OPEN ABAL VISUAL LIBRARY

Programmers Reference Manual Version 5.1a

Abstract

This document provides an exhaustive programmer's reference to the Abal Visual Dynamic Library for the creation of Graphical User Interfaces.

Jamie Marshall

ijm@amenesik.com

Table of Contents

Introduction

The Visual Dynamic Library was added to the Abal Language which provides a powerful collection of APIs for the use of the underlying portable graphical engine GIGO. This compliments the more traditional textual interface CICO, of the Abal Runtime EXA. This library was designed to facilitate the creation of graphical user interfaced business applications that could be easily ported between Windows and Linux systems.

This document provides an exhaustive programmer's reference manual describing the various functions and features that this dynamic library provides. Extensive examples can be found after the reference section.

Overview

The Abal language provides a powerful collection of screen management and display functions in the form of the PRINT, ASK, OP, KBF, CONF and EVENT instructions. To better understand the use of these instructions it helps to understand the different sub systems through which they operate. The CICO and GIGO sun systems will be described later in this document.

Graphical Application Design

The development of a user-friendly and welcoming business application work environment requires a minimum amount of thought and preparation to ensure that the resulting user interface is consistent, not only in its appearance, but also in its behaviour. The Abal Visual Library provides a collection of coherent functions which facilitate not only the preparation phase but also the realisation of the various application dialog components comprising the global user interface.

The following aspects are of particular importance and need to be decided prior to engaging in the actual development process.

Colour Schemes

The preparation or selection of a suitable Visual Palette comprising the collection of color values that are appropriate to the subject, user mood, or even daytime usage of the application.

Element Styles

The preparation or selection of a collection of suitable visual styles for each of the different user interface controls, ensuring consistency not only of appearance but also behavioural aspects.

Logos and Images

The preparation or selection of the collection of logos, textures and background images that are required on support of the resulting style, and in accordance with an eventual variety of palleted colour schemes.

Text Fonts

The preparation or selection of a collection of suitable text fonts or a variety of sizes with both fixed and proportional spacing, again in accordance with the nature and quantity of information to be displayed via the corresponding element styles.

Operational Mode

The way in which the application will operate, either as **event driven** "in response to" or **menu driven** "in control of" its user interface components, will depend to a great extent on the nature of the application itself and the actual problems that it is designed to solve.

Menu Driven Design

This design pattern is suitable for a standard business application approach to subjects such as stock control, accountancy, and customer relation management. A structured collection and sequence of nested menus, and their options, allows the user to rapidly gain access to specific functionalities such as invoice creation, customer support or product creation.

The preparation of the list of functionalities and the organisation of their efficient access, is of particular importance and has been a domain of predilection in which traditional ABAL application development has been particularly suitable.

Event Driven Design

This design pattern is particularly prevalent and suitable for the development of modern, end user, application tools. These include spread sheets, word processers, web browsers, and other domain or subject specific tools. The widespread adoption of object-oriented programming technics has facilitated the development of such complex event driven scenarios. These applications usually afford immediate and often permanent access through their central work environment, assisted by both event driven extensions and global and contextual menu systems, allowing direct access to underlying functionalities.

Hybrid Design Pattern

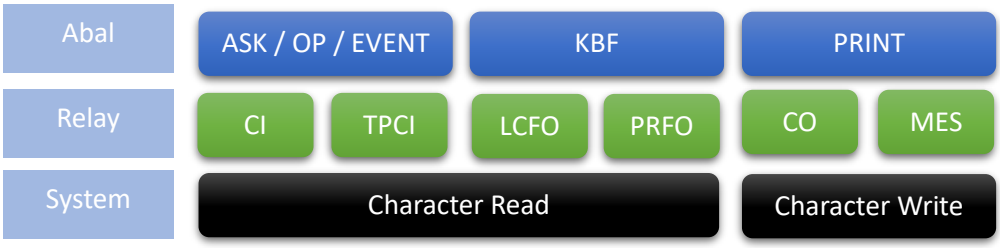
The two preceding design patterns can be successfully combined for the development of business applications. The menu driven design allows primary navigation of functional subjects with the event driven approach being employed for the development of the tools required for the different functional domains.

Abal Visual Applications

The Abal Visual Library provides all the functionality required for the efficient manual development of complex business applications of all three afore mentioned design patterns, though the use of the Abal Visual Application Design Tool, known as SING, is strongly recommended.

CICO

The portable textual screen interface of Abal, known as CICO, for Console Input Console Output, provides a collection of text-character based input output functions that are made available for use by the Abal programmer via the PRINT, ASK, OP, KBF, CONF and EVENT instructions. Each Abal Runtime environment comprises a CICO sub section which is adapted and optimised for operation on the corresponding host operating system.



The CI, TPCI, LCFO, PRFO, CO, and MES relays provide a generic abstraction for the ABAL instructions to access the underlying terminal management, through the corresponding system dependent read and write calls. Since all PRINT output is directed through the CO and MES relays, the overloading of these relays can be performed by ABAL visualisation libraries, such as ASFUN and VISUAL, facilitating the seamless integration of their enhanced windowing and graphical functionalities.

The configuration of the CICO sub system is performed through the CICO environment variable which comprises the following four fields:

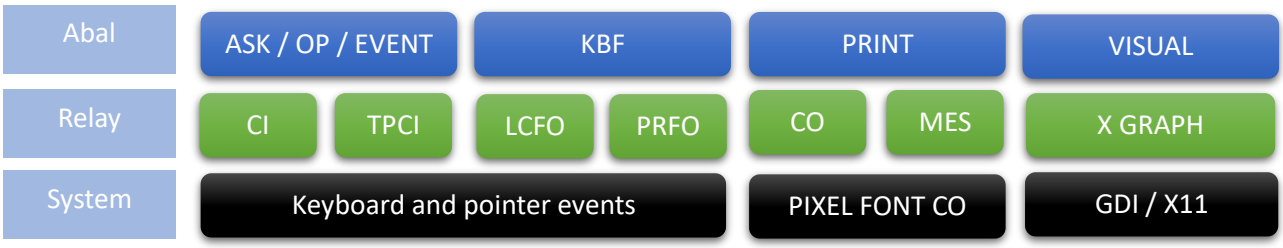
- Terminal emulation parameter file name
- The full path name to the parameter file
- Escape Sequence timer expressed in seconds.
- Default colour configuration comprising three hexadecimal digits representing, in order, the ABAL colour codes for the default text colour, the default background colour and the default delete colour.

```
$ export CICO=vt100:/home/abal64/cico:1:F00
```

The preceding CICO environment variable definition shows the configuration of use of a VT100 terminal emulation, retrieved from the **/home/abal64/cico** folder, with an ESC interpretation delay of 1 second and the default colours of white foreground, black background, and black default fill.

GIGO

The portable graphical screen interface, known as GIGO, for Graphical Input Graphical Output, provides an equivalent collection of graphical character-font based input output functions, equally available to the Abal programmer via the corresponding text instructions. Each Abal Runtime environment comprises a GIGO sub system which is adapted and optimised for the corresponding operating system.



With the addition of the GIGO subsystem, an extra relay was added to provide the abstraction for the graphical operations. The other relays are overloaded and connected through to the graphical CI and CO emulation using the underlying PIXEL font management and operating keyboard and pointer device event management.

The configuration of the GIGO sub system is performed through the GIGO environment variable which comprises the following four fields:

- Graphical Resolution in mode or explicit screen dimensions
- Standard base font filename and path
- Standard colour palette filename and path
- Default colour configuration

```
$ export GIGO=3:/home/abal64/fonts/standard.fmf:/home/abal64/images/standard.rgb:F00
```

This first GIGO environment variable definition shows the configuration of use of the VESA display mode 3 (800x600), the standard text font and colour palette, and sets the default colours of white foreground, black background, and black delete fill.

```
$ export GIGO==2000,1000:/home/abal64/fonts/standard.fmf:/home/abal64/images/standard.rgb:F0F
```

The second GIGO environment variable definition shows the configuration of use of the specific graphical resolution of 2000x1000, the standard text font and colour palette and sets the default colours of white foreground, black background, and white delete fill.

