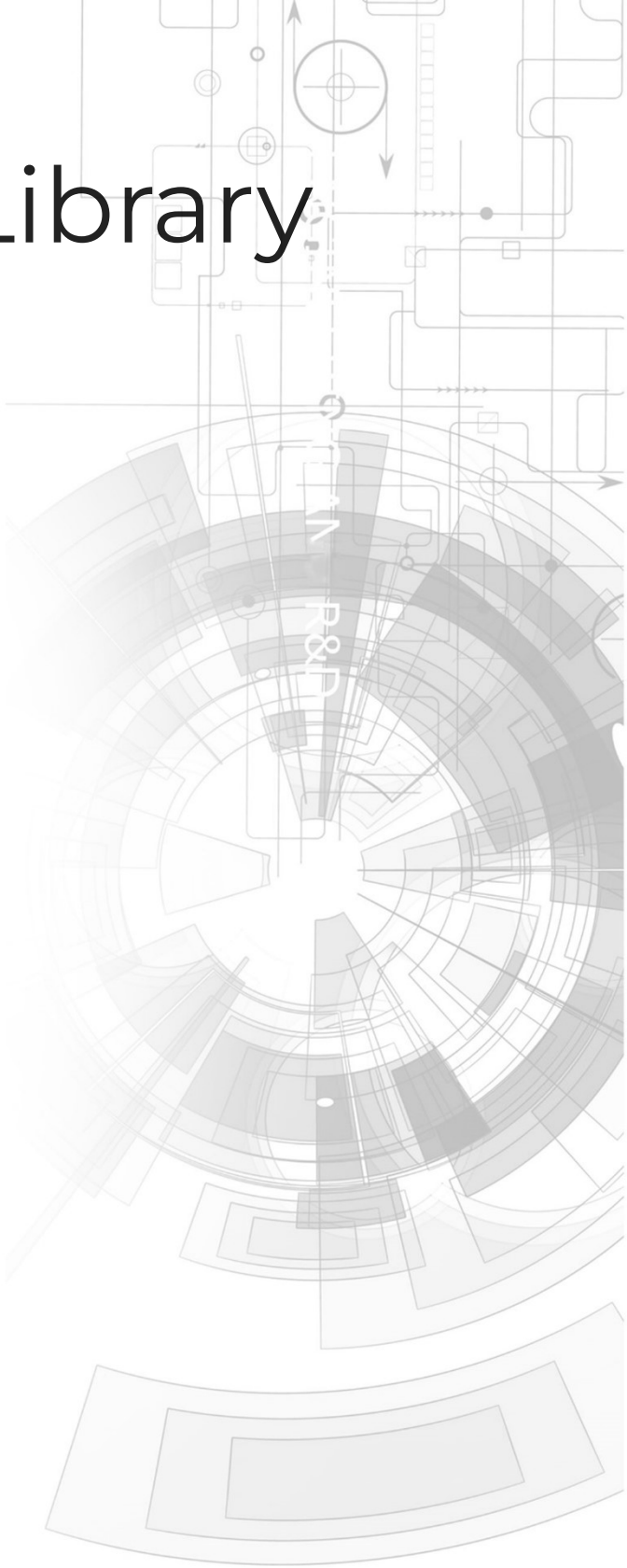
GFX4dESP32 Library



Manual

Revision 1.0

Contents

1. Introduction

The GFX4dESP32 library is provided by 4D Systems for use with gen4-ESP32-XX product series. This library provides users access to the graphics, touch, and WiFi functionalities of 4D Systems' ESP32-S3 display modules.

Note however that some functionalities might not be supported by a certain product types, depending on its specifications. For instance, non-touch variants have no access to all touch-related functions. For more information on the specifications of a product, refer to its datasheet.

To install the library, please refer to the instructions in Workshop4 ESP32 Development Manual.

It is recommended to use Workshop4 IDE to get the most of out the library functions.

> **Note**
>
> Workshop4 is a **Windows-only** application.

2. Library Setup

When using Workshop4, the library is automatically included in the code.

To setup the library for use with Arduino IDE or other Arduino-compatible IDEs, the correct header file for the target display must be included and an instance of the library needs to be created.

A sample code should look like this:

```
#include "gfx4desp32_%%displaynm%%.h"

gfx4desp32_%%displaynm%% gfx = gfx4desp32_%%displaynm%%();
```

where `%%displaynm%%` is the name of the target display module with underscores (`_`) as separators instead of a hyphen (`-`).

Here's a table listing all the available products and their respective header files.

