INTERGRAPH
RASTER FILE FORMAT
REFERENCE GUIDE

March 1994 - Version 3.2.0

This document replaces SDN 84-007

INTERGRAPH

Copyright

Intergraph Corporation
Huntsville, Alabama 35894-0001

Trademarks

Restrictions

This document is available by contacting:

Intergraph Corporation
Scanning Systems
Huntsville, AL 35894-0001

raster@ingr.com

TABLE OF CONTENTS

INTERGRAPH RASTER FILE FORMAT REFERENCE GUIDE

This document is the Intergraph Raster File Format Reference Guide. It is intended to be of use to Intergraph and other application programmers in developing software to access and create Intergraph raster image files used in conjunction with Intergraph systems.

These formats are considered the property of Intergraph Corporation, Huntsville, Alabama. No warrantee is implied nor expressed. See usage restrictions on page two of this document.

For further information contact:

Intergraph Corporation
Scanning Systems
Huntsville, Alabama USA 35894-0001
raster@ingr.com

INTRODUCTION

This document is designed to assist application developers within Intergraph and customers who have agreed to protect Intergraph proprietary information. It will assist the development of software designed to read or write Intergraph raster files. This document defines the standard location of information in the raster file and includes a brief history of raster file development as well as a short glossary of raster related terminology.

BACKGROUND INFORMATION

The following section provides pertinent background information about how Intergraph has approached development of raster applications.

Raster files are used to store regular, rectangular arrays of data. Raster files are most commonly used to store pictorial image information obtained from scanners or created with raster editors.

As the image elements of scanners pass over the area to be imaged, individual sensors capture the intensity of light reflected from the surface. This is true of engineering drawing scanners, and for "scanners" used to image the earth's surface from satellites.

The Intergraph system uses a rectangular grid of matrix locations composed of a series of lines. "Lines" of raster data are comprised of pixels. The data is organized in the file one line at a time.

Each Intergraph raster file consists of a number of "Header Blocks," optional application specific information in the header, and the raster data.

"Blocks" in this context refers to contiguous 512 byte sections of the raster file. This terminology originated with the DEC VAX disk file structure of "blocks" of 512 bytes.

"Words" in this document refer to 16 bit data items regardless of the actual word size of the computer system used.

The header blocks contain information about the raster data, including size, orientation, and storage format. Some information in the header is optional, and applications should either ignore this data, or copy it intact to the output file. A minimal amount of information is necessary to decode, display and manipulate the raster data. Data items in the header blocks are at fixed locations. See the sections on header blocks for a full discussion of their structure.

Raster image data can consist of bi-level data, where the pixel is either "black" or "white," "on" or "off," or "foreground" or "background". Typically this data is the result of a scanner imaging a piece of paper. Byte per pixel data is usually a single (monochrome) sensor observation at each pixel location. In addition to monochrome images, byte-per-pixel data can be "palette color" images where the value of the pixel is an index into a color look-up table that contains the actual red, green and blue intensities of the image.

Other types of data can be stored as raster as well. The elevation of the earth's surface is one example of data that can be stored in a raster file. Much of Intergraph's historical raster data is information that is not image related.

Since raster data can be very voluminous, various compression techniques are used to minimize valuable computer resources. Intergraph has developed a large number of compression techniques, and uses many of the industry standards as well. Run length encoding techniques have been used in the Intergraph environment to store bi-level image data, and color palette data. More recently, CCITT Group 4 compression techniques are used to store bi-level information.

RASTER FILE VERSIONS

As the raster file format has evolved, it has become necessary to identify three major versions of the format. Significant numbers of Version 1 and Version 2 files are in the customer base, and applications must be prepared to accept these files as input

data. All new files that are created must follow this standard, and indicate that they are Version 3 files.

Version 1 raster files are assumed to have two block headers, with DEC-VAX format strings (a leading character count, followed by the string), and VAX floating point formats.

Version 2 was a transition period, and the format is not consistent. Those applications that have written data prior to approximately 1987 are likely to be in VAX format. and since then, are likely to be compatible with the Version 3 formats. However, for imaging applications, the important applicable data is in neither the described string, nor floating point format.

Version 3 formats are mandated to have IEEE floating point formats, and use ASCII null terminated strings. In addition, any software applications that read Version 3 formats should accept files with arbitrary length headers.

DATA ORIENTATION

The orientation word is a bit-mapped word that indicates the major direction of the raster data. In a rectangular coordinate system there are four corners, and two directions from each corner, yielding eight possible pixel orientations. These are commonly referred to by defining the corner of the first pixel in the file, and the orientation of the scanlines. For example, "Upper-Left- Horizontal," indicates that the first pixel in the file is in the upper left corner, and the scanlines are horizontally oriented. This definition is dependent on the information in the image, and on the normal viewing orientation. Such a convention is excellent for documents, but when considering imaging the earth's surface, there is no "horizontal" or "vertical" direction for the scanlines, and the selection is somewhat arbitrary.

Scanners and plotters have preferred directions. These devices generally have a sensor or writing system that can process data much faster in one direction. The scanlines in the raster file can be oriented in a direction that is preferable. The decision of

orientation should be part of an overall systems design. To reduce the number of possible orientations, Intergraph has chosen Upper-Left-Horizontal (ULH) and Lower-Left-Vertical (LLV) as the default orientations. These two orientations differ only in a 90 degree rotation relative to the image content.

Typically the smallest scanners and plotters possible are purchased for an application. If scanning is done in such a manner as to produce either ULH or LLV data, that data is in an ideal orientation for the plotter. Display applications can usually display data in either orientation in an efficient manner. But scanners and plotters are very inefficient if they have to rotate raster data during the scanning or plotting process. These pieces of equipment represent a major purchase, and need to be as fully used as possible.

RESOLUTION

The resolution of the image is the distance between adjacent sensors on the original object. It is defined in the Intergraph file header as "dots-per-inch" or in microns. If the signed word in the resolution location is negative, the absolute value of the number indicates image resolution in dots per inch. Should the value be positive, it indicates the number of microns between one scanline and the next.

SCANLINE HEADERS

Intergraph has used special "headers" on graphic elements for many years to help software skip through a graphic file in a very efficient manner. The raster file format optionally includes these headers as well. The "scanline header" includes the length of the graphic element (scanline) to allow software to easily skip to the next scanline.

Scanline headers help find a random location in some compressed data formats, and add nothing to the file in others. In an uncompressed file, it is a trivial process to find any arbitrary line and pixel coordinate in the file. In run length

compressed data, the scanline headers can be used to skip over un-wanted data to find the desired line.

COLOR TABLES

The header also contains a data structure that can be used to map the raster values to color triplets for display. The color display uses three (red, green, and blue) colors to create almost any color available.

Two different types of color table structures are defined for the Intergraph raster file. The first of these was defined in the IGDS vector display system. IGDS defined 256 colors, each with eight bits of intensity. The color value for the first slot (the 0^{th} slot) contains the intensity triplet for the "background," followed by 254 color triplets for the vectors in the file. The 256^{th} color slot is used for the cursor.

The second type of color tables are used in what is called the "standard" color table format. This color table structure does not assume that all the color values, or slots, are defined, and as such the color table contains quadruples consisting of a value (slot number), and the red, green, and blue intensities. Only those values required are included in the color table, and there is no restriction on their order.

It is important to note that the color table is not mandatory; some imaging and scanning applications will create files without color tables. The default behavior for applications when they encounter a file without a color table is to assume that the values in the file are image data, and should be manipulated as if there was a "identity" color table in the header. For example, if the file had an IGDS format color table, it's contents would be the values from 0 to 255. In the case of greyscale images, the value of 0 represents the darkest part of the image, and the value of 255 represents the part of the image of the highest intensity.

The second half of Block Two, and all of Block Three are reserved for color table usage. Should an application not use

this area for color tables, or not use all of this area, it is available for application specific information.

HEADER CONTENTS

The Intergraph file format uses a fixed header scheme to store information about the raster data in the file. Specific, fixed locations are assigned by this document to contain specific data elements.

The raster file formats were first defined when Intergraph was using VAX systems as the primary graphics processors. Many of the concepts are based on VAX conventions and are more easily discussed with these concepts in mind.

As discussed in the section on raster file versions, the VAX floating point format is used in Version 1 and in some Version 2 files. Floating point formats used in later versions of Version 2 and all Version 3 files meets the IEEE Floating point standard. Applications that need to use the floating point formats in the header must accept data in either of these formats.

As a minimum requirement, all files must contain at least two blocks of header information. The term "block" in this document relates to the VAX disk blocking factor of 512 bytes per physical disk block. The convention of allocating the header size in "blocks" will be continued. However, all applications must be able to handle an arbitrary number of header blocks. Some early VAX applications could only handle data with two blocks of header information.

Every raster file has a minimum amount of information necessary to decode and interpret the raster data. All applications can depend on this minimum information being present in any Intergraph Raster File. Other information is included at the discretion of the application and should not be relied on, but applications must handle any information that is present.

FOREIGN SYSTEM INTERCHANGE FORMATS

The interchange of raster data with "foreign" systems is of increasing importance to our customer base. There are two existing (and emerging) standards supported at this point in time. Intergraph is committed to support both of these standards, and to supply translators to each of these formats. Application developers are advised to consider the use of these formats as alternatives to writing their own converters.

TIFF, or "Tag Image File Format," is widely accepted in the personal computer industry as a standard interchange format. The TIFF format uses a series of "tags," or small data structures that identify various characteristics of the raster file. TIFF files can store various types of raster data from one bit per pixel through color raster imagery.

The United States Federal Government has initiated a series of standards in the CALS, or Computer Aided Acquisition and Logistic System. One aspect of these standards defines raster file formats for exchange of images and scanned drawings. The current standards use CCITT Group 4 compression of bi-level images.

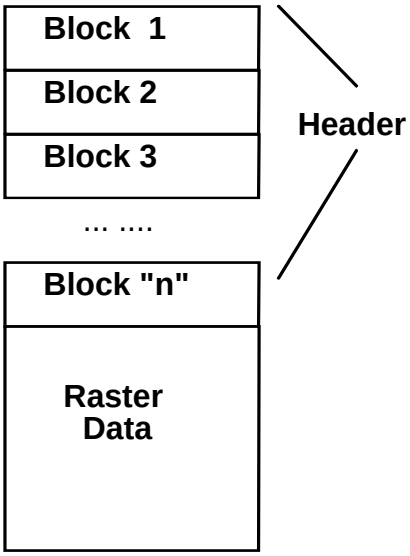
FILE FORMAT

This section describes the format of raster data files. Each raster data file consists of a file header and associated raster data.

The raster data formats described are various compression techniques used to store the raster data. Not all of the formats described are used currently, but application developers should be aware of their existence. Some of the formats are not used by imaging systems, but are used by other applications. Application developers should exit gracefully when they encounter formats that they are not prepared to read.

RASTER FILE HEADER BLOCKS

Raster file headers contain information about the raster data format in the file. The first part of this section provides information on the content, structure, and format of these raster file headers. Each raster file contains a header at least two blocks (a total of 1024 bytes) long that stores information used by different applications. This section describes the format of these header fields. To maintain compatibility with older files, headers have at least two blocks.



Raster File Structure

Raster file headers contain information about the raster data format in the file. The first part of this section provides information on the content, structure, and format of these raster file headers. Each raster file contains a header at least two blocks (1024 bytes) long that stores information used by different applications.

This document uses the convention of 512 bytes per "block," and numbers blocks starting with the sequence number "1". Byte locations within the file and within the blocks start with the sequence number "0". This custom is deeply rooted in the VAX FORTRAN philosophy of starting with "1" and indicates our conversion to the "C" conventions of starting with "0". It may appear confusing at times, but at this juncture it appears unwise to define header blocks starting with "0".

Earlier applications assumed a fixed size header of two blocks, however new applications cannot assume the raster data in the file begins in byte 0 of Block Three. The length of the header varies greatly depending on the application that created the raster data file. Use the Words to Follow field in Block One (byte 2) to calculate the number of words in the header. Refer to "Header Fields Block One," for information about the Words to Follow field.

This document refers to "words". These words are 16 bit signed or unsigned data elements. Specific element formats will be described in detail.

Not all applications use every field in the raster file header. If an application does not use a particular field, it places an integer zero value in the field.

Applications should assume that an integer zero indicates that this field may be not used, or the value may be a valid zero. The context of usage should indicate the difference. This helps to prevent problems if the fields are defined later.

Applications that read raster files should only rely on required header information. Fields that are required should be sanity checked for invalid values. For example, if an application uses only the raster data types described in this document, the header of the raster file should be checked for a valid raster data types. If the header contains an invalid data type, the application should take appropriate action and display a message. Refer to the section on Header Fields Block One for information on required fields.

On Intergraph workstations, the elements of complex data structures are aligned on "natural" boundaries. Not all the elements in the header are aligned on these boundaries. The definition of a complex structure followed by a block read from the disk file leads to erroneous results. The recommended method of reading the header blocks is to read each block into a buffer and perform individual memory copies into data elements from the buffer.

