

image data, although truecolor TGA image files may contain a palette that is used to store the color information from a paint program. Truevision reserves the ImageType values 0 to 127 for its own use and allots values 128 to 255 for use by developers.

Valid ImageType values are listed below:

ImageType	Image Data Type	Colormap	Encoding
0	No image data included in file	No	No
1	Colormapped image data	Yes	No
2	Truecolor image data	No	No
3	Monochrome image data	No	No
9	Colormapped image data	Yes	Yes
10	Truecolor image data	No	Yes
11	Monochrome image data	No	Yes

The next three fields are known collectively as the Color Map Specification; the information contained in these fields is used to manipulate the image palette. If the ColorMapType field value is zero, then all three of these fields have a value of zero.

CMapStart defines the offset of the first entry in the palette. Although all palette entries must be contiguous, the entries may start anywhere in the palette; for example, 16-color values may be stored in a 64-element palette starting at entry 31 rather than at entry 0.

CMapLength defines the number of elements in the colormap. If an image contains only 57 colors, then it is possible to construct a 57-element palette using this field.

CMapDepth contains the number of bits in each palette entry. The value is typically 15, 16, 24, or 32 and need not be the same value as the image data pixel depth. Table TGA-1 shows valid entries for different types of TGA palettes.

Table TGA-1: TGA Palette Entry Sizes

Truevision Bits Per Display Adapter	Attribute Colormap Entry	Bits Per Pixel	Color Formats Supported
Targa M8	24	0	Pseudo
Targa 16	15	1	True
Targa 24	24	0	True
Targa 32	32	8	True

ICB	0	0	True
VDA	16	0	Pseudo
VDA/D	16	0	Pseudo
Vista	24 or 32	0 or 8	True, Pseudo, Direct

The next six fields in the header (the last 10 bytes) are referred to collectively as the image specification. The data in these fields is used to describe the image data found in the TGA file.

XOffset and YOffset describe the position of the image on the display screen. Normally, the coordinate 0,0 defaults to the lower-left corner of the screen, but any of the four corners may be designated the origin point by the ImageDescriptor field (the last field in the header).

Width and Height are the size of the image in pixels. The maximum size of a TGA image is 512 pixels wide by 482 pixels high.

PixelDepth is the number of bits per pixel, including attribute bits, if any. Typical PixelDepth values are 8, 16, 24, and 32, although other depth values may be specified, as shown in Table TGA-1.

ImageDescriptor contains two pieces of information. Bits 0 through 3 contain the number of attribute bits per pixel. Attribute bits are found only in pixels for the 16- and 32-bit flavors of the TGA format and are called alpha channel, overlay, or interrupt bits. Bits 4 and 5 contain the image origin location (coordinate 0,0) of the image. This position may be any of the four corners of the display screen. When both of these bits are set to zero, the image origin is the lower-left corner of the screen. Bits 6 and 7 of the ImageDescriptor field are unused and should be set to 0.

Image ID

The Image ID (image description) field is an optional field that may appear immediately after the TGA header. The Image ID field stores information that identifies the image in some way (filename, author name, serial number, and so on). This field is not required to be NULL-terminated, although it should be if it is used to store string data. The size of this field is indicated by the value of the IDLength field in the header. This value may be in the range 0-255. A value of 0 indicates that no Image ID field is present in the TGA file.

Colormap

The TGA format defines three methods of arranging image data: psuedocolor, direct-color, and truecolor.

Pseudocolor images store an index value into a palette in each pixel value of data. It is the palette that contains the actual pixel values that are displayed. Pseudocolor image palettes store each pixel value as a single element in the

palette. The color channels of each pixel value are not accessible individually.

Direct-color images are similar to pseudocolor images, except that each color channel (red, green, and blue) is stored in separate elements and may be individually altered. Each pixel value of direct-color image data contains three index values, one for each color channel in the colormap.

Truecolor images store the pixel color information directly in the image data and do not use a palette.

The presence of a palette and the format of the image data found in a TGA file is determined by the type of Truevision hardware that was used to create the image data (see Table TGA-1). TGA images created with a Targa 24 are only truecolor images and therefore never use a palette. TGA images created with a VDA/D card are only in the pseudocolor format and will therefore always use a palette. The Vista series of cards (ATVista and NuVista) may create and store TGA data in any of the three color formats. A palette is present in a TGA file if the ColorMapType field is set to 1. A value of 0 indicates that no palette is present in the TGA file.