The activation of the GIGO subsystem is controlled by the presence and value of the GRAPHON and GRAPHOFF environment variables. If either is subsequently positioned, then its value will produce the corresponding effect on the GIGO sub system, if this is correctly configured in terms of the GIGO variable, and any other system specific environment variables such as the X/11 DISPLAY variable.

VISUAL

The VISUAL library was designed to facilitate the development of portable, interactive, graphical user interfaces for the ABAL language.

The VISUAL library overloads the X GRAPH relay that is subsequently used for the pure pixel-oriented functions that it provides.

The functions were designed for use by both manual programmers and users of the ABAL graphical user interface generation tool, known as SING.

The VISUAL library functions are designed and intended for the construction of a graphical user interfaced application comprising a collection of dialog boxes or Visual Frames. Each of these frames are usually autonomous but may also be combined in the construction of complex multi-paged dialog boxes.

Library Functions

This section of this documentation will describe each of the functions of the Abal Visual Dynamic Library, describing the objectives, parameters, and all other pertinent information. The dynamic library interface description is provided in the “visual.def” file which must be included prior to the PROGRAM, MODULE or PROGRAM OVERLAY instruction to allow access to the visual library functions.

VisualZone

Syntax

```
E% = VisualZone(x%, y%, w%, h%, type%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the zone.
Y	The Y pixel position of the top left corner of the zone.
W	The overall width of the zone in pixels.
H	The overall height of the zone in pixels.
Type	The type of the zone to be drawn as described below.

Description

This function is to be used for the display of a graphical frame of the type specified by the final parameter, that fits exactly within the area defined by the parameters **x**%, **y**%, **w**% and **h**%.

Clipping will be performed if either the start or end coordinates are outside of the current graphical visualisation space.

The value of the **type** parameter must be one of the following values:

Type	Description	Value
_OUTSET_FRAME	will display an outset relief frame using the standard highlight and lowlight colours of the palette.	0
_INSET_FRAME	will display an inset relief frame using the standard lowlight and highlight colours of the palette.	1
_GROOVE_FRAME	will display a groove relief frame using the standard highlight and lowlight colours of the palette.	2
_RIDGE_FRAME	will display a ridge relief frame using the standard highlight and lowlight colours of the palette.	3
_EDIT_FRAME	Will display a standard edit box frame using the standard black and standard white colours of the palette.	4

VisualLine

Syntax

```
E% = VisualLine(x1%, y1%, x2%, x3%, thickness%, pixel%)
```

Parameters

Parameter Name	Description
X1	The X pixel position of the start of the line.
Y1	The Y pixel position of the start of the line.
X2	The X pixel position of the end of the line.
Y2	The Y pixel position of the end of the line.
Thickness	The thickness of the line in pixels.
Pixel	The colour value to be used to draw the line.

Description

This function will draw a straight line from the graphical point coordinate described by the parameters X1, Y1 to the graphical point coordinate described by the parameters X2, Y2. Clipping will be performed if either the start or end coordinates are outside of the current graphical visualisation space.

VisualFont

Syntax

```
E% = VisualFont(filename$, length%)
```

Parameters

Parameter Name	Description
Filename$	This string parameter provides the full path prefix name of the font file to be loaded.
Length%	The usable length in characters of the font file name parameter.

Description

This function will load the graphical font from the disk file identified by the full path and file name provided by the value of the first parameter. The length parameter determines the useable length of the font file name parameter string.

This function will return the number of the font slot into which the font will have been loaded and should be used in subsequent visual functions where a font identifier parameter is required.

VisualText

Syntax

```
E% = VisualText(x1%, y1%, w%, h%, fontid%, fg%, bg%, message$, length%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the text zone.
Y	The Y pixel position of the top left corner of the text zone.
W	The width of the text zone in pixels.
H	The height of the text zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
Options	Text display options as described below.

Description

This function will draw a graphical textual message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **options** parameter influences the way in which the text message will be displayed as shown below:

Option	Description	Value
Align Left	Aligns the text with the left edge of the graphical zone.	2
Align Right	Aligns the text with the right edge of the graphical zone.	1
Align Center	The text will be displayed centred within the graphical zone.	3
Justify	Justifies the text within the graphical zone adding spaces to ensure the text fills the zone completely.	4
Bold	The text will be displayed using a bold font.	/0100
Line	The text will be displayed underlined.	/0200
Shadow	The text will be displayed using an appropriate shadow colour.	/0400
Bar	The text will be displayed overlined.	/0800
Strike	The text will be displayed with a strike through line.	/1000
Space	Leading space will be preserved when this option is set.	/2000

VisualEdit

Syntax

```
E% = VisualEdit(x1%, y1%, w%, h%, fontid%, buffer$, length%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the edit zone.
Y	The Y pixel position of the top left corner of the edit zone.
W	The width of the edit zone in pixels.
H	The height of the edit zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Buffer	The variable from which and into which the edit will be performed.
Length	The useable length of the buffer variable.
Options	Edit function options as described below.

Description

This function will perform graphical edit operation in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The initial value of the edit field will be taken from the provided buffer length and will be returned upon successful completion.

The **options** parameter influences the way in which the edit operation will be performed as shown below:

Option	Description	Value
SECRET	When this option is set, the characters will be displayed as stars such that the input is hidden from view.	512
REFRESH	When this option is set the contents of the zone will be refreshed at the start of the Visual Edit operation.	1024
NOFOCUS	When this option is set the visual edit will not enter the text input loop and will exit immediately after all display operations have been completed.	2048
VLINE CURSOR	The text entry point cursor will be displayed as a vertical line just before the character entry point.	/20
BLOCK CURSOR	The text entry point cursor will be displayed as a highlighted block over the entire character cell.	/10
HLINE CURSOR	The text entry point cursor will be displayed as an underline across the bottom of the entire character cell.	/30
OUTLINE CURSOR	The text entry point cursor will be displayed as outline frame around the entire character cell.	/40

VisualWindow

Syntax

```
E% = VisualWindow(x1%, y1%, w%, h%, fontid%, message$, length%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the window.
Y	The Y pixel position of the top left corner of the window.
W	The width of the window in pixels.
H	The height of the window in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Message	The window title text string to be displayed.
Length	The useable length of the title text.
Options	This parameter is a BIT MASK of window options as described below.

Description

This function will draw a graphical window with title in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **options** parameter influences the way in which the window buttons will be displayed as shown below:

Option	Description	Value
EXIT	This option activates display of the window EXIT ICON button in the top righthand corner.	1
HELP	This option activates display of the window HELP ICON button in the top righthand corner.	2
LANGUAGE	This option activates display of the window NATIONAL LANGUAGE FLAG icon button in the top righthand corner.	4
MINIMISE	This option activates display of the WINDOW MINIMISE icon button in the top righthand corner.	8
MAXIMISE	This option activates display of the WINDOW MAXIMISE icon button in the top righthand corner.	16
SHADOW	When this option is specified, the window will cast a shadow below its position.	/0400

VisualButton

Syntax

```
E% = VisualButton(x1%, y1%, w%, h%, fontid%, fg%, bg%, message$, length%, state%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the text zone.
Y	The Y pixel position of the top left corner of the text zone.
W	The width of the text zone in pixels.
H	The height of the text zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
State	The state of the button to be displayed.

Description

This function will draw a push button control with the provided text message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **state** parameter influences the way in which the button frame and text message will be displayed as shown below:

Option	Description	Value
IMAGE	When this option is set, the message or payload will be used as an image id.	16
FRAME	When this option is set the display of the button frame will be inhibited.	32
TRANSPARENT	When this option is set the display of the button background will be inhibited.	64
EVENT	Performs a visual event click simulation.	8
PRESS	When this option is set the button frame will be displayed depressed.	4
FOCUS	When this option is set the button frame will be displayed focused.	2
BOLD	When this option is set the button text will be bold.	/0100
SPRITE	When this option is set the button image be a sprite image.	/0400
SHADOW	When this option is set the button text will be shadowed.	/0400

VisualCheck

Syntax

```
E% = VisualCheck(x1%, y1%, w%, h%, fontid%, fg%, bg%, message$, length%, value%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the text zone.
Y	The Y pixel position of the top left corner of the text zone.
W	The width of the text zone in pixels.
H	The height of the text zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
Value	The value of the radio button.

Description

This function will draw a graphical check box and associated **label** message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. When the **value** parameter is not ZERO then the check control will be displayed as **checked**, otherwise it will be **unchecked**.

