INTERNATIONAL STANDARD

ISO/IEC 9636-6

First edition
1991-12-15

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification —

Part 6: Raster

Technologies de l'information — Infographie — Interfaces pour l'infographie — Spécifications fonctionnelles —

Partie 6: Raster

Reference number
ISO/IEC 9636-6:1991(E)

Contents

Page

Foreword iv

Introduction v

1 Scope 1

2 Normative references 2

3 Raster concepts 3
- 3.1 Introduction 3
- 3.2 Architectural concepts 3
  - 3.2.1 Bitmaps 3
  - 3.2.2 Displayable bitmaps 3
  - 3.2.3 Non-displayable bitmaps 4
  - 3.2.4 Bitmap identifiers 5
- 3.3 Control of bitmap manipulations 5
  - 3.3.1 The drawing bitmap 5
  - 3.3.2 Two-operand bitblts 5
  - 3.3.3 Tile three-operand bitblt 6
  - 3.3.4 Bitmaps regions used as patterns 6
  - 3.3.5 Drawing modes 6
  - 3.3.6 Transparency 7
  - 3.3.7 Raster operation functions with mapped bitmaps 7
  - 3.3.8 Rendering in full-depth bitmaps 7
  - 3.3.9 Rendering in mapped bitmaps 7
- 3.4 Pixel array 7
- 3.5 The VDC-to-Device Mapping and clipping 8
  - 3.5.1 Determining the position and size of created bitmaps 8
  - 3.5.2 Clipping 9
- 3.6 Inquiry 10

4 Interactions with other parts of ISO/IEC 9636 11
- 4.1 Interactions with ISO/IEC 9636-2 (Control) 11
- 4.2 Interactions with ISO/IEC 9636-3 (Output) 11
- 4.3 Interactions with ISO/IEC 9636-4 (Segments) 11
- 4.4 Interactions with ISO/IEC 9636-5 (Input) 11

5 Abstract specification of functions 12
- 5.1 Introduction 12
  - 5.1.1 Data types employed 12
  - 5.1.2 Validity of returned information 12
- 5.2 Raster control functions 12
  - 5.2.1 GET NEW BITMAP IDENTIFIER 12
  - 5.2.2 CREATE BITMAP 12
  - 5.2.3 DELETE BITMAP 14
  - 5.2.4 DRAWING BITMAP 14
  - 5.2.5 DISPLAY BITMAP 15
  - 5.2.6 MAPPED BITMAP FOREGROUND COLOUR 15
  - 5.2.7 MAPPED BITMAP BACKGROUND COLOUR 15
  - 5.2.8 TRANSPARENT COLOUR 15
- 5.3 Raster attribute functions 16
  - 5.3.1 DRAWING MODE 16
  - 5.3.2 FILL BITMAP 17
- 5.4 Raster operation functions 18
  - 5.4.1 PIXEL ARRAY 18

5.4.2 GET PIXEL ARRAY 19
5.4.3 GET PIXEL ARRAY DIMENSIONS 20
5.4.4 SOURCE DESTINATION BITBLT 20
5.4.5 TILE THREE OPERAND BITBLT 21

6 Raster inquiry functions 25
6.1 Introduction 25
6.1.1 Data types employed 25
6.1.2 Validity of returned information 25
6.2 Raster description table 25
6.2.1 INQUIRE RASTER CAPABILITY 25
6.2.2 INQUIRE LIST OF SUPPORTED DRAWING-MODE/TRANSPARENCY PAIRS 26
6.2.3 INQUIRE LIST OF SUPPORTED DRAWING-MODE-3/TRANSPARENCY PAIRS 26
6.3 Raster state list 26
6.3.1 INQUIRE RASTER STATE 26
6.3.2 INQUIRE LIST OF NON-DISPLAYABLE BITMAP IDENTIFIERS 26
6.3.3 INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS 27
6.4 Bitmap state list 27
6.4.1 INQUIRE BITMAP STATE 27

7 Raster description tables and state lists 28
7.1 Raster description table 28
7.2 State lists 29
7.2.1 Raster state list 29
7.2.2 Bitmap state list 29

A Formal grammar of the functional specification 31

B Raster errors 38

C Guidelines for CGI implementors 39

D List of BOOLEANOP class drawing-mode-3 values 41

E Some raster operation examples 45

F Algorithmic explanation of raster operations 47

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9636-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9636 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification*:

— *Part 1: Overview, profiles, and conformance*
— *Part 2: Control*
— *Part 3: Output*
— *Part 4: Segments*
— *Part 5: Input and echoing*
— *Part 6: Raster*

Annexes A and B form an integral part of this part of ISO/IEC 9636. Annexes C, D, E, and F are for information only.

Introduction

This part of ISO/IEC 9636 describes the functions of the Computer Graphics Interface concerned with raster graphic specific devices.

The functional capability incorporated in this part of ISO/IEC 9636 is concerned with creating, manipulating, displaying and retrieving information stored as pixel data below the CGI in a device independent, yet efficient manner.

The functionality described in this part of ISO/IEC 9636 pertains to Virtual Devices of class OUTPUT and OUTIN with display type RASTER.

This page intentionally left blank

Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification –

Part 6:
Raster

1 Scope

This part of ISO/IEC 9636 describes those functions of the Computer Graphics Interface concerned with creating, modifying, retrieving, and displaying portions of an image stored as pixel data. It includes functionality for combining such images.

This part of ISO/IEC 9636 is part 6 of ISO/IEC 9636 and should be read in conjunction with ISO/IEC 9636-1, ISO/IEC 9636-2, and ISO/IEC 9636-3. The relationship of this part of ISO/IEC 9636 to the other parts of ISO/IEC 9636 is described in ISO/IEC 9636-1 (see ISO/IEC 9636-1, 5.2.1 and figures 6 and 7) and in clause 4.

The functionality described in this part of ISO/IEC 9636 pertains to Virtual Devices of class OUTPUT and OUTIN with display type RASTER.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9636. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9636 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 9636-1 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 1: Overview, profiles, and conformance.*

ISO/IEC 9636-2 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 2: Control.*

ISO/IEC 9636-3 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 3: Output.*

ISO/IEC 9636-4 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 4: Segments.*

ISO/IEC 9636-5 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 5: Input and echoing.*

ISO/IEC 9637-1 : -[1] *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 1: Character encoding.*

ISO/IEC 9637-2 : -[1] *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding.*

ISO/IEC TR 9973 : 1988 *Information processing — Procedures for registration of graphical items.*

1) To be published.

3 Raster concepts

3.1 Introduction

This part of ISO/IEC 9636 defines a set of functions for creating, modifying, retrieving, and displaying information stored as pixel data below the CGI. This functionality is divided into the following areas

- *Raster control functions*, including functions for the creation and deletion of bitmaps, and the selection of drawing and display bitmaps, and the control of raster transparency and mapped bitmap expansion.
- *Raster attribute functions*, for setting particular attributes which have significance with other graphical output, defined in ISO/IEC 9636-3 and this part of ISO/IEC 9636, when used in conjunction with raster functionality.
- *Raster operation functions*, including display and retrieval of pixel array data, and various forms of bitmap manipulation operations (bitblts) including movement, combination, and replication of bitmap regions.
- *Raster inquiry functions*, which provide access to the description tables and state lists defined in this part of ISO/IEC 9636.

3.2 Architectural concepts

3.2.1 Bitmaps

A bitmap is a region of computer memory that can be treated as if it were a rectangular array of pixels. Bitmaps are created and maintained below the CGI, and their internal format is hidden from the CGI client. Bitmaps never share common memory. Functions are provided to allow a CGI client to create and manage bitmaps.

Bitmaps may be INDEXED, DIRECT, and MIXED, as specified by the Array of Supported Bitmap Mode Combinations entry in the Raster Description Table, indicating the type of colour values that may be contained in a bitmap. When MIXED, both indexed and direct colour values may exist simultaneously in a bitmap.

3.2.2 Displayable bitmaps

Displayable bitmaps are special bitmaps that can be displayed on the display surface. The client can select which displayable bitmap is to be displayed on the display surface and a different displayable bitmap may be selected by the client at any time. There are from 1 to N predefined displayable bitmaps, where N is defined in the Raster Description Table. Predefined displayable bitmaps are all created to be the same size as the display surface and may not be deleted. Additional displayable bitmaps may be created at any arbitrary rectangular size by the client. Client created bitmaps may be deleted.

In some environments the displayed bitmap is subject to spontaneous change in dimensions; for example, in window-managed environments. Whether or not such spontaneous change in the displayable bitmap dimensions is allowed is indicated by an entry in the Output Device Description Table defined in ISO/IEC 9636-2. If such spontaneous change in dimensions is allowed, the device coordinate information in the Bitmap State List for the currently displayed bitmap will be modified to reflect the change. Thus, to detect such a spontaneous change, periodic polling by the client of this information is necessary. Alternatively, the CGI client may receive asynchronous notification of a change of dimensions by the environment via a non-CGI interface.

When the currently selected display bitmap does not completely cover the display surface, the CGI allows latitude about whether the contents of previous displayable bitmaps are visible in those regions of the display surface not covered by the currently selected display bitmap. The Previous Display Bitmap Data entry in the Raster Description Table specifies the implemented behaviour, which may be either CLEARED or PRESERVED. A value of PRESERVED indicates that it is possible for the display surface to be dirty even though the drawing surface is clear. If the contents of a previous displayable bitmap are visible and the bitmap is selected as the current drawing bitmap, it is implementation-dependent whether changes to the bitmap are visible.

3.2.3 Non-displayable bitmaps

These bitmaps cannot be displayed directly, but the information within a non-displayable bitmap may be moved to or combined with a displayable bitmap.

Pixels in a non-displayable bitmap are treated as if their aspect ratio is the same as that of pixels in a displayable bitmap for the device.

Non-displayable bitmaps may be one of two types: FULL DEPTH or MAPPED, as illustrated in figure 1.

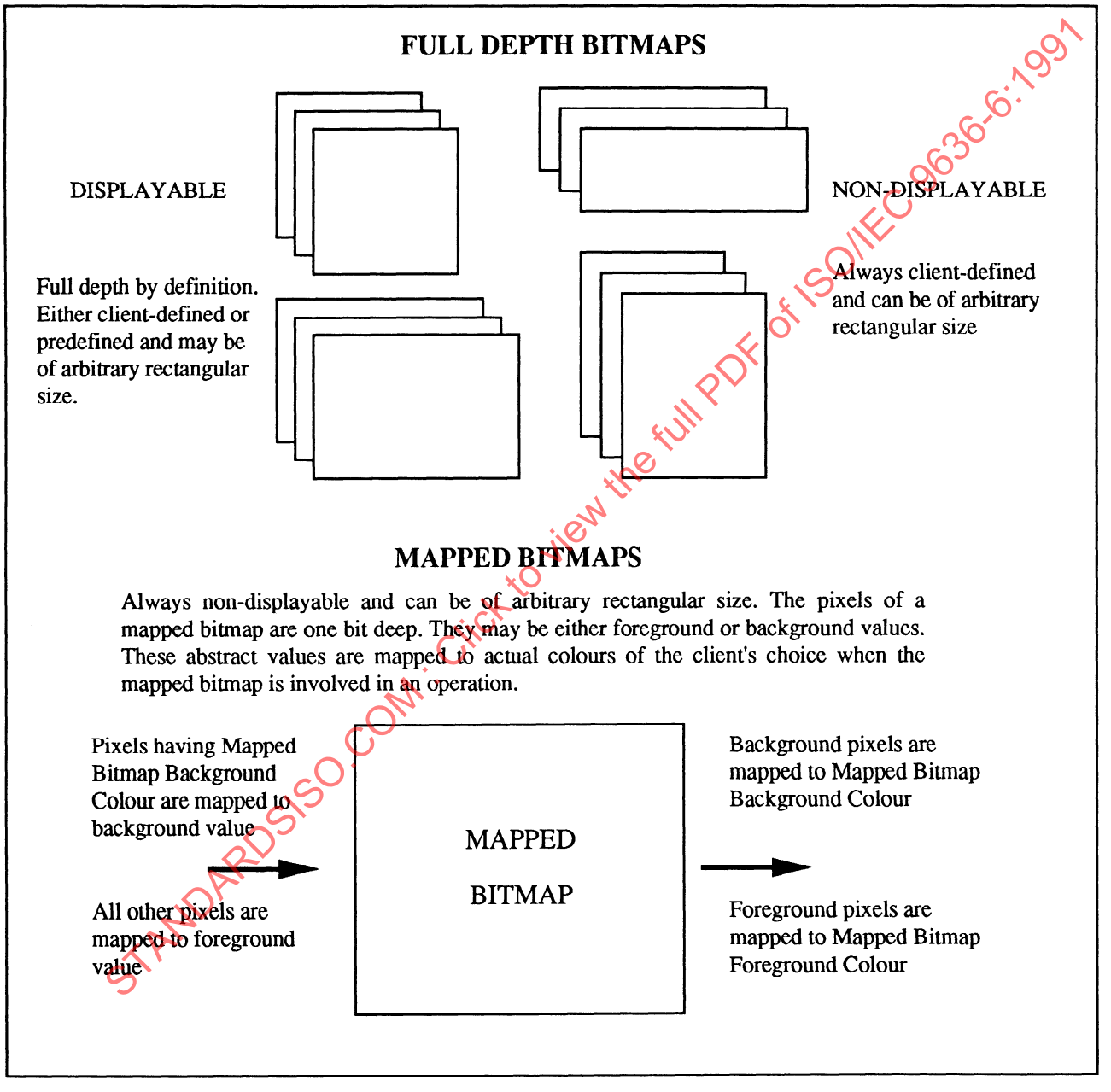

Figure 1 – Types of bitmaps

The pixels of full-depth bitmaps have the same number of bits per pixel as displayable bitmaps. Thus, a full-depth bitmap has the same colour capabilities as the physical device, and maintains the pixel values as CI or CD colour specifiers in the same manner as displayable bitmaps.