It is important to realize that a palette may be present in a TGA image file even if it is not used by the image data. All TGA image files created by the TIPS paint program contain a palette that stores the 256 colors found in the TIPS color palette. This palette is not actually used to display the image data but is instead used by TIPS. A TGA reader should therefore never assume that truecolor TGA files never contain a palette.

The TGA format supports variable-size palettes. Most other formats require a palette to have a fixed number of entries based on the pixel depth of the image data. Thus, an 8-bit image contains a 256-element colormap even if only four colors are needed to reproduce the image. The TGA format, however, does not determine the number of colormap elements based on the pixel depth, so an image with 57 colors may only have a 57-element palette. The number of elements in the palette is contained in the CMapLength field in the header.

The size of each palette element in bits is found in the CMapDepth field of the header. The depth of a pixel and the depth of a palette element are not always the same. A 24-bit image may contain a 256-element palette, with each element having a depth of 24 bits, but it may contain pixel data with only an 8-bit depth. This is because 8 bits is all that is required to index a 256-element palette. It is also possible for a TGA image to contain a 4096-element palette where each element is eight bits in depth. Each pixel value of the image data therefore needs to have a minimum depth of 12 bits for indexing into the palette, although 16-bit pixel values are easier to read and write. The depth of a palette element always includes alpha channel, overlay, or interrupt bit information, if any.

Image Data Encoding

Image data stored in a TGA file is normally raw (unencoded). For this reason, TGA files tend to be quite large, especially when the bitmap data is 24 or 32 bits deep. To address this problem, the TGA specification incorporates a simple, but

effective, RLE compression scheme. For more detailed information on run-length encoding, see Chapter 9, *Data Compression*.

The RLE encoding method used by the TGA format encodes runs of identical pixels rather than runs of identical bits or bytes. This achieves a higher compression ratio over a bit-wise or byte-wise RLE scheme, because TGA pixel data often occurs as multiple-byte values rather than single-byte values. Therefore, contiguous runs of identical bytes in TGA image data often occur only in very short lengths.

Data encoded using the TGA RLE scheme may contain two types of encoded data packets. The first type is a run-length packet that is used to encode multiple runs of the same pixel value into a single data packet. A run-length packet begins with a single byte used as the pixel count. The value for the lower seven bits of this byte is in the range 0 to 127, and the count is always one plus this value (1 to 128). There can never be an encoded run length of zero pixels. The high bit of the pixel count value is always set to 1 to indicate that this is a run-length encoded packet. Following the pixel count is the pixel data value. This value is the number of bits equal to the Pixel Depth value in the Image Specification section of the header. Because the size of TGA pixels can range from one to four bytes in size, this value varies from between two to five bytes in length, depending upon the type of TGA image data encoded.

The second type of packet is the raw or non-run-length encoded packet. When a run of pixel values is too short in length to justify using the run-length packet format, the run is encoded using the raw packet format. Raw packets start with a byte that is used as the pixel count. Just as with the encoded packet, the count value is in the range of 0 to 127, with the actual pixel count being one plus this value (1 to 128). A run length of zero pixels can never be encoded. Raw packets, however, have the high bit of the count byte always set to zero. This differentiates raw packets from encoded packets, in which the high byte is always one. Following the count byte is the number of pixels equal to the count. The number of bytes that follows the raw count is equal to the count value multiplied by the number of bytes per pixel.

The following TGA RLE pseudocode algorithm is used to encode a pixel run using an encoded packet:

```
Set counter to zero
Read a pixel of scan-line data
Read a second pixel of scan-line data
If the first pixel is the same as the second pixel
        increment the counter
Else
        write the counter value (with high bit ON)
        write the pixel value
```

The following TGA RLE pseudocode algorithm is used to encode a pixel run using a raw packet:

```
Set counter to zero
```

```
Read a certain number of pixels of scan-line data
Increment the counter for each pixel read
Write the counter value (with high bit OFF)
Write all pixel values read
```

Figure TGA-1 shows the RLE packet types for various pixel sizes.