VisualRadio

Syntax

```
E% = VisualRadio(x1%, y1%, w%, h%, fontid%, fg%, bg%, message$, length%, storage%, value%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the text zone.
Y	The Y pixel position of the top left corner of the text zone.
W	The width of the text zone in pixels.
H	The height of the text zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
Storage	The number of radio buttons in the group.
Value	The value of the radio button.

Description

This function will draw a graphical radio button and associated **label** message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. When the **value** parameter and the **storage** parameter are equal then the button will be displayed as **checked**, otherwise it will be **unchecked**.

VisualImage

Syntax

```
E% = VisualImage(x1%, y1%, w%, h%, imagename$, length%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the image display zone.
Y	The Y pixel position of the top left corner of the image display zone.
W	The width of the image display zone in pixels.
H	The height of the image display zone in pixels.
Imagename	The full file and path name of the image file to be loaded.
Length	The useable length of the image name.
State	The image display options as described below.

Description

This function will draw the image loaded from the named image file in the graphical zone described by the x, y, w, and h parameters, using the option as shown below:

Option	Description	Value
TILE	When this option is selected the image will be repeated to cover or tile the output zone.	4
SPRITE	When this option is selected the zero pixels in the image will be transparent and allow the background to shine through.	/0400
MAKEFIT	When this option is selected the image will be re-dimensioned to cover the output zone.	5
BESTFIT	When this option is selected the image will be re-dimensioned respecting its original proportions to best fill the output zone.	6
NONE	The image will be displayed and may be clipped.	0
LEFT	The image will be left aligned in the output zone.	2
RIGHT	The image will be aligned with the right edge of the output zone.	1
CENTER	The image will be centred between the left and right edges of the output zone.	3
TOP	The image will be aligned with the top edge of the output zone.	/0010
BOTTOM	The image will be aligned with the bottom edge of the output zone.	/0020

VisualTabPage

Syntax

```
E% = VisualTabPage(x1%, y1%, w%, h%, fontid%, title$, length%, offset%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the tab page zone.
Y	The Y pixel position of the top left corner of the tab page zone.
W	The width of the tab page zone in pixels.
H	The height of the tab page zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Title	The message text string to be displayed.
Length	The useable length of the message text.
Offset	The offset in pixels from the left edge of the frame at which the tab button will be displayed.
Options	The Boolean value of the radio button.

Description

This function will draw a graphical tab page frame and associated label message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **offset** parameter determines the tab button offset and the **options** parameter describes the state of the tab page button.

Option	Description	Value
CONCAVE	When this option is set, the tab button frame will be concave.	/0010
CONVEX	When this option is set, the tab button frame will be convex.	/0020
RELIEF	When this option is set the display of the tab button framer will be a standard relief frame.	0
FOCUS	When this option is set the tab button frame will be displayed focused.	2
BOLD	When this option is set the tab button text will be bold.	/0100
LINE	When this option is set the tab button text will be underlined.	/0200
SHADOW	When this option is set the tab button text will be shadowed.	/0400

VisualSelect

Syntax

```
E% = VisualSelect(x%, y%, w%, h%, fontid%, fg%, bg%, message$, length%, storage%, state%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the select zone.
Y	The Y pixel position of the top left corner of the select zone.
W	The width of the text select zone in pixels.
H	The height of the text select zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
Storage	The integer value of the chosen option before and after the select operation.
State	The Boolean value of the state of the select box, open or closed.

Description

This function will draw a graphical selection control and associated label messages in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **storage** parameter provides the current and final selection value, and the **state** parameter provides display opens and indicates if the selection box is currently open or closed.

Option	Description	Value
INHIBITED	When this option is set the select control will be displayed inhibited.	/0010
OPEN	When this option value is set the select control will be displayed open otherwise it will be displayed closed.	8
PRESS	When this option is set the button frame will be displayed depressed.	4
FOCUS	When this option is set the button frame will be displayed focused.	2

VisualScroll

Syntax

```
E% = VisualScroll(x%, y%, w%, h%, fontid%, fg%, bg%, item%, limit%, total%, style%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the scroll bar zone.
Y	The Y pixel position of the top left corner of the scroll bar zone.
W	The width of the scroll bar zone in pixels.
H	The height of the scroll bar zone in pixels.
Font Id	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Item	The value of the scroll bar within the limit.
Limit	The limit value of the scroll bar.
Total	The total number of items represented by the scroll bar.
Style	The Boolean value of the radio button.

Description

This function will draw a graphical scroll bar in the zone described by the x, y, w, and h parameters, using the font identified by the font id parameter and the colours. The **item** parameter determines the position of the slider compared to the **limit** value. If the zone is wider than it is high then the scroll bar will be a horizontal scroll bar, otherwise it will be a vertical scroll bar. The **style** parameter controls the presence of the top, end, page up and page down buttons.

Style	Description	Value
LEFT	When this option is set the left hand or top scroll icon will be displayed.	2
RIGHT	When this option is set the right hand or bottom scroll icon will be displayed.	1
PRESS	When this option is set the slider bar will be displayed depressed.	4
FOCUS	When this option is set the slider bar will be displayed focused.	/0100

VisualPalette

Syntax

```
E% = VisualPalette(filename$, length%)
```

Parameters

Parameter Name	Description
Filename$	This string parameter provides the full path prefix name of the colour palette file to be loaded.
Length%	The usable length in characters of the palette file name parameter.

Description

This function will load the graphical colour palette from the disk file identified by the full path and file name provided by the value of the first parameter. The length parameter determines the useable length of the palette file name parameter string.

This function will position the loaded palette as the currently active palette, comprising a table of 256 colour values each described using the standard R8G8B8 pixel format standard.

The palette is logically divided into three zones.

- The first zone, comprising the colour codes from **0** to **15**, contains the pixel colour values for the sixteen standard PAINT colours, that may be used to set both FOREGROUND and BACKGROUND colours.
- The second zone, comprising the colour codes from **16** to **31**, contains the pixel colour values for the standard user interface components, relief frames and their labels and focus bars.
- The remaining colour codes from **32** to **255** are free for use by the application for its own specific colour requirements.

VisualInitialise

Syntax

```
E% = VisualInitialise(mode%)
```

Parameters

Parameter Name	Description
Mode%	This integer parameter indicates the VESA graphical resolution code.

Description

This function will activate the GIGO graphical subsystem with a standard screen buffer described by the following VESA mode resolution code:

VESA Code	Width	Height
0	320	200
1	640	400
2	640	480
3	800	600
4	1024	769
5	1280	1024

VisualLiberate

Syntax

```
E% = VisualLiberate()
```

Description

This function will release the GIGO graphical subsystem and any allocated visual buffers and the complete style tree. The screen will be returned to standard CICO text mode.

VisualFreeze

Syntax

```
E% = VisualFreeze()
```

Description

This function will suspend output mirroring of changes to the currently connected graphical buffer. Modifications performed will not be visible until a Visual Thaw has been performed.

VisualThaw

Syntax

```
E% = VisualThaw(x%, y%, w%, h%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the zone to thaw.
Y	The Y pixel position of the top left corner of the zone to thaw.
W	The width of the zone to thaw in pixels.
H	The height of the zone to thaw in pixels.

Description

This function will commit any modifications performed on the buffer area described by the x, y, w, and h parameters to the output for display.