Because of various word lengths on different computers, the following table defines the bit length and sign use of each data type in the header field descriptions:

DATA ITEM DEFINITIONS

DATA	TYPE	DESCRIPTION	Intergraph "C" Format
uint8	8 bit	unsigned integer	unsigned char
uint16	16 bit	unsigned integer	unsigned short
uint32	32bit	unsigned integer	unsigned int
int8	8 bit	signed integer	char
int16	16 bit	signed integer	short
real32	32 bit	IEEE floating point	float
real64	64 bit	IEEE floating point	double

In multiple byte data elements, the bytes are ordered in the least significant to most significant order. This is commonly referred to as the "Intel" format.

HEADER FIELDS DEFINITIONS - BLOCK ONE

The Header Block One table shows the name, symbol, type of data, and other specifications for each field in the first block (Block One) of a standard Intergraph raster file. The fields indicated are required in any raster file. Refer to "Header Field Descriptions Block One" following the table for additional information on the header fields.

HEADER BLOCK ONE DATA ITEMS

Element Acronym	Notes	Symbol	Element Type	Number of Elements	Beginning Byte
Header Type Code	1	HTC	uint16	1	0
Words to Follow	1	WTF	uint16	1	2
Data Type Code	1	DTC	uint16	1	4
Application Type	1	UTC	uint16	1	6
X View Origin		XOR	real64	1	8
Y View Origin		YOR	real64	1	16
Z View Origin		ZOR	real64	1	24
X View Extent		XDL	real64	1	32
Y View Extent		YDL	real64	1	40
Z View Extent		XDL	real64	1	48
Transformation Matrix		TRN	real64	16	56
Pixels Per Line	1	PPL	uint32	1	184
Number of Lines	1	NOL	uint32	1	188
Device Resolution		DRS	int16	1	192
Scanline Orientation	1,3	SLO	uint8	1	194
Scannable Flag	1	SCN	uint8	1	195
Rotation Angle		ROT	real64	1	196
Skew Angle		SKW	real64	1	204
Data Type Modifier	1	DTM	uint16	1	212
Design File Name		DGN	byte	66	214
Data Base File Name		DBS	byte	66	280
Parent Grid File Name		PRN	byte	66	346
File Description		DES	byte	80	412
Minimum Value	2	MN1	uint8	1	492
Minimum Value	2	MN2	uint16	1	492
Minimum Value	2	MN4	uint32	1	492
Minimum Value	2	MNR	real32	1	492
Minimum Value	2	MN8	real64	1	492
Maximum Value	2	MX1	uint8	1	500
Maximum Value	2	MX2	uint16	1	500
Maximum Value	2	MX4	uint32	1	500
Maximum Value	2	MXR	real32	1	500
Maximum Value	2	MX8	real64	1	500
Reserved		RV1	byte	3	508
Grid File Version	1	VER	uint8	1	511

NOTES

1 -> Required field

2 -> Interpretation of this field is application dependent.

3 -> SLO - bits 0,1,2 are used in this byte

Field Descriptions

This section contains descriptions of the file header fields listed in the Header Section 1 table. Refer to the table for information on required fields, data types, number of units, and beginning byte.

Header Element Type Word (HTC)

The Header Element Type Word field has the following format:

IGDS element type = 9 (bits 8 through 15)
"3D" = 3 or "2D" = 0 (bits 6 and 7)
IGDS major version level = 8 (bits 0 through 5)

Applications should check for IGDS element type 9, version 8. If this element is not in the file, the file is not a valid Intergraph raster file. This 16bit integer should be 0x0908.

Words to Follow (WTF)

The Words to Follow field helps determine how many blocks are in raster data file header. The number in the field represents the number of 16 bit words that follow this field. If the header has only two blocks, the number of words to follow is 510. (There are 256 words in Block two and 254 words to follow for Block One.) If the number in the field is greater that 254, use the following formula:

$$a = \frac{WTF + 2}{256}$$

(where *"a"* is the total number of header blocks)

Applications should always read the first block of the header and then calculate the number of additional blocks in the header. If the number of header blocks is not an integral number, then the file is not a valid raster file.

Data Type Code (DTC)

The Data Type Code field contains a code which identifies the format of the raster data portion of the file. The following list describes the codes used for this field and the raster data format they indicate. Refer to the section on Raster Data Formats for format descriptions.

RASTER FORMAT TYPES

Code	Format	Notes
1	Packed Binary	1 bit / pixel
2	Byte Integer	8 bits / pixel
3	Word Integers	16 bits / pixel
4	32 bit Integers	
5	32 bit Floating Point	
6	64 bit Floating Point	
7	Complex	64 bits / pixel
8	Double Precision Complex	
9	Run length Encoded	Bi-level Images
10	Run length Encoded	Gray Scale, Color
11	Figure of Merit	FOM
12	DTM Flags	
13	RLE Variable Values with Z	Simple
14	RLE Binary Values	w/ Edge Type
15	RLE Variable Values	w/ Edge Type
16	RLE Variable Values with Z	w/ Edge Type
17	RLE Variable Values	Color Table and Shade
18	RLE Variable Values	w/ Normals
19	Quad Tree Encoded	
24	CCITT Group-4	Bi-level Images
25	Run Length Encoded RGB	Full Color
26	Variable Run Length	
27	Adaptive RGB	Full Color
28	Uncompressed 24 bit	Full Color
29	Adaptive Gray Scale	
30	JPEG	Gray Scale
31	JPEG	Full Color RGB
32	JPEG	CYMK
67	Continuous Tone	CYMK
68	Line Art	CYMK/RGB

Applications can use an application data type of 0 in order to use the standard header format as the header containing only application specific data. In this case, the application specific data type can be indicated in the application header. This is now a nonstandard use of the raster file format, and software that cannot handle this variant should exit gracefully.

Application Type Code (UTC)

The Application Type Code (UTC) identifies the application that created the file. It is used to determine the applicability of application specific data in the header. If the software does not recognize the application type, only

the information in Block One, the first 128 words of Block Two, and the color table information is reliable. The application should ignore rest of the header.

The following is a list of the applications using raster data files and the corresponding codes used in the Application Type code header field.

RASTER APPLICATION TYPES

Code	Application Type
0	"Generic" Raster Image File
1	Digital Terrain Modeling
2	Grid Data Utilities
3	Drawing Scanning
4	Image Processing
5	Hidden Surfaces
6	Imagitex Scanner Product
7	Screen Copy Plotting
8	I/IMAGE and MicroStation Imager
9	ModelView

In the past, the last 128 words in Block Two of the header were reserved for specific application information. If the application required further storage space for specific data, additional blocks can be added. If a file contains more than 2 or 3 header blocks, and has a defined application type code reading applications should copy intact any header information, and append application data packets to the end of the existing header.

This technique is obsolete. Applications must use Applications Packets to store application specific information. Each application group is responsible for documenting Application Packets specific to the application and releasing that information to others. For more information on Applications Packets, refer to the section describing Application Packets.

View Origin (XOR, YOR, ZOR) and View Extent (XDL, YDL, ZDL)

The View Origin and View Extent fields are application dependent and were used primarily by the Digital Terrain Mapping applications in file versions prior to Version 3 to orient raster data with relationship to an IGDS design file.

Proper interpretation of the homogeneous transformation matrix (used by most applications) essentially replaces this field for imaging applications. Refer to the Homogeneous Transformation Matrix (TRN) field description for more information. Either the Transformation Matrix, or the View Origin and Extent fields must be filled in to maintain the specified relationship between the raster information and vectors in a design file display. If the View Origin and Extent fields are integer zero, the transformation matrix will define the relationship between the raster and the design file coordinates.

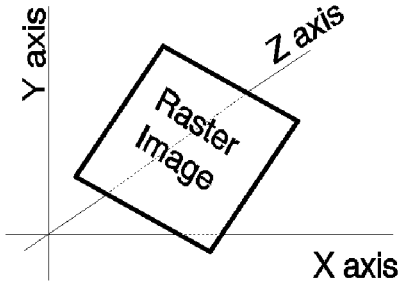
The relationship of raster (grid) data to a specific vector design file is defined by the numbers in these two fields. The numbers are currently interpreted as Units of Resolution (UORs).

The view origin (specifically x and y origin) defines the relationship of the grid file to the vector graphics database. (The Design File Name (DGN) field at byte 214 of the raster file header specifies the design file name.) The x and y origin point is the upper left corner of the grid file in UOR space.

The view extents are used to store the delta X (column spacing) and delta Y (row spacing) respectively. These values are also in UORs.

Homogeneous Transformation Matrix (TRN)

The Homogeneous Transformation Matrix (TRN) contains the 4 x 4 transformation matrix which completely specifies the relationship between the raster pixels and the IGDS world coordinate system. It includes the origin in the application coordinate space and scaling and rotation matrices. Raster data files are often used within the structure of an application coordinate system, referred to as a world coordinate system. Raster files include a coordinate system of their own, referred to as a local coordinate system. The raster data file may include appropriate information for a specific application to match the pixels with the world coordinate system.



The raster data in the file is a planar orthogonal matrix of pixels, the relationship with a three dimensional world coordinate system is totally defined by the pixels per line (PPL), the number of lines (NOL), the

scanline orientation (SLO), and the homogeneous transformation matrix (TRN).

The scanline orientation (SLO) serves as a standard to flip the position of the data in the raster file to another orientation. The transformation matrix can also be used to represent this reorientation. Raster data generated by scanners and plotters have natural boundaries. Raster data should be used with the natural orientation for these devices. All scanners and plotters have a preferred orientation.

The scan head in a line oriented scanner produces pixels in one direction faster than it will in the other directions. Line oriented scanners are generally designed in such a manner that the pixel direction is oriented along the scan head direction, and the lines are orthogonal to the scan head direction. The same is true for plotters, which are much more efficient in plotting a series of lines in one direction when the raster data is naturally oriented along that plot head.

Raster file orientations should be consistent from scanner to plotter. Raster display and editing software should accept the data oriented in the "natural" direction for the plotter and scanner. Software should transform the data, if necessary, for display. Orientations should be consistent for tiled files as well.

To obtain full transformation from local coordinates to world coordinates, catenate the homogeneous transformation matrix (TRN) with a flip matrix (T=TRN * FLIP). The concatenated matrix (T) now contains all the information required to map a particular pixel to the world coordinate system including the scanline orientation (SLO).

If a raster data file has a local coordinate system where (row, col)=(0,0) occurs at the origin and the first row of data runs along the X axis and the first column of data runs along the Y axis, the local coordinate representation of a row/column pair is (c,r,0). For example, the fourth pixel in the sixth line of raster data has local coordinates (3,5,0) and maps the following transformation to the world coordinates:

$$\begin{vmatrix} x_w \\ y_w \\ z_w \\ 1 \end{vmatrix} = T \times \begin{vmatrix} 3 \\ 5 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{vmatrix} \times \begin{vmatrix} 3 \\ 5 \\ 0 \\ 1 \end{vmatrix}$$

The origin of the raster data expressed as world coordinates occurs in the fourth column of the T matrix. Because the perspective is usually defined as a viewing transformation, it should not occur in the transformation above and the fourth row should always be (0,0,0,n), where n is the global scale factor, normally equal to one.

In the raster file header, the transformation matrix occurs in the following order:

t_{11}
t_{12}
t_{13}
t_{14}
t_{21}
t_{22}
etc

The View Origin (XOR, YOR, ZOR), View Extent (XDL, YDL, ZDL), Rotation Angle (ROT), and Skew Angle (SKW) header fields are redundant since the homogeneous transformation matrix completely describes the local and world coordinates. These fields were used in an application dependent manner. The transformation matrix is the accepted method of defining this information.

In a raster file it is recommended that either the scanline orientation (SLO) be used, or the transformation matrix to indicate rotation. For orthogonal rotations the scanline orientation with a unity transformation matrix is much preferable and should be used. As an acceptable option, a scanline orientation of "4" (upper left horizontal) and a transformation matrix indicating orthogonal rotation is acceptable. For non-orthogonal rotations, set the scanline orientation to "4" and include transformation matrix information.

Pixels Per Line (PPL)

The Pixels Per Line field specifies the number of pixels in one line of raster data.

Number of Lines (NOL)

The Number of Lines field specifies the number of lines of raster data in the image.

Device Resolution (DRS)

The Device Resolution field defines the scan resolution if the file was scanned or the distance between pixels in real word "paper" units. A positive value indicates the size of the pixel in microns. A negative value indicates the number of pixels per inch (or lines per inch). Some applications place a zero in this field when the "resolution" in paper units is meaningless (satellite images are an excellent example).

Scan Line Orientation (SLO)

The Scan Line Orientation field defines the data origin and line orientation for the raster file.