The pixels of a mapped bitmap can assume only the two abstract values *foreground* or *background*, which are mapped to actual colours of the client's choice when the bitmap is involved in an operation. Mapped bitmaps are thus convenient for such purposes as storing bitmap character fonts.

The effect of PREPARE DRAWING SURFACE on a mapped bitmap is to set all pixels of the mapped bitmap to the background value.

3.2.4 Bitmap identifiers

Bitmaps are referenced through an identifier. The first N consecutive identifiers identify the N predefined displayable bitmaps. The value of N is contained in the Raster Description Table and is an invariant resource of the CGI Virtual Device. The list of predefined displayable bitmap identifiers is given by the default value of the List Of Bitmap Identifiers entry (for displayable bitmaps) in the Raster State List.

The client can define its own identifiers for bitmaps it creates or it may have identifiers defined for it by using the function GET NEW BITMAP IDENTIFIER. The identifier is passed as an input parameter to the CREATE BITMAP function which then creates a bitmap with the given identifier.

3.3 Control of bitmap manipulations

3.3.1 The drawing bitmap

The client can select which bitmap is affected by drawing operations. This is referred to as the *drawing bitmap*. Any bitmap may be selected as the drawing bitmap regardless of its depth type or displayability: FULL DEPTH or MAPPED, NON DISPLAYABLE or DISPLAYABLE. It is the currently selected drawing bitmap, not the current display bitmap, that is affected by functions such as PREPARE DRAWING SURFACE, VDC EXTENT, DEVICE VIEWPORT, POLYLINE, DRAW ALL SEGMENTS. The client should select explicitly the current displayed bitmap as the drawing bitmap if such functions are to apply to the displayed bitmap.

3.3.2 Two-operand bitblts

A two-operand bitblt function is provided to support operations that move and combine the contents of rectangular regions of bitmaps in memory. The regions of interest are specified using points in VDC. The actual movement of data takes place without regard to the VDC coordinate systems (see figure 2). If the destination for these operations is also the currently selected display bitmap then these operations may affect the displayed picture. The two-operand bitblt function combines the source and destination pixels according to the bitmap type and drawing modes in tables 1 and 2. The particular drawing mode used is specified as one of the parameters of the two-operand bitblt function.

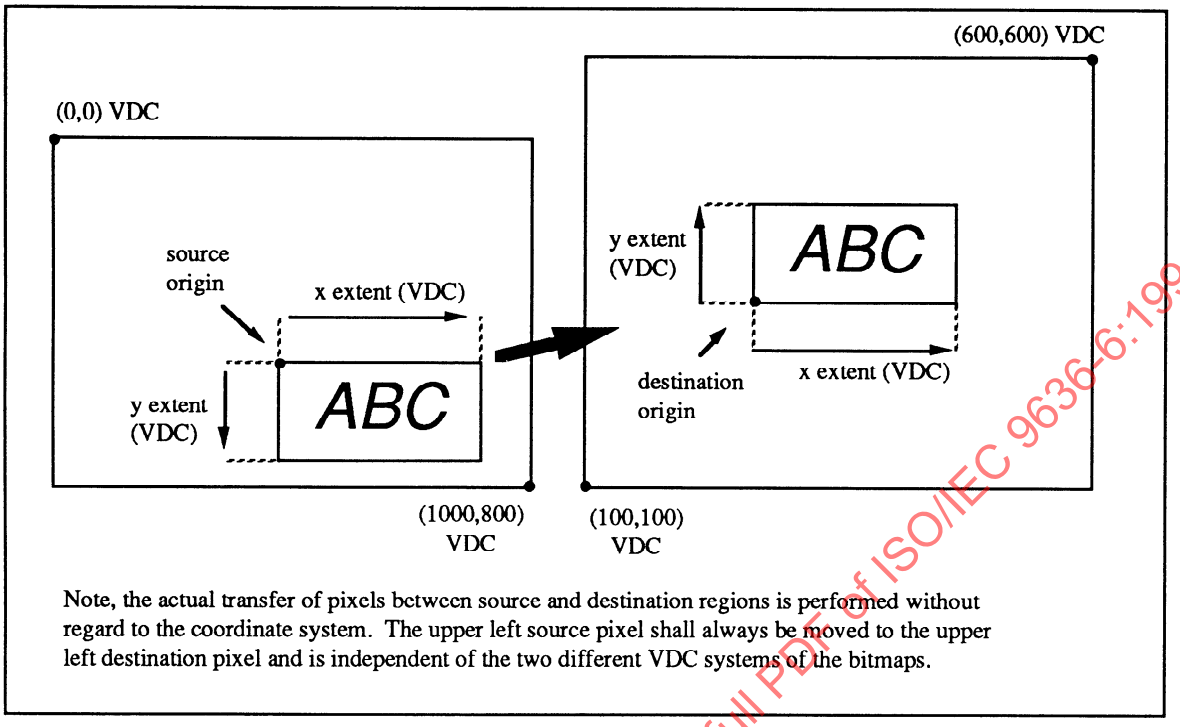

Figure 2 – Bitblt regions

3.3.3 Tile three-operand bitblt

Tile three-operand bitblts support the combination of two sources and a destination for the data movement. One of the sources may be used as a replicating tile and is referred to as the pattern. The drawing mode-3 operation used is specified as one of the parameters of the tile three-operand bitblt function, (refer to 5.4.5 and annex D). Annex E.3 illustrates the use of a tile three-operand bitblt in tiling a filled area (the letter P) in the destination bitmap corresponding to the filled area given by the source bitmap.

3.3.4 Bitmaps regions used as patterns

In addition to using bitmap regions as tiles for a tile three-operand bitblt, they may also be used to provide the pattern data for fill objects defined in ISO/IEC 9636-3. Figure 3 illustrates the use of a bitmap region as a pattern for filling.

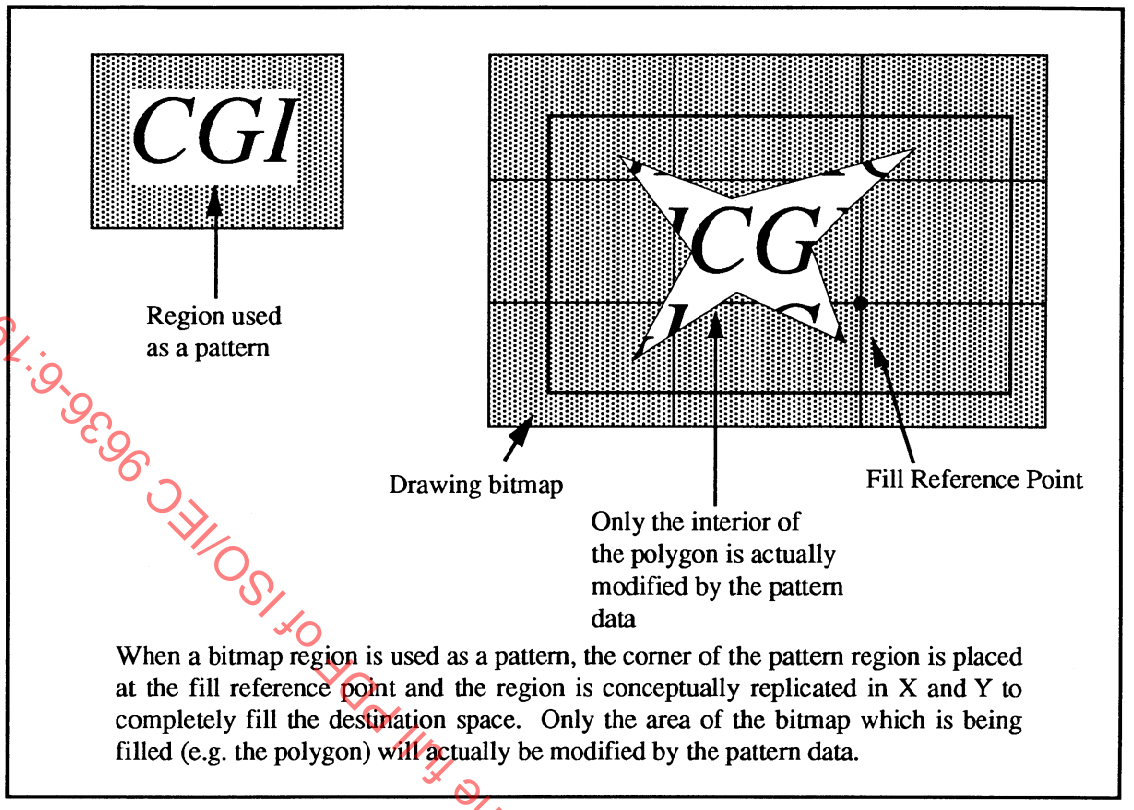

Figure 3 – Bitmap regions used as patterns

There is no restriction on the size of a bitmap region that may be used as a pattern. By using a bitmap region as large as the area being filled and properly aligning it by setting the Fill Reference Point, the pattern bitmap region may be used to achieve the effect of a stencil, rather than a tile. Bitmap regions include their boundary.

The default Fill Bitmap is a mapped bitmap of one pixel having the Mapped Bitmap Foreground Colour.

3.3.5 Drawing modes

Drawing modes are used to select the way in which pixels are combined during rendering or bitblt operations. In particular, it applies to the rendering of graphic objects (defined in ISO/IEC 9636-3) into the drawing bitmap. In this case, the drawing mode used to determine the combining operation is the object's DRAWING MODE attribute value. During object creation, this attribute value is taken from the Drawing Mode entry in the Raster State List. The drawing mode concept also applies to raster operation functions where the specification of a particular drawing mode is supplied as a separate parameter.

A device-dependent colour index value may be used if the result of a combining operation using indexed colour values is undefined. The RGB components of direct colour values are combined separately. A device-dependent colour value may be used if the result of a combining operation on any of the individual RGB components results in an overflow.

Implementation-dependent colour value conversions may occur when using the raster operation functions if the pixel values of the source and destination bitmaps are not all specified as indexed colour values nor all specified as direct colour values. An implementation-dependent colour value may be placed in the drawing bitmap if the colour selection mode of the colour value resulting from a combining operation differs from the Bitmap Mode of the drawing bitmap.

3.3.6 Transparency

The concept of transparency is also applicable to raster operation functions. For these operations, the transparency value is supplied as a parameter of the raster operation functions. If the value of the transparency parameter is OPAQUE, all pixels transferred to the destination bitmap region will affect the destination. If the value of the transparency parameter is TRANSPARENT, only those pixels that either have, or expand to (in the case of mapped bitmaps), a value that is not that of the Transparent Colour entry in the Raster State List will affect the destination. (See also ISO/IEC 9636-3, 3.4.1.5.)