VisualFill

Syntax

```
E% = VisualFill(x%, y%, w%, h%, pixel%, mode%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the zone to fill.
Y	The Y pixel position of the top left corner of the zone to fill.
W	The width of the zone to fill in pixels.
H	The height of the zone to fill in pixels.
Pixel	The pixel value to use for the fill.
Mode	The type of fill operation required.

Description

This function will fill the zone described by the **x**, **y**, **w**, and **h** parameter values with the colour provided by the value of the **pixel** parameter as described by the value of the **mode** parameter.

Mode	Palette Mode Fill Description	Value
FILL	This operation fills the target zone with the pixel value.	0
RELIEF	This operation draws a relief frame with transparent fill.	1
CIRCLE FRAME	This operation draws a circle frame with pixel value.	2
ELLIPSE FRAME	This operation draws an elliptical frame with pixel value.	258
DISK FILL	This operation draws a filled circle with pixel value.	3
ELLIPSE FILL	This operation draws a filled ellipse with pixel value.	259
VISUAL CONVEX	This operation draws a horizontal convex fill.	4
VERTICAL CONVEX	This operation draws a vertical convex fill.	5
VISUAL CONCAVE	This operation draws a horizontal concave fill.	6
VERTICAL CONCAVE	This operation draws a vertical concave fill.	7
COLOUR FILL	This operation draws a frame using low byte of pixel colour and fills with high byte of pixel colour.	8
FRAMED CIRCLE	This operation draws a circular frame using the low byte of the pixel colour and fills with the high byte pixel colour.	9
FRAMED ELLIPSE	This operation draws an elliptical frame using the low byte of the pixel colour and fills with the high byte pixel colour.	267
CONVEX RELIEF	This operation draws a standard relief frame with a convex fill.	10
CONCAVE RELIEF	This operation draws a standard relief frame with a concave fill.	11
INSIDE FILL	This operation draws an inside fill using pixel colour.	12
ROUNDED FRAME	This operation draws a rounded frame with the pixel colour	13
ROUNDED FILL	This operation draws a rounded fill using the pixel colour.	14
X RELIEF	This operation draws an X relief frame using the pixel colour.	15

SHADED ZONE	This operation draws a shaded zone using the pixel colour value.	16
SHADED HOLE	This operation draws a shaded hole using the pixel colour value.	17
Mode	True Colour Mode Fill Description	Value
TRUE VERTICAL CONVEX	This operation draws a true colour vertical convex fill with standard relief colours.	18
TRUE HORIZONTAL CONVEX	This operation draws a true colour horizontal convex fill with standard relief colours.	19
TRUE VERTICAL CONCAVE	This operation draws a true colour vertical concave fill with standard relief colours.	20
TRUE HORIZONTAL CONCAVE	This operation draws a true colour horizontal concave fill with standard relief colours.	21
TRUE COLOUR RELIEF	This operation draws a true colour relief zone with standard relief colours.	22
TRUE COLOUR DISK	This operation draws a true colour filled circle using the provided pixel value.	23
TRUE COLOUR CIRCLE	This operation draws a true colour circle frame using the provided pixel value.	24
TRUE COLOUR HOLE	This operation draws a true colour filled hole using the provided pixel value.	25

VisualEvent

Syntax

```
E% = VisualEvent(buffer$)
```

Parameters

Parameter Name	Description
Buffer	The output buffer into which the event data will be stored.

Description

This function will return the next event in the visual event queue in the provided output buffer which will be used as an array of four short integer values, for the event click, the event button, the event column and the event row values respectively. The function will return the standard HOTKEY EVENT code of 256.

VisualKey

Syntax

```
E% = VisualKey(mode%)
```

Parameters

Parameter Name	Description
Mode	The operation mode of the key collection operation.

Description

This function will either a Boolean flag or an actual key code depending on the value of the **mode** parameter as described below.

Mode Value	Description
0	Test for key pending and return Boolean flag.
1	Wait for a return Key code value.
2	Test for key and return key if key pending.

VisualControl

Syntax

```
E% = VisualControl(command%, buffer$, length%)
```

Parameters

Parameter Name	Description
Command	The visual control operation code.
Buffer	The buffer parameter.
Length	The usable length of the buffer parameter.

Description

This function will perform the Visual Control operation as described by the value of the **command** parameter with the contents of the **length**-controlled **buffer** parameter.

The following visual control operations are currently available:

Command	Description	Value
VERSION	Returns the version of the visual subsystem in the buffer.	0
CONFIG	Re-initialises the visual subsystem using the visual configuration file named by the value of the buffer parameter.	1
RECORD	Activation of recording of visual events.	2
REPLAY	Replay recorded visual events.	3
BUFFER WIDTH	Returns the width, in pixel columns, of the visual buffer identified by the integer value provided by the buffer parameter.	15
TEXT LENGTH	Returns the pixel width of the text provided by the buffer parameter when displayed with the font identified by the length parameter.	26
BUFFER HEIGHT	Returns the height, in pixel columns, of the visual buffer identified by the integer value provided by the buffer parameter.	16
FONT WIDTH	Returns the width of the text font identified by the integer value provided by the buffer parameter.	4
FONT HEIGHT	Returns the height of the text font identified by the integer value provided by the buffer parameter.	5
TRACE	Output a screen trace message using the contents of the buffer parameter.	6
TITLE	Set the window title using the value of the buffer parameter.	7
ICON	Set the process icon using the name taken from the buffer parameter.	8
HELP	Display a visual help bubble with the text provided from the buffer parameter.	9
SET KEYS	Set The collection of visual edit keys from the buffer parameter.	43
GET KEYS	Collect the collection of visual edit keys to the buffer parameter.	42
SET EDIT	Set The collection of visual edit keys from the buffer parameter.	49
GET EDIT	Collect the collection of visual edit keys to the buffer parameter.	48
LANGUAGE	Set the national language code of the visual library.	11
TRIGGER	Return the visual library trigger code.	12
PUSH VIEWPORT	Push view port description data to the buffer parameter.	13
POP VIEWPORT	Pop view port description data from the buffer parameter.	14
PAGE VIEWPORT	Allows a view port to be defined in an allocated page buffer.	29
IMAGE NAME	Return the name of the image identified by the number value extracted from the buffer parameter.	17

FONT NAME	Return the name of the font identified by the number value extracted from the buffer parameter.	27
IMAGE WIDTH	Return the width of the image identified by the number value extracted from the buffer parameter.	18
IMAGE HEIGHT	Return the height of the image identified by the number value extracted from the buffer parameter.	19
IMAGE COUNT	Return the number of loaded image files.	24
IMAGE INFO	Returns the image height and width in the buffer after extracting the image name from the buffer.	28
TABLE FOCUS	Returns the visual table focus value using the **length** parameter.	52
DROP IMAGE	Release the image identified by the image name in the buffer parameter.	25
VISUAL ATTRIBUT	Declare the visual attribute as described in the buffer parameter.	20
VISUAL PAGE	Returns the current visual buffer page number.	55
STYLE HEIGHT	Returns the effective height of the visual style expression described by the buffer parameter.	37
STYLE WIDTH	Returns the effective width of the visual style expression described by the buffer parameter.	38
STYLE FONT	Resolve the font identifier of the visual style expression described by the buffer parameter.	39
DROP STYLE	Releases the style information corresponding to the style name in the buffer parameter.	31
STYLE	Loads the style described in the buffer parameter.	30
DOMAIN	Set the style domain to the name provided in the buffer.	35
CLASS	Define the style described by the buffer parameter.	36
ECHO	Activate the ECHO state with the value from the length parameter.	21
SAVEBMP	Perform a graphical screen capture to the BMP file named by the buffer parameter.	23
SAVEGIF	Perform a graphical screen capture to the GIF file named by the buffer parameter.	22
SAVEPNG	Perform a graphical screen capture to the PNG file named by the buffer parameter.	33
SAVEJPG	Perform a graphical screen capture to the JPG file named by the buffer parameter.	32
CONVISO	Convert an ISO encoded string from buffer to the prologue encoded equivalent.	40
CONVPRL	Convert a prologue encoded string from buffer to the ISO equivalent.	41
STATUS	Returns the visual library status.	50
DEFAULT	Set and get the default overload style.	51
EMERGENCY	Generate a timestamped emergency trace message provided by buffer parameter value.	999
GETCLIP	Gets the clip board contents to the buffer.	44
SETCLIP	Sets the clip board contents from the buffer.	45
GET CLIP NO CONV	Gets the clip board contents to the buffer without conversion, windows only.	46
SET CLIP NO CONV	Sets the clip board contents from the buffer without conversion, windows only.	47
PRN BUFFER	Prints a Visual Buffer image as Postscript as described by the buffer parameter.	70

PLAY SOUND	Plays the audio file named by the buffer parameter if the audio subsystem is operational.	99
CALLBACK	Send an application callback message with the data from the buffer parameter.	666

VisualProgress

Syntax

```
E% = VisualProgress(x%, y%, w%, h%, fontid%, fg%, bg%, limit%, value%, option%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the select zone.
Y	The Y pixel position of the top left corner of the select zone.
W	The width of the text select zone in pixels.
H	The height of the text select zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Limit	The integer value of this parameter specifies the upper limit.
Value	The integer value of this parameter specifies the current value.
Option	This integer parameter specifies the progress bar display options.