A raster data file is rectangular and the origin of the file is in one of the four corners. Starting from the origin, an application reads lines of raster data vertically or horizontally depending on the line orientation. Therefore, there are eight different possible orientations for a raster data file.

A code in the SLO file indicates both the origin and line orientation. Bits 0 and 1 indicate the origin. Bit 2 indicates the line orientation.

SCAN LINE ORIENTATION CODES:

Data Origin	Line Orientation	bit 2	bit 1	bit 0	Value
Upper Left	Horizontal	1	0	0	4
Upper Left	Vertical	0	0	0	0
Upper Right	Horizontal	1	0	1	5
Upper Right	Vertical	0	0	1	1
Lower Left	Horizontal	1	1	0	6
Lower Left	Vertical	0	1	0	2
Lower Right	Horizontal	1	1	1	7
Lower Right	Vertical	0	1	1	3

The most common orientations are upper-left-horizontal (4), lower-left-vertical (2) and upper-left-vertical (0). All applications should support at least these orientation types. Some applications may require the application provide raster in a specific orientation to improve performance of critical or expensive interface devices. For example, the standard electrostatic plotter assumes that data supplied for plotting is oriented with the raster lines perpendicular to the paper direction. This requirement is imposed to eliminate the need for time consuming rotation of raster data during the plotting process. Appropriate work flows should be designed to prevent unnecessary manipulation of raster data. Each scanner will provide raster data consistent with the mechanical/optical design, and a separate process step may be required to rotate the data from a specific scanner prior to storage or plotting.

Scannable Flag (SCN)

In the IGDS environment, raster "lines" could consist of one or many "scanlines," each of which had a unique position in the design coordinate space. This concept of one or many scanlines making up the larger image line has carried into the raster file formats.

The Scannable Flag field defines how the application should move to the next "scanline" of raster data without necessarily decompressing the raster data. The SCN field in the header contains a code that indicates which indexing method is used.

SCANNABLE FLAG FIELD CODES:

Code	Indexing Method Description
1	Every line of raster data has a 4 word raster line header at the beginning of the line. In the line header, the Words to Follow field specifies the amount of data following the field, indicating the start of the next scanline of raster data
0	No raster line headers exist. The application must calculate where lines of raster data start and end. This process is simple for non-run length encoded data. It is a fixed value; therefore, the line length can be calculated from the data type (DTC) and the number of pixels per line (PPL). In a run length compression case, the data must be decoded to find the end of a raster line.

Most run length encoded raster files contain scanline headers to assist the application in moving through the file. This is not a requirement of this standard, nor is its use recommended for future applications. In current systems, where large amounts of memory are generally available, alternative indexing methods can be generated at the time the file is read into memory. These methods make it much simpler for the application to keep track of the raster data and update file changes when necessary.

All applications should handle raster data with or without scanline headers. A significant number of files with scanline headers are currently in use. New applications should not depend on these headers, they may not be applicable in the future.

Some types of raster compression techniques are inconsistent with the concept of scanlines and scanline headers. CCITT Group-4 compression uses a "bit-stream" concept, and as a result, does not have well defined image line beginning and endings. CCITT Group-4 files should not contain scanlines nor scanline headers.

Rotation Angle (ROT)

The Rotation angle field specifies the counterclockwise angle of the raster data coordinate system. The use of this field is application dependent and not recommended since the homogeneous transformation matrix (TRN field) contains the full description of the relationship between the raster pixels and the world coordinate system.

For Intergraph RISC systems, the Rotation Angle field may not align "correctly." The field is a 64 bit number and is long word aligned. Application code should declare this field an array of unsigned elements and move the data to or from a data structure to prevent alignment problems.

Skew Angle (SKW)

The information in this field is application dependent since the homogeneous transformation matrix (TRN field) contains the full description of the relationship between the local and world coordinate systems.

For Intergraph RISC systems, the Skew Angle field may not align correctly. The field is a 64 bit number and long word aligned. Application code should declare this field an array of unsigned elements and move the data to or from a data structure to prevent alignment problems.

Data Type Modifier (DTM)

The Data Type Modifier field contains a modifier for data types not fully described by the code in the Data Type Code (DTC) field. This field should be set to zero for data types fully described in the DTC field. The only data type that uses a modifier is type 26 (Variable Run length Indexed Color).

This field is obsolete, and with the exception of the type 26 file, it should not be used by any new raster applications.

Design File Name (DGN)

The Design File Name field specifies the name of a design file if there is one associated with this raster data file.

This field is not used in imaging applications. It is extensively used by non-imaging applications. For example, Mapping software uses this field to obtain important information regarding the coordinate system definition.

Data Base File Name (DBS)

The Data Base File Name field specifies the name of a data base if there is one associated with the raster data file. The entity and attribute values refer to the data base specified in the field.

This field is obsolete, and is not required.

Parent Grid File (PRN)

The Parent Grid File field specifies the original raster data file from which the current file was extracted.

This field is obsolete, and is not required.

File Description (DES)

The File Description field provides space for an optional ASCII description of the raster data. The description limit is 80 characters. This field is optional, applications can place relevant text strings in this location.

Minimum Value (MN1, MN2, MN4, MNR, MN8)

The Minimum Value field contains the minimum raster data value in the file. It is an application dependent data structure.

Maximum Value (MX1, MX2, MX4, MXR, MX8)

The Maximum Value Field contains the maximum raster data value in the file. It is an application dependent structure.

Reserved (RV1)

Reserved fields are fields reserved for information to be added to the header. An unused reserved field should always be initialized to zero by any application that writes raster files.

Grid File Version (VER)

The Grid File Version field identifies the version number of the raster file. **This is a mandatory field.**

HEADER FIELD DEFINITIONS - BLOCK TWO

The following table shows the name, symbol, type of data, and other specifications for each field in the second block (Block Two) of the standard two block header.

BLOCK TWO FIELD DESCRIPTIONS:

Element Acronym	Notes	Symbol	Element Type	Number of Elements	Beginning Byte
Gain		GAN	uint8	1	0
Offset / Threshold		OFT	uint8	1	1
View #	Screen #1	VF1	uint8	1	2
View #	Screen #2	VF2	uint8	1	3
View Number		VNO	uint8	1	4
Reserved		RV2	uint8	1	5
Reserved		RV3	uint16	1	6
Aspect Ratio		ASR	real64	1	8
Catenated File Pointer		CFP	uint32	1	16
Color Table Type		CTV	uint16	1	20
Reserved		RV8	uint16	1	22
Number of CT Entries		CTE	uint32	1	24
Application Packet Pointer		APP	uint32	1	28
Application Packet Length		APL	uint32	1	32
Reserved		RV9	uint16	110	36
Application Data		USE	uint16	128	256

Field Descriptions

This section contains descriptions of the file header fields previously listed in the Header Block Two table. Refer to the table for information on required fields, data element types and number of units, and beginning bytes.

Gain (GAN)

The Gain field stores the gain of the analog to digital converter used during scanning.

Offset/Threshold (OFT)

The Offset/Threshold field defines the thresholding parameters or the analog to digital conversion parameters for scanning devices.

View Number (VF1, VF2, VNO)

The View Number fields define which view a raster data file represents and whether the view is in "quadrant mode."

Reserved (RV2 and RV3)

Reserved fields are fields reserved for information to be added to the header.

An unused reserved field should always be zero.

Aspect Ratio (ASR)

The aspect ratio is the ratio of the width and height of the pixel. It is the floating point number represented by dividing the physical width of the pixel to the physical height of the pixel.

Catenated File Pointer (CFP)

The Catenated File Pointer field represents the byte offset from the beginning of the file to the next image in the file. A value of zero indicates that this is the last image in the file. See the appendix for more information on multi-image file formats.

Color Table Type (CTV)

The Color Table Type field specifies the application independent color table associated with the file.

COLOR TABLE VALUES (CTV)

Code	Color Table Type
0	No Color Table
1	IGDS Color Table
2	Environ-V Color Table

IGDS Color Table

The IGDS color table is located in the second half of Block Two and Block Three (256*3 bytes total). The format consists of an array of 256 elements, each being three bytes. For each color, there are eight bits for red, eight bits for green, and eight bits for blue. When combined, these three components create other colors. Varying the values for any of the RGB components varies the color created by their combination. The following is an example of the beginning of a color table index. Each byte is a separate color component.

Index 0, Red Value
Index 0, Green Value
Index 0, Blue Value
Index 1, Red Value
Index 1, Green Value
Index 1, Blue Value
Index 2, Red Value

etc.

Index 255, Red Value
Index 255, Green Value
Index 255, Blue Value

When IGDS loads a color table, the background color is assumed to be at the first index, followed by the remaining 255 color indexes.

If the file has an IGDS format color table, the second half of Block Two is not available for application data. Again, application data packets should be used for ANY application specific data.

Environ-V Color Tables

An Environ-V color table consists of an array of indices starting at Block Three.

The following is the format of the Environ-V color table slot as it would appear in the native Intergraph-C language.

```
struct vlt_slot
{
unsigned short v_slot;
unsigned short v_red;
unsigned short v_green;
unsigned short v_blue;
};
```

The Color Table Entries header field defines the number of entries. All entries in the color table should have a value before adding any additional application data blocks. The Environ-V format does not have a fixed size; it specifies a slot or index number, and the associated color components. Individual workstations have different sizes of color indexing tables, and only those necessary entries in the file format are included. In addition, the

index or slot numbers are not in any specific order, but can be randomly placed in the table.

Should a color value be eight bits in length, the color value appears in the ***most significant*** byte of each slot value.

Reserved (RV8)

Reserved fields are fields reserved for information to be added to the header.

An unused reserved field should always be zero.

Color Table Entries (CTE)

The Color Table Entries field specifies the number of entries in an Environ-V color table. Each entry is four bytes long. IGDS color tables by definition, always have 256 entries.

Offset of Start of First Application Packet (APP)

The application packet offset is the **byte** offset from the beginning of the file to the first byte of the first application packet. Each application packet contains a "words to follow" field, allowing applications to index through the application packets. All application packets are located in the header of the raster file. Refer to the appendix for more detailed discussion of the application packets.

All application data packets are quad-word aligned in the raster file.

Length of Data Packets (APL)

This field is the total length of the application data. This field is an unsigned 32 bit integer. It is the total length of all the application packets in **words**.

Reserved (RV9)

Reserved fields are fields reserved for information to be added to the header.

An unused reserved field should always be zero.

Application Data (USE)

The Application Data field is application dependent unless an IGDS color table is defined (CTV=1) in which case it is the color table. If this field is unused, it should be set to zero.

The preferred method of storing application specific data is to use application packets. The storage of application specific data in this location is non-standard, and must be avoided. Refer to the appndix for more information on application packets.

RASTER DATA TYPES

This section describes the raster data type formats Intergraph uses to store raster data, including each data type's suggested use and compression technique. The formats vary from no compression at all to the maximum compression. The Data Type Code (DTC) field (byte 4, Block One of the raster file header) identifies the raster data format for the file. Refer to the section on the header fields of Block One for more information on the Data Type Code (DTC) field and a list of the codes.

RASTER DATA TYPES -- BACKGROUND

Before producing any software that reads and/or creates any of the formats described in this section, read the introduction in the Raster File Format Reference Guide, to determine the applicability of each specific format. Some of these formats are rarely used, and their documentation in this form is not a commitment by Intergraph to support these formats in the future.

RECOMMENDED STORAGE FORMATS

Image Type	Recommended	Currently Acceptable
Bi-Level	24	9, 24
Gray Scale	29, 30	2, 29
Palette Color	29	10, 29
24 Bit Color	27, 31	27

Those formats specifically defined below as "OBSOLETE" are not to be used! The information provided is a guide to assist in interpreting existing files, but cannot be relied on as sufficient documentation to create any of these files!!!

Scanline Headers

All lines of raster data can begin with scanline headers, though this method is not recommended. In the past, scanline headers provided a rapid method of indexing through a raster file if the file was compressed causing the exact length of the compressed data to be indeterminable before expanding each line. With the current memory and tiling techniques available, the use of scanline headers is limited and discouraged. New applications should not create files with scanline headers, but should be able to read files with or without headers.

Scanline headers are not valid with file types 27, 30, 31, 32, 67, 68 and 69. They should not be used in any of these file types.

Each scanline header begins with a file processor compatible identifier (value 0x5900), which identifies the header. All software that uses scanline headers should check this value to ensure the data integrity of each raster line.

SCANLINE HEADERS

Bits	Description	Value (Hex)
15	Delete Flag	0
8-14	IGDS Element type	59
7	Complex Flag	0
6	Reserved	0
0-5	IGDS Level #	0

Immediately following the identifier is a 16 bit "Words to Follow" field that indicates the number of words remaining in this raster line. The Words to Follow field is followed by a line number, a sequential number beginning at 1 for the first line of the image. This line number resets to 1 (or zero) after 65535 lines and then resumes incrementing. The last 16 bit word is the pixel offset within the line. This is commonly set to zero.