Figure TGA-1: Run-length encoding packet types

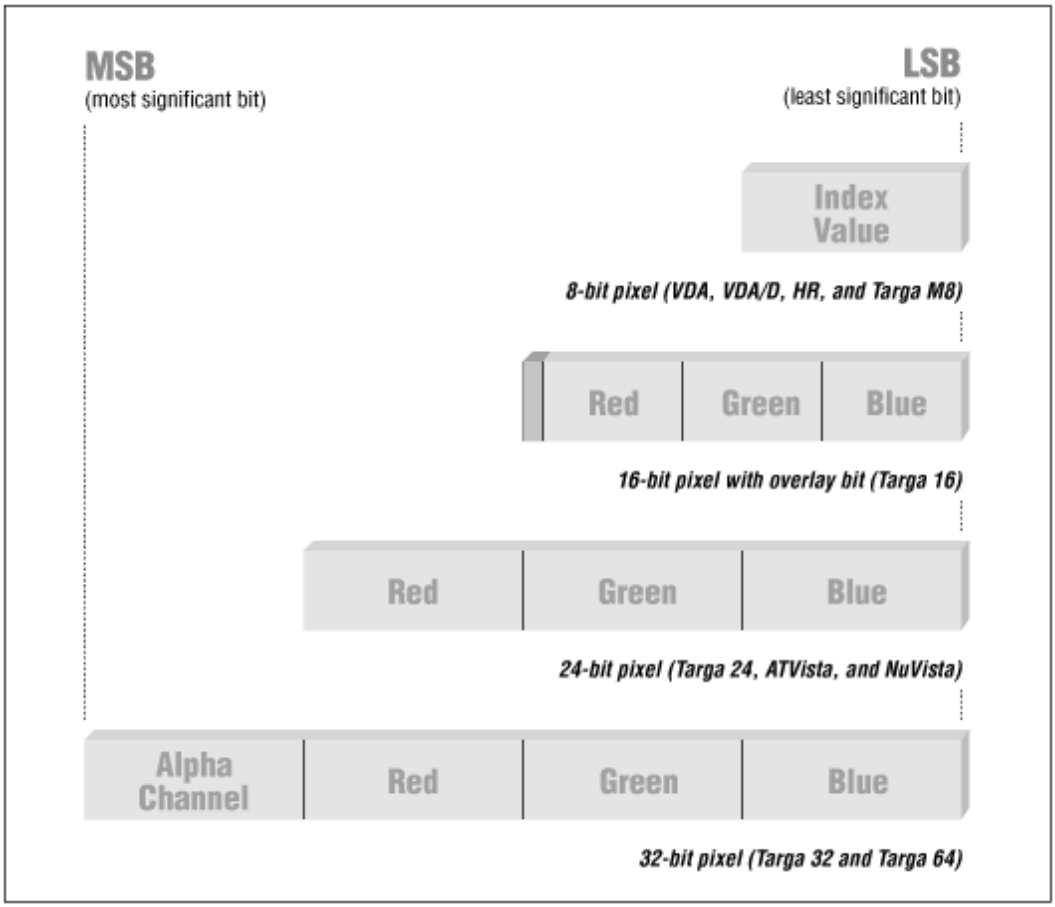


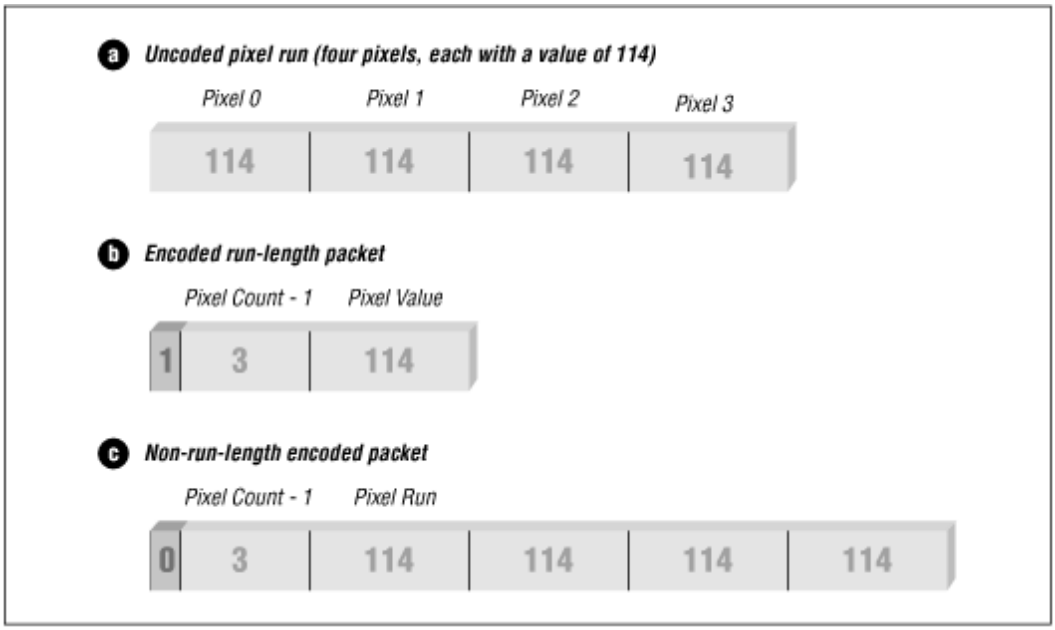
Image Data

Image data is usually found following the header, but may occur after a palette or Image ID field if these are present in the TGA file. For this reason, never read image data from a TGA file without first checking for the presence of palette and Image ID fields. If you do, you will quickly notice that the displayed image is skewed, because the image was read starting at the wrong offset. (See the section called "Colormap" above.)

The size of a TGA image is limited to 65,535 pixels high by 65,535 pixels wide. This is because a 16-bit field is used to store the size of the image in the header. Otherwise, the size of a TGA image would be unlimited. A typical size for Targa 16, 24, and 32 images is 512x482 pixels; the NuVista is 640x480 pixels; and the ATVista is 756x486 pixels.

Figure TGA-2 shows different pixel data formats.

Figure TGA-2: Pixel data formats



Most of the Truevision display adapters store pixel data in 8-, 16-, 24-, or 32-bit increments. Reading or writing pixel information for these formats is as simple as reading and writing bytes of data. Targa 16-bit pixel data, however, is slightly more complicated, as described below.

When a value of 15 appears in the PixelDepth field of the Image Specification section of the header, there are five bits each of red, green, and blue pixel data and one bit of overlay data in each pixel (see the section below called "Pixel Attribute Bits"). This is the format the Targa 16 uses to store data. Because these 16 bits are stored in two bytes of data, a little shifting and masking is required to read and write these pixel data values.