Description

This function will draw a graphical progression bar in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **value** parameter indicates the current value compared to the upper **limit** value. The foreground and background colour values will be used to colour the **value** region and the **remaining** region accordingly.

VisualTable

Syntax

```
E% = VisualTable(x%, y%, w%, h%, font%, fg%, bg%, titles$, tl%, val$, vl%, opt%, fmt$, fl%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the select zone.
Y	The Y pixel position of the top left corner of the select zone.
W	The width of the text select zone in pixels.
H	The height of the text select zone in pixels.
Font	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Title	The string value of this parameter provides the pipe separated list of column title texts to be displayed at the top of the table.
Tl	The effective length of the preceding titles parameter buffer.
Val	The string value of this parameter provides the pipe separated list of column values texts to be displayed in the rows of the table.
Vl	The effective length of the preceding values parameter buffer.
Opt	The integer value of this parameter provides the table display options.
Formats	The string value of this parameter provides the pipe separated list of style format names to be used for the display of the data in the different columns of the rows of the table.
Fl	The effective length of the preceding formats parameter buffer.

Description

This function will draw a graphical table object comprising the upper column titles followed by the rows of column wise data. The formats parameter allows the format of the different columns of the table to be specified as visual style class names.

Option	Description	Value
FORMAT	When format information is provided for the columns, then it will be used when this option is set.	/4000
ONSHOT	When this option is set then the cell data will be displayed in one single text output call and must correspond exactly to the table formatting. Otherwise, the data will be displayed one cell at a time.	/8000
CONVEX	When this option is set, the tab button frame will be convex.	/0020
BOLD	When this option is set the table text will be bold.	/0100
LINE	When this option is set the table text will be underlined.	/0200
SHADOW	When this option is set the Title cell texts will be displayed with a test shadow.	/0400
INHIBIT FRAME	When this option is set, the table frame is inhibited and only the data cells are displayed.	/2000
WHITE	Cells will be uniform background filled with the standard WHITE colour of the current palette.	0
ZONED	Standard Relief colours of the current palette will be used to perform the alternating row zoning effect.	1
COLOURED	The specified background colour will be used to colour the table cells.	2
COLOUR ZONED	The specified foreground colours will be used to perform the alternating row zoning effect.	3

COLUMNED	Standard Relief colours of the current palette will be used to perform the alternating column zoning effect.	4
COLOUR COLUMNED	The specified foreground and background colours will be used to perform the alternating column zoning effect.	5
CHEQUERED	Standard Relief colours of the current palette will be used to perform the alternative cell chequering effect.	6
COLOUR CHEQUERED	The specified foreground and background colours will be used to perform the alternating cell chequering effect.	7

VisualColour

Syntax

```
E% = VisualColour(command%, buffer$, first%, last%)
```

Parameters

Parameter Name	Description
Command	The visual colour operation code.
Buffer	The buffer parameter.
First	The integer value of the first parameter.
Second	The integer value of the first parameter.

Description

This function will perform the Visual Colour operation as described by the value of the **command** parameter with the contents of the **buffer** parameter and the secondary **first** and **second** integer parameter values.

The following visual colour operations are currently available:

Command	Description	Value
GET COLOURS	Returns the colour information from the current palette into the buffer parameter from the first to second colours.	0
PUT COLOURS	Sets the colour information into the current palette from the buffer parameter from the first to second colours.	1
GET RED VALUE	Returns the red portion of the colour information from the current palette into the buffer parameter from the first to second colours.	2
GET GREEN VALUE	Returns the green portion of the colour information from the current palette into the buffer parameter from the first to second colours.	3
GET BLUE VALUE	Returns the blue portion of the colour information from the current palette into the buffer parameter from the first to second colours.	4
PUT RED VALUE	Sets the red portion of the colour information into the current palette from the buffer parameter from the first to second colours.	5
PUT GREEN VALUE	Sets the green portion of the colour information into the current palette from the buffer parameter from the first to second colours.	6
PUT BLUE VALUE	Sets the blue portion of the colour information into the current palette from the buffer parameter from the first to second colours.	7

VisualStyle

Syntax

```
E% = VisualStyle(x%, y%, w%, h%, style$, sl%, message$, ml%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the styled zone.
Y	The Y pixel position of the top left corner of the styled zone.
W	The width of the styled zone in pixels.
H	The height of the styled zone in pixels.
Style	The string value of this parameter provides name of the style to be used for the formatting of the message.
Sl	The effective length of the preceding style parameter buffer.
Message	The string value of this parameter provides message value to be styled by the instruction.
Ml	The effective length of the preceding message parameter buffer.

Description

This function will display the provided payload string using the indicated style within the x, y, w, h defined graphical zone. The payload string may be interpreted as an image URL if the explicit content of the style has been set accordingly.

Please consult the section of this document describing the Abal Visual Style for precise and detailed information about the fill collection of style properties that may be used to define the Abal Visual Style classes that may be used with this function.

VisualBuffer

Syntax

```
E% = VisualBuffer(width%, height%)
```

Parameters

Parameter Name	Description
Width	The width of the visual buffer in pixels.
Height	The height of the visual buffer in pixels.

Description

This function will allocate visual buffer and the associated pixel storage units required for a virtual visualisation space defined by the width and height parameters.

The visual buffer identifier will be returned to the caller of the function allowing it to be used in Visual Page instructions.

VisualDrop

Syntax

```
E% = VisualDrop(number%)
```

Parameters

Parameter Name	Description
Number	The visual buffer identification number.

Description

This function will release the indicated visual buffer and all associated pixel storage pages.

VisualPage

Syntax

```
E% = VisualPage(number%)
```

Parameters

Parameter Name	Description
Number	The visual buffer identification number.

Description

This function will position the indicated visual buffer as the currently active target to capture all subsequent visual function output. This functionality is important for performance reasons and allows background screen capture to be performed with subsequent display performed from the visual buffer contents when the preparation phase has completed.

VisualGet

Syntax

```
E% = VisualGet(number%, x%, y%)
```

Parameters

Parameter Name	Description
Number	The visual buffer identification number.
X	The column value of the top left corner of the source position.
Y	The row value of the top left corner of the source position.

Description

This function will capture pixels from the x, y position of the current screen or positioned visual buffer, into the specified visual buffer. The dimensions of the target visual buffer will dictate the dimensions of the capture. Used in conjunction with the Visual Page function, this allows a background display operation to be suitably prepared using a capture of the current screen state.

VisualPut

Syntax

```
E% = VisualPut(number%, x%, y%)
```

Parameters

Parameter Name	Description
Number	The visual buffer identification number.
X	The column value of the top left corner of the target position.
Y	The row value of the top left corner of the target position.

Description

This function will transfer pixels to the x, y position of the current screen or positioned visual buffer, from the specified visual buffer. The dimensions of the source visual buffer will dictate the dimensions of the capture. This function may be used to complete the background screen buffering optimisation operation and performs the final display of the collection of output pixels.

VisualGetRow

Syntax

```
E% = VisualGetRow(row%, buffer$, length%)
```

Parameters

Parameter Name	Description
Row	The row number between 1 and N.
Buffer	The buffer into which the row of pixels will be transferred.
Length	The number of pixels to be transferred.

Description

This function will return the indicated row pixels from current screen or positioned visual buffer, into the specified buffer. The dimensions of the target visual buffer will dictate the dimensions of the capture.

VisualPutRow

Syntax

```
E% = VisualPutRow(row%, buffer$, length%)
```

Parameters

Parameter Name	Description
Row	The row number between 1 and N.
Buffer	The buffer from which the row of pixels will be transferred.
Length	The number of pixels to be transferred.

Description

This function will output the indicated row of pixels to the current screen or positioned visual buffer, from the specified buffer. The dimensions of the source visual buffer will dictate the dimensions of the capture.

VisualTransform

Syntax

```
E% = VisualTransform(number%, command%, p1%, p2%)
```

Parameters

Parameter Name	Description
Number	The visual buffer identification number.
Command	The transformation command to be performed.
P1	The first command specific parameter.
P2	The second command specific parameter.

Description

This function will perform the transformation described by **command** on the indicated visual buffer **number** using the command specific parameter values **p1** and **p2**.