3.3.7 Raster operation functions with mapped bitmaps

In performing a raster operation function which involves a combining operation over pixel values from one or more mapped bitmaps, then, prior to the pixel combining operation, pixels in mapped bitmaps which have "foreground value" are expanded to the value of Mapped Bitmap Foreground Colour in the Raster State List and those which have "background value" are expanded to the value of Mapped Bitmap Background Colour in the Raster State List. Such expansion also occurs whenever the destination bitmap is full-depth, even when there is no effective combining operation (e.g. $d' <- s$).

In performing a raster operation function in which the destination bitmap is a mapped bitmap, then, after any pixel combining operations, destination pixels are set to "background value" whenever the result of the combining operation was a pixel with value equal to that of Mapped Bitmap Background Colour and to "foreground value" otherwise.

3.3.8 Rendering in full-depth bitmaps

The associated DRAWING MODE attribute value of a graphic object affects the way that object is rendered in the drawing bitmap. For each pixel affected in rendering an object, the drawing colour may either be the colour attribute value associated with the object (or edge) or the AUXILIARY COLOUR attribute value (if the associated TRANSPARENCY attribute is OPAQUE). The drawing colour is combined with the destination pixel in accordance with the associated DRAWING MODE attribute value and the result replaces the former value of the destination pixel. The Transparent Colour entry in the Raster State List has no affect on this rendering operation.

3.3.9 Rendering in mapped bitmaps

The rendering of an object in the drawing bitmap when that bitmap is a mapped bitmap, is affected by the Mapped Bitmap Background Colour and Mapped Bitmap Foreground Colour in the Raster State List. Conceptually, an affected destination pixel is expanded to a full-depth value according to whether it is foreground or background. It is then combined with the drawing colour (determined by the object's colour and transparency attributes) in accordance with the associated DRAWING MODE attribute value. If the result is equal to the value of Mapped Bitmap Background Colour then the affected destination pixel is set to the "background value", otherwise it is set to the "foreground value".

3.4 Pixel array

The PIXEL ARRAY function is related to the Bitblt's described above, in that it is not considered to create a graphic object and is treated in a manner similar to a Bitblt where the source array is provided by the client and the destination is the drawing bitmap. The colour information in the pixel array maps directly to the pixels of the destination bitmap. Thus the starting point and colour information are specified in a device-independent fashion, but the appearance of the final image depends directly on the resolution and aspect ratio of the target device. An MxN array of device-independent colour specifiers are assigned to an MxN array of pixels (assuming the x and y scale parameters are both 1). The differences between CELL ARRAY, PIXEL ARRAY, and Bitblt are illustrated in figure 4.

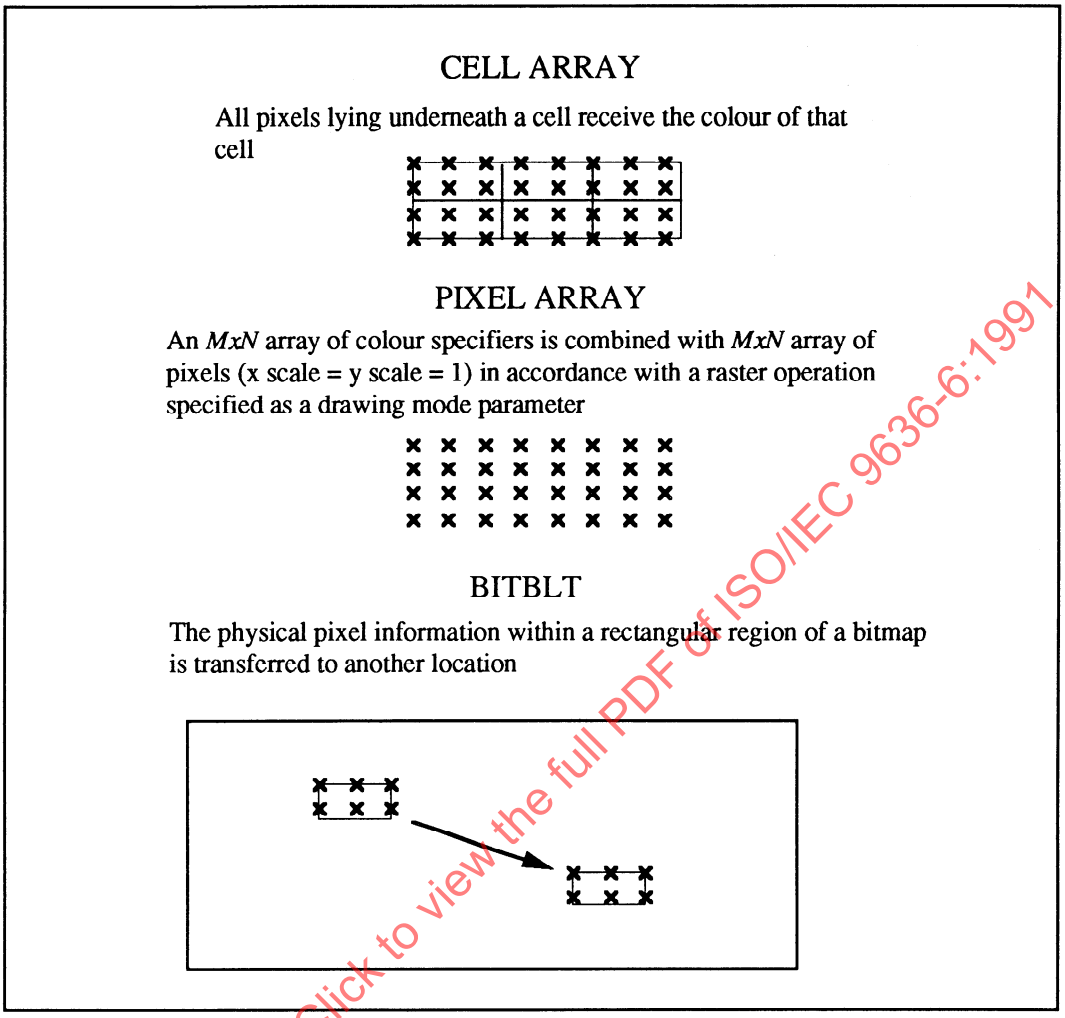

Figure 4 – A comparison of CELL ARRAY, PIXEL ARRAY, and Bitblt

The GET PIXEL ARRAY function returns a rectangular array of colour values from the identified source bitmap to the client.

3.5 The VDC-to-Device Mapping and clipping

3.5.1 Determining the position and size of created bitmaps

The position on the display surface and size, in pixels, of the bitmap to be created are derived from the VDC coordinates of the bitmap extent (specified as a parameter of the function CREATE BITMAP) transformed by the VDC-to-Device Mapping of the current drawing bitmap (see figure 5). In essence, the size, in pixels, is determined by passing the corner points of the bitmap extent through the VDC-to-Device Mapping. The x and y displacements from the first DC point to the second give the dimensions in pixels of the bitmap that the client desires to have created.

While the bitmap dimensions of an existing bitmap cannot be modified by the client, the client may modify the VDC-to-Device Mapping for the bitmap by selecting the bitmap as the drawing bitmap and then invoking the functions VDC EXTENT and DEVICE VIEWPORT (see ISO/IEC 9636-2).

Once created, the VDC-to-Device Mappings of different bitmaps are completely independent. The client might use the VDC EXTENT function to specify several different bitmaps with the same VDC Extents (even if the dimensions in pixels of the bitmaps were different) or might specify different VDC Extents for bitmaps having the same bitmap dimensions.

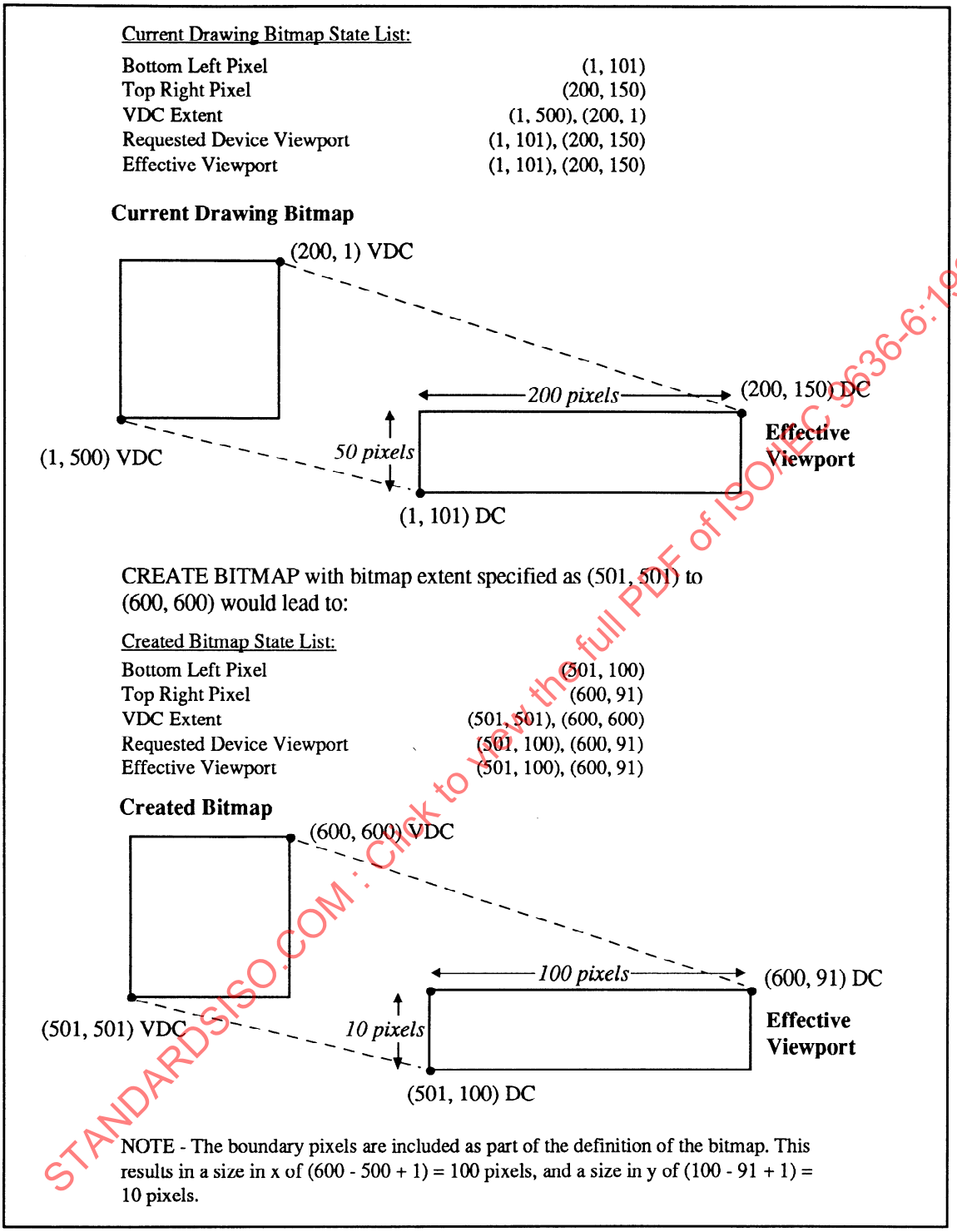

Figure 5 – Creating bitmaps

3.5.2 Clipping

A CGI implementation supporting raster functionality renders into the currently selected drawing bitmap. Graphic objects are clipped as described in ISO/IEC 9636-2 and ISO/IEC 9636-3 using the Drawing Surface Clipping entries in the Control State List and the clip attributes associated with each object. However, when utilizing the functionality defined in this part of ISO/IEC 9636, the inquiry functions defined in ISO/IEC 9636-2 that return the Drawing Surface Clip Rectangle and Drawing Surface Clip Indicator will return this information from the state list of the currently selected drawing bitmap. Each bitmap,

whether predefined by the implementation or client-defined, has its own Drawing Surface Clip Rectangle and Drawing Surface Clip Indicator as described in 4.1. When a bitmap is selected as the current drawing bitmap (using the function DRAWING BITMAP) the Drawing Surface Clip entries in the Control State List are effectively the same as those entries in the state list of the currently selected drawing bitmap. Additionally, where the description of a clipping function defined in ISO/IEC 9636-2 refers to "the limits of the device's drawing surface" or "the physical device limits", these terms should be interpreted as being equivalent to the drawing bitmap's dimensions in pixels.