Product	Header File
gen4-ESP32-24	`gfx4desp32_gen4_ESP32_24.h`
gen4-ESP32-24CT	`gfx4desp32_gen4_ESP32_24CT.h`
gen4-ESP32-24CT-CLB	`gfx4desp32_gen4_ESP32_24CT-CLB.h`
gen4-ESP32-28	`gfx4desp32_gen4_ESP32_28.h`
gen4-ESP32-28CT	`gfx4desp32_gen4_ESP32_28CT.h`
gen4-ESP32-28CT-CLB	`gfx4desp32_gen4_ESP32_28CT-CLB.h`
gen4-ESP32-32	`gfx4desp32_gen4_ESP32_32.h`
gen4-ESP32-32CT	`gfx4desp32_gen4_ESP32_32CT.h`
gen4-ESP32-32CT-CLB	`gfx4desp32_gen4_ESP32_32CT-CLB.h`
gen4-ESP32-35	`gfx4desp32_gen4_ESP32_35.h`
gen4-ESP32-35CT	`gfx4desp32_gen4_ESP32_35CT.h`
gen4-ESP32-35CT-CLB	`gfx4desp32_gen4_ESP32_35CT-CLB.h`
gen4-ESP32-43	`gfx4desp32_gen4_ESP32_43.h`
gen4-ESP32-43T	`gfx4desp32_gen4_ESP32_43T.h`
gen4-ESP32-43CT	`gfx4desp32_gen4_ESP32_43CT.h`
gen4-ESP32-43CT-CLB	`gfx4desp32_gen4_ESP32_43CT-CLB.h`
gen4-ESP32-50	`gfx4desp32_gen4_ESP32_50.h`
gen4-ESP32-50T	`gfx4desp32_gen4_ESP32_50T.h`
gen4-ESP32-50CT	`gfx4desp32_gen4_ESP32_50CT.h`
gen4-ESP32-50CT-CLB	`gfx4desp32_gen4_ESP32_50CT-CLB.h`

5-inch Resistive Touch Variant Example

```
#include "gfx4desp32_gen4_ESP32_50T.h"

gfx4desp32_gen4_ESP32_50T gfx = gfx4desp32_gen4_ESP32_50T();
```

Note

When using Workshop4, a new project will include the snippet of code with `%%displaynm%%` above. Workshop4 automatically replaces `%%displaynm%%` with the correct name for the current display selected in the project before the project is compiled.f!!

To start using the created `gfx` instance, add the line `gfx.begin()` before using any other library functions.

Common Setup Routine

```
void setup() {
    gfx.begin(); // Initialize the display
    gfx.Cls();
    gfx.ScrollEnable(true);
    gfx.BacklightOn(true);
    gfx.Orientation(PORTRAIT);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE); gfx.Font(2); gfx.TextSize(1);
}
```

3. General and Utility Functions

3.1. begin

Initialize the display module and all onboard components.

Syntax: `gfx.begin();`

Return: None (*void*)

Example

```
void setup()
{
    gfx.begin();
    gfx.Cls();
    gfx.ScrollEnable(false);
    gfx.BacklightOn(true);
    gfx.Orientation(LANDSCAPE);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE, BLACK); gfx.Font(2);  gfx.TextSize(1);
    gfx.Open4dGFX("sample");        // Opens DAT/GCI files using filename w/o extension
    gfx.touch_Set(TOUCH_ENABLE);    // Global touch enabled
}
```

3.2. getHeight

Returns the height of the display in pixels at the current orientation

Syntax: `gfx.getHeight();`

Return: Height of the display in pixels at the current orientation (*int16_t*)

Example

```
gfx.Orientation(LANDSCAPE);
int16_t displayHeight = gfx.getHeight();
// Get display height then print its value
gfx.print("Height: "); gfx.println(displayHeight);
```

3.3. getWidth

Returns the width of the display in pixels at the current orientation

Syntax: `gfx.getWidth();`

Return: Width of the display in pixels at the current orientation (*int16_t*)

Example

```
gfx.Orientation(PORTRAIT);
int16_t displayWidth = gfx.getWidth();
// Get display Width then print its value
gfx.print("Width: "); gfx.println(displayWidth);
```

3.4. Transparency

Enables or disables transparent colours.