The following transformation commands are possible:

Command	Description
ROTATE	Performs a 90, 180, or 270, degree rotation as defined by the p1 parameter.
REFLECT	Performs a vertical reflection or a horizontal reflection as defined by the p1 parameter (0 or 1).
COLOUR	Performs a colour hue transformation on red, green, or blue portions as defined by the p1 (0=red, 1=green,2=blue) and p2 (= portion) values.
RESIZE	Resize the indicated visual buffer to the dimensions provided by p1 (width) and p2 (height) and clip or fill the result.
MAKEFIT	Resize the indicated visual buffer to the dimensions provided by p1 (width) and p2 (height) and rescale the source to fit the target.

VisualPutZone

Syntax

```
E% = VisualPutZone(x%, y%, w%, h%, buffer$)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the target zone.
Y	The Y pixel position of the top left corner of the target zone.
W	The width of the target zone in pixels.
H	The height of the target zone in pixels.

Description

This function will output the contents of the source buffer into the area defined by the x, y, w, and h parameter values. The buffer must contain exactly h rows of w pixels and the pixel format must correspond to the output pixel format of 1, 2 or 4 bytes.

VisualSwitch

Syntax

```
E% = VisualSwitch(x%, y%, w%, h%, fontid%, fg%, bg%, message$, length%, state%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the switch zone.
Y	The Y pixel position of the top left corner of the switch zone.
W	The width of the switch zone in pixels.
H	The height of the switch zone in pixels.
Fontid	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Message	The message text string to be displayed.
Length	The useable length of the message text.
State	The state of the switch to be displayed.

Description

This function will draw a switch button control with the provided text message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **state** parameter influences the way in which the switch frame and text message will be displayed as described for the Visual Button function.

VisualGraph

Syntax

```
E% = VisualGraph(x%, y%, w%, h%, fontid%, fg%, bg%, data$, length%, options%)
```

Parameters

Parameter Name	Description
X	The X pixel position of the top left corner of the graph zone.
Y	The Y pixel position of the top left corner of the graph zone.
W	The width of the graph zone in pixels.
H	The height of the graph zone in pixels.
Font id	The font identifier of the font loaded via Visual Font.
Fg	The foreground (text) colour value.
Bg	The background (wash) colour value.
Data	The buffer containing the data values to be graphed.
Length	The useable length of the data text.
Options	Graph specific display options.

Description

This function will draw a switch button control with the provided text message in the graphical zone described by the x, y, w, and h parameters, using the font identified by the font id parameter. The **options** parameter influences the way in which the data buffer will be interpreted as shown below.

Command	Description	Value
BYTE	The data buffer is interpreted as a pointer to an ABAL INT8 array.	1
WORD	The data buffer is interpreted as a pointer to an ABAL INT16 array.	2
LONG	The data buffer is interpreted as a pointer to an ABAL INT32 array.	4
HUGE	The data buffer is interpreted as a pointer to an ABAL INT64 array.	8

VisualActivate

Syntax

```
E% = VisualActivate(width%, weight%, style%)
```

Parameters

Parameter Name	Description
Width	The width of the graphical screen in pixel columns.
Height	The height of the screen on pixel rows.
Style	The style of the graphics mode and pixel type.

Description

This function allows the GIGO graphic session to be started for screen resolutions other than the standard VESA modes offered by the Visual Initialise function. Here the precise width and height can be specified along with the nature of the individual pixels which may be 8bit colour mapped (default) or RGB 332, 555 or 777 requiring 1 byte 2 bytes or four bytes respectively.

VisualViewPort

Syntax

```
E% = VisualViewPort(x%, y%, w%, h%, f%)
```

Parameters

Parameter Name	Description
X	The top left corner X position of the viewport.
Y	The top left corner X position of the viewport.
W	The width in pixels of the viewport.
H	The height in pixels of the viewport.
F	The font ID to be used when calculating the text positioning within the viewport.

Description

This function allows a CICO text viewport to be declared in a region of a graphical screen.

The standard CICO rules will be imposed within this area as if it were a physical screen.

The textual dimensions will be calculated using the zone width and font dimension information.

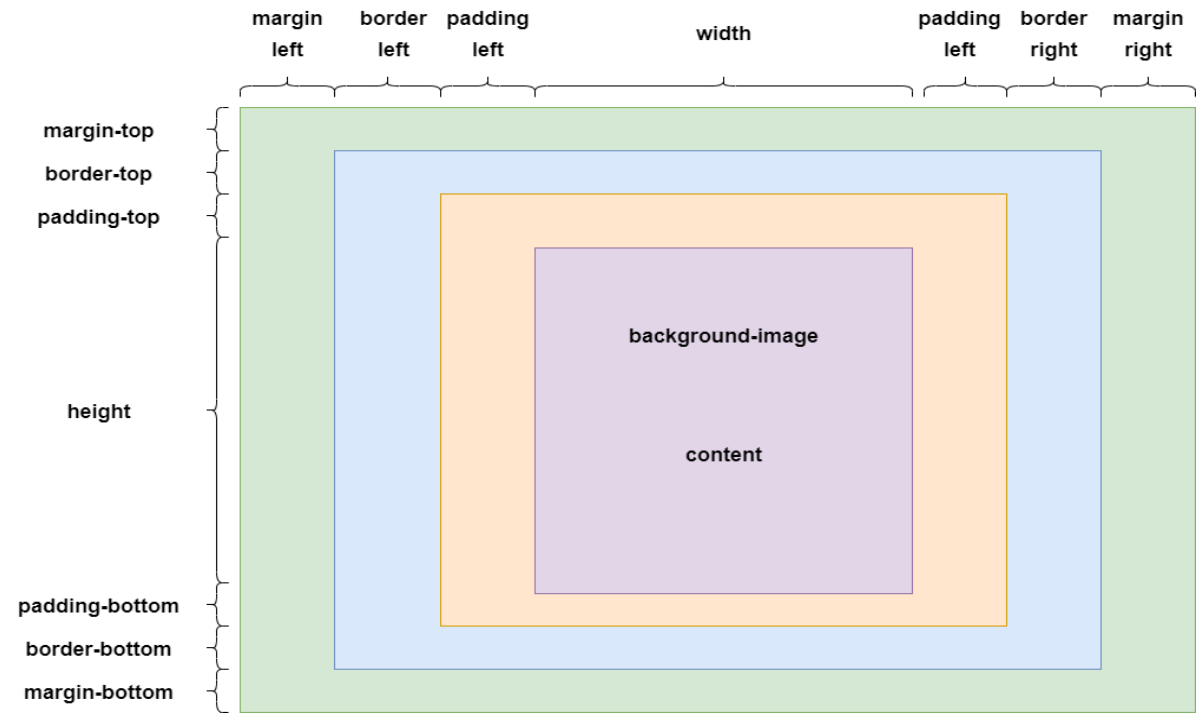
The PRINT, TAB, PAINT, ASK and CONF functions will be restricted to operation within the resulting viewport.

Abal Visual Style

This section of the documentation presents the style domains, classes and instructions that are referred to in Visual functions described above.

Style Model

The Visual Library allows graphical user interface elements to be styled, using a descriptive syntax, Abal Visual Style (AVS) that is very similar to the cascading style sheet (CSS) language used in web page authoring. AVS, like CSS, is based on the margin, border, padding, content model shown below:



The following important differences are to be noted:

- Abal Visual Style is not **cascading** since its usage is intended for Abal Visual application and not for the authoring of nested web page elements.
- Style classes are regrouped under application domains, like name spaces.
- An explicit style class hierarchy may be established between style classes.
- Style classes may aggregate style classes instances.