The raster operation functions, PIXEL ARRAY, SOURCE DESTINATION BITBLT, and TILE THREE OPERAND BITBLT, are not treated as graphic objects, and as such, object clipping is not applied. The drawing surface clipping is applied if the Drawing Surface Clip Indicator is on (i.e. either DSCRECT or VIEWPORT). This clip only applies when modifying the destination bitmap; it does not apply to either the source or pattern bitmap regions. The effective drawing surface clip rectangle includes the intersection with the destination region.

A conforming CGI raster implementation is permitted to produce device-dependent effects when drawing outside of the dimensions, in pixels, of the drawing bitmap if clipping has not been requested. If portions of the source or pattern bitmap regions lie outside of the bitmap it is permitted to produce device-dependent effects; the preferred behaviour is that the source and pattern bitmap regions which extend outside the bitmap be truncated to that portion which does lie in the bitmap. Should the pattern bitmap region be truncated then the truncated portion is replicated as the file. If the source bitmap region is truncated then only an area the size of the source bitmap region is affected in the destination bitmap. The implemented behaviour is indicated by the Source Bitmap Truncation Capability entry in the Raster Description Table.

If a bitmap exceeds the limits of the display surface, that part of the bitmap outside of the display surface is not drawn, which is equivalent to clipping the contents of the bitmap to the limits of the display surface.

3.6 Inquiry

Raster inquiry functions, as defined in clause 6, provide the client with the means to access the information in the Raster Description Table, and Raster and Bitmap State Lists. These description tables and state lists provide information about the current state and capabilities of the CGI Virtual Device.

4 Interactions with other parts of ISO/IEC 9636

This clause discusses the interactions of the functions and features defined in this part of ISO/IEC 9636 with the other parts of ISO/IEC 9636.

4.1 Interactions with ISO/IEC 9636-2 (Control)

The INITIALIZE function sets all state list entries to their default values and all dynamically created entities are deleted. It is recommended that, when possible, the content of the drawing surface is not affected by the INITIALIZE function.

There are no state restrictions on the use of INITIALIZE or TERMINATE, i.e. INITIALIZE and TERMINATE may be used at any time (see ISO/IEC 9636-2, 5.2.1).

All functions in ISO/IEC 9636-2 that address the drawing surface (for example, EXECUTE DEFERRED ACTIONS, PREPARE DRAWING SURFACE, END PAGE, VDC EXTENT, etc.) only affect the current drawing bitmap. If the drawing bitmap is a mapped bitmap then PREPARE DRAWING SURFACE will set all pixels of the drawing bitmap to the "background value".

For a CGI implementation which supports the functionality defined in this part of ISO/IEC 9636, the VDC-to-Device Mapping entries of the current drawing bitmap determine the VDC-to-Device Mapping. The corresponding values of the Control State List defined in ISO/IEC 9636-2 in such implementations are actually the corresponding entries in the current Drawing Bitmap's Bitmap State List. Inquiry of the Control State List defined in ISO/IEC 9636-2 shall therefore always return values for VDC-to-Device Mapping and Drawing Surface Clip entries from the current drawing bitmap's Bitmap State List.

When a function defined in ISO/IEC 9636-2 that affects these values is invoked, it is the Bitmap State List or the current drawing bitmap which is updated. If another drawing bitmap is selected, the values previously set in its Bitmap State List will come into use in determining the VDC-to-Device Mapping and Drawing Surface Clip for subsequent output into the bitmap.

Where a description of a clipping function defined in ISO/IEC 9636-2 refers to "the limits of the device's drawing surface" or "the physical device limits", then these terms should be interpreted as being equivalent to the drawing bitmap's dimensions in pixels.

4.2 Interactions with ISO/IEC 9636-3 (Output)

All functions in ISO/IEC 9636-3 that address the drawing surface (for example, POLYLINE, RECTANGLE, CIRCLE, TEXT, CELL ARRAY, etc.) only affect the current drawing bitmap.

Deleting a bitmap that is used in conjunction with fill interior style BITMAP by fill objects defined in ISO/IEC 9636-3 will result in these objects using the default fill bitmap.

Predefined bundles shall not specify interior style BITMAP as there are no predefined fill bitmaps.

Certain graphic objects formed by functions of ISO/IEC 9636-3 may have the raster attributes DRAWING MODE and FILL BITMAP associated with them.

4.3 Interactions with ISO/IEC 9636-4 (Segments)

All segment rendering operations, including implicit segment regeneration, only affect the current drawing bitmap.

4.4 Interactions with ISO/IEC 9636-5 (Input)

There are no interactions between ISO/IEC 9636-5 and this part of ISO/IEC 9636. In particular, echoing is conceptually performed downstream of any drawing or display bitmap and shall not produce client detectable effects in these bitmaps.

5 Abstract specification of functions

5.1 Introduction

This clause describes the abstract functional capabilities of the raster part of the CGI. The descriptions of individual functions are grouped on the basis of functional association as follows:

- Raster control functions
- Raster attribute functions
- Raster operation functions

5.1.1 Data types employed

The abstract specifications of functions detail the functions in terms of input and output parameters. The data type of each parameter is selected from a standard set and is identified in the functional specification by a standard abbreviation.

The data types and the abbreviations used in this part of ISO/IEC 9636 are taken from the complete list of data types given in ISO/IEC 9636-1, 5.2.10.

5.1.2 Validity of returned information

For all of the functions specified in this clause which solicit a response from the Virtual Device, a response validity flag is returned as INVALID if an error was detected in executing the function. In such cases, other output parameters are undefined and no meaning should be applied to any of these parameter values.

5.2 Raster control functions

5.2.1 GET NEW BITMAP IDENTIFIER

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	bitmap identifier		BN

Effect:

The CGI returns a unique *bitmap identifier*. This *bitmap identifier* may be used as a parameter given to the CREATE BITMAP function.

GET NEW BITMAP IDENTIFIER shall not return a bitmap identifier which is associated with an existing bitmap, either a predefined displayable bitmap or a client-defined bitmap which has been created and not deleted.

If no new bitmap identifier is available, a *response validity* value of INVALID is returned.

NOTE – This function does not have to be used to obtain a bitmap identifier; however, it can be used to minimize the tracking of bitmap identifiers by the client.

5.2.2 CREATE BITMAP

Parameters:

In	bitmap identifier		BN
In	bitmap extent		2P
In	depth type	(MAPPED, FULL DEPTH)	E

In	bitmap displayability	(NON DISPLAYABLE, DISPLAYABLE)	E

Effect:

This function creates a bitmap with the given *bitmap identifier*. The client can specify a *bitmap identifier* that is client determined or use the function GET NEW BITMAP IDENTIFIER to obtain a unique bitmap identifier. Displayable bitmaps may or may not be created depending on the value of Display Bitmap Creation Supported in the Raster Description Table.

The Depth Type, Displayability, Bitmap Mode, Bottom Left Pixel, Top Right Pixel, VDC-to-Device Mapping group, and Drawing Surface Clipping group of entries in the Bitmap State List are set directly or indirectly by this function. In particular the Bottom Left Pixel, Top Right Pixel, and the VDC-to-Device Mapping group of entries are derived from the *bitmap extent* parameter as follows:

- The VDC Extent is set to the value of the *bitmap extent* parameter.
- The points derived from the *bitmap extent* parameter (see 3.5.1) are used to set the Bottom Left Pixel and Top Right Pixel entries. These values are derived by the implementation after it has created the bitmap. For displayable bitmaps, the Bottom Left Pixel and Top Right Pixel may differ from the specified bitmap extent and consequently an implementation may create a bitmap that is smaller than that specified. Bitmaps do not share physical memory with one another.
- The Requested Device Viewport and Effective Viewport entries are set to contain the entire bitmap as follows:
  - If the value of the Specification Mode of Device Viewport in the state list of the currently selected drawing bitmap is PHYSICAL DEVICE COORDINATES, then these entries are set to the coordinates of the Bottom Left Pixel and Top Right Pixel DC values.
  - If the value of the Specification Mode of Device Viewport in the state list of the currently selected drawing bitmap is FRACTION OF DISPLAY SURFACE, then these entries are set to (0.0, 0.0), (1.0, 1.0).
  - If the value of the Specification Mode of Device Viewport in the state list of the currently selected drawing bitmap is MILLIMETRES WITH SCALE FACTOR, then the VP coordinates are derived utilizing the Size of Pixel entry in the Raster Description Table.

It should be noted that since the aspect ratios are the same, isotropy has no effect. Refer to ISO/IEC 9636-2, 3.3 for a more detailed description of the VDC-to-Device Mapping.

Additionally, the following are also set in the Bitmap State List:

- The Drawing Surface Clip Indicator, Drawing Surface Clip Rectangle, Specification Mode of Drawing Surface Clip Rectangle, and Metric Scale Factor of Drawing Surface Clip Rectangle are set to the default values as described in the Control State List defined in ISO/IEC 9636-2.
- The Specification Mode of the Current Device Viewport is set to the value that is associated with the current drawing bitmap.

See also 3.2.2 and 3.2.3 for a description of bitmap types.

NOTE – an implementation may only be able to provide displayable bitmaps that are the same size as the display surface.

Errors:

Error identifier: 3:602
Cause: Unsupported bitmap depth type requested
Reaction: Function ignored.

Error identifier: 3:605
Cause: Bitmap identifier already in use
Reaction: Function ignored.

Error identifier: 3:609
Cause: Unsupported bitmap displayability requested.
Reaction: Function ignored.

Error identifier: 6:601
Cause: Bitmap memory overflow.
Reaction: Function ignored.

5.2.3 DELETE BITMAP

Parameters:

In bitmap identifier BN

Effect:

The Bitmap State List and bitmap of the associated *bitmap identifier* are deallocated.

Some bitmap identifiers may be specified in bundle tables or in FILL BITMAP to address filling operations (see ISO/IEC 9636-3). Deleting one of these identifiers shall not generate an error. The default fill bitmap will be used at the time any function defined in ISO/IEC 9636-3 attempts to perform a fill operation with a non-existing bitmap pattern.

NOTE – DELETE BITMAP informs the CGI Virtual Device that the client has no further need of the specified bitmap. The Virtual Device may use this information as an indication to free the memory allocated to the state list and bitmap in a dynamic memory allocation implementation.

Errors:

Error identifier: 3:606
Cause: Cannot delete the current drawing bitmap
Reaction: Function ignored.

Error identifier: 3:607
Cause: Bitmap identifier is assigned to a predefined displayable bitmap.
Reaction: Function ignored.

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

5.2.4 DRAWING BITMAP

Parameters:

In bitmap identifier BN

Effect:

Selects the bitmap into which all subsequent graphic output will be drawn.

NOTE – When a bitmap is selected as the drawing bitmap, it becomes the output area for all subsequent CGI functions and the destination bitmap for all bitblt functions. When a bitmap is selected as the drawing bitmap, the Control State List defined in ISO/IEC 9636-2 inherits the VDC-to-Device Mapping and Drawing Surface Clipping entries in the state list of the currently selected drawing bitmap. Thus, for example, the client may specify a clipping rectangle, draw segments, lines, polygons, etc., with all output going to the selected drawing bitmap.

It is the drawing bitmap (and not the displayed bitmap) that is affected by the functions defined in ISO/IEC 9636-2 (e.g. PREPARE DRAWING SURFACE, VDC EXTENT, and DEVICE VIEWPORT) as well as by the rendering functions from other parts of ISO/IEC 9636.

Selecting a drawing bitmap may implicitly change the VDC Extent and thus change the VDC-to-Device Mapping and Drawing Surface Clipping.

Errors:

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

5.2.5 DISPLAY BITMAP

Parameters:

In bitmap identifier BN

Effect:

Selects the displayable bitmap identified by the *bitmap identifier* to be displayed.

The CGI allows latitude concerning whether the contents of uncovered regions of previously displayable bitmaps are visible when the currently selected display bitmap does not completely cover the display surface (see 3.2.2).

See also 5.2.4.

Errors:

Error identifier: 3:601
Cause: Bitmap identifier is not assigned to a displayable bitmap
Reaction: Function ignored.