Syntax: `gfx.Transparency(mode);`

Argument	Type	Description
mode	boolean	Use `true` to enable or `false` to disable transparent colours

Return: None (*void*)

Example

```
gfx.TransparentColor(BLACK);
gfx.Transparency(true);     // Enable transparency
gfx.DrawImageFile("sample.gci");
gfx.Transparency(false);
```

Note

This doesn't apply to pushColors function

3.5. TransparentColor

Sets the colour to be treated as transparent when transparency is enabled

Syntax: `gfx.TransparentColor(colour);`

Argument	Type	Description
colour	uint16_t	RGB565 colour that won't be drawn in all functions when transparency is enabled

Return: None (*void*)

Example

See gfx.Transparency for an example

Note

This doesn't apply to pushColors function

3.6. AlphaBlend

Enables Alpha Blending of new foreground colour to existing background colour

Syntax: `gfx.AlphaBlend(mode);`

Argument	Type	Description
mode	bool	Specifies whether to enable (`true`) or disable (`false`) alpha blending

Return: None (*void*)

Example

```
gfx.AlphaBlendLevel(true);
gfx.AlphaBlend(true); // Enable alpha blending
gfx.DrawImageFile("sample.gci");
gfx.AlphaBlend(false);
```

3.7. AlphaBlendLevel

Sets the level of alpha blending between new foreground colour and existing background colour. gfx.AlphaBlend must be used and set to `true` or `ON` for this to take effect.

Syntax: `gfx.AlphaBlendLevel(value);`

Argument	Type	Description
value	uint32_t	Specifies the intensity for alpha blending (0 to 255)

Return: None (*void*)

Example

See gfx.AlphaBlend for an example

3.8. BacklightOn

Turns the backlight **ON** or **OFF**.

Syntax: `gfx.BacklightOn(mode);`

Argument	Type	Description
mode	boolean	Use `true` to turn **ON** and `false` to turn **OFF**

Return: None (*void*)

Example

```
gfx.BacklightOn(false); // Turns the backlight OFF
delay(3000);            // Wait for approx. 3 seconds
gfx.BacklightOn(true);  // Turns the backlight ON
```

3.9. Contrast

Sets the backlight intensity

Syntax: `gfx.Contrast(level);`

Argument	Type	Description
level	int	0 = backlight is OFF, 1-15 = backlight brightness level

Return: None (*void*)

Example

```
for (i = currentLevel; i >= 0; i--) {
    delay(200);          // Slowly dim the backlight
    gfx.Contrast(i);     // until it is off
}
```

3.10. Invert

Inverts the colours displayed on the screen or returns to original.

Syntax: `gfx.Invert(mode);`

Argument	Type	Description
mode	boolean	Use `true` to invert display colours and `false` to display original

Return: None (*void*)

Example

```
gfx.RectangleFilled(0, 0, 50, 50, BLACK);
gfx.RectangleFilled(100, 100, 150, 150, BLUE);
delay(2000);
gfx.Invert(true);   // Inverts colours displayed on screen
delay(2000);
gfx.Invert(false);  // Revert back to original colours
```

3.11. Cls

Clear the screen and fill with the specified colour. If no colour value was specified, the function will use **BLACK**.

This function also brings some settings back to default.

- Cursor position is reset to (0, 0)
- Scroll is set to 0 pixels

Syntax: `gfx.Cls([colour]);`

Argument	Type	Description
colour (optional)	uint16_t	Specifies the colour to clear the screen with

Return: None (*void*)

Example

```
gfx.Cls();      // Clears the screen with BLACK
gfx.Cls(LIME);  // Clears the screen with LIME
```

3.12. FillScreen

Fills the screen with the specified colour.

Syntax: `gfx.FillScreen(colour);`

Argument	Type	Description
colour	uint16_t	Specifies the colour to fill the screen with

Return: None (*void*)

Example

```
gfx.FillScreen(LIME);   // Fills the screen with LIME
```

3.13. Orientation

3.13.1. Get Orientation

Get the current display orientation. See Set Orientation for possible values

Syntax: `gfx.Orientation();`

Return: Current orientation (*uint8_t*)

Example

```
gfx.Orientation(PORTRAIT);
int8_t orientation = gfx.Orientation();
// Get orientation then print its value
gfx.print("Orientation: "); gfx.println(orientation);
```

3.13.2. Set Orientation

Sets the orientation of the display the the mode specified.

Constants	Value
`LANDSCAPE`	0
`LANDSCAPE_R`	1
`PORTRAIT`	2
`PORTRAIT_R`	3

Syntax: `gfx.Orientation(orient);`

Argument	Type	Description
orient	uint8_t	Specifies the orientation, refer to table above

Return: None (*void*)

Example

```
gfx.Orientation(PORTRAIT);  // Sets Orientation to PORTRAIT
```

Note

The cursor position is not altered in any way by changing the orientation.

3.14. CheckSD

Check if a uSD card is properly mounted to the display module.

If the uSD Card is properly mounted during the execution of `gfx.begin()`, this function will return `true`. Otherwise, this will return `false`.

Syntax: `gfx.CheckSD();`

Return: `true` if uSD is properly mounted, `false` otherwise (*boolean*)

Example

```
gfx.begin();
if (!gfx.CheckSD()) {
    gfx.print("uSD Card not mounted.");
    gfx.print("Please insert uSD Card and restart module");
    while(1);
} // Check if the uSD is mounted
```

3.15. Orbit

Calculate the position of the pixel relative to the current cursor position using the given angle and length.