Style Domains

Abal Visual Style allows style domains to be defined for the regrouping of application domain specific style classes with similar or identical class naming schemes. The domain definition is shown below:

```
@domain domain_name
```

Style Definitions

Abal Visual Style allows symbolic names to be defined to represent values that will subsequently be substituted wherever a style token of the same name is encountered. The style define syntax is shown below:

```
@define symbolic_name value
```

Values defined in this way can then be used in style property instructions and their values will be substituted whenever they are encountered by the style parser. The following example demonstrates the use of this to allow the font size to be handled by a definition.

```
@define myfontsize 14

Example { text-font: myfontsize }
Another { text-font: myfontsize }
```

Style Import

Abal Visual Style classes can be regrouped in physical files and then these collections of files can be imported to a master style class or used in a subsequent overloaded definition. The style import syntax is shown below:

```
@import style_class_file
```

Style Classes

-Abal Visual Style classes are named collections of style property instructions, and their name is unique within a style domain. The class definition syntax is shown below.

```
classname { style property instructions }
```

Style Property Instructions

Abal Visual Style instructions fall into the logical categories shown in the preceding style model, those that influence the overall style **cell**, the **margin**, the **border**, the **padding**, and the **content** areas. Each individual style property instruction comprises a name value pair, separated from each other by the colon character and terminated by the semi-colon character, as can be seen below.

```
Property-name: property-value;
```

In the description of the style property instructions of each of the regions, the following standard value types are referenced:

- **Color Value**
  These values may be a define in one of the following ways:
  - "#" prefixed sequence of 6 hexadecimal digits of the following form **#RRGGBB**
  - A valid style colour name (**black**, red, navy, green, cyan, magenta, brown, silver, grey, pink, blue, lime, sky, purple, yellow, white) corresponding to the first 16 colours of the GIGO palette and associated with the standard 16 ABAL PAINT colours.
  - An RGB expression comprising three comma separated three digit values for each of the hues of the following form **rgb(rrr, ggg, bbb)**.
  - An RGBA expression comprising three comma separated three digit values for each of the hues and a fourth for the alpha channel representing the degree of transparency, of the following form **rgba(rrr, ggg, bbb, ttt)**

  In all cases both spellings **color** and **colour** are accepted.

- **Size Value**
  Size values are used to specify the sizes of regions and comprise a numeric portion and one of the following measurement symbols and names:
  - **+**: increase
  - **%**: percentage value
  - **mm**: millimetre value
  - **cm**: centimetre value
  - **in**: inch value
  - **em**: element value
  - **px**: pixel value
  - **pt**: point value
  - **pc**: point value

  in the absence of an explicit type-specifier then the value is presumed to be expressed in pixels.

- **Alignment Value**
  The following are valid alignment values:
  - **none**: the image or text will be displayed “as is”, with no pre-trimming of any space characters, and may be clipped.
  - **left**: The image or text will be aligned with the edge of the lefthand padding.
  - **right**: The image or text will be aligned with the edge of the righthand padding area.
  - **center**: The image or text will be centred between the left and righthand edges of the padding area.
  - **justify**: The text will be space filled to ensure it is aligned perfectly and filling the space between the left and righthand edges of the padding area.

  And for image accommodation within the cellular space the following terms apply:

  - **repeat** and **tile**: These are synonyms that will cover the area by vertical and horizontal repetition of the image pixels.
  - **makefit:** This will cover the area by a simple deformative re-dimensioning of the image to fit the cellular space.
  - **bestfit:** This will cover the area by a proportional re-dimensioning of the image to fit the cellular space allowing background colour to shine through as appropriate.
  - **transparent:** This will allow the zero-value pixels of the image data to allow the current background to shine through.

- **Url Value**
  These values are required for the correct description of image and font names and are composed as shown below:
  - **url**(font path name)
  - **url**(image path name)

Cell Properties

- **width**: allows the required width of the cell to be described as a size value.
- **height**: allows the required height of the cell to be described as a size value.

- **word-spacing**: This allows a size value to be specified to be used for the inter word spacing of all text displayed within the style cell.
- **letter-spacing**: This allows a size value to be specified to be used for the inter letter spacing of all text displayed within the style cell.
- **line-spacing**: This allows a size value to be specified to be used for the inter line spacing of all text displayed within the style cell.
- **vertical-align**: This allows an alignment value to be specified to be used for the vertical alignment of any content that will be displayed within the style cell.
- **file**: This allows the original filename from which the styled was sourced, to be preserved.
- **alias**: This allows an ordered, comma separated, list of style names for which this style class definition is an alias. The list of classes will be rendered, in order, overlaying the previous in the list, when styling the provided content.
  ** Since the alias property is defined for at the style class level, it will be loaded along with any other property values, into the current style definition, by the **class** property instruction. **

- **class**: This allows the single class style name to be used as the base class upon which this class is derived. All properties of the specified style class will be copied into the style container of the class under definition and may be overloaded as required by subsequent style property instructions.

- **float**: This allows the float property of the cell to be established as one of the following:
  - **none**: The float effect of the cell will be cancelled.
  - **left**: The cell will attempt to float towards the left edge.
  - **right**: The cell will attempt to float the right edge.
  - **top**: The cell will attempt to float towards the top edge.
  - **bottom**: The cell will attempt to float towards the bottom edge.
  - **home**: The cell will attempt to float toward the top left corner.

- **content**: This property allows the nature of the content to be pre-determined as one of the following:
  - **url**: indicates that the content string is a URL for the display of an image.
  - **text**: indicates that the content string is normal text to be displayed in the cell.
  - **trigger**: indicates that the content string is trigger styled text to be displayed in the cell.
  - **grip**: indicates that the standard scroll bar grip image is to be displayed in the cell.
  - **up**: indicates that the standard scroll bar up arrow image is to be displayed in the cell.
  - **down**: indicates that the standard scroll bar down arrow image is to be displayed in the cell.
  - **top**: indicates that the standard scroll bar top arrow image is to be displayed in the cell.
  - **bottom**: indicates that the standard scroll bar bottom arrow image is to be displayed in the cell.
  - **left**: indicates that the standard scroll bar left arrow image is to be displayed in the cell.
  - **right**: indicates that the standard scroll bar right arrow image is to be displayed in the cell.

- **auto**: This value indicates that the nature of the content should be auto determined from the value received.
- **none**: This value indicates that no content should be displayed, only the background colour and or image will be processed.

Margin Properties

The following properties may be specified collectively (in clockwise order) or individually for each of the four margin region sides, namely top, right, bottom, and left):

- Size

```
margin-size-top: size-value
margin-size-right: size-value
margin-size-bottom: size-value
margin-size-left: size-value
margin-size: top-size right-size bottom-size left-size
```

- Color

```
margin-color-top: color-value
margin-color-right: color-value
margin-color-bottom: color-value
margin-color-left: color-value
margin-color: top-color right-color bottom-color left-color
```

- Image

```
margin-image-top: url-value
margin-image-right: url-value
margin-image-bottom: url-value
margin-image-left: url-value
margin-image: top-image right-image bottom-image left-image
```

Border Properties

The following property instructions may be used to define the characteristics of the margin area of an Abal Visual Style class or element.

- Width

```
border-width-top: size-value;
border-width-right: size-value;
border-width-bottom: size-value;
border-width-left: size-value;
border-width: size-value [ size-value [ size-value [ size-value] ] ];
```

- Style

```
border-style-top: Style;
border-style-right: Style;
border-style-bottom: Style;
border-style-left: Style;
border-style: style [ Style [ Style [ Style] ] ];
```

- Color

```
border-color-top: color-value;
border-color-right: color-value;
border-color-bottom: color-value;
border-color-left: color-value;
border-color: color-value [ color-value [ color-value [ color-value] ] ];
```

- All

```
border-top: style size-value color-value;
border-right: style size-value color-value;
```

```
border-bottom: style size-value color-value;
border-left: style size-value color-value;
```

- Align

```
align: align-value;
```

- **Border Style**

The following list shows the collection of legal values that may be used to define border style properties:

- **none**
  The border display is inhibited.
- **url**
  The border will be displayed using the image described by the URL element.
- **rounded**
  A rounded frame will be used for the corners.
- **outset**
  An outset relief frame will be used for the border element.
- **inset**
  An outset relief frame will be used for the border element.
- **disk**
  A colour filled circle will be drawn.
- **fill**
  The border area will be colour filled.
- **shade**
  The area will alternate between background and foreground colours.
- **hole**
  This will produce the inverse of the disk, the area outside of the disk will be colour filled.
- **edit**
  A standard edit frame will be drawn.
- **ridge**
  A standard ridge frame will be drawn.
- **groove**
  A standard groove frame will be drawn.
- **convex**
  A convex relief fill will be drawn.
- **concave**
  A concave relief fill will be drawn.
- **double**
  A double solid line will be drawn.
- **dashed**
  A standard dashed line will be drawn.
- solid
  A single solid line will be drawn.
- **dotted**
  A standard dotted line will be drawn.
- **vconvex**

A vertical convex fill will be drawn.
- **hconvex**
  A horizontal convex fill will be drawn.
- **vconcave**
  A vertical concave fill will be drawn.
- **hconcave**
  A horizontal concave fill will be drawn.