Type 1: Packed Binary

The type 1 format uses the packed binary storage technique. A single bit represents the value of each pixel in the image.

Each 16 bit word in the raster file represents 16 pixels in a raster data line. The most significant bit is the pixel farthest to the left on the line; the least

significant bit is the pixel farthest to the right. If the length of each raster line is not a multiple of 16, the remaining bits are filled with zeros.

This compression technique is used with bi-level images. The "ON" bits (value = 1) represent foreground information, and the "OFF" bits (value=0) represent background.

OBSOLETE - No application currently uses this format. The CCITT-Group IV compression technique is much more efficient, and is to be used in place of this format.

Type 2: Byte per Pixel

The type 2 format uses a single byte to store each pixel in the image.

Each 8 bit byte represents a single pixel in the raster file. No word order exists, and each sequential byte is the next pixel in the file. The interpretation of the sign bit is application dependent. The default is "unsigned." This value can be the gray scale intensity, or it can be an entry into a color lookup table either in the image or external. This format is most frequently used to store gray scale continuous tone files. It is generally referred to as either type 2, contone, or continuous tone data.

A large number of files exist in the type 2 format. Type 29 includes an adaptive compression technique that, in most cases, is more efficient than type 2, and is to be used for current and future raster storage. Applications should be prepared to read this format and in most cases store this data uses the type 29 format.

Type 3: Word

The type 3 format uses each 16 bit word to store the value of the pixel. The interpretation of the sign bit is application dependent.

Type 4: 32Bit Integer

The type 4 format uses each 32 bit integer to store the value of the pixel. The interpretation of the data is application dependent.

Type 5: 32Bit Floating Point

The type 5 format uses each 32 bit floating point values to store the value of each pixel. The interpretation of the data is application dependent.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 6: 64Bit Floating Point

With the type 6 format, every 64bit element is a floating point value representing the value of that information.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 7: Complex

With the type 7 format, each 64bits is a complex number representing the value of that pixel.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 8: Double Precision Complex

With the type 8 format, every 128 bit data element is a double precision complex value representing each pixel value.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 9: Run Length Encoded

The type 9 format compresses bi-level raster data. Each 16 bit unsigned word stores the number of pixels with an identical level.

This format consists of a series of on and off runs. Each line starts with an "OFF" run (background level) followed by an "ON" run (foreground level). This OFF/ON pattern repeats until the end of the line and a new line starts. This continues until the end of the file. Each line ends with an "OFF" run. A zero length "OFF" run is placed at the end of a line if necessary. Zero length runs are valid runs any where in the file. Runs are always less than 65536 pixels in length. It is strongly recommended that applications cut the run at 32767 pixels for consistency with the majority of applications.

Data stored in this format is generally referred to as run length, RLE or type 9 data. The most popular file extension is ".RLE".

A large number of files exist in this format throughout the Intergraph installed customer base. It has been primarily the storage format for bi-level image data. It is not as efficient as CCITT-Group IV compression. This format should be read by all applications, and only written when specifically required. The preferred storage format is type 24 or type 65/24.

Type 10: Color Run Length

This format is used to store gray scale or color images where there are a large number of adjacent pixels of any one color or shade. The format is a run length format, where the color is stored, followed by the number of pixels in the line that are of that color. Two 16 bit unsigned integers are used. The "color" is the index value to use in the color table to determine the actual RGB values for the pixels.

This format is a standard format for storage of scanned data, and for specific map publishing applications. Additional application information about "color" tables is required to interpret this format. The most popular names are type 10 and "CRL" data, and common file extensions are ".CRL," ".TPE" and ".LSR".

This format is not as efficient as type 29 for storage of color palette and continuous tone images, and is currently supported only in specific applications. Applications should be prepared to read this format and write it only in specific applications.

Type 11: Not used (reserved)

Type 12: Not used (reserved)

Type 13: RLE Variable Values with Z (Simple)

The RLE variable values with Z (simple) raster format is useful for storing gray scale shaded data or color data that may be merged later. This format was primarily used by one application area. They should be contacted for details of the exact data format.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 14: RLE Binary Values (with Edge Type)

The RLE binary values (with edge type) raster format is useful for storing gray scale data or any data consisting of only two states (0 or 1). It was used primarily by one application.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 15: RLE Variable Values (with Edge Type)

The RLE variable values (with edge type) raster format is useful for storing gray scale shaded data or color data. Only one application used this format.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 16: RLE Variable Values with Z (with Edge Type)

The RLE variable values with Z (with edge type) raster format is useful for storing gray scale shaded data or color data that may be merged at some later time.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 17: RLE Variable Values with Separate Color and Shade

The RLE variable values with separate color and shade format is useful for the number of shade run lengths. It is assumed that color and shade may be separated on the output device involved.

NOTE: This data type is obsolete and should not be used in general applications. Readers may ignore this data type.

Type 18: RLE Variable Values with Normals

The RLE variable values with normals raster format is useful for storing color, shade, and normal values in run length encoded form.

Type 19: Quad Tree Encoded Variable Values

The quad tree encoded variable values raster format data type is useful for storing raster data intended for display on a random access device such as a terminal, or in some cases provides more economical storage than typical RLE schemes. Type 19 format has limited usefulness for such run length oriented devices as plotters and optical scanners.

A quad tree encoded record is actually a major subdivision record (where the definition of major depends on the particular data case or application) which contains all nodes in the quad tree that belong to that subdivision.

Each pair of bits in the 28 bit subdivision location represents an encoding of one of four possible subdivisions at a particular level in the quad tree. The encoding starts with the high order pair of bits representing the first level in the quad tree (after the root) and ends with the low order pair of bits representing the lowest possible level in the quad tree. Therefore, there can be at most 15 levels including the root (level 0). In addition, if a subdivision appears at level ***n*** in the quad tree, then only the first ***n*** pairs of subdivision location bits are relevant; the other pairs may be ignored. Fifteen levels in the quad tree permit a maximum resolution of 16K by 16K pixels.

Type 24: CCITT Group 4

The CCITT Group 4 format is a highly compressed format that has become a standard for transmission and storage of bi-level facsimile images.

The format uses a modified Huffman code scheme to encode each line in terms of that line's relationship to the previous line. Data in this format will not have the four word raster line header since the Group 4 specification defines the complete data format. Skipping to a particular line of data would be no use in this case because a line of data cannot be decoded without first decoding all lines that precede it.

The compression technique is complicated and beyond the scope of this document. For a complete description refer to the following standards documents:

CCITT Draft Recommendation T.5, General Aspects of Group 4 Facsimile Apparatus

CCITT Draft Recommendation T.6, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus

The tiled version of this format is the recommended method of storing bi-level raster data. It is strongly suggested that this format be used by all new applications.

Type 25: Simple 24bit RGB RLE

The simple 24 bit RGB run length encoded raster format is a format consisting of runs of composite RGB values. The data in a raster line record consists of a series of four byte fields, each consisting of an 8 bit red, green, and blue intensity and an 8 bit run length. For a more compact RGB format, refer to the information on Type 27: Compressed RGB.

Type 26: Variable Run length Indexed Color

The variable run length raster format is defined to be a more flexible and compressed color run length format. The format was designed so that different applications can use a different number of bits to represent the data. Most applications have been using the type 10 format with the 16 bit value simply representing a color. This format is more compressed for images that are not 16 bits per pixel.

This data type is parameterized. The Data Type Modifier (DTM) raster header field is a parameter that has valid values 0 to 15. This represents the number of bits in the run length. Refer to the section on header Block One for more information on the Data Type Modifier (DTM) header field.

Type 27: Compressed RGB

Type 27 format is a compressed RGB format for 24-bit color data. A file of this format can contain both run length encoded data and unencoded data for separate red, green, and blue components.

The RGB data is band interleaved by line. Each line of RGB data is stored as a line of red data, followed by a line of green data and a line of blue

data. The end of line is reached when the appropriate number of pixels have been represented.

Each line of raster data is composed of one or more atoms. An atom is a string of bytes consisting of an atom head and an atom tail. The first byte, or the atom head, is a signed value that determines the size and format of the remaining byte(s), the atom tail. If the atom is positive (1 to 127), then the atom tail contains that number of bytes. These bytes are to be interpreted as a string of individual intensity values. If the atom head is negative (-1 to -128), then it signifies a constant shade run length. In this case, the absolute value of the atom head is the length of the run, and the atom tail is a single byte that specifies the intensity of the run. Atom head values of zero are ignored, and the atom is skipped.

The tiled version of this format is the recommended method of storing 24 bit data color raster data. It is strongly suggested that this format be used by all new applications.

Type 28: Uncompressed RGB

Type 28 is uncompressed RGB raster data. The main use of this data type is the storage of unprocessed color scan data. This file can be smaller in size if there are many runs of pixels of different value. Each line consists of triplets of red, green and blue intensities, one byte for each intensity.

> For some short length of time, files may have been created that assumed each line was an integral number of words long. Should the line be an odd number of bytes, the line was padded to the next word boundary with zero.

Type 29: Compressed 8 Bit

The type 29 format is a compressed 8 bit data format that combines a run length encoding technique with straight byte per pixel format. This format is the 8 bit counterpart to the 24 bit compressed RGB format (type 27).

Each line of raster data is composed of one or more atoms. An atom is a string of bytes consisting of an atom head and an atom tail. The first byte, or the atom head, is a signed value that determines the size and format of the remaining byte(s), the atom tail. If the atom is positive (1 to 127), then the atom tail contains that number of bytes. These bytes are to be interpreted as a string of individual intensity values. If the atom head is negative (-1 to -128), then it signifies a constant intensity run length. In

this case, the absolute value of the atom head is the length of the run, and the atom tail is a single byte that specifies the intensity of the run. Atom heads of zero are ignored and the atom is skipped.

The tiled version of this format is the recommended method of storing 8 bit raster data. It is strongly suggested that this format be used by all new applications.

Type 30, 31 and 32: JPEG

JPEG is a compression technique defined by the "Joint Photographic Experts Group," that is used primarily for full color, pictorial images. It uses a block by block conversion to frequency space, and stores a discrete cosine series representation of the frequency space. Compression is achieved by defining the number of terms to retain in the cosine series. The image is converted (if necessary) into HSI color space, and then compressed. The Hue and Saturation images can be greatly compressed without loosing pictorial quality. This is the only "lossy" compression technique Intergraph supports.

Type 30 is JPEG compressed greyscale imagery with eight bits per pixel in the original image. By default, no color table association exists for this type of data. Type 31 is JPEG compressed RGB data, 24 bit color raster images, and type 32 is JPEG compressed CYMK, four band imagery.

As the JPEG formats contain a private application packet that contains critical information necessary to decode the JPEG data, contact the Intergraph Raster Review Board Chairman at the address in the beginning of this document for information about this application packet.

Tiled JPEG files must have "full" tiles for proper compression and decompression without image anomalies. Tiles should be padded with the last pixel value in each line to give the least distortion.

Type 65: Tiled raster data

Should a type "65" occur in the Data Type Code (DTC) in the raster header, it indicates that the data contained in the raster part of the file is organized into logical "tiles". The tiled image is preceded by a tile application data packet, including the specification of the actual raster data type in the file. The tile raster format is defined in the appendix.

Type 66: Not Used (reserved)

Type 67: Continuous Tone CMYK

Type 67 files are used to store continuous tone CMYK images. These images are representative of images used in the lithographic printing process. Each color ink used in the printing process is represented by an intensity in the image file. The three subtractive primaries of Cyan, Magenta and Yellow, and a fourth printing ink in Black are used in this image format. Each pixel in the file is represented as a set of eight bit values corresponding to these four colors.

In the Type 67 file, an eight bit value of zero represents 100% ink coverage, and a value of 255 represents 0% ink. Even though this is a continuous tone format, it includes a facility for run length encoding to allow large contiguous areas of solid color to be represented compactly.

Each row is made up of a line of cyan atoms, followed by a line of magenta atoms, followed by a line of yellow, and finally black. Atoms are composed of a string of bytes which represent a run length. The first byte of the atom is the atom head. The atom head determines the length of the atom tail.

If the atom head positive, then that number of bytes follow the atom head and the bytes are individual pixel values.

A negative atom head indicates that the next bytes represent a string of pixels that are all the same intensity. The absolute value of the head is the length of the run, The single unsigned byte value immediately following the atom head is the value of the run.

When the number of pixels per row are represented for each color, the current line is terminated. Note that an atom head with a zero value is meaningless.

Type 68: Linework CYMK and RGB

Type 68 files are used to store CYMK and RGB "linework" files. These files represent images that are generally not continuous tone, but are "spot" color that contain large areas of a single color intensity. The type 68 file is similar to the Type 67 in general structure, but includes the option to have very long runs stored as single atoms.

Type 68 files use additional data elements in the header.

TYPE 68 HEADER ELEMENT VALUES

File Byte Offset	Value
212	1
213	1
508	8
509	0 (Note)

Note: byte offset 509 may not be zero (0) in many current files, but should be set to zero in new files.