Syntax: `gfx.Orbit(angle, length, output);`

Argument	Type	Description
angle	int	Angle in degrees relative to the current cursor position
length	int	Pixel distance from current cursor position
output	int *	Pointer to an *int* array with at least 2 elements for x and y

Return: None (*void*)

Example

```
int xy[2];
gfx.MoveTo(gfx.getWidth() / 2, gfx.getHeight() / 2);
gfx.Orbit(135, 50, xy);
gfx.printf("Pixel is in (%d, %d)\n", xy[0], xy[1]);
```

3.16. XYposToDegree

Calculate the angle between the current cursor position and the pixel position specified and return it in degrees.

Syntax: `gfx.XYposToDegree(x, y);`

Argument	Type	Description
x	int	Horizontal pixel position of the pixel
y	int	Vertical pixel position of the pixel

Return: Angle in degrees relative to current cursor position (*int16_t*)

Example

```
int16_t deg = gfx.XYposToDegree(100, 150);
gfx.printf("(100, 150) is at %d degrees from current cursor position\n", deg);
```

3.17. ScreenCapture

Capture and save to uSD an area of the screen and returns `true` if successful.

The operation can fail under two conditions:

- uSD is not mounted properly
- File with the filename specified already exists

Syntax: `gfx.ScreenCapture(x, y, w, h, fname);`

Argument	Type	Description
x	int16_t	Top left horizontal position in pixels
y	int16_t	Top left vertical position in pixels
w	int16_t	Width of the capture area in pixels
h	int16_t	Height of the capture area in pixels
fname	String	Filename to save the image with

Return: `true` if successful, `false` otherwise (*boolean*)

Example

```
gfx.begin();

// draw a few things here...

gfx.ScreenCapture(0, 0, gfx.getWidth(), gfx.getHeight(), "fullpage.bmp");
```

3.18. getSdFatInstance

Returns a reference to the initialized SdFat instance of the uSD. This allows access full access to the mounted uSD card.

Please refer to the SdFat documentation for more information.

Syntax: `gfx.getSdFatInstance();`

Return: Instance of SdFat (*SdFat &*)

Example

```
SdFat& uSD;

void setup()
{
    gfx.begin();
    gfx.Cls();
    gfx.ScrollEnable(false);
    gfx.BacklightOn(true);
    gfx.Orientation(LANDSCAPE);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE, BLACK); gfx.Font(2);  gfx.TextSize(1);
    gfx.Open4dGFX("sample");        // Opens DAT/GCI files using filename w/o extension
    gfx.touch_Set(TOUCH_ENABLE);    // Global touch enabled
    uSD = gfx.getSdFatInstance();

    // use "uSD" as needed here
}

void loop()
{
    // use "uSD" as needed here
}
```

Note

Version 1.0.0 of the GFX4dESP32 library depends on SdFat version 2.2.2 or higher.

The library is tested to work specifically with SdFat v2.2.2, therefore using this version is mainly recommended if any issues are encountered with newer versions.

3.19. DisplayControl

Executes various display initialization functions as listed:

- `DISP_CTRL_RE_INIT`
- `DISP_CTRL_RESET`
- `DISP_CTRL_NEW`
- `DISP_CTRL_INIT`
- `DISP_CTRL_STOP`
- `DISP_CTRL_START_TX`
- `DISP_CTRL_DEL`
- `DISP_CTRL_START`
- `DISP_CTRL_FLUSH`

Syntax: `gfx.DisplayControl(action);`

Argument	Type	Description
action	uint8_t	Specifies the initialization function to perform

Return: None (*void*)

Example

```
gfx.DisplayControl(DISP_CTRL_RESET);
```

4. Primitive Shapes

4.1. PutPixel

Writes the pixel colour to the specified position

Syntax: `gfx.PutPixel(x, y, colour);`

Argument	Type	Description
x	int16_t	Horizontal position in pixels
y	int16_t	Vertical position in pixels
colour	uint16_t	16-bit RGB565 colour to be drawn to the specified position

Return: None (*void*)

Example

```
gfx.PutPixel(5, 10, RED); // Draws a RED pixel at (5,10)
```

4.2. Vline

Draws a vertical line from point **(x, y)** with length equal to **height** using the specified colour.

Direction is specified by the sign of **height**: positive (+) draws downwards and negative (-) draws upwards

Syntax: `gfx.Vline(x, y, height, colour);`

Argument	Type	Description
x	int16_t	Horizontal position in pixels
y	int16_t	Vertical position in pixels
height	int16_t	Length in pixels and direction of the vertical line
colour	uint16_t	16-bit RGB565 colour of the line

Return: None (*void*)

Example

```
gfx.Vline(5,10,100,RED);
// Draws a 100-pixel RED Vline from (5,10) downwards
gfx.Vline(5,10,-100,BLUE);
// Draws a 100-pixel BLUE Vline from (5,10) upwards
```

4.3. Hline

Draws a horizontal line from point **(x, y)** with length equal to **width** using the specified colour.