In the following example, a scan line of unencoded pixel data is stored in the byte array pixeldata[]. The red, green, and blue values are five bits in size, and the overlay attribute is a single bit in size. Data from the first pixel (array elements 0 and 1) are read and stored in the variables defined:

```
/*
** Reading and writing 16-bit pixel data stored
** in an 8-bit BYTE array.  Note that the green
** value is split between two bytes.
*/
BYTE red, green, blue, overlay;
BYTE pixeldata[SCAN_LINE_LENGTH];
/* Read */
red   =  (pixeldata[0] & 0xf8) >> 3;
green = ((pixeldata[0] & 0x07) << 2) | ((pixeldata[1] & 0xfb) >> 6);
blue  =  (pixeldata[1] & 0x3e) >> 1;
overlay =   pixeldata[1] & 0x01;
/* Write */
pixeldata[0] =  (red  << 3)          | (pixeldata[0] & 0x07);
pixeldata[0] = ((green & 0x1c) >> 2) | (pixeldata[0] & 0xf8);
pixeldata[1] = ((green & 0x03) << 6) | (pixeldata[1] & 0xf8);
pixeldata[1] =  (blue << 1)          | (pixeldata[1] & 0xc1);
```

```
pixeldata[1] =   overlay             | (pixeldata[1] & 0xfe);
/*
** Reading and writing 16-bit pixel data stored
** in a 16-bit WORD array
*/
BYTE red, green, blue, overlay;
WORD pixeldata[SCAN_LINE_LENGTH];
/* Read */
red      = (pixeldata[0] & 0xfc00) >> 11;
green    = (pixeldata[0] & 0x07e0) >> 6;
blue     = (pixeldata[0] & 0x003f) >> 1;
overlay  =  pixeldata[0] & 0x0001;
/* Write */
pixeldata[0] = (red   << 11) | (pixeldata[0] & 0x03ff);
pixeldata[0] = (green << 6)  | (pixeldata[0] & 0xfc1f);
pixeldata[0] = (blue  << 1)  | (pixeldata[0] & 0xffc1);
pixeldata[0] =  overlay      | (pixeldata[0] & 0xfffe);
```