5.2.6 MAPPED BITMAP FOREGROUND COLOUR

Parameters:

In colour specifier CO

Effect:

The Mapped Bitmap Foreground Colour entry in the Raster State List is set to the value specified and will be used as the colour to which the foreground pixels of a mapped bitmap are expanded.

When mapped bitmaps are used in bitblt operations, the background pixels are expanded to Mapped Bitmap Background Colour and the foreground pixels are expanded to Mapped Bitmap Foreground Colour before performing any pixel combining operations. Additionally, when the destination bitmap is a mapped bitmap, if the result of the operation has the same value as Mapped Bitmap Background Colour the corresponding pixel value is set to "background value"; otherwise, the pixel value is set to "foreground value".

5.2.7 MAPPED BITMAP BACKGROUND COLOUR

Parameters:

In colour specifier CO

Effect:

The Mapped Bitmap Background Colour entry in the Raster State List is set to the value specified and will be used as the colour to which the background pixels of a mapped bitmap are expanded.

When mapped bitmaps are used in bitblt operations, the background pixels are expanded to Mapped Bitmap Background Colour and the foreground pixels are expanded to Mapped Bitmap Foreground Colour before performing any pixel combining operations. Additionally, when the destination bitmap is a mapped bitmap, if the result of the operation has the same value as Mapped Bitmap Background Colour the corresponding pixel value is set to "background value"; otherwise, the pixel value is set to "foreground value".

5.2.8 TRANSPARENT COLOUR

Parameters:

In colour specifier CO

Effect:

The Transparent Colour entry in the Raster State List is set to the value specified.

When a raster operation function is performed in which the transparency parameter is TRANSPARENT, the pixels of the destination bitmap region are not modified in positions corresponding to pixels in the source bitmap region whose value is the same as that of Transparent Colour.

5.3 Raster attribute functions

5.3.1 DRAWING MODE

Parameters:

In	drawing mode	((-n..-1,1..n),(-n..n))	[IX,IX]

Effect:

The Drawing Mode entry in the Raster State List is set to the specified value. The current value of Drawing Mode is associated with graphic objects created by the graphic primitive functions defined in ISO/IEC 9636-3.

Drawing mode is a compound data type that defines the pixel operation between the source and destination during all output operations. The first component specifies a class of drawing modes, while the second component specifies a particular element within the class, see table 1. Pixel operations are performed on the actual pixel values stored within the bitmap. A pixel operation performed on a specified destination pixel affects no other pixel in the destination bitmap region. For example, an (1, 3) operation, *d' = d MINUS s*, which underflows will not borrow from an adjacent pixel.

The standardized classes of drawing mode are:

1) ADDITIVEOP: A class of drawing mode pixel operations based on normal unsigned two's complement operations between source and destination pixel values. The operations include: PLUS and MINUS, which are standard addition and subtraction allowing overflow or underflow; ADDCAP, addition which does not overflow, but "caps" the value at the highest possible (unsigned) pixel value; MINUSCAP subtraction which does not underflow, but "caps" the value at the lowest possible (unsigned) pixel value.

2) BOOLEANOP: A class of drawing mode pixel operations based on the standard boolean operations between source and destination pixel values. The operations include: NOT, one's complement; AND, bitwise "and"; OR, bitwise "inclusive-or"; XOR, bitwise "exclusive-or", and "replace".

3) COMPARATIVEOP: A class of drawing mode pixel operations based on unsigned comparisons between source and destination pixel values. The operations include: MAX, the maximum of a source and destination pixel; MIN, the minimum of a source and destination pixel.

Class values above 3 are reserved for registration; negative values are available for private use.

The operands for the standardized operations are defined in table 1. For the boolean operations, each bit *n* of a pixel value can be determined algorithmically by applying the following formula:

$$D'_n = R_3 * (-S_n * -D_n) + R_2 * (-S_n * D_n) + R_1 * (S_n * -D_n) + R_0 * (S_n * D_n)$$

where, $R_3..R_0$ is the 4-bit value of the boolean drawing mode specification (0 to 15) where R_3 is the most significant bit of the value.

S_n, D_n refer to the nth bit of the Source and initial Destination pixel values respectively.

* indicates logical AND.
\+ indicates logical OR.
\- indicates logical NOT.

Values above those given for the second component of the drawing mode specifications in table 1 are reserved for registration; negative values are available for private use.

Table 1 – Drawing modes

Drawing Mode	Operation
(BOOLEANOP, 0)	d' = 0
(BOOLEANOP, 1)	d' = s AND d
(BOOLEANOP, 2)	d' = s AND (NOT d)
(BOOLEANOP, 3)	d' = s
(BOOLEANOP, 4)	d' = (NOT s) AND d
(BOOLEANOP, 5)	d' = d
(BOOLEANOP, 6)	d' = s XOR d
(BOOLEANOP, 7)	d' = s OR d
(BOOLEANOP, 8)	d' = NOT (s OR d)
(BOOLEANOP, 9)	d' = NOT (s XOR d)
(BOOLEANOP, 10)	d' = NOT d
(BOOLEANOP, 11)	d' = s OR (NOT d)
(BOOLEANOP, 12)	d' = NOT s
(BOOLEANOP, 13)	d' = (NOT s) OR d
(BOOLEANOP, 14)	d' = NOT (s AND d)
(BOOLEANOP, 15)	d' = 1
(ADDITIVEOP, 0)	d' = s PLUS d
(ADDITIVEOP, 1)	d' = s ADDCAP d
(ADDITIVEOP, 2)	d' = s MINUS d
(ADDITIVEOP, 3)	d' = d MINUS s
(ADDITIVEOP, 4)	d' = s MINUSCAP d
(ADDITIVEOP, 5)	d' = d MINUSCAP s
(COMPARATIVEOP, 0)	d' = MAX(s,d)
(COMPARATIVEOP, 1)	d' = MIN(s,d)
Note – (d = original destination pixel value, s = original source pixel value, d' = resulting destination pixel value)	

Errors:

Error identifier: 3:603
Cause: Unsupported drawing mode / transparency requested
Reaction: The Drawing Mode entry in the Raster State List is set to the specified value; at interpretation time, the default drawing mode and transparency is used.

5.3.2 FILL BITMAP

Parameters:

In	bitmap identifier	BN
In	pattern bitmap region	2P

Effect:

The Fill Bitmap Identifier and Fill Bitmap Region entries of the Fill Attributes State List defined in ISO/IEC 9636-3 are set to the specified values. If the Fill Bitmap Identifier specifies a bitmap that does not exist, the default Fill Bitmap will be used until such time as the referenced bitmap is created.

The *pattern bitmap region* is defined in the Fill Bitmap's VDC space and is not subject to transformations other than the VDC-to-Device Mapping.

When the fill bitmap is being used as a pattern for drawing, pixel (1, ny) of the Fill Bitmap Region is positioned at the Fill Reference Point in the current drawing bitmap's VDC space, (see figure 3) as specified in ISO/IEC 9636-3, 3.10.2.1. The data of the Fill Bitmap Region is then conceptually replicated as a tile over the entire drawing bitmap's VDC Extent for use in the fill operation. The associated attributes of TRANSPARENCY and AUXILIARY COLOUR are applied. If the value of the associated TRANSPARENCY attribute is OPAQUE, the Fill Bitmap Region is replicated without regard to

the AUXILIARY COLOUR attribute. If the value of the associated TRANSPARENCY attribute is TRANSPARENT, then those pixels in the Fill Bitmap Region that have the same colour as the AUXILIARY COLOUR attribute value shall not affect the pixels in the drawing bitmap.

5.4 Raster operation functions

5.4.1 PIXEL ARRAY

Parameters:

In	origin point		P
In	nx, ny	(1..n)	2I
In	x scale, y scale	(1..n)	2I
In	x direction	(INCREASING VDC, DECREASING VDC)	E
In	y direction	(INCREASING VDC, DECREASING VDC)	E
In	drawing mode	((-n..-1,1..n), (-n..n))	[IX,IX]
In	transparency	(OPAQUE, TRANSPARENT)	E
In	local colour precision requirement	(≥ 0)	3I
In	colour specifiers		nx*ny CO

Effect:

A rectangular array of pixels is combined with the pixels in the drawing bitmap using the rectangular array of device-independent *colour specifiers*.

The *origin point* specifies the location in VDC space of the first pixel to be drawn. The parameters *nx* and *ny* specify the number of colour specifiers for each row, and the number of rows respectively. The parameter *x scale* specifies how many successive pixels in the x direction are used for each colour specifier. Similarly, *y scale* specifies how many successive rows of pixels are to be used for each row of colour specifiers. The parameter *x direction* specifies whether successive pixels in a row are selected in the direction of increasing or decreasing VDC X. Similarly, *y direction* specifies whether successive rows of pixels are selected in the direction of increasing or decreasing VDC Y.

The *local colour precision requirement* parameter specifies the precision requirement for each colour component of the given *colour specifiers*, in the case that Colour Selection Mode is DIRECT. A positive value specifies the minimum number of bits required to represent the corresponding component of an RGB triple, and a value of zero specifies that the corresponding component is to be represented as specified by the Colour Precision Log2 Upper Bound entry in the Control State List.

The first integer of the *local colour precision requirement* parameter specifies the precision requirement for the given *colour specifiers*, in the case that the Colour Selection Mode is INDEXED. A positive value specifies the minimum number of bits required to specify a colour index, and a value of zero specifies that a colour index is to be represented as specified by the Colour Index Precision Log2 Upper Bound entry in the Control State List.

The pixels are combined with the drawing bitmap in the same manner as SOURCE DESTINATION BITBLT, where the pixel array colour specifiers are used as the source bitmap and the destination is the current drawing bitmap. The combination is formed using the specified *drawing mode* and *transparency* parameters. See 5.3.1 for definitions of the various drawing modes that may be used as values for the *drawing mode* parameter. Also see annex C.2.

NOTE – The aspect ratio of the rendered pixel array is dependent on the aspect ratio of the pixels, which may vary between devices.

Errors:

Error identifier: 3:603
Cause: Unsupported drawing mode / transparency requested
Reaction: At interpretation time, the default drawing mode and transparency is used.

Error identifier: 3:610
Cause: Local colour precision requirement not achievable
Reaction: Function ignored.

Error identifier: 6:602
Cause: Too many colour specifiers for PIXEL ARRAY.
Reaction: Function ignored.

5.4.2 GET PIXEL ARRAY

Parameters:

In	source bitmap identifier		BN
In	origin point		P
In	nx, ny	(1..n)	2I
In	x direction	(INCREASING VDC, DECREASING VDC)	E
In	y direction	(INCREASING VDC, DECREASING VDC)	E
In	local colour precision	(≥ 0)	3I
Out	response validity	(INVALID, VALID)	E
Out	pixel validity flag	(NONE, SOME, ALL)	E
Out	vx range	(0..n)	2I
Out	vy range	(0..n)	2I
Out	colour specifiers		dx*dy CO

Effect:

A rectangular array of pixels from the bitmap designated by the *source bitmap identifier* is returned to the client as a rectangular array of device-independent *colour specifiers*.

The *origin point* specifies the location in VDC space of the first pixel to be returned. The parameters *nx* and *ny* specify the number of colour specifiers for each row, and the number of rows respectively to return. The parameter *x direction* specifies whether successive pixels in a row are selected in the direction of increasing or decreasing VDC X. Similarly, *y direction* specifies whether successive rows of pixels are selected in the direction of increasing or decreasing VDC Y.

The *local colour precision* parameter specifies the precision of the returned *colour specifiers*. If the current Colour Selection Mode is DIRECT, the parameter specifies the number of bits representing each colour component of the returned *colour specifiers*. If the Colour Selection Mode is INDEXED, the first integer specifies the number of bits representing each colour index of the returned *colour specifiers*.

If the *colour specifiers* in the bitmap designated by the *source bitmap identifier* are in a different mode than the current Colour Selection Mode then the values of the returned *colour specifiers* are implementation-dependent.

GET PIXEL ARRAY only returns the values for requested pixels which actually exist in the *source bitmap*. Clipping is not applied and has no effect on this function.