Direction is specified by the sign of **width**: positive (+) draws to the right and negative (-) draws to the left

Syntax: `gfx.Hline(x, y, width, colour);`

Argument	Type	Description
x	int16_t	Horizontal position in pixels
y	int16_t	Vertical position in pixels
height	int16_t	Length in pixels and direction of the horizontal line
colour	uint16_t	16-bit RGB565 colour of the line

Return: None (*void*)

Example

```
gfx.Hline(5,10,100,RED);
// Draws a 100-pixel RED Hline from (5,10) to the right
gfx.Hline(5,10,-100,BLUE);
// Draws a 100-pixel BLUE Hline from (5,10) to the left
```

4.4. VlineD

Draws a vertical line from point **(x, y1)** to **(x, y2)** using the specified colour.

Syntax: `gfx.VlineD(x, y1, y2, colour);`

Argument	Type	Description
x	int	Horizontal position in pixels
y1	int	Vertical position in pixels of first endpoint
y2	int	Vertical position in pixels of second endpoint
colour	int	16-bit RGB565 colour of the line

Return: None (*void*)

Example

```
gfx.VlineD(5,10,100,RED);
// Draws a vertical RED line from (5,10) to (5, 100)
```

4.5. HlineD

Draws a horizontal line from point **(x1, y)** to point **(x2, y)** using the specified colour.

Syntax: `gfx.HlineD(y, x1, x2, colour);`

Argument	Type	Description
y	int	Vertical position in pixels
x1	int	Horizontal position in pixels of first endpoint
x1	int	Horizontal position in pixels of second endpoint
colour	int	16-bit RGB565 colour of the line

Return: None (*void*)

Example

```
gfx.HlineD(10, 5, 100, RED);
// Draws a horizontal RED line from (5, 10) to (100, 10)
```

4.6. Line

Draws a line from point **(x0, y0)** to point **(x1, y1)** using the specified colour.

Syntax: `gfx.Line(x0, y0, x1, y1, colour);`

Argument	Type	Description
x0	int16_t	Horizontal position of starting point in pixels
y0	int16_t	Vertical position of starting point in pixels
x1	int16_t	Horizontal position of endpoint in pixels
y1	int16_t	Vertical position of endpoint in pixels
colour	uint16_t	16-bit RGB565 colour of the line

Return: None (*void*)

Example

```
gfx.Line(0,0,50,50,RED);
// Draws a RED line from (0,0) to (50,50)
```

4.7. RectangleFilled

Draws a solid rectangle having a diagonal with endpoints at **(x1, y1)** and **(x2, y2)**.

Syntax: `gfx.RectangleFilled(x1, y1, x2, y2, colour);`

Argument	Type	Description
x1	int	Horizontal position of first endpoint of one diagonal of the rectangle
y1	int	Vertical position of first endpoint of one diagonal of the rectangle
x2	int	Horizontal position of second endpoint of one diagonal of the rectangle
y2	int	Vertical position of second endpoint of one diagonal of the rectangle
colour	uint16_t	16-bit RGB565 colour of the solid rectangle

Return: None (*void*)

Example

```
gfx.RectangleFilled(0, 0, 50, 50, YELLOW);
// Draws a YELLOW solid rectangle with:
// a diagonal whose end points are (0, 0) and (50, 50)
```

4.8. Rectangle

Draws an outlined rectangle having a diagonal with endpoints at **(x1, y1)** and **(x2, y2)**.

Syntax: `gfx.Rectangle(x1, y1, x2, y2, colour);`

Argument	Type	Description
x1	int16_t	Horizontal position of first endpoint of one diagonal of the rectangle
y1	int16_t	Vertical position of first endpoint of one diagonal of the rectangle
x2	int16_t	Horizontal position of second endpoint of one diagonal of the rectangle
y2	int16_t	Vertical position of second endpoint of one diagonal of the rectangle
colour	uint16_t	16-bit RGB565 colour of the outlined rectangle

Return: None (*void*)

Example

```
gfx.Rectangle(0, 0, 50, 50, CYAN);
// Draws a CYAN rectangle with:
// a diagonal whose end points are (0, 0) and (50, 50)
```

4.9. CircleFilled

Draws a solid-coloured circle with the specified radius and colour with the center at **(x, y)**