Each row of raster data in a type 68 file is made up of a line of atoms. Atoms are composed of a string of bytes which represent a run length. The first byte of an atom is the atom head. The atom head determines the length of the atom tail.

If the atom head is positive, then that number of bytes follow the atom head and the bytes are to be interpreted as a string of individual pixel values. A negative atom head signifies a constant value run length. The absolute value of the atom head is the length of the run.

The single byte following the atom head is the value of the run. The next two bytes (a 16 bit unsigned word) contain the length of the run. When the number of pixels per row are represented, the current line is terminated. There is no run length encoding across row boundaries.

Color Table

The color table for the type 68 file is contained in a standard application packet. The application packets are fully described in the appendix of this document.

The color definition for each of the 256 possible pixel values is an array of structures (RSvc_entry) The array starts at the file offset indicated in the last field (colors) in the application data packet.

RSVC_ENTRY STRUCTURE

```
struct RSvc_entry
{
        uint8       type;
        uint8       flags;
       uint16       rgb[3];
       uint16       c[4];
        uint8       spot[12];
        int32       trans_flag;
        uint8       spot_index;
        uint8       res2;
       uint16       res3;
       uint32       res4;
        uint8       name[RSVC_MAX_NAME];
}
```

The field "type" indicates the type of the color which may be 0 or 1. A value of 0 indicates that the color is an RGB color table and that the values are stored in the rgb[3] location in the entry. The color red value is stored in rgb[0], green in rgb[1] and blue in rgb[2].

A value of 1 in the "type" field indicates that the color is a CYMK color stored in the c[4] portion of the structure. The order of the unsigned 16 bit integer is cyan, magenta, yellow and black in c[0] through c[3], respectively. The 8 bit color value for both RGB and CYMK are stored in the most significant byte of the unsigned 16 bit integer.

A value of 2 in the "type" field indicates that the color is a spot color stored in the spot[spot_index] portion of the structure. A value of 3 in the "type" field indicates that the color is a combo color. A combo color can exist on all 16 films. The color values are stored in the c[4] and spot[12] portions of the structure.

The "trans_flag" is used to indicate that the separation is transparent. Each bit starting with the least significant bit is one (1) if a particular separation is transparent. The bits are assigned cyan, magenta, yellow, black, spot[1], spot[2], ... spot[11] from the low order bit. Un-used bits must be zero, and are currently reserved.

The remaining fields of "res2," "res3," and "res4" are reserved at this point, and must be set to zero.

A color may have a name contained in the "name" field. The name is an ASCII null terminated string. If there is no name for the color, the first uint8 of the field contains the ASCII null character. The constant RSVC_MAX_NAME is defined to be 32.

The virtual color table packet includes the following header immediately following the application data packet header.

RSPTCLSUB14_DATA FORMAT

```
struct RSptclsub14_data
{
        uint16          version;
        uint16          num_colors;
        uint64          res1;
        uint64          colors;
}
```

The version field contains the value of 2 for the current implementation, the num_colors are the number of colors in the color table (maximum of 256). The res1 field is reserved and must be set to zero. The colors field is the byte offset for the first color in the color table, consisting of num_colors entries of RSvc_entry structures contiguously located in the file.

The application packet for the color table has Application Code = 1, and Sub-Type Code = 14. The Words to Follow field in the application packet should include all the RSvc_entry and RSptclsub14_data structures. See **Appendix B** for more information on the application packets.

Type 69: Continuous Tone CYMK (Non-Compressed)

Type 69 files are use to store non-compressed continuous tone CYMK images. The pixels are stored as 32-bit per pixel values. The first byte is cyan, the second magenta, the third yellow, and the fourth is black. An eight bit value of zero represents 100% ink and a value of 255 represents 0% ink.

APPENDICES

APPENDIX A: GLOSSARY:

atom: A group of bytes used in the adaptive compression technique. It consists of a "head" that indicates either the run length, or number of uncompressed pixels that follow in the "tail."

block: A section of the file containing 512 bytes. [The VAX allocated disk space in 512 byte sections called "blocks."]

byte: a data element consisting of 8 bits

horizontal: the "left to right" direction in an image, as the viewer is looking at the image oriented for natural viewing.

line: A group of contiguous pixels in a single direction. The image is composed of a regular set of lines, each of which is composed of a regular set of pixels.

longword: A data element consisting of 32 bits. May either be signed or un-signed. [uint32 or int32]

pixel: the smallest element in a digital image. The image is comprised of a regular set of lines, each composed of a regular set of pixels.

quadword: A data element consisting of 64 bits. May either be signed or un-signed. [uint64 or int64]

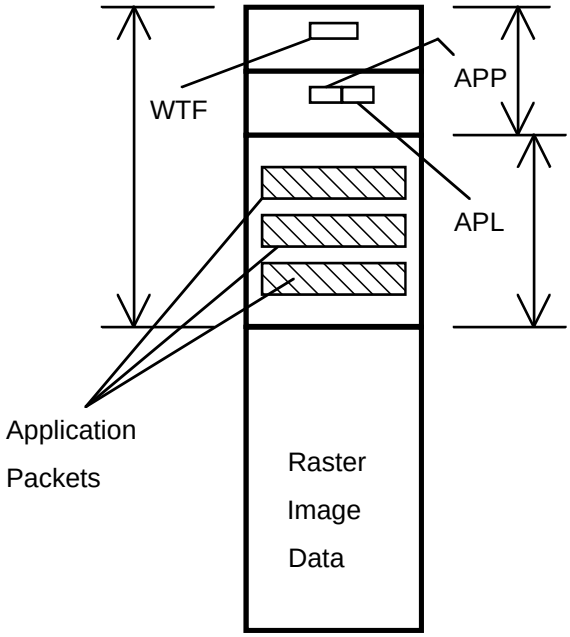
quadword alignment: The byte boundary that corresponds to an integral number of quadwords from the beginning of the file (or data structure).

word: a data element consisting of 16 bits. May be either signed or un-signed. [uint16 or int16]

APPENDIX B: APPLICATION PACKET FORMAT

Application packets are the preferred method of storing application specific information. Each packet contains a header followed by application defined data. See the section on application packets for more background information.

This is a graphic representation of the raster file including the header, and application packets:

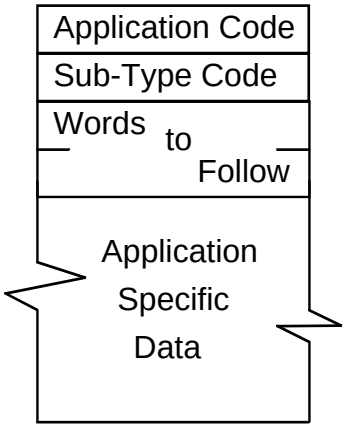


Raster File with Application Packets

The packets are located in the last part of the header. As long as each application uses the words to follow (WTF) as the length of the raster file header, application packets can be ignored. If an application is cognizant of application packets, the application packet pointer (APP) and the application packet length (APL) fields indicate the location and length of each application packet.

Application packets consist of a short, 8 byte "tag" that identifies the packet type code, sub-type code, and the remaining length of the application packet. The Intergraph Raster Review Board Chairman is

responsible for maintaining a list of application packet type codes, and allocating new codes to application developers. Each application is responsible for allocating a sub-type code for each of their individual packet types.



Application Data Packet

By reading the application type, software can determine if it understands the format of the application specific data to follow, and whether it needs to interpret the information contained in the packet. If not, the application uses the last field in this structure to skip the "foreign" information, and searches for application packets that are applicable. Should an application produce a "derived" file from this raster file, all application packets are copied to the new file, and any new application data packets appended to the existing packets, all within the raster file header.

APPLICATION PACKET DATA TYPE CODES

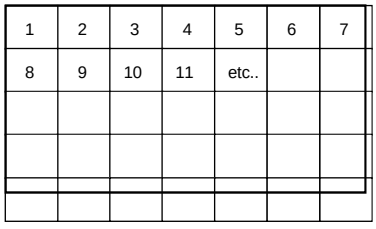
Code	Application
1	IGE (Intergraph Graphics Environment)
2	Image Processing Applications
3	DMANDS
4	GRASS
5	MapPublishing Applications
6	Mapping Applications
7	DTM (Digital Terrain Modeling Applications)
8	Scanning

Other application packets are non-standard and should be ignored by all applications. For the latest list of application packets, or to add your applications to the list, please contact the Intergraph Raster Review Board Chairman.

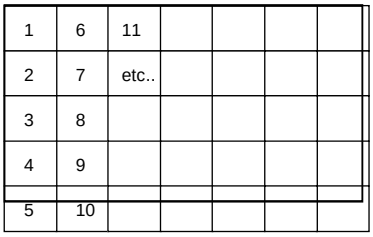
APPENDIX C: TILED FILE FORMAT

When raster data becomes very large, it may be necessary to subdivide the raster data into smaller sections or "tiles" that can be loaded into memory and operated on individually. Raster data type code (DTC) 65 is used to specify the data format of a tiled raster file. The tiled raster file consists of the standard raster file header with additional data specifying the tile information stored immediately after the standard header. The application data packet format is used to store the tile information as specified below. The actual tiled raster data is contained immediately following this tile directory.

The tiles are stored in the file immediately after the tile directory. Tiles are "numbered" or logically thought of as appearing in the image in the same order as the pixels. For example, if the data is upper-left-horizontal, the first tile is in the upper left corner, the second immediately to its right, and so on. Tiles in new images are allocated in this manner, however, no assumptions should be made about the physical order of the tiles in the file. Tiles may be stored in any order. After editing, for example, a tile that will no longer fit in the originally allocated area, can be placed at the end of the image, and only the tile directory updated.



Horizontal Upper Left Orientation



Vertical Lower Left Orientation

TILE DIRECTORY FORMAT

Data Type	Number of Elements	Description	Comments
uint16	1	Application Type	Must be 1
uint16	1	Sub-Type Code	Must be 7
uint32	1	Words to follow	Multiple of 4
uint16	1	Packet Version	Must be 1
uint16	1	Identifier	Must be 1
uint16	2	Reserved	Must be 0
uint16	1	Properties	bit 0 = 1 if tiles not in order bit 1 = 1 if overview packet present
uint16	1	Data Type Code	Raster data compression (DTC)
uint8	100	Reserved	Must be 0
uint32	1	Tile size	Square Tiles
uint32	1	Reserved	Must be 0
uint32	1	S_0	Starting Byte Offset, Tile 0, relative to the end of the raster file header.
uint32	1	A_0	Allocated Length, Tile 0
uint32	1	U_0	Used Length, Tile 0
		etc	
uint32	1	Sn	Starting Offset, Tile n
uint32	1	An	Allocated Length, Tile n
uint32	1	Un	Used Length, Tile n
uint8	AS REQUIRED	Padding	Pad to quadword boundary

NOTE:
offset, allocated, and used lengths are in bytes
offset is relative to the end of the raster file header

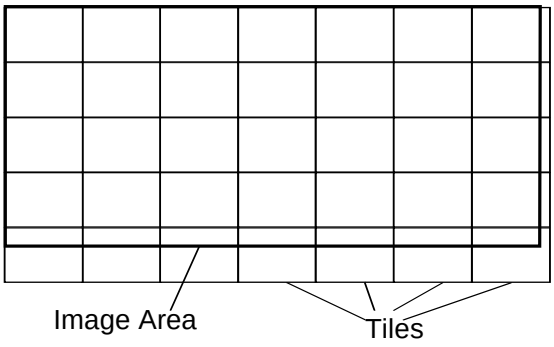
Each tile is a contiguous area of the image with the location and size characterized by the start location, allocated length for the tile, and used length of the tile. The tile information contains a "directory" of starting position, and the allocated and used length for each tile. The tile "directory" is really an application packet, (See **Appendix B** for a complete discussion of application packets) located directly **after** the header in the raster file; preceding the overview image data in the case of a tiled overview. **The tile directory is the one exception to the rule that all application packets reside in the header.**

Since the tile directory is a non-conforming application packet, the APP (Application Packet Pointer) entry in Block One of the header should not point to the tile directory, but should point to the "real" application packets in the header. For a type 65 file, the tile directory is at the end of the header, and can be found by using the WTF (Words To Follow) entry in Block One.(See the appendix on Application Packets for more information about the Application Packet Pointer).

The offset in the tile directory is the number of bytes between the end of the file header and the beginning of the tile. It is specifically **not** relative to the tile directory, but to the end of the header as defined in the WTF entry in the first block.

The allocated length of the tile is that area in the file that can be used by the data representing this particular tile. Should a raster editor modify a tile, and it will still fit within the allocated space, the data should be written in that location. Should the edited data be too long to fit in the allocated area, it should be written at the end of the tile area, and the directory updated appropriately.