The *pixel validity flag* indicates, in general terms, how many valid pixels are actually returned as colour specifiers at the local colour precision. If ALL, then all *nx* * *ny* requested pixels are returned and are valid. If SOME, then only a sub-rectangular array of the requested pixels are returned. If NONE, then no pixels are returned.

The parameters *vx range* and *vy range* specify the range within the requested *nx* * *ny* array which contained valid pixels. *vx range* specifies the starting and ending indices (sx and ex) for each row and *vy range* specifies the starting and ending rows (sy and ey). Note that the first index and row is numbered 0. If the pixel validity flag is ALL, then *vx range* = (0, nx-1) and *vy range* = (0, ny-1). If the pixel validity flag is NONE, then *vx range* = *vy range* = (0, 0). If the pixel validity flag is SOME, then *vx range* = (sx, ex) and *vy range* = (sy, ey) and the number of total returned pixels equals *dx* * *dy* where *dx* = (ex - sx + 1) and *dy* = (ey - sy + 1).

Errors:

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

Error identifier: 3:611
Cause: Local colour precision not achievable
Reaction: Function ignored.

5.4.3 GET PIXEL ARRAY DIMENSIONS

Parameters:

In	source bitmap identifier		BN
In	region		2P
In	local colour precision requirement	(≥ 0)	3I
Out	response validity	(INVALID, VALID)	E
Out	local colour precision	(≥ 0)	3I
Out	nx, ny	(1..n)	2I

Effect:

Determines the number and precision of bitmap pixels in x and y dimensions, *nx* and *ny* respectively, that are in the interior of the rectangular region defined by *region* in the bitmap identified by the *source bitmap identifier*. The *local colour precision* output parameter is a value appropriate for use in the function GET PIXEL ARRAY.

The *local colour precision requirement* parameter specifies the precision requirement for each colour component of the pixel array colour specifiers, in the case that Colour Selection Mode is DIRECT. A positive value specifies the minimum number of bits required to represent the corresponding component of an RGB triple, and a value of zero specifies that the corresponding component is to be represented as specified by the Colour Precision Log2 Upper Bound entry in the Control State List.

The first integer of the *local colour precision requirement* parameter specifies the precision requirement for the pixel array colour specifiers, in the case that the Colour Selection Mode is INDEXED. A positive value specifies the minimum number of bits required to specify a colour index, and a value of zero specifies that a colour index is to be represented as specified by the Colour Index Precision Log2 Upper Bound entry in the Control State List.

The value of *local colour precision* is the actual precision that each colour specifier in the bitmap would be returned, based on the specified value of the *local colour precision requirement*. The *local colour precision* parameter specifies the number of bits required to represent the corresponding component of an RGB triple in the case that the Colour Selection Mode is DIRECT. The first integer of the *local colour precision* parameter specifies the number of bits required to represent a colour index value in the case that the Colour Selection Mode is INDEXED.

The values of *nx* and *ny* returned are the total number of pixels required to fill the rectangle in each dimension. Clipping is not applied, even if the region extends beyond the VDC Extent of the drawing bitmap.

See 5.4.2

Errors:

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

Error identifier: 3:610
Cause: Local colour precision requirement not achievable
Reaction: Function ignored.

5.4.4 SOURCE DESTINATION BITBLT

Parameters:

In	source bitmap identifier		BN
In	source origin		P
In	destination origin		P
In	x offset		VDC
In	y offset		VDC
In	drawing mode	((-n..-1,1..n), (-n..n))	[IX,IX]
In	transparency	(OPAQUE, TRANSPARENT)	E

Effect:

This function is used to move a bitmap region of the identified source bitmap to the current drawing bitmap, (specified by the DRAWING BITMAP function), using the specified *drawing mode* and *transparency*. See 5.3.1 for definitions of the various drawing modes which may be used as values for the *drawing mode* parameter.

The bitmap regions in the source and destination bitmaps are defined by two points representing the diagonal corners of a rectangle. One corner in each bitmap is defined explicitly with the source/destination origin parameters. The other corner point in the source bitmap is found by adding the signed *x offset* and *y offset* parameters to the coordinates of the source origin point. The destination bitmap region will be the same size and orientation as the source bitmap region even if the two bitmaps have different VDC-to-Device Mappings (see figure 2). Should pixels of the source bitmap region lie outside of the source bitmap then truncation (as described in 3.5.2) may occur.

The source data is not affected by the transfer unless the source and destination bitmap regions overlap. In this case, the resulting source and destination bitmap regions will appear in the same way they would have had the source data first been copied to a temporary buffer and from there combined with the destination.

Table 2 describes the operations for all combinations of source and destination bitmap depth types:

Table 2 – Two-Operand Bitblt rules

Source Depth Type	Destination Depth Type	Operation
FULL DEPTH	FULL DEPTH	Normal operation (Note 1).
MAPPED	FULL DEPTH	Expand source bitmap, then apply "normal operation" (Note 1).
FULL DEPTH	MAPPED	Expand destination bitmap and conceptually combine with the source. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.
MAPPED	MAPPED	Expand both bitmaps and conceptually combine the pixel values. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.
NOTE – 1) "Normal operation" refers to the application of drawing mode and transparency rules to full-depth pixels.		

(See also annex C.2.)

Errors:

Error identifier: 3:603
Cause: Unsupported drawing mode / transparency requested
Reaction: At interpretation time, the default drawing mode and transparency is used.

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

5.4.5 TILE THREE OPERAND BITBLT

Parameters:

In	pattern bitmap identifier	BN
In	pattern region	2P
In	reference point	P
In	source bitmap identifier	BN

Raster operation functions **Abstract specification of functions**

In	source origin		P
In	destination origin		P
In	x offset		VDC
In	y offset		VDC
In	drawing-mode-3	((-n..-1,1..n), (-n..n))	[IX,IX]
In	transparency	(OPAQUE, TRANSPARENT)	E

Effect:

This function utilizes three bitmap regions and up to three separate bitmaps. The pattern, source and destination are combined in a manner specified by the *drawing-mode-3* and *transparency* parameters. The destination bitmap is the current drawing bitmap.

If *transparency* is TRANSPARENT, the pixels of the source bitmap region are used to determine transparency (see 5.2.8).

The *pattern region* specifies the location and extent of the pattern bitmap region in the pattern bitmap. The *reference point* specifies the location in the destination bitmap of the corner of the pattern bitmap region (see figure 3). The pattern conceptually extends as a tile over the entire destination VDC space. Pattern data is not affected by the transfer unless the pattern bitmap region overlaps the destination bitmap region. In this case, the result on the pattern and destination will be device-dependent. Some devices may be especially efficient in applying the pattern bitmap region if the size in pixels of the pattern is of a certain size in either or both x and y. The Preferred Bitblt Pattern Size entry in the Raster Description Table indicates if there is such a preferred pattern size. A value of one for either dimension indicates there is no particular size that is more efficient to use than another. If one or both of the values is greater than one, the pattern application is most efficient if its size in that dimension is the preferred size.

The source and destination bitmap regions are defined by two points representing the diagonal corners of a rectangle. One corner in each bitmap is defined explicitly with the *source origin* and *destination origin* parameters. The other corner point in the source bitmap is found by adding the signed *x offset* and *y offset* parameters to the coordinates of the *source origin* point. The destination bitmap region will be the same size and orientation as the source bitmap region even if the two bitmaps have different VDC-to-Device Mappings (see figure 2). Should pixels of the pattern bitmap region lie outside of the pattern bitmap or pixels of the source bitmap region lie outside of the source bitmap, then truncation (as described in 3.5.2) may occur.

The source data is not affected by the transfer unless the source and destination bitmap regions overlap. In this case, the resulting source and destination bitmap regions will appear in the same way they would have had the source data first been copied to a temporary buffer and from there combined with the destination.

Drawing-mode-3 is a compound data type parameter that defines the way the pattern, source, and destination are combined to form the result. The first component specifies a class of drawing modes, while the second component specifies a particular element within the class.

The only standardized class is:

1) BOOLEANOP: There are 256 possible boolean values for drawing-mode-3, which define all possible boolean combinations of three logical variables. An itemized list of all 256 values defined in terms of the boolean operations (AND, OR, XOR, and NOT) is provided in annex D.

The principle used is to define, for each of the eight possible combinations of three bits (pattern, source, destination) what the result will be. By specifying the eight single-bit values for the result, the boolean operation is uniquely defined.

P	S	D	Result
0	0	0	R7
0	0	1	R6
0	1	0	R5
0	1	1	R4
1	0	0	R3
1	0	1	R2
1	1	0	R1
1	1	1	R0

By reading $R_7..R_0$ as an unsigned 8-bit integer (i.e. R_7 is the most significant), the 256 possible boolean combinations can be uniquely defined by the hex values *00* through *FF*. For the boolean operations, each bit *n* of a pixel value can be determined algorithmically by applying the following formula:

$$\begin{aligned} D'_n = \; & R_7 * (-P_n * -S_n * -D_n) + R_6 * (-P_n * -S_n * D_n) + \\ & R_5 * (-P_n * S_n * -D_n) + R_4 * (-P_n * S_n * D_n) + \\ & R_3 * (P_n * -S_n * -D_n) + R_2 * (P_n * -S_n * D_n) + \\ & R_1 * (P_n * S_n * -D_n) + R_0 * (P_n * S_n * D_n) \end{aligned}$$

where, $R_7..R_0$ is the 8-bit drawing-mode-3 boolean result specification.

P_n, S_n, and D_n refer to the *nth* bit of the Pattern, Source, and initial Destination pixel values respectively.

* indicates logical AND
\+ indicates logical OR
\- indicates logical NOT

Class values above 1 and operation values above 255 are reserved for registration; negative values for these components are available for private use. Table 3 describes the combination rules for pattern, source and destination bitmap regions for the bitmap depth types:

Table 3 – Three-Operand BITBLT rules

Pattern Depth Type	Source Depth Type	Destination Depth Type	Operation
FULL DEPTH	FULL DEPTH	FULL DEPTH	Normal operattion (Note 1).
MAPPED	FULL DEPTH	FULL DEPTH	Expand the pattern and combine normally.
FULL DEPTH	MAPPED	FULL DEPTH	Expand the source and combine normally.
MAPPED	MAPPED	FULL DEPTH	Expand the pattern and source and combine normally.
FULL DEPTH	FULL DEPTH	MAPPED	Expand the mapped bitmap and then conceptually combine with the full-depth bitmaps. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.
MAPPED	FULL DEPTH	MAPPED	Expand the mapped bitmaps and then conceptually combine with the full-depth bitmap. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.
FULL DEPTH	MAPPED	MAPPED	Expand the mapped bitmaps and then conceptually combine with the full-depth bitmap. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.
MAPPED	MAPPED	MAPPED	Expand all three bitmaps and then conceptually combine the expanded values. Resulting pixels having the value of Mapped Bitmap Background Colour are written back into the destination as background pixels. All other pixel colours are written as foreground pixels.

NOTE –
1) "Normal operation" refers to the application of drawing mode-3 and transparency rules to full-depth pixels.

Errors:

Error identifier: 3:604
Cause: Unsupported drawing mode-3 / transparency requested
Reaction: At interpretation time, the default drawing-mode-3 and transparency are used.

Error identifier: 3:608
Cause: Bitmap identifier is not assigned to a bitmap
Reaction: Function ignored.

6 Raster inquiry functions

6.1 Introduction

This clause describes the abstract functional specification of the Raster Inquiry functions of the CGI.

The abstract names of these functions begin with INQUIRE and will only return values of one or more entries in a description table or state list.

Refer to ISO/IEC 9636-1, 5.2.7.

6.1.1 Data types employed

The abstract specifications of functions detail the functions in terms of input and output parameters. The data type of each parameter is selected from a standard set and is identified in the functional specification by a standard abbreviation. Both the data types and the abbreviations are taken from the complete list of data types given in ISO/IEC 9636-1, 5.2.10.

6.1.2 Validity of returned information

For all the inquiry functions specified in this clause, if any of the inquired information is available, the response validity flag is returned as VALID and the values specified in the output parameters are returned. In the case of a VALID response, if there is any possibility that any of the individual parameters within the response are not valid, then the response itself will contain additional (always valid) parameters which indicate which of the other returned parameters are valid.

If the inquired information is not available or the inquiry function is unsupported, the response validity flag is returned as INVALID and the specified output parameters are undefined. No other meaning should be applied to these other output parameters.