Syntax: `gfx.CircleFilled(x, y, r, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the circle
y	int16_t	Vertical position of the centre of the circle
r	int16_t	Radius of the filled circle in pixels
colour	uint16_t	16-bit RGB565 colour of the filled circle

Return: None (*void*)

Example

```
gfx.CircleFilled(50, 50, 10, RED);
// Draws a RED filled circle with:
// radius of 10 and center @(50, 50)
```

4.10. Circle

Draws an outlined circle with the specified radius and colour with the center at **(x, y)**

Syntax: `gfx.Circle(x, y, r, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the circle
y	int16_t	Vertical position of the centre of the circle
r	int16_t	Radius of the outlined circle in pixels
colour	uint16_t	16-bit RGB565 colour of the filled circle

Return: None (*void*)

Example

```
gfx.Circle(50, 50, 10, RED);
// Draws a RED circle w/ radius of 10 and center at (50, 50)
```

4.11. EllipseFilled

Draws a solid coloured ellipse with the specified x radius (**radx**), y radius (**rady**), and colour with the center at **(x, y)**

Syntax: `gfx.EllipseFilled(x, y, radx, rady, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the ellipse
y	int16_t	Vertical position of the centre of the ellipse
radx	int16_t	Radius of the ellipse along the x-axis
rady	int16_t	Radius of the ellipse along the y-axis
colour	uint16_t	16-bit RGB565 colour of the filled ellipse

Return: None (*void*)

Example

```
gfx.EllipseFilled(50, 50, 10, 5, RED);
// Draws a RED filled ellipse with:
// x-radius of 10, y-radius of 5 and center @(50, 50)
```

4.12. Ellipse

Draws an outlined ellipse with the specified x radius (**radx**), y radius (**rady**), and colour with the center at **(x, y)**

Syntax: `gfx.Ellipse(x, y, radx, rady, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the ellipse
y	int16_t	Vertical position of the centre of the ellipse
radx	int16_t	Radius of the ellipse along the x-axis
rady	int16_t	Radius of the ellipse along the y-axis
colour	uint16_t	16-bit RGB565 colour of the outlined ellipse

Return: None (*void*)

Example

```
gfx.Ellipse(50, 50, 10, 5, RED);
// Draws a RED ellipse with:
// x-radius of 10, y-radius of 5 and center @(50, 50)
```

4.13. ArcFilled

Draw 90-degree Arcs with center at (x, y) and gap between left and right parts. This is a support function used for filled rounded rectangles and circles.

Syntax: `gfx.ArcFilled(x, y, r, topBottom, gap, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the arc
y	int16_t	Vertical position of the centre of the arc
r	int16_t	Radius in pixels of the filled arc
topBottom	int16_t	Indicates whether to draw the top arcs (1), bottom arcs (2) or both sets (3)
gap	int16_t	Indicates the gap between the left and right arcs
colour	uint16_t	16-bit RGB565 colour of the filled arc

Return: None (*void*)

Example

```
gfx.ArcFilled(100, 100, 50, 3, 0, RED);
```

4.14. Arc

Draw 90-degree Arcs with center at (x, y) for the selected quadrants This is a support function used for outlined rounded rectangles and circles.

Multiple quadrants can be selected by setting the following bits for the value of quadrants:

- 0x01 - Top left
- 0x02 - Top right
- 0x04 - Bottom right
- 0x08 - Bottom left

Syntax: `gfx.Arc(x, y, r, quadrants, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of the centre of the arc
y	int16_t	Vertical position of the centre of the arc
r	int16_t	Radius in pixels of the outlined arc
quadrants	uint16_t	The selected quadrants
colour	uint16_t	16-bit RGB565 colour of the outlined arc

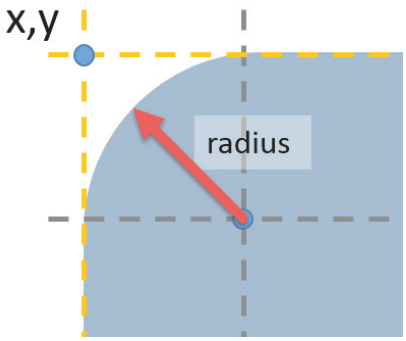
Return: None (*void*)

Example

```
gfx.Arc(100, 100, 50, 0x8 | 0x4, RED);
// Draw the bottom left and bottom right arcs with
// center at (100, 100), a radius of 50 pixels
```

4.15. RoundRectFilled

Draws a solid round-cornered rectangle having an outer rectangle diagonal with endpoints at **(x, y)** and **(x1, y1)** and with a corner radius of **r**.