Pixel Attribute Bits

The names of the Targa display adapters include a designation that indicates the number of bits per pixel that they are capable of storing. This seems logical for the Targa 16 and the Targa 24, but not for the Targa 32 and Targa 64. It's difficult to believe 32 and 64 bits per pixel until you realize that color data is not the only information you can store in a pixel.

The number of bits a pixel may contain that are not directly associated with the color value of the pixel are stored in bits 0 through 3 of the ImageDescriptor field of the header (attribute bits per pixel). In the case of the Targa 16, only 15 of the 16 pixel bits are used for color information. The sixteenth bit, also called the overlay bit, is used to indicate whether the pixel is transparent (invisible) or opaque (visible) when displayed on a video monitor. The ICB board also uses 15 bits per pixel for color information and a single bit for overlay control. The VDA/D board is similar in that it uses five bits per primary color and uses the sixteenth bit for interrupt control. The Targa 32 and the TrueVista boards (ATVista and NuVista) each use 32 bits per pixel. Color information is stored in 24 bits, and the additional eight attribute bits in each pixel are used as an alpha channel value.

Alpha channel is a nondescript name that indicates the degree of transparency of a displayed pixel. Alpha channel and overlay values are used when one image is overlaid onto another image or onto a live video picture. A single overlay bit (as in the Targa 16) can only indicate that the pixel is visible or invisible. Eight bits of precision can vary the visibility of a pixel from completely transparent (0) to completely opaque (255).

The alpha channel value also describes the degree to which a pixel from an image is mixed with a live video source. An alpha value of 0 displays the pixel entirely from the graphics image stored in the frame buffer memory. An alpha value of 255 displays the pixel entirely from the live video source. An alpha value of 84 displays the pixel as 33 percent graphics image (85 of 256) and 67 percent live video (171 of 256). A pixel with an alpha value of 84 appears as a translucent graphics image overlaid on a field of live video.

The ability to control graphical and live video images using 256 alpha channel levels allows the superimposing of graphical text over video, fading into and out of an image, and cross-fading between graphical images and live video. All of these effects can be rendered using the Truevision Targa+ and NuVista+ graphical display boards.

When storing pixel data or pixel size, the attribute bits are always included in any reads, writes, or calculations. Attribute bits are also stored in colormaps and lookup tables, although in these cases their values are usually set to zero and ignored.

The New TGA Format

Version 2.0 adds several features to the original TGA format, which increases the amount of information the TGA format can support. The original TGA format is fairly simple in design, which allows it to be quickly and easily implemented in software. However, it does not contain many features needed by developers--features that are found in several other image file formats. Therefore, extensions have been added to the TGA format to allow the storage of additional image information and to create a customizable area for the storage of developer-specific information.

The extensions added by the new TGA format are called the Developer Area, the Extension Area, and the TGA File Footer. These areas were appended to the original TGA format without making any changes to the TGA file header. Applications that read only original TGA image files should have no problems reading new TGA format files unless the Developer or Extension Areas contain information necessary to read or display the image data properly. For newer applications that support the new TGA format, it is important to correctly interpret as much of the Extension and Developer Areas information as possible.

Footer

The footer is 26 bytes in length and is always the last piece of information found in a v2.0 TGA format file. It contains a total of three fields that are represented in the following structure:

```
typedef struct _TgaFooter
{
  DWORD ExtensionOffset;        /* Extension Area Offse */
  DWORD DeveloperOffset;        /* Developer Directory Offset */
  CHAR Signature[18];           /* TGA Signature */
} TGAFOOT;
```

ExtensionOffset is a 4-byte offset value of the extension area of the TGA file. If this value is zero, then there is no extension area present in the TGA file.

DeveloperOffset is the number of bytes from the beginning of the file (byte 0) to the first byte of the developer directory. If this value is zero, then the TGA file does not contain a Developer Area.