The recommended tile size is 512 X 512 pixels for bi-level images and 128X128 for other formats. This appears to be an "optimum" size for the current applications and hardware. If these sizes are not possible, the tiles must be divisible by 32. Some applications use a set of subroutines called the "RCE" library, which requires that tiles are divisible by 32.



Tiles along the "right" and "bottom" edges of the original raster are not "padded" to provide full tiles, unless required by specific applications The JPEG compression technique requires all tiles must be "full," and the recommended technique of completing the edge tiles is to pad the tile with the last pixel value. The number of pixels and/or number of lines of these tiles may be less than the rest of the tiles in the image. Applications should

always use the number of pixels and lines from the tile directory to determine the size of an individual tile.

If the "scannable" flag is set in the header, all the raster in the file, including all the tiles must contain scanline headers. Each tile is considered a separate file and the pixel line numbers should start with "1" in the pixel line number field. Tiled files are "scannable" only in exceptional circumstances.

There are some application alignment restrictions on the tiles and on the application packet tile directory: for CCITT Group 3 or Group 4 files, the lines in the tiles are padded to a "longword boundary" (an even multiple of 32 bits). And the last line in the file is also padded to a longword boundary.

If the tile is all one color or shade, it is not necessary to store the entire tile. All that is stored is the color of the tile in the data packet "directory." For these "un-instantiated" tiles the offset, allocated and used structure becomes:

UNINSTANTIATED TILES

Data Type	Number of Elements	Description	Notes
uint32	1	Sn	Sn = 0 indicates un-instantiated tile
uint32	1	An	Reserved, Must be 0
uint32	1	Un	Color Word

When $S_n = 0$, the color (uint32) indicates the color of the tile. For color values of 8 bits (or less) per pixel, the least significant byte is the color. Full color data uses the lower three bytes to store the red, green and blue intensities. The format is:

For 8 bit (or less) pixels, the color uint32:

0000	0000	0000	0000	0000	0000	CCCC	CCCC

Where "CCCC" is the color shade information.. And For 24 bit color data the color uint32:

0000	0000	RRRR	RRRR	GGGG	GGGG	BBBB	BBBB

Un-instantiated tiles are strongly recommended to all applications that create tiled data, and should be handled by all products that read tiled data.

APPENDIX D: OVERVIEW PACKETS AND OVERVIEWS

Overviews are small "postage stamp" raster images that represent the large image contained within the raster file. They are used by applications to improve the apparent speed in displaying a raster image, or used as "static" displays for operator reference and selection. Within the Intergraph raster file, multiple overviews are permitted, allowing applications to create and use "multiple resolution data sets" that provide pre-decimated representations of the raster image for rapid retrieval and display.

The overview is located at the end of the raster file to allow applications that do not know about overviews the ability to display the full resolution image, and completely ignore the overview. This provides backward compatibility with earlier raster applications.

The overview application packet contains the file offsets for each overview in the image, the number of lines and pixels in each image, and the sampling method used to reduce the image resolution for each overview. The actual length of each overview, and the allocated length are also indicated. Applications can allocate more length than required for each of the overviews, leaving room for larger images if the file is edited.

TILED OVERVIEWS

Overviews can also be tiled. Some reduced resolution images are still very large, and the application can achieve significant performance improvements by including tiles in the overviews. Tiling is described in detail in **Appendix C**.

The overview application packet contains a flag that indicates if the overview is tiled. Should the overview be tiled, the allocated and used length fields in the overview application packet include the tile packet.

Raster Data Compression in Tiled Overviews

If the image includes tiles and/or if the overview is tiled, the compression technique (raster data type) is included in a different location in the image file.

Untiled Image with Untiled Overviews

The algorithm used to compress the main image in a file is the same technique used to compress the overviews. For untiled images and untiled overviews, the "Data Type Code (DTC)" in the main image header indicates the compression technique used.

Untiled Image with Tiled Overviews

The tile packet (described in detail in the tile appendix) also has a "Data Type Code (DTC)." For tiled overviews, the DTC values in the main image header and in the tiled overviews must be the same. Compression of the tiled overview with a different algorithm is not permitted.

Tiled Image with Tiled Overviews

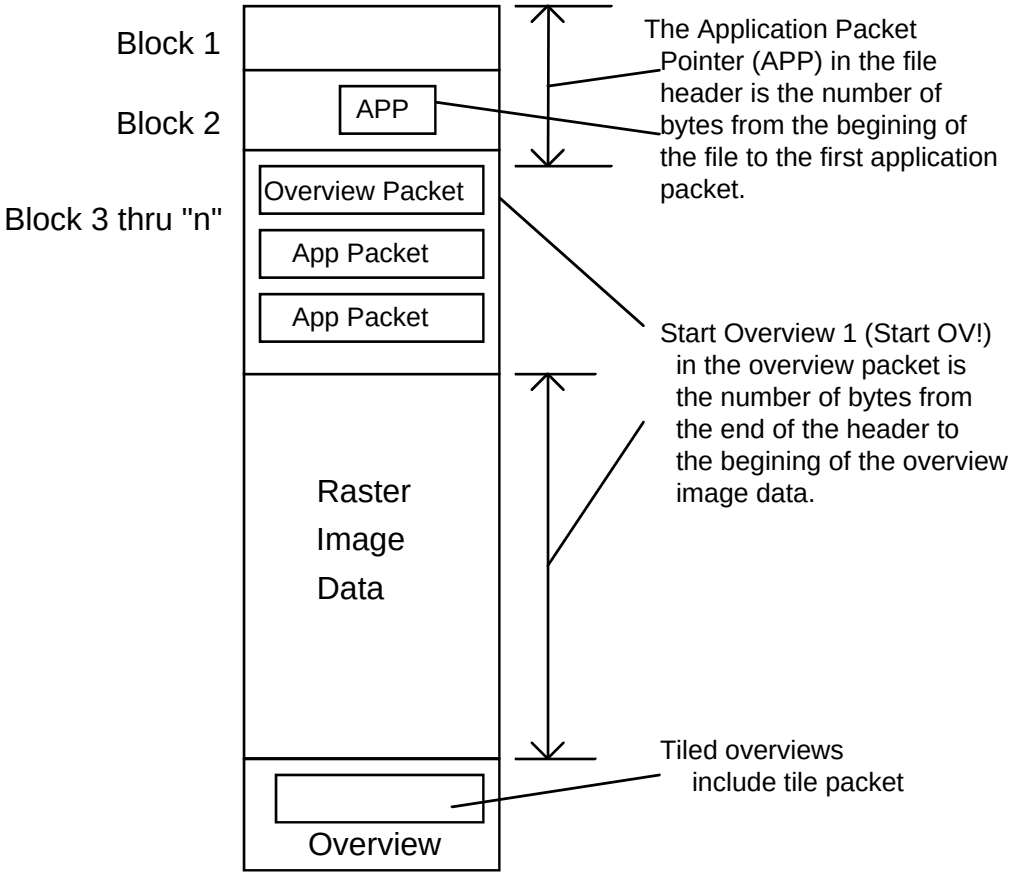
If an image is tiled, the Data Type Code in the main image contains the decimal value of "65." The tile packet contains another "Data Type Code" that indicates the actual compression of the data in the tiles. The DTC in the tile packet for the main image is used to indicate the compression algorithm for the raster data in the overview and for the main image.

Tiled Image with Untiled Overviews

In the case of tiled images with untiled overviews, the DTC in the main image tile packet is used to indicate the data compression used.

Overview Location and Contents

The overview is defined in an application packet in the header of the raster file. The structure of this header is described below. The typical file structure with an overview is as follows:



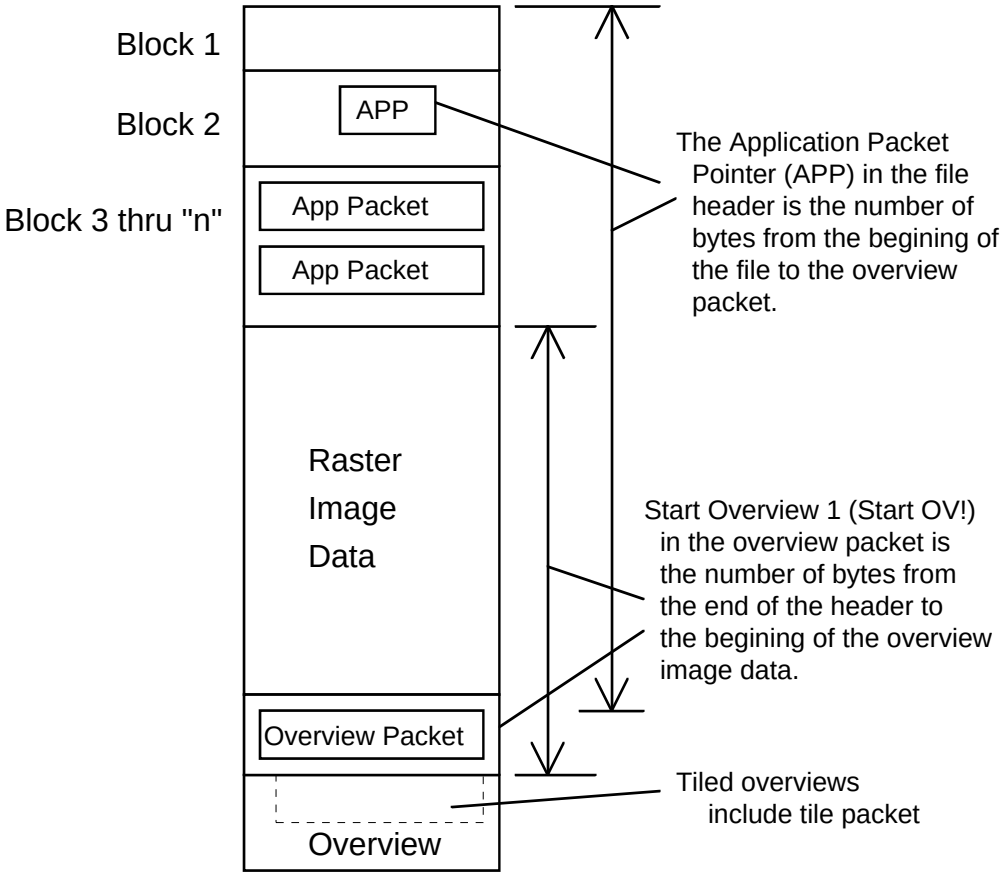
Raster File with Overview

OVERVIEW PACKET CONTENTS

Data Type	Number of Elements	Description	Note
uint16	1	Application Type	Must be 1
uint16	1	Sub-Type Code	Must be 10
uint32	1	Words to follow	Multiple of 4, words in packet, not the data
uint16	1	Packet Version	Must be 1
uint16	1	Identifier	Must be 1
uint32	1	Reserved	Must be 0
uint32	1	Number of Overviews	
uint32	1	Number Lines, OV1	Lines in first overview
uint32	1	Number Pixels, OV1	Pixels in first overview
uint16	1	Sampling Method, OV1	0 = Subsample 1 = Logical "OR" 2 = Averaging
uint16	1	Flags	bit0 1 = tiled overview
uint32	1	Reserved	Must be 0
		. . .	
uint32	1	Number Lines, OVn	Lines in n^{th} overview
uint32	1	Number Pixels, OVn	Pixels in n^{th} overview
uint16	1	Sampling Method, OVn	0 = Subsample 1 = Logical "OR" 2 = Averaging
uint16	1	Flags	bit0 1 = tiled overview
uint32	1	Reserved	Must be 0
uint32	1	Start, OV1	Starting byte offset for first overview
uint32	1	Allocated, OV1	Allocated length of the first overview
uint32	1	Used, OV1	Used length of the first overview
		¤¤¤	
uint32	1	Start, OVn	Starting byte offset for the n^{th} overview
uint32	1	Allocated, OVn	Allocated length of the n^{th} overview
uint32	1	Used, OVn	Used length of the n^{th} overview
uint8	AS REQUIRED	Padding	End of application data packet must be on 32 bit boundary

NOTE:
All the lengths and offsets indicated in the format are in bytes.
Offsets are relative to the end of the file header, not the application packet.
Length of overviews include any application packet.

There is an alternative method of placing overview packet information in the raster image file. **<u>This technique is not recommended,</u>** but applications should be prepared to accept data in this format. A graphic representation of this positioning follows:



<u>Alternate Overview Location</u>

In the alternate scheme, other application packets are assumed to start at the beginning of the third block. These files do not contain color table

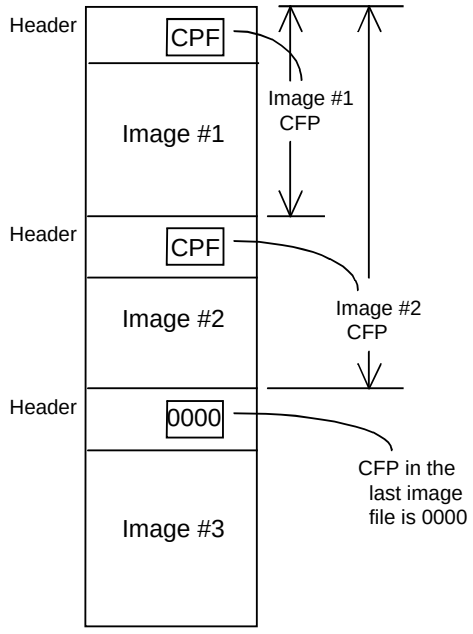
information that extends beyond the second block. The APP field in the header is used to point to the overview application packet, and not to other application packets.