Syntax: `gfx.RoundRectFilled(x, y, x1, y1, r, colour);`

Argument	Type	Description
x	int16_t	Horizontal position of first endpoint of one diagonal of the outer rectangle
y	int16_t	Vertical position of first endpoint of one diagonal of the outer rectangle
x1	int16_t	Horizontal position of second endpoint of one diagonal of the outer rectangle
y1	int16_t	Vertical position of second endpoint of one diagonal of the outer rectangle
r	int16_t	Corner radius. This is the distance in pixels extending from the corners of the **inner** rectangle.
colour	uint16_t	16-bit RGB565 colour of the filled rounded rectangle

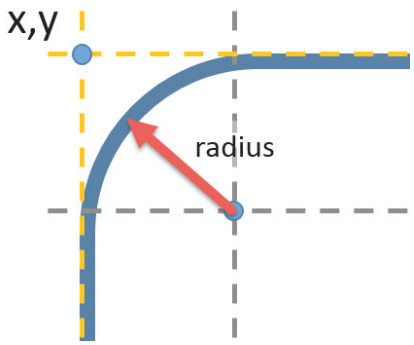
Return: None (*void*)

Example

```
gfx.RoundRectFilled(0, 0, 50, 50, 10, RED);
// Draws a solid RED round-cornered rectangle with:
// a diagonal whose end points are (0, 0) and (50, 50)
// and with a corner radius of 10
```

4.16. RoundRect

Draws an outlined round-cornered rectangle having an outer rectangle diagonal with endpoints at **(x, y)** and **(x1, y1)** and with a corner radius of **r**.



Syntax: `gfx.RoundRect(x, y, x1, y1, r, colour);`

Argument	Type	Description
x0	int16_t	Horizontal position of first endpoint of one diagonal of the outer rectangle
y0	int16_t	Vertical position of first endpoint of one diagonal of the outer rectangle
x1	int16_t	Horizontal position of second endpoint of one diagonal of the outer rectangle
y1	int16_t	Vertical position of second endpoint of one diagonal of the outer rectangle
r	int16_t	Corner radius. This is the distance in pixels extending from the corners of the **inner** rectangle.
colour	uint16_t	16-bit RGB565 colour of the outlined rounded rectangle

Return: None (*void*)

Example

```
gfx.RoundRect(0,0,50,50,10,GREEN);
// Draws a GREEN round-cornered rectangle with:
// a diagonal whose end points are (0,0) and (50,50) // and corner radius of 10
```

4.17. TriangleFilled

Draws a solid triangle between vertices **(x1, y1)**, **(x2, y2)** and **(x3, y3)** using the specified colour **c**.

Syntax: `gfx.TriangleFilled(x1, y1, x2, y2, x3, y3, c);`

Argument	Type	Description
x1	int16_t	Horizontal position of first vertex of the triangle
y1	int16_t	Vertical position of first vertex of the triangle
x2	int16_t	Horizontal position of second vertex of the triangle
y2	int16_t	Vertical position of second vertex of the triangle
x3	int16_t	Horizontal position of third vertex of the triangle
y3	int16_t	Vertical position of third vertex of the triangle
c	uint16_t	16-bit RGB565 colour of the filled triangle

Return: None (*void*)

Example

```
gfx.TriangleFilled(0, 0, 10, 50, 50, 50, CYAN);
// Draws a solid CYAN triangle with:
// the vertices (0, 0), (10, 50), and (50, 50)
```

4.18. Triangle

Draws a triangle outline between vertices **(x0, y0)**, **(x1, y1)**, and **(x2, y2)** using the specified colour **c**.

Syntax: `gfx.Triangle(x0, y0, x1, y1, x2, y2, c);`

Argument	Type	Description
x0	int16_t	Horizontal position of first vertex of the triangle
y0	int16_t	Vertical position of first vertex of the triangle
x1	int16_t	Horizontal position of second vertex of the triangle
y1	int16_t	Vertical position of second vertex of the triangle
x2	int16_t	Horizontal position of third vertex of the triangle
y2	int16_t	Vertical position of third vertex of the triangle
c	uint16_t	16-bit RGB565 colour of the outlined triangle

Return: None (*void*)

Example

```
gfx.Triangle(0, 0, 10, 50, 50, 50, CYAN);
// Draws a CYAN triangle with:
// the vertices (0, 0), (10, 50), and (50, 50)
```

4.19. GradTriangleFilled

Produce a triangle with or without a gradient

Syntax: `gfx.GradTriangleFilled(x0, y0, x1, y1, x2, y2, colour, ncCol, h, ypos, lev, erase);`