Signature contains an identifying signature string. The TGA format does not

contain a field in the header indicating the version of the format. Instead, v2.0 includes a footer containing a 16-byte character string that identifies the version of the file.

To determine the version of a TGA file, read the file footer and check bytes 8 through 23 of the footer for the presence of the character string TRUEVISION-XFILE. If this signature string is present, the TGA file is v2.0 and thus may contain a Developer and Extension Areas.

Following the signature is the ASCII value 2Eh (period) and the ASCII value 00h (NULL). All of these fields must contain the correct information for the file to be recognized as a v2.0 TGA format file.

A v1.0 TGA file may be converted to a v2.0 TGA file simply by appending a footer with the appropriate signature string, a period, and NULL characters. In this case, you could set the ExtensionOffset and DeveloperOffset to 0. Be cautious about doing this, however. This conversion adds nothing to the functionality of the file unless an Extension or Developer Area is added. It will not, however, interfere with the ability of older, pre-v2.0 TGA format software to read the file.

Developer Area

The Developer Area begins with a directory that resembles the structure of the Image File Directory found in the TIFF format. The first entry in the directory is the number of directory entries, or tags, that the directory contains. This field is two bytes in size. The offset value of this field is stored in the footer described in the previous section.

Following the number of directory entries is a series of 10-byte tags, one for each entry specified. The structure of a Developer Area tag is as follows:

```
typedef struct _TgaTag
{
  WORD  TagNumber;      /* ID Number of the tag */
  DWORD DataOffset;     /* Offset location of the tag data */
  DWORD DataSize;       /* Size of the tag data in bytes */
} TGATAG;
```

TagNumber is the identification number of the tag. Tag number values from 0 to 32767 are reserved for developer use, while tag number values 32768 to 65535 are reserved for use by Truevision only. Tags may be registered with the Truevision Developer Services to assure permanent and exclusive use by your application.

DataOffset contains the offset location of the data in the TGA file. Note that offsets are always calculated from the beginning of the file.

DataSize is the size of the data in bytes.

The tags in the developer directory are always stored as a contiguous block, and the tags do not have to be sorted by tag number. Tags do not indicate the type of

data pointed to by the tag (BYTE-, WORD-, or DWORD-oriented data), so an application that is reading tag data is required to have prior knowledge of the type of data the tag points to.

Extension Area

The Extension Area can be thought of as a second header that contains information not found in the original TGA format header. The offset of the Extension Area is stored in the TGA footer. The size of the Extension Area in v2.0 is 495 bytes. The structure of the Extension Area is as follows:

```
typedef struct _TgaExtension
{
  WORD  Size;                  /* Extension Size */
  CHAR  AuthorName[41];        /* Author Name */
  CHAR  AuthorComment[324];    /* Author Comment */
  WORD  StampMonth;            /* Date/Time Stamp: Month */
  WORD  StampDay;              /* Date/Time Stamp: Day */
  WORD  StampYear;             /* Date/Time Stamp: Year */
  WORD  StampHour;             /* Date/Time Stamp: Hour */
  WORD  StampMinute;           /* Date/Time Stamp: Minute */
  WORD  StampSecond;           /* Date/Time Stamp: Second */
  CHAR  JobName[41];           /* Job Name/ID */
  WORD  JobHour;               /* Job Time: Hours */
  WORD  JobMinute;             /* Job Time: Minutes */
  WORD  JobSecond;             /* Job Time: Seconds */
  CHAR  SoftwareId[41];        /* Software ID */
  WORD  VersionNumber;         /* Software Version Number */
  BYTE  VersionLetter;         /* Software Version Letter */
  DWORD KeyColor;              /* Key Color */
  WORD  PixelNumerator;        /* Pixel Aspect Ratio */
  WORD  PixelDenominator;      /* Pixel Aspect Ratio */
  WORD  GammaNumerator;        /* Gamma Value */
  WORD  GammaDenominator;      /* Gamma Value */
  DWORD ColorOffset;           /* Color Correction Offset */
  DWORD StampOffset;           /* Postage Stamp Offset */
  DWORD ScanOffset;            /* Scan-Line Table Offset */
  BYTE  AttributesType;        /* Attributes Types */
} TGAEXTEN;
```

Size specifies the number of bytes in the extension area. This value is 495 for v2.0 of the TGA format.