6.2 Raster description table

6.2.1 INQUIRE RASTER CAPABILITY

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	number of predefined displayable bitmaps	(1..n)	I
Out	displayable bitmap creation supported	(NO, YES)	E
Out	bitmap formats supported	(FULL DEPTH, MAPPED AND FULL DEPTH)	E
Out	number of bits per full depth pixel	(1..n)	I
Out	drawing mode/transparency support	(NONE, SOME, ALL)	E
Out	drawing mode-3/transparency support	(NONE, SOME, ALL)	E
Out	number of supported bitmap mode combinations	(1..3)	I
Out	array of supported bitmap mode combinations	(INDEXED, DIRECT, MIXED)	3E
Out	size of pixel	(> 0)	2R
Out	preferred bitblt pattern size	(1..n)	2I
Out	source bitmap truncation capability	(NOT TRUNCATED, TRUNCATED)	E
Out	previous display bitmap data	(CLEARED, PRESERVED)	E

Effect:

See 6.1.2.

6.2.2 INQUIRE LIST OF SUPPORTED DRAWING-MODE/TRANSPARENCY PAIRS

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of list elements in description table	(0..n)	I
Out	list of supported pairs	((-n..-1,1..n), (-n..n), (OPAQUE, TRANSPARENT))	n[2IX,E]

Effect:

See 6.1.2.

6.2.3 INQUIRE LIST OF SUPPORTED DRAWING-MODE-3/TRANSPARENCY PAIRS

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of list elements in description table	(0..n)	I
Out	list of supported pairs	((-n..-1,1..n), (-n..n), (OPAQUE, TRANSPARENT))	n[2IX,E]

Effect:

See 6.1.2.

6.3 Raster state list

6.3.1 INQUIRE RASTER STATE

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	display bitmap identifier		BN
Out	drawing bitmap identifier		BN
Out	drawing mode	((-n..-1,1..n), (-n..n))	[IX,IX]
Out	selection mode of mapped bitmap foreground colour	(INDEXED, DIRECT)	E
Out	mapped bitmap foreground colour		CO
Out	selection mode of mapped bitmap background colour	(INDEXED, DIRECT)	E
Out	mapped bitmap background colour		CO
Out	selection mode of transparent colour	(INDEXED, DIRECT)	E
Out	transparent colour		CO

Effect:

See 6.1.2.

6.3.2 INQUIRE LIST OF NON-DISPLAYABLE BITMAP IDENTIFIERS

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E

Out	total number of list elements in state list	(0..n)	I
Out	list of bitmap identifiers		nBN

Effect:
See 6.1.2.

6.3.3 INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of list elements in state list	(0..n)	I
Out	list of bitmap identifiers		nBN

Effect:
See 6.1.2.

6.4 Bitmap state list

6.4.1 INQUIRE BITMAP STATE

Parameters:

In	bitmap identifier		BN
Out	response validity	(INVALID, VALID)	E
Out	depth type	(MAPPED, FULL DEPTH)	E
Out	displayability	(NON DISPLAYABLE, DISPLAYABLE)	E
Out	bitmap mode	(INDEXED, DIRECT, MIXED)	E
Out	bottom left pixel		DP
Out	top right pixel		DP
Out	VDC extent		2P
Out	isotropy	(NOT FORCED, FORCED)	E
Out	horizontal alignment	(LEFT, CENTRE, RIGHT)	E
Out	vertical alignment	(BOTTOM, CENTRE, TOP)	E
Out	specification mode of current device viewport	(FRACTION OF DISPLAY SURFACE, MILLIMETRES WITH SCALE FACTOR, PHYSICAL DEVICE COORDINATES)	E
Out	metric scale factor of current device viewport	(>0)	R
Out	requested device viewport		2VP
Out	effective viewport		2VP
Out	drawing surface clip indicator	(OFF, DSCRECT, VIEWPORT)	E
Out	drawing surface clip rectangle		2VP
Out	specification mode of drawing surface clip rectangle	(FRACTION OF DISPLAY SURFACE, MILLIMETRES WITH SCALE FACTOR, PHYSICAL DEVICE COORDINATES)	E
Out	metric scale factor of drawing surface clip rectangle	(>0)	R

Effect:
See 6.1.2.

Errors:

Error identifier:	3:608
Cause:	Bitmap identifier is not assigned to a bitmap
Reaction:	Function ignored.

7 Raster description tables and state lists

This clause contains the description tables and state lists that define the portion of the Virtual Device relating to Raster functionality.

If this part of ISO/IEC 9636 allows latitude for certain description table entries, the preferred entries are designated by the entry being underlined.

The information in the tables and lists is available to a client by means of inquiry functions. The abstract specifications of these functions are found in clause 6.

7.1 Raster description table

This describes the characteristics and capabilities of the Virtual Device, with respect to the raster functionality.

Table 4 – Raster Description Table

Entry	Possible Values	Data
Raster Capability:		
Number of Predefined Displayable Bitmaps (N)	(1..n)	I
Displayable Bitmap Creation Supported	(NO, YES)	E
Bitmap Formats Supported	(FULL DEPTH, MAPPED AND FULL DEPTH)	E
Number of Bits Per Full Depth Pixel	(1..n)	I
Drawing-Mode/Transparency Support	(NONE, SOME, ALL)	E
Drawing-Mode-3/Transparency Support	(NONE, SOME, ALL)	E
Number of Supported Bitmap Mode Combinations	(1..3)	I
Array of Supported Bitmap Mode Combinations	(INDEXED, DIRECT, MIXED)	E
Size of Pixel	(>0) (Note 1)	2R
Preferred Bitblt Pattern Size	(1..n) (Note 2)	2I
Source Bitmap Truncation Capability	(NOT TRUNCATED, TRUNCATED)	E
Previous Display Bitmap Data	(CLEARED, PRESERVED)	E
Supported Drawing-Mode/Transparency Pairs:		
List of Drawing-Mode/Transparency Pairs containing:		
Drawing Mode Class	(-n..-1,1..n)	IX
Class Element	(-n..n)	IX
Transparency	(OPAQUE, TRANSPARENT)	E
Supported Drawing-Mode-3/Transparency Pairs:		
List of Drawing-Mode-3/Transparency Pairs containing:		
Drawing Mode Class	(-n..-1,1..n)	IX
Class Element	(-n..n)	IX
Transparency	(OPAQUE, TRANSPARENT)	E

NOTES

1) Size in millimetres.

2) First value is x length, second is y width. Both represent numbers of pixels.

7.2 State lists

7.2.1 Raster state list

This describes the current state (and default values) of those values that can be set by the raster control and attribute functions.

Table 5 – Raster State List

Entry	Possible Values	Data Type	Default
Raster State:			
Display Bitmap Identifier	(imp.dep, Note 1)	BN	(imp.dep, Note 2)
Drawing Bitmap Identifier	(imp.dep, Note 1)	BN	(imp.dep, Note 2)
Drawing Mode containing:			
Drawing Mode Class	(-n..-1,1..n)	IX	1
Class Element	(-n..n)	IX	3
Selection Mode of Mapped Bitmap Foreground Colour	(INDEXED, DIRECT)	E	INDEXED
Mapped Bitmap Foreground Colour		CO	1
Selection Mode of Mapped Bitmap Background Colour	(INDEXED, DIRECT)	E	INDEXED
Mapped Bitmap Background Colour		CO	0
Selection Mode of Transparent Colour	(INDEXED, DIRECT)	E	INDEXED
Transparent Colour		CO	0
Non-Displayable Bitmap Identifiers:			
List of Bitmap Identifiers	(Note 1)	nBN	empty
Displayable Bitmap Identifiers:			
List of Bitmap Identifiers		nBN	(Note 3)

NOTES
1) The range of values of the data type BN is defined by the binding or encoding employed by the implementation. A subrange of N bitmap identifiers is reserved for predefined displayable bitmaps (where N is defined in the Raster Description Table).
2) The default for the Display Bitmap Identifier and Drawing Bitmap Identifier shall be the same.
3) A list of bitmap identifiers associated with N predefined displayable bitmaps.

7.2.2 Bitmap state list

A copy of the Bitmap State List is created for each predefined displayable bitmap when the CGI Virtual Device is initialized. A copy of the Bitmap State List is created on each occasion the function CREATE BITMAP is invoked.

Table 6 – Bitmap State List

Entry	Possible Values	Data Type	Default
Bitmap State:			
Depth Type	(MAPPED, FULL DEPTH)	E	(Note 1)
Displayability	(NON DISPLAYABLE, DISPLAYABLE)	E	(Note 2)
Bitmap Mode	(INDEXED, DIRECT, MIXED)	E	(imp.dep)
Bottom Left Pixel		DP	(Note 6)
Top Right Pixel		DP	(Note 6)
VDC-to-Device Mapping:			
VDC Extent		2P	(Note 7)
Isotropy	(NOT FORCED, FORCED)	E	(Note 3)
Horizontal Alignment	(LEFT, CENTRE, RIGHT)	E	(Note 3)

Table 6 – Bitmap State List — (concluded)

Entry	Possible Values	Data Type	Default
Vertical Alignment	(BOTTOM, CENTRE, TOP)	E	(Note 3)
Specification Mode of Current Device Viewport (Note 5)	(FRACTION OF DISPLAY SURFACE, MILLIMETRES WITH SCALE FACTOR, PHYSICAL DEVICE COORDINATES)	E	(Note 3)
Metric Scale Factor of Current Device Viewport	(> 0)	R	1.0
Requested Device Viewport		2VP	(Note 3,4)
Effective Viewport		2VP	(Note 3,4)
Drawing Surface Clipping:			
Drawing Surface Clip Indicator	(OFF, DSCRECT, VIEWPORT)	E	VIEWPORT
Drawing Surface Clip Rectangle		2VP	(Note 3)
Specification Mode of Drawing Surface Clip Rectangle (Note 5)	(FRACTION OF DISPLAY SURFACE, MILLIMETRES WITH SCALE FACTOR, PHYSICAL DEVICE COORDINATES)	E	(Note 3)
Metric Scale Factor of Drawing Surface Clip Rectangle	(> 0)	R	1.0

NOTES

1) Defined at CGI initialization time for predefined displayable bitmaps to be FULL DEPTH. Defined at CREATE BITMAP time for client-defined bitmaps.
2) Defined at CGI initialization time for predefined displayable bitmaps to be DISPLAYABLE. Defined at CREATE BITMAP time for client-defined bitmaps.
3) Defined at CGI initialization time for predefined displayable bitmaps to be the default values for VDC-to-Device Mapping in the Control State List. For client-defined bitmaps the state list of the current drawing bitmap are used to determine these values at CREATE BITMAP time.
4) The entire bitmap in the same device viewport specification mode units as the current Drawing Bitmap at bitmap creation time is used.
5) Applies to both the requested device viewport and the effective viewport.
6) Defined at CGI initialization time for predefined displayable bitmaps to be equal to the values of the Display Surface Bottom Left-Corner and Display Surface Top-Right Corner entries in the Output Device Description Table defined in ISO/IEC 9636-2. Defined at CREATE BITMAP time for client-defined bitmaps. May be subject to spontaneous change for displayable bitmaps (see the Output Device Description Table in ISO/IEC 9636-2). Non-displayable bitmaps are not subject to spontaneous change.
7) Defined when the CGI Virtual Device is initialized as the default values of the VDC Extent in the Control State List. Otherwise, the values of the bitmap extent parameter at bitmap creation time.

Annex A

(normative)

Formal grammar of the functional specification

A.1 Introduction

This grammar is a formal definition of the Raster portion of standard CGI syntax. It shows all productions regardless of the encoding scheme. The terminal symbols correspond to the CGI basic abstract data types. Encoding and representation details of these can be found in ISO/IEC 9637.