Corner Properties

The following properties can be specified at the style cell level and affect the corners of the border region of the entire cell.

- **top**: This property, qualified with either **left** or **right** allows the corresponding border corner characteristics to be defined.
- **bottom**: This property, qualified with either **left** or **right** allows the corresponding border corner characteristics to be defined.

Padding Properties

The following property instructions may be used to define the characteristics of the padding area of an Abal Visual Style class or element.

```
padding-top: size-value;
padding-right: size-value;
padding-bottom: size-value;
padding-left: size-value;
padding: top-value right-value bottom-value left-value;
```

Text Properties

The following properties can be specified at the style cell level and affect the text of the entire cell.

- **text-font**: This property allows a URL value to be specified to be used for the display of textual content.
- **text-align**: This property allows an alignment value to be specified for the text within the style cell.
- **text-style** and **text-decoration** are synonyms that allow a comma separated list of terms to be used to define the text content attributes, from the following list of values (bold, underline, overline, blink, shadow, line)
- **text-indent**: This option allows an indentation value to be specified and included at the start of the content text.
- **text-colour**: This option allows a colour value to be specified for the characters of textual content.

Shadow Properties

The following properties can be specified at the style cell level and affect the nature of shadow that may be activated via the text decoration property instruction.

- **shadow-color**: This property allows a colour value to be specified to be used for the display of textual shadow.
- **shadow-width**: This property allows a size value to be specified for the horizontal displacement of the shadow.
- **shadow-height**: This property allows a size value to be specified for the vertical displacement of the shadow.

Background Properties

The following properties can be specified at the style cell level and affect the background of the entire cell.

- **background**: This property allows either a URL value to be specified to be used for the display of the background image, or the keyword **none**.

Default Style Class Names

If Abal Visual Style classes are defined with any of the following names, they will be detected and used to style the standard Visual Function equivalents.

- "windowframe"
- "buttonframe"
- "pageframe"
- "checkframe"
- "radioframe"
- "switchframe"
- "tableframe"
- "progressframe"
- "outsetframe"
- "insetframe"
- "grooveframe"
- "ridgeframe"
- "editframe"
- "formframe"
- "selectframe"
- "selectframeup"
- "selectframedown"
- "selectframebtn"
- "selectframescroll"
- "imageframe"
- "vscrollframe"
- "hscrollframe"

EXAMPLES

This section of the documentation will provide simple examples demonstrating the above library functions.

SIMPLE GUI

This first example shows the construction of a trivial graphical user interface for a user login dialog frame.

```
PROGRAM "LOGIN"
DCL E%, F%
DCL USERNAME$=16
DCL PASSWORD$=16
DCL FONTNAME$=256
DCL PALETTE$=256
SEGMENT 0
        E = VISUALINITALISE(3)
        FONTNAME = home/abal/images/standard.rgb"
        F = VISUALFONT(FONTNAME,LEN$(FONTNAME))
        PALETTE  = "/home/abal64/fonts/arial16.fmf"
        E = VISUALPALETTE(PALETTE,LEN$(PALETTE))
        E = VISUALWINDOW(100,100,600,400,F, "LOGIN", 5, 0)
        E = VISUALTEXT(200, 200, 100, 24, F, 16,0, "USERNAME", 8, 2)
        E = VISUALTEXT(200, 250, 100, 24, F, 16,0, "PASSWORD", 8, 2)
        E = VISUALEDIT(320, 200, 160, 24, F, USERNAME, 16, 0)
        E = VISUALEDIT(320, 250, 160, 24, F, PASSWORD, 16, 0)
        E = VISUALLIBERATE()
        STOP
ESEG 0
END
```

VISUAL LIBRARY CLASSES

The following collection of classes allows the visual library to be encapsulated for use from an Abal Object oriented programming scenario and alleviates the need to include the VISUAL Library definitions file.

```
#ifndef _visual_cls
#define _visual_cls

;**     ----------------------------------------
;**     C L A S S   :   v i s u a l _ l i b r a r y
;**     ----------------------------------------

#class  visual_library

        Library visual

        public integer strict user function VisualZone:(x%,y%,w%,h%,style%)
        public integer strict user function VisualLine:(x1%,y1%,x2%,y2%,thickness%,pixel%)
        public integer strict user function VisualFont:(fontname$,namelength%)
        public integer strict user function
VisualText:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,options%)
        public integer strict user function
VisualEdit:(x%,y%,w%,h%,fid%,buffer$,length%,options%)
        public integer strict user function
VisualWindow:(x%,y%,w%,h%,fid%,title$,length%,options%)
        public integer strict user function
VisualButton:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,options%)
        public integer strict user function
VisualCheck:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,value%)
        public integer strict user function
VisualRadio:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,storage%,value%)
        public integer strict user function VisualImage:(x%,y%,w%,h%,name$,length%,options%)
        public integer strict user function
VisualTabPage:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,offset%,options%)
        public integer strict user function
VisualSelect:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,storage%,options%)
        public integer strict user function
VisualScroll:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,item%,limit%,total%,value%)
        public integer strict user function VisualPalette:(filename$,length%)
        public integer strict user function VisualInitialise:(mode%)
        public integer strict user function VisualLiberate:()
        public integer strict user function VisualFreeze:()
        public integer strict user function VisualThaw:()
        public integer strict user function VisualFill:(x%,y%,w%,h%,pixel%,mode%)
```

```
        public integer strict user function VisualEvent:(buffer$)
        public integer strict user function VisualKey:(mode%)
        public integer strict user function VisualControl:(command%,buffer$,length%)
        public integer strict user function
VisualProgress:(x%,y%,w%,h%,fid%,fg%,bg%,limit%,value%,option%)
        public integer strict user function
VisualTable:(x%,y%,w%,h%,fid%,fg%,bg%,titles%,tl%,data$,dl%,format$,fl%,options%)
        public integer strict user function VisualColour:(command%,buffe^$,first%,last%)
        public integer strict user function VisualBuffer:(width%,height%)
        public integer strict user function VisualPage:(number%)
        public integer strict user function VisualGet:(number%,x%,y%)
        public integer strict user function VisualGetRow:(row%,buffer$,length%)
        public integer strict user function VisualPutRow:(row%,buffer$,length%)
        public integer strict user function VisualPut:(number%,x%,y%)
        public integer strict user function VisualDrop:(number%)
        public integer strict user function VisualTransform:(number%,command%,p1%,p2%)
        public integer strict user function VisualPutZone:(x%,y%,w%,h%,buffer$)
        public integer strict user function
VisualSwitch:(x%,y%,w%,h%,fid%,fg%,bg%,message$,length%,state%)
        public integer strict user function
VisualGraph:(x%,y%,w%,h%,fid%,fg%,bg%,data$,length%,option%)
        public integer strict user function VisualActivate:(width%,height%,option%)
        public integer strict user function VisualViewPort:(x%,y%,w%,h%,fid%)
        public integer strict user function VisualStyle:(x%,y%,w%,h%,style$,sl%,message$,ml%)

#endclass       visual_library

;**     ---------------------------------------------------------------
;**     C L A S S   :   v i s u a l _ l i b r a r y _ i n t e r f a c e
;**     ---------------------------------------------------------------

#class          visual_library_interface(public visual_library)

public  common  dcl     visual_library_status%

public  inline constructor      initialise_visual_library:
        if ( visual_library_status = 0 )
                if ( VisualInitialise(3) <> 0 ) :: endif
        endif
        visual_library_status += 1
end

public  inline destructor       liberate_visual_library:
        if ( visual_library_status > 0 )
                visual_library_status -= 1
                if ( visual_library_status = 0 )
                        if ( VisualLiberate() <> 0 ) :: endif
                endif
        endif
end
#endclass       visual_library_interface

;**     ---------------------------------------------------------------
;**     C L A S S   :   v i s u a l _ l i b r a r y _ c l i e n t
;**     ---------------------------------------------------------------

#class          visual_library_client(public common visual_library_interface)
#endclass

#endif  _visual_cls
```