APPENDIX E: MULTI-IMAGE FILES

Multi-image files are simply the catenation of standard image files. These files can be any compliant image format, mixed types, with or without application packets, etc..

The second block of the raster file header contains an unsigned longword called "Catenated File Pointer (CFP)." If this value is non-zero, it is the byte offset of the next image header from the beginning of the multi-image file.

The last image in a multi-image file is indicated by placing a zero in the CFP field. Note that this means that a single image file is really a multi-image file with only one image.



Multi Image File

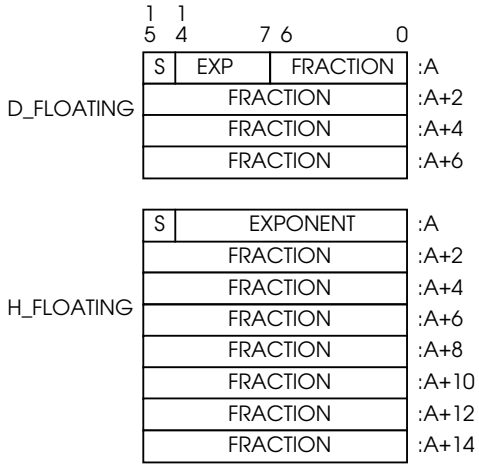
There is no restriction on placing images in the file in sequential order. Indeed images can be inserted in the logical document by adding the new image to the end of the file and changing pointers to the next image in the sequence.

Images can be deleted by eliminating all pointers to those images that are no longer required. However, for optimal access, applications should include utilities to re-sort and pack ("garbage-collect") the multi-image file.

Applications can use the catenated file pointer to build an internal linked list of the location of each image in the file. The application can then seek directly to the required image.

APPENDIX F: VAX FLOATING POINT FORMAT

Version 1 and some Version 2 files will have floating point data in the VAX D_FLOATING or in the H_FLOATING formats. These formats were called "REAL*4" and "REAL*8," in VAX FORTRAN. These formats are depicted in the associated diagram.



VAX Floating Point Format

APPENDIX G: RASTER ELEMENT BY REFERENCE

Introduction

A number of IGDS-based applications have a need to display and plot both vector and raster data. The current IGDS type 87 and 88 elements do not include a number of raster formats used today (for example 24 bit color). Furthermore, they are inefficient in that they require an IGDS header for each element, and each element can only be three blocks long.

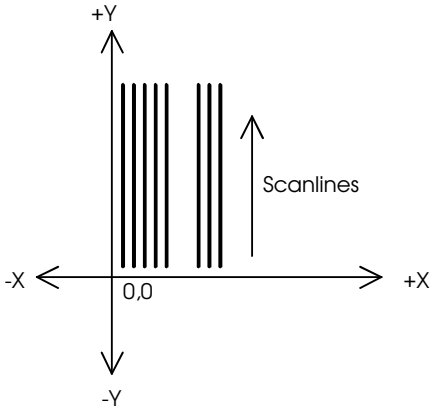
This appendix is the definition of a new IGDS element, Raster Element by Reference, that allows raster data in an external file to be referenced from within an IGDS vector design file.

The external raster file is in the Intergraph Raster File Format described elsewhere in this document.

RASTER TYPES SUPPORTED IN RASTER ELEMENT BY REFERENCE

Raster Data Type	Description	Reference "Class"
1	bit-packed bi-level	bi-level
2	byte per pixel	contone
3	word (16 bit)	contone
9	run length-encoded bi-level	bi-level
10	run length-encoded byte	contone
25	run length-encoded RGB	RGB
26	run length-encoded variable contone	contone
27	compressed RGB	RGB
65 (sub-type 24)	Tiled Group-4	bi-level

The scanline orientation in the raster data file is ignored and the raster data is assumed to be bottom left origin with vertical scanlines in reference to the IGDS coordinate system.



Raster Scanline Orientation for Plotting

Reference Element Definition

A raster element by reference will be a set of IGDS elements of type 90. The first element of this set will have a component type of zero (0). It will have the complex bit clear in its header.

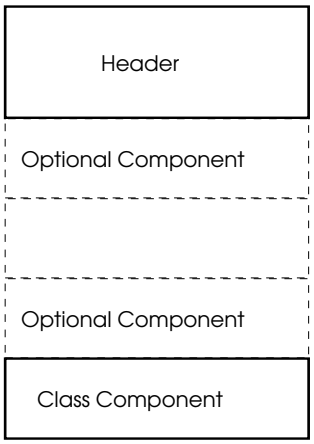
The header element can be followed by optional components, if any. The only optional components defined are the clip polygon component and the color table component. These components are defined completely in this appendix.

Ending the complex element, there will be a required class component. The class component contains information specific of the raster data. For

example a transparent color for RGB data or foreground and background colors for bi-level class raster.

The raster data type and the class component must agree or the rendering of the raster data will be unpredictable.

The overall structure of the raster element by reference is:



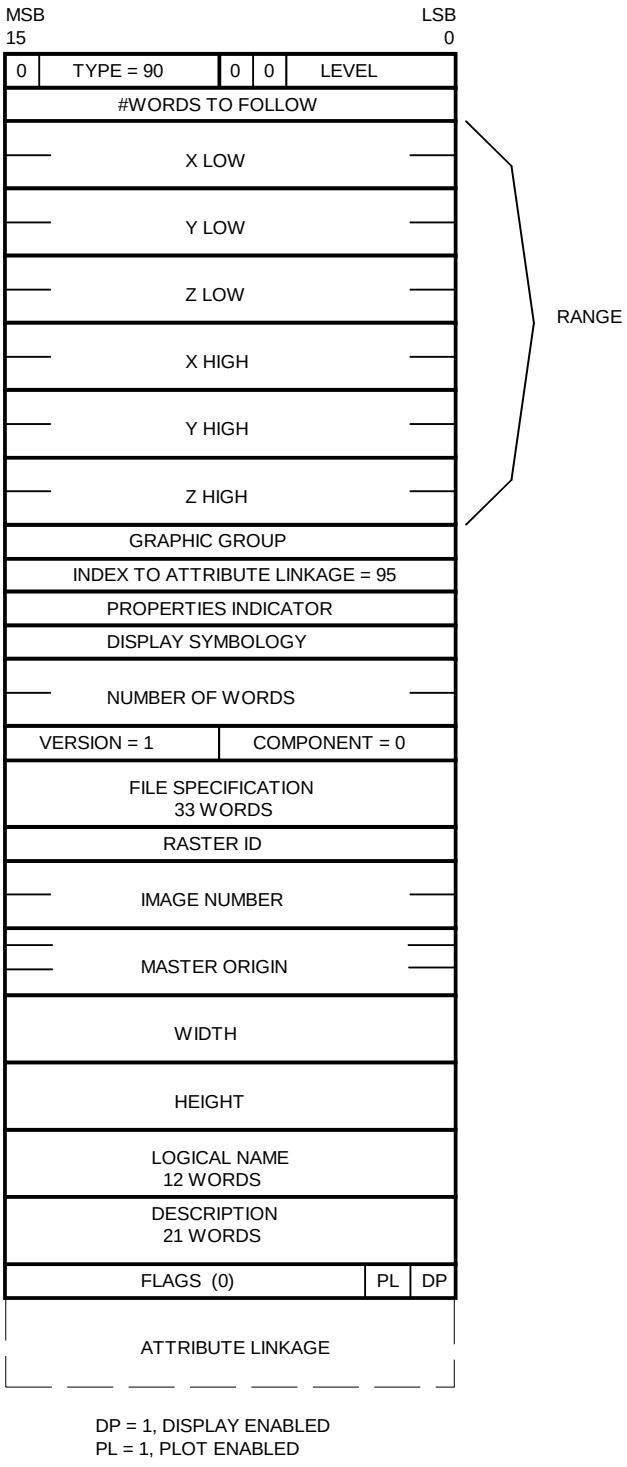
Raster Element by Reference

Raster Element by Reference Header Format

The Header is an IGDS header; it contains type, level, range, index to attribute linkage, properties, and symbology information. For more information about IGDS header elements, consult the current MicroStation documentation.

Since a Raster Element by Reference is plot and view rotation independent, the range of the header element is the cube containing the sphere formed by rotating the raster about its origin in all possible directions. The range of the components are the same as the range of the header element.

The symbology (color, weight, and line style) is ignored.



Element Header

The number of words is an unsigned integer defining the 16 bit word length of the complex element description. It is the sum of the lengths of all component elements plus the number of words existing in the complex header element after this word.

The version is an unsigned byte number, the first version being 0 (zero), with the current version being 1 (one). As improvements will be made in the future, the element format is likely to change. The version number is the mechanism to let the software know how to interpret the information. All care will be taken to make any future changes backward compatible, but this is not guaranteed.

DATA ITEM DEFINITIONS

Item Type	Description	Comments
uint32	Number of words	Length of the data following this field
uint8	Component Type	0, for the main Type 90 header
uint8	Version Number	Must be one (1)
uint8 * 66	File Specification [66]	File specification, ASCII Null terminated string
uint16	Raster ID	Raster Identification
uint32	Image Number	For multi-image files
real64 * 3	Master Origin [3]	Position of first pixel (X, Y, Z)
real64	Width	Width of the raster in UOR's
real64	Height	Height of the raster in UOR's
uint8 * 24	Logical Name [24]	Application logical name
uint8 * 42	Description [42]	Application description
uint16	Flags	Plot and Display enable

The file specification contains the raster file specification as a null-terminated string. The raster file can be specified in three distinct methods: logical name, environment variable, or complete path name.

For example, suppose that two raster files are named /usr2/raster/mary.rgb and /usr2/raster/marlene.rgb. Using the logical name method, the file specification could consist of "RASTER_FILES:marlene.rgb", where RASTER_FILES is an environment variable containing "/usr2/raster". As an environment variable, the file specification could contain just "WIFES_FILE", where the environment variable "WIFES_FILE" would contain "/usr2/raster/mary.rgb", or "/usr2/raster/marlene.rgb". Or, the file specification could contain just "/usr2/raster/mary.rgb".

The raster ID is a number that provides a shorthand method of identifying a raster element. It is used primarily by display applications for quickly referring to the raster.

In multi-image files, the Image Number refers to the file sequence number of the image to be displayed, or plotted. Refer to multi-image files elsewhere in this document for more information.

The master origin is the point in the design file that matches the first pixel in the first line in the raster file. Width and height are the design space UORs in the raster/view plane for the raster file dimensions.

Logical name and description are both null terminated strings for user information.

The flags word contains settings for different options available in the processing of the raster element. The least significant bit (bit 0) is the display flag. If this bit is set, the raster will be displayed. The second least significant bit is the plotting bit. Plotting will plot the referenced raster only if this bit is set.

Raster Element by Reference Components

Raster class components are used to identify specific information required to display or plot the raster information in combination with the associated vector data. For continuous tone and full color images, a transparent color is identified, and for bi-level images, the color number is specified for both the background and foreground. Optional components are also defined to provide a color table and a raster clip polygon.