A.2 Notation used

<symbol>	- nonterminal
<SYMBOL>	- terminal
<symbol>*	- 0 or more occurrences
<symbol>+	- 1 or more occurrences
<symbol>o	- 0 or 1 occurrences
<symbol>(n)	- exactly n occurrences; n=2,3,...
<symbol-1> ::= <symbol-2>	- symbol-1 has the syntax of symbol-2
<symbol-1> | <symbol-2>	- symbol-1 or alternatively symbol-2
<symbol: meaning>	- symbol with the stated meaning
comment	- explanation of a symbol or a production
returned: <symbol>	- output parameter(s)

A.3 Detailed grammar

```
<part 6 function>   ::=   <raster operation functions>
                      |   <raster attribute functions>
                      |   <raster control functions>
                      |   <raster inquiry functions>
```

A.4 Raster operation functions

```
<raster operation functions>  ::=   <pixel array>
                                |   <get pixel array>
                                |   <get pixel array dimensions>
                                |   <source destination bitblt>
                                |   <tile three operand bitblt>

<pixel array>   ::=   <PIXEL ARRAY>
                      <point: origin point>
                      <BDT_INTEGER: nx, ny>(2)
                      <BDT_INTEGER: x scale, y scale>(2)
                      <vdc direction: x, y>(2)
                      <drawing mode pair>
```

<transparency>
<local colour prec requirement>
<colour specifier>*

<point> ::= <vdc value>(2)

<vdc value> ::= <BDT_REAL> | <BDT_INTEGER>

<vdc direction: enumerated> ::= <INCREASING VDC> | <DECREASING VDC>

<colour specifier> ::= <BDT_COLOUR_INDEX> | <BDT_COLOUR_DIRECT>

<drawing mode pair> ::= <BDT_INDEX: drawing mode class>
<BDT_INDEX: drawing mode operation>

<transparency: enumerated> ::= <OPAQUE> | <TRANSPARENT>

<get pixel array> ::= <GET PIXEL ARRAY>
<BDT_CSN_VALUE: source bitmap identifier>
<point: origin point>
<BDT_INTEGER: nx, ny>(2)
<vdc direction: x, y>(2)
<local colour prec>
returned:
<response validity>
<pixel validity flag>
<BDT_INTEGER: vx, vy range>(2)
<colour specifier>*

<local colour prec> ::= <BDT_INTEGER>(3)

<local colour prec requirement> ::= <BDT_INTEGER>(3)

<response validity> ::= <validity flag>

<validity flag: enumerated> ::= <INVALID> | <VALID>

<pixel validity flag: enumerated> ::= <NONE> | <SOME> | <ALL>

<get pixel array dimensions> ::= <GET PIXEL ARRAY DIMENSIONS>
<BDT_CSN_VALUE: source bitmap identifier>
<point: region>(2)
<local colour prec requirement>
returned:
<response validity>
<local colour prec>
<BDT_INTEGER: nx, ny>(2)

<source destination bitblt> ::= <SOURCE DESTINATION BITBLT>
<BDT_CSN_VALUE: source bitmap identifier>
<point: source, destination origin>(2)
<point: reference point>
<vdc value: x, y offset>(2)
<drawing mode pair>
<transparency>

<tile three operand bitblt> ::= <TILE THREE OPERAND BITBLT>
<BDT_CSN_VALUE: pattern bitmap identifier>
<point: pattern region>(2)
<BDT_CSN_VALUE: source bitmap identifier>
<point: source, destination origin>(2)
<vdc value: x, y offset>(2)

<drawing mode-3 pair>
<transparency>

<drawing mode-3 pair> ::= <BDT_INDEX: drawing mode-3 class>
<BDT_INDEX: drawing mode-3 operation>

A.5 Raster attribute functions

<raster attribute function> ::= <DRAWING MODE>
<drawing mode pair>
| <FILL BITMAP>
<BDT_CSN_VALUE: bitmap identifier>
<point: pattern bitmap region>(2)

A.6 Raster control functions

<raster control function> ::= <GET NEW BITMAP IDENTIFIER>
returned:
<response validity>
<BDT_CSN_VALUE: bitmap identifier>
| <CREATE BITMAP>
<BDT_CSN_VALUE: bitmap identifier>
<point: bitmap extent>(2)
<depth type>
<displayability>
| <DELETE BITMAP>
<BDT_CSN_VALUE: bitmap identifier>
| <DRAWING BITMAP>
<BDT_CSN_VALUE: bitmap identifier>
| <DISPLAY BITMAP>
<BDT_CSN_VALUE: bitmap identifier>
| <MAPPED BITMAP FOREGROUND COLOUR>
<colour specifier>
| <MAPPED BITMAP BACKGROUND COLOUR>
<colour specifier>
| <TRANSPARENT COLOUR>
<colour specifier>

<depth type: enumerated> ::= <MAPPED> | <FULL DEPTH>

<displayability: enumerated> ::= <NON DISPLAYABLE> | <DISPLAYABLE>

A.7 Raster inquiry functions

<raster inquiry function> ::= <inquire raster capability>
| <inquire list of supported drawing mode/transparency pairs>
| <inquire list of supported drawing mode 3/transparency pairs>
| <inquire raster state>
| <inquire list of non-displayable bitmaps>
| <inquire list of displayable bitmaps>
| <inquire bitmap state>

<inquire raster capability> ::= <INQUIRE RASTER CAPABILITY>
returned:
<response validity>
<BDT_INTEGER: number of predefined displayable bitmaps>
<yes no flag: displayable bitmap creation support>
<depth types supported>
<BDT_INTEGER: number of bits per full depth pixel>
<drawing mode support>(2)
<BDT_INTEGER: number of supported bitmap mode combinations>
<bitmap mode: supported combinations>(3)
<BDT_REAL: size of pixel>(2)
<BDT_INTEGER: preferred bitblt pattern size>(2)
<bitmap truncation support>
<previous display bitmap data>

<depth types supported: enumerated> ::= <FULL DEPTH> | <MAPPED AND FULL DEPTH>

<drawing mode support: enumerated> ::= <NONE> | <SOME> | <ALL>

<previous display bitmap data: enumerated> ::= <CLEARED> | <PRESERVED>

<bitmap mode: enumerated> ::= <INDEXED> | <DIRECT> | <MIXED>

<yes no flag: enumerated> ::= <NO> | <YES>

<bitmap truncation support: enumerated> ::= <NOT TRUNCATED> | <TRUNCATED>

<inquire list of supported drawing mode/transparency pairs> ::=
<INQUIRE LIST OF SUPPORTED DRAWING-MODE/TRANSPARENCY PAIRS>
<BDT_INTEGER: number of list elements requested>
<BDT_INTEGER: index within list>
returned:
<response validity>
<BDT_INTEGER: total number of list elements in description table>
<drawing mode / transparency pair>*

<drawing mode / transparency pair> ::= <drawing mode pair>
<transparency>

<inquire list of supported drawing mode 3/transparency pairs> ::=
<INQUIRE LIST OF SUPPORTED DRAWING-MODE-3/TRANSPARENCY PAIRS>
<BDT_INTEGER: number of list elements requested>
<BDT_INTEGER: index within list>
returned:
<response validity>
<BDT_INTEGER: total number of list elements in description table>
<drawing mode-3 / transparency pair>*

<drawing mode-3 / transparency pair> ::= <drawing mode-3 pair>
<transparency>

<inquire raster state> ::= <INQUIRE RASTER STATE>
returned:
<response validity>
<BDT_CSN_VALUE: display and drawing bitmaps>(2)
<drawing mode pair>
<colour selection mode: foreground>
<colour specifier: mapped bitmap foreground colour>
<colour selection mode: background>
<colour specifier: mapped bitmap background colour>

<colour selection mode: transparent>
<colour specifier: transparent colour>

<colour selection mode: enumerated>::= <INDEXED> | <DIRECT>

<inquire list of non-displayable bitmaps> ::=
<INQUIRE LIST OF NON-DISPLAYABLE BITMAP IDENTIFIERS>
<BDT_INTEGER: number of list elements requested>
<BDT_INTEGER: index within list>
returned:
<response validity>
<BDT_INTEGER: total number of list elements in state list>
<BDT_CSN_VALUE: bitmap identifier>*

<inquire list of displayable bitmaps> ::=
<INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS>
<BDT_INTEGER: number of list elements requested>
<BDT_INTEGER: index within list>
returned:
<response validity>
<BDT_INTEGER: total number of list elements in state list>
<BDT_CSN_VALUE: bitmap identifier>*

<inquire bitmap state>::= <INQUIRE BITMAP STATE>
<BDT_CSN_VALUE: bitmap identifier>
returned:
<response validity>
<depth type>
<displayability>
<bitmap mode>
<device point: bottom left, top right>(2)
<point: VDC extent>(2)
<isotropy>
<horizontal alignment>
<vertical alignment>
<viewport specification mode>
<BDT_REAL: metric scale factor>
<point: requested device viewport>(2)
<point: effective viewport>(2)
<drawing surface clip indicator>
<point: drawing surface clip rectangle>(2)
<viewport specification mode: drawing surface clip>
<BDT_REAL: metric scale factor of drawing surface clip>

<device point> ::= <fixed_integer>(2)

<viewport specification mode: enumerated> ::= <FRACTION OF DISPLAY SURFACE>
| <MILLIMETRES WITH SCALE FACTOR>
| <PHYSICAL DEVICE COORDINATES>

<isotropy: enumerated> ::= <NOT FORCED> | <FORCED>

<horizontal alignment: enumerated> ::= <LEFT> | <CENTRE> | <RIGHT>

<vertical alignment: enumerated> ::= <BOTTOM> | <CENTRE> | <TOP>

<drawing surface clip indicator: enumerated>::= <OFF> | <DSCRECT> | <VIEWPORT>

<fixed_integer> ::= <BDT_FIXED_INTEGER_8>
| <BDT_FIXED_INTEGER_16>
| <BDT_FIXED_INTEGER_32>

A.8 Terminal symbols

The following are the terminals in this grammar. They correspond directly to the CGI basic abstract data types. Their representation depends on the encoding scheme used. Encoding and representation details can be found in ISO/IEC 9637.

<BDT_COLOUR_DIRECT>
<BDT_COLOUR_INDEX>
<BDT_CSN_VALUE>
<BDT_FIXED_INTEGER_8>
<BDT_FIXED_INTEGER_16>
<BDT_FIXED_INTEGER_32>
<BDT_INDEX>
<BDT_INTEGER>
<BDT_REAL>

The enumerated values used in this part of ISO/IEC 9636 are as follows:

<ADDITIVEOP>
<ALL>
<BOOLEANOP>
<BOTTOM>
<CENTRE>
<CLEARED>
<COMPARATIVEOP>
<DECREASING VDC>
<DIRECT>
<DISPLAYABLE>
<DSCRECT>
<FORCED>
<FRACTION OF DISPLAY SURFACE>
<FULL DEPTH>
<INCREASING VDC>
<INDEXED>
<INVALID>
<LEFT>
<MAPPED>
<MAPPED AND FULL DEPTH>
<MIXED>
<MILLIMETRES WITH SCALE FACTOR>
<NO>
<NONE>
<NOT DISPLAYABLE>
<NOT FORCED>
<NOT TRUNCATED>
<OFF>
<OPAQUE>
<PHYSICAL DEVICE COORDINATES>
<PRESERVED>
<RIGHT>
<SOME>
<TOP>

<TRANSPARENT>
<TRUNCATED>
<VALID>
<VIEWPORT>
<YES>

The CGI opcodes are encoding dependent. A list of those defined in this part of ISO/IEC 9636 can be found in the productions for <part 6 function name> below:

<part 6 function name> ::= <CREATE BITMAP>
| <DELETE BITMAP>
| <DISPLAY BITMAP>
| <DRAWING BITMAP>
| <DRAWING MODE>
| <FILL BITMAP>
| <GET NEW BITMAP IDENTIFIER>
| <GET PIXEL ARRAY DIMENSIONS>
| <GET PIXEL ARRAY>
| <INQUIRE BITMAP STATE>
| <INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS>
| <INQUIRE LIST OF NON-DISPLAYABLE BITMAP IDENTIFIERS>
| <INQUIRE LIST OF SUPPORTED DRAWING-MODE-3/TRANSPARENCY PAIRS>
| <INQUIRE LIST OF SUPPORTED DRAWING-MODE/TRANSPARENCY PAIRS>
| <INQUIRE RASTER CAPABILITY>
| <INQUIRE RASTER STATE>
| <MAPPED BITMAP BACKGROUND COLOUR>
| <MAPPED BITMAP FOREGROUND COLOUR>
| <PIXEL ARRAY>
| <SOURCE DESTINATION BITBLT>
| <TILE THREE OPERAND BITBLT>
| <TRANSPARENT COLOUR>