AuthorName allows the name of the TGA file creator to be stored using up to 40 characters. Unused characters are padded with spaces and the field is NULL-terminated.

AuthorComment is also a string field containing 324 bytes (four 80-character lines) in which to store information. This field is similar to the Image ID field that follows the header. This field is also padded with spaces and is NULL-terminated.

The six Stamp fields contain the time and date the image was created or last modified. These fields may have the following values:

StampMonth	1 to 12
StampDay	1 to 31
StampYear	0000 to 9999
StampHour	0 to 23
StampMinute	0 to 59
StampSecond	0 to 59

Unused fields are set to zero.

JobName is a 4-byte, NULL-terminated string identifying the production job with which the image is associated. JobHour, JobMinute, and JobSecond indicate the amount of time expended on the job.

The SoftwareId field is a 40-character, NULL-terminated string that identifies the software application that created the file. The VersionNumber and VersionLetter fields contain the version of the software application.

KeyColor contains the background color of the image. This is the pixel color used to paint the areas of the display screen not covered by the image or the color used to clear the screen if the image is erased. The default value for this field is 0, which corresponds to black.

PixelNumerator and PixelDenominator store the aspect ratio of the pixels used in the image. If no aspect ratio is specified, then these fields are set to 0.

GammaNumerator and GammaDenominator are the gamma correction values to be used when displaying the image. If both fields are 0, then a gamma value is not used.

ColorOffset contains an offset value of the color correction table. If the file does not contain a color correction table, then the value of this field is set to 0.

The StampOffset field contains an offset value of the postage-stamp image included in the TGA file. If no postage-stamp image is present, the value of this field is 0.

ScanOffset contains the offset value of the scan-line offset table.

AttributesType describes the type of alpha channel data contained within the pixel data. A value of 00h indicates that the image data contains no alpha channel value. A value of 01h, 02, 04h, or 08h indicates the presence of alpha channel data. (See the TGA specification for more information on this field.)

Scan-line table, postage-stamp image, and color correction table

A new TGA format file may contain three additional data structures not found in the original TGA format. These structures are the scan-line table, the postage-

stamp image, and the color correction table. There may only be one of each of these data structures per TGA file, and offsets to these structures appear in the Extension Area.

The scan-line table is a method of accessing scan lines at any location within raw or compressed image data. The table is an array of DWORD values. Each value is the offset location from the beginning of the file to the beginning of the corresponding scan line in the image data. There is one entry in the scan-line table per scan line in the image. Entries are written to the table in the order in which the scan lines appear within the image.

The postage-stamp image is a smaller rendering of the primary image stored within the TGA file. The first byte of the postage-stamp data specifies the width of the stamp in pixels, and the second byte specifies the height, also in pixels. Postage stamps should not be larger than 64x64 pixels, are typically stored in the same format as the primary image, and are never compressed.

The color correction table is an array 1000 bytes in length, which contains 256 entries used to store values used for color remapping. The entire table has the following format:

```
typedef struct _TGAColorCorrectionTable
{
  SHORT Alpha;
  SHORT Red;
  SHORT Green;
  SHORT Blue;
} TGACCT[256];
```

The fields Alpha, Red, Green, and Blue store the color values for each entry. The range of each value is 0 to 65535. Black is 0,0,0,0, and white is 65535,65535,65535,65535.

For Further Information

For further information about the TGA format, see the specification and code example included on the CD-ROM.

The TGA format is maintained by Truevision, Inc. Prior to this writing, Truevision was acquired by RasterOps, Inc., but for the time being Truevision remains an independent subsidiary. Copies of the latest TGA specification, including a sample code disk, may be obtained directly from Truevision:

Truevision Incorporated
7340 Shadeland Station
Indianapolis, IN 46256-3925
Voice: 317-841-0332
FAX: 317-576-7700
BBS: 317-577-8783
FTP: *ftp://ftp.truevision.com/*
WWW: *http://www.truevision.com/Truevision.html*

Also on their Internet sites, you can get the TGA v2.0 file format specifications, as well as tools in C that read and write the format, and sample images.

More Resources