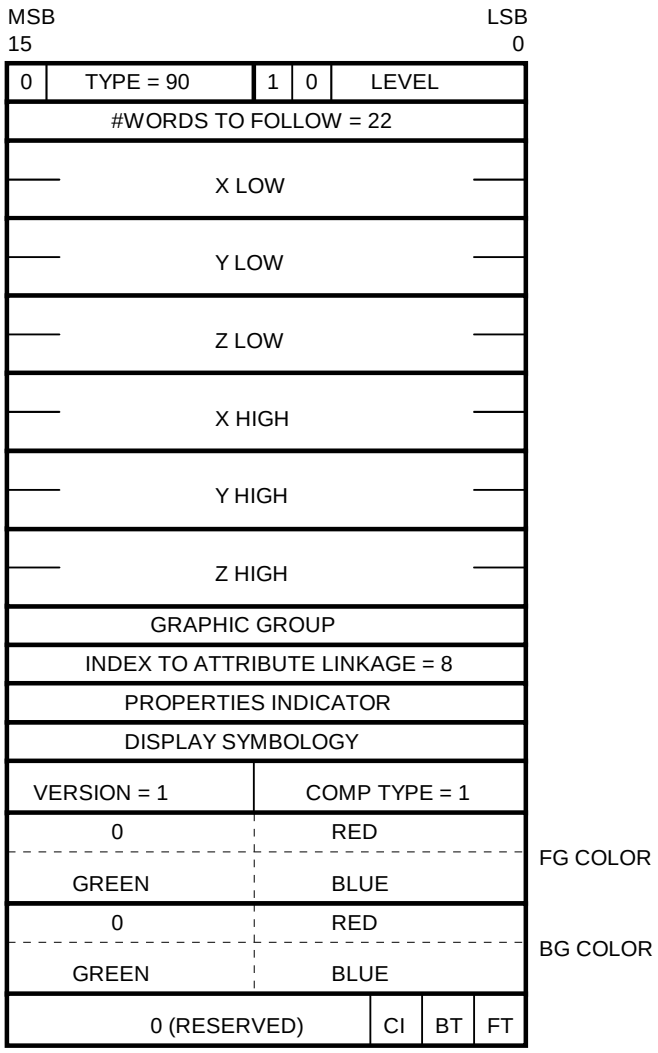
The class component includes an IGDS element header, a version number and a component number that identifies the component type. The following components have been defined:

COMPONENT TYPES

Component Type	Component Description
0	Header Component
1	Bi-Level Component
2	Contone Component
3	RGB Class Component
4	Clip Polygon Optional Component
5	Color Table Optional Component
6-255	Reserved

Binary Raster Class Component

The binary, or bi-level class component includes the RGB color values for both the foreground and the background pixels in the image. A bit in the flags item indicates that the RGB colors in the component are either direct color intensities, or indices into the color table, where the exact RGB components can be found. In addition,. two bits in the flag item indicate the transparency of the foreground and background pixels.



FT = 1, If FG Color is transparent

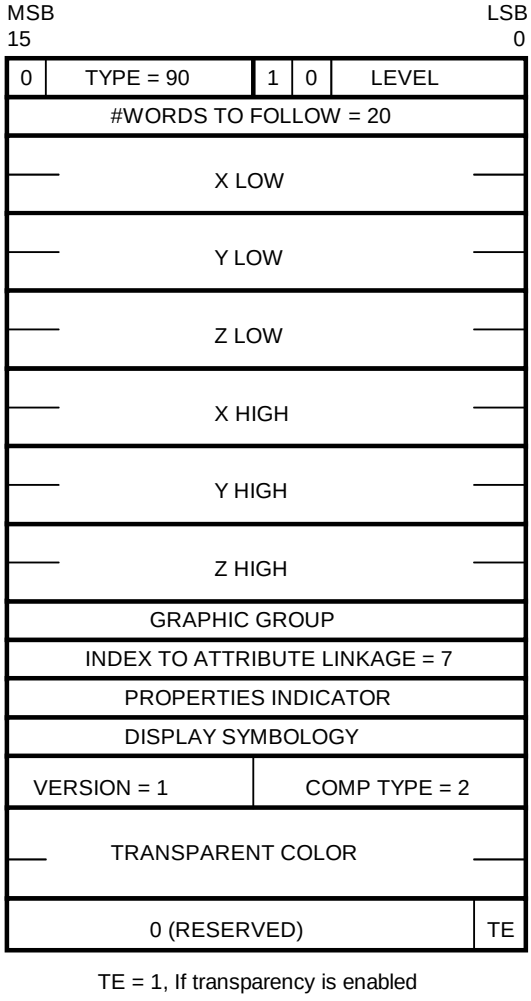
BT = 1, If BG Color is transparent

CI = 1, If FG and BG Colors are indicies

Bi-Level Component Structure

Continuous Tone Raster Class Component

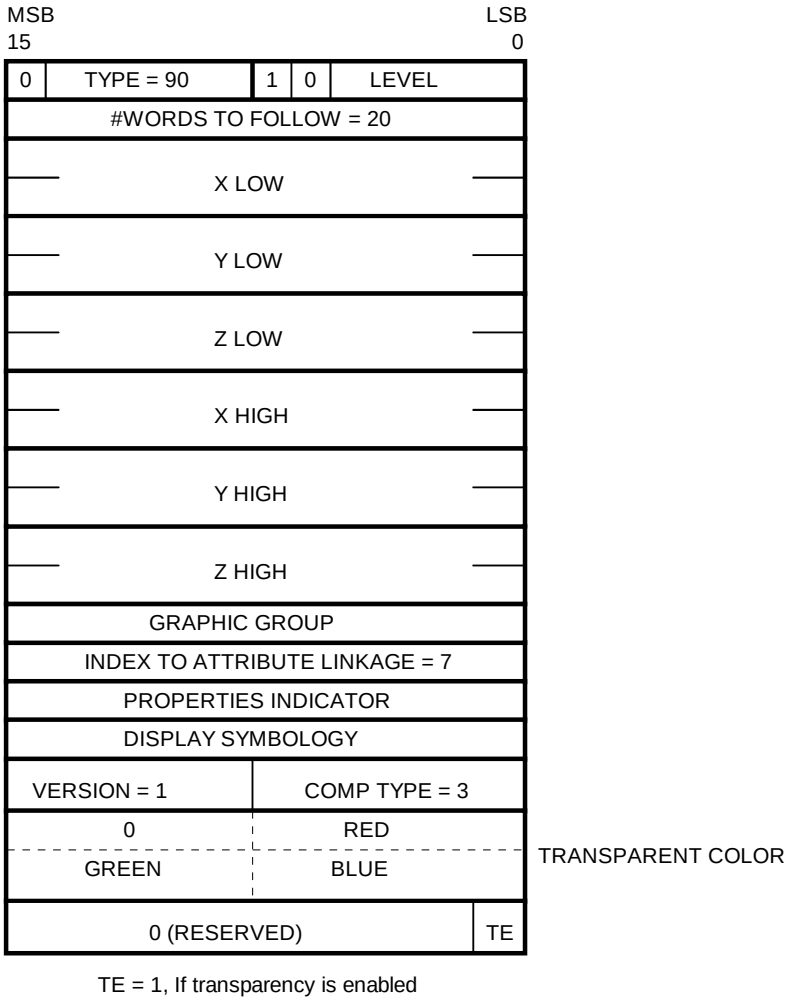
For continuous tone raster, a single digital value is used to identify a single transparent color. The least significant bit in the flags item is used to indicate that the transparency is enabled. The remaining bits in the flag item are reserved and must be zero.



Continuous Tone Component Structure

RGB Raster Class Component

A specific RGB triplet can be identified if a transparent color is identified for the RGB Raster. The LSB in the flags item is used to indicate that transparency is enabled for the raster component.



RGB Component Structure

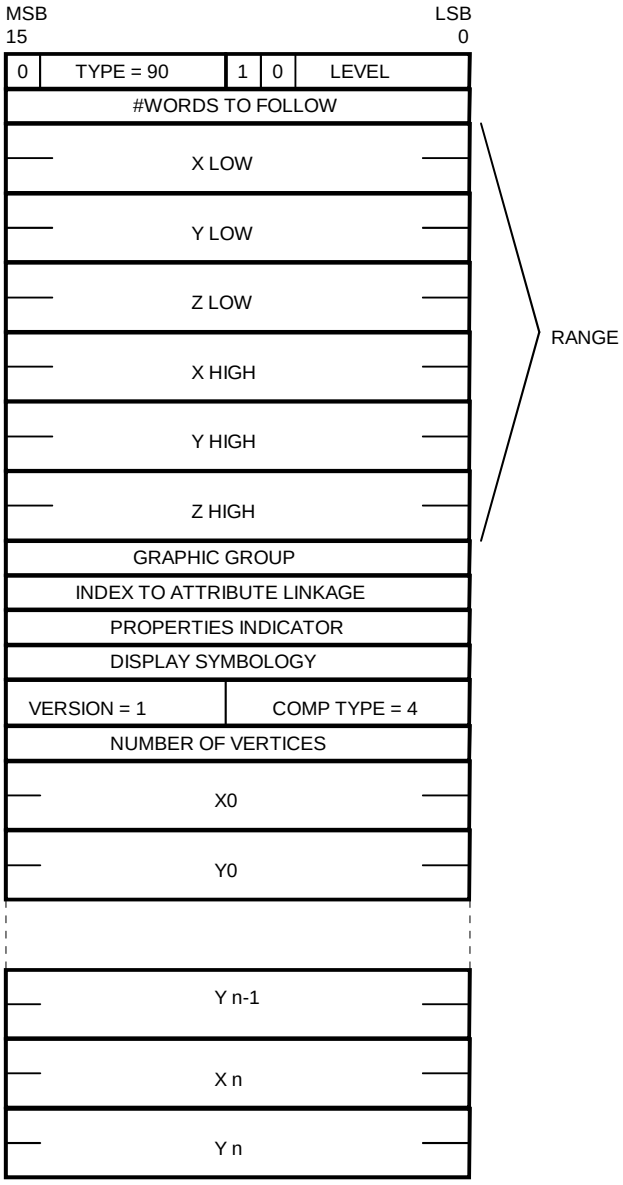
Clip Polygon Optional Component

The optional clip polygon component identifies a polygon in the design space that is used to clip the raster during plotting and display. This is an optional component, if it is not present, the entire raster is displayed or plotted.

The component contains a list of vertices of the polygon in design file UOR coordinates. These UORs are relative to the raster origin (lower left corner of the raster) and are aligned with the raster plane. The range of these UORS are from 0 to width-1 and from 0 to height-1. The first point of the polygon is repeated as the last element.

Plotting does not have a multi-point polygon clipping element implemented, and during the plot process, the minimum enclosing rectangle of the clip polygon will be calculated and used to clip the raster data.

If the Optional Clip Polygon component is not present, the view extents are used to clip the raster data. If the clip polygon is present, but reaches outside the view, the raster data is still clipped to the view extents.



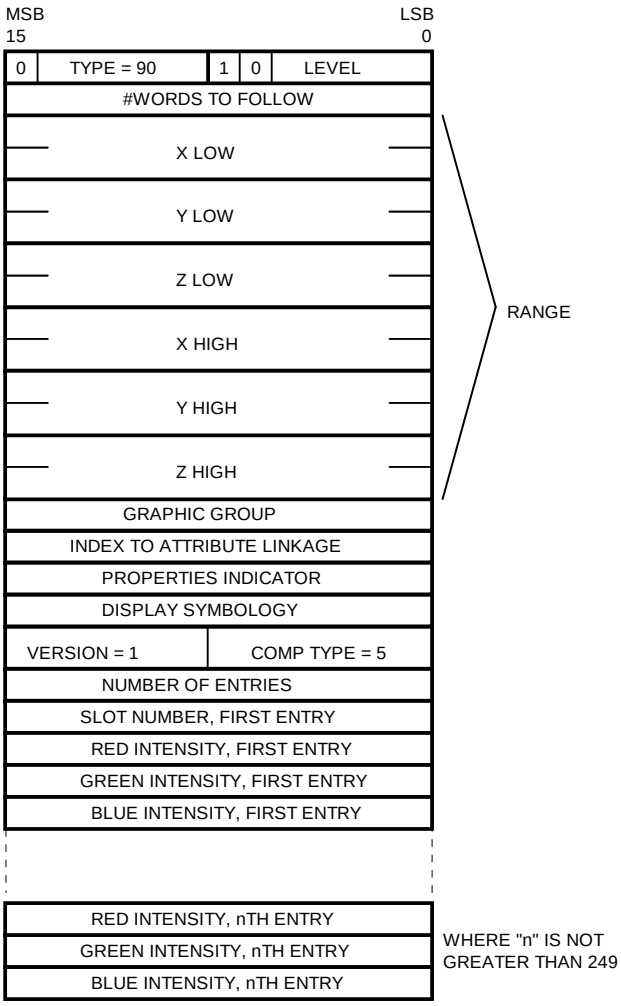
Clip Polygon Component Structure

Color Table Optional Component

The plotting software uses the color table stored in the first Type 5, Level 4 element to determine the vector color table during plotting. This color table will also be used if there is no color table in the type 90 element.

The color table consists of a standard IGDS header and a starting slot number, followed by a list of RGB color values. Because of limitations on the length of an IGDS element, the number of RGB triplets are limited to 249 in any single color table component. Multiple color table components can be used to specify color tables longer than 249. There is no limit to the number of color table components that can be included in the type 90.

Each color table list of triplets begins with the specified slot number, and each succeeding triplet populates the next contiguous slot number.



Color Table Component Structure

REVISION HISTORY

Version	Comments	Date
1.0.1	First Internal Release	12/29/91
2.0.1	Supplemental Information Separated	1/7/92
2.0.2	Final Draft Released For Comment	1/20/92
3.0.0	Editorial Changes	6/13/92
3.0.1	Final First Draft Release	6/22/92
3.0.2	Added Raster Element by Reference	6/26/92
3.0.3	Final Draft Edits	9/7/92
3.0.4	Internal Review	2/15/93
3.0.5	Software Planning Board Approval	4/13/93
3.1.0	System Executive Board Approval	4/19/93
3.2.0	First Public Release	2/1/94