Argument	Type	Description
x0	int	Horizontal position of first vertex of the triangle
y0	int	Vertical position of first vertex of the triangle
x1	int	Horizontal position of second vertex of the triangle
y1	int	Vertical position of second vertex of the triangle
x2	int	Horizontal position of third vertex of the triangle
y2	int	Vertical position of third vertex of the triangle
colour	int	Colour that will be used if the Solid or Gradient parameter is set to 0
ncCol	int	Colour that will be used if the Solid or Gradient parameter is set to 1
h	int	Height of the area that the gradient will be calculated. Can be larger than the triangle
ypos	int	Position on the Y axis that the gradient will be calculated from with respect to triangle position
lev	int	Level of gradient applied
erase	int	Select whether solid triangle or gardient triangle is drawn

Return: None (*void*)

Example

```
gfx.GradTriangleFilled(10, 10, 10, 100, 100, 100, YELLOW, DARKKHAKI, 100, 10, 30, 1);
```

4.20. gradientShape

Produce a shaped color gradient using the supplied parameters

Syntax: `gfx.gradientShape(vert, ow, xPos, yPos, w, h, r1, r2, r3, r4, darken, color, sr1, gl1, colorD, sr3, gl3, gtb);`

Argument	Type	Description
vert	int	Horizontal or Vertical gradient -- 0 or 1
ow	int	Outer gradient width
ow	int	Outer gradient width
xPos	int	x co-ordinate
yPos	int	y co-ordinate
w	int	Width
h	int	Height
r1	int	Top left corner radius
r2	int	Top right corner radius
r3	int	Bottom left radius
r4	int	Bottom right radius
darken	int	Darken both colours by a value. Can be a negative to lighten instead
color	int	Outer Gradient colour
sr1	int	Outer Gradient type (0 - 3 horizontal, +4 vertical) **0** - Raised **1** - Sunken **2** - Raised flatter middle **3** - Sunken flatter middle
gl1	int	Outer Gradient level 0 - 63
colorD	int	Inner Gradient colour
sr3	int	Outer Gradient type (0 - 3 horizontal, +4 vertical) **0** - Raised **1** - Sunken **2** - Raised flatter middle **3** - Sunken flatter middle
gl3	int	Inner Gradient level 0 - 63
gtb	int	Split gradient

Return: None (*void*)

Example

```
gfx.gradientShape(HorzVert, OuterWidth, X, Y, W, H, TLrad, TRrad, BLrad, BRrad,
    Darken, OuterColor, OuterType, OuterLevel,
    InnerColor, InnerType, InnerLevel, Split);
```

5. Primitive Widgets

5.1. Panel

Draws a raised 3 dimensional rectangular panel at a screen location defined by **x** and **y** parameters (top left corner). The size of the panel is set with the **w** and **h** parameters. The colour is defined by colour **c**

Syntax: `gfx.Panel(x, y, w, h, c);`

Argument	Type	Description
x	int16_t	Horizontal position of the top left pixel of the panel
y	int16_t	Vertical position of the top left pixel of the panel
w	int16_t	Horizontal position of second endpoint of one diagonal of the panel
h	int16_t	Vertical position of second endpoint of one diagonal of the panel
c	uint16_t	16-bit RGB565 colour of the panel

Return: None (*void*)

Example

```
gfx.Panel(0, 0, 100, 50, GRAY);
// Draws a 100x50 GRAY raised panel @(0, 0)
```

5.2. PanelRecessed

Draw recessed Panel at **(x, y)**, with dimensions **w** and **h** and colour **c**.

Syntax: `gfx.Panel(x, y, w, h, c);`

Argument	Type	Description
x	int16_t	Left most pixel position of the panel
y	int16_t	Vertical position of first endpoint of one diagonal of the panel
w	int16_t	Horizontal position of second endpoint of one diagonal of the panel
h	int16_t	Vertical position of second endpoint of one diagonal of the panel
c	uint16_t	16-bit RGB565 colour of the panel

Return: None (*void*)

Example

```
gfx.PanelRecessed(0, 0, 100, 50, GRAY);
// Draws a 100x50 GRAY recessed panel @(0, 0)
```

5.3. ButtonXstyle

Sets the style of primitive buttons with the following defined styles

- `BUTTON_SQUARE`
- `BUTTON_ROUNDED`
- `BUTTON_CIRCULAR`
- `BUTTON_CIRCULAR_TOP`

Syntax: `gfx.ButtonXstyle(style);`

Argument	Type	Description
style	byte	Selected button style

Return: None (*void*)

Example

```
gfx.ButtonXstyle(BUTTON_ROUNDED);
```

5.4. drawButton

This function draws a button at the specified state using the size, color and text parameters provided. This is a support function for ButtonX functions.

Syntax:

```
gfx.drawButton(updn, x, y, w, h, colourb, btext, tfont, tfontsize, tfontsizeht, tcolour [, compressed]);
```

Argument	Type	Description
updn	uint8_t	State of the button
x	int16_t	Horizontal pixel position of the left side of the button
y	int16_t	Vertical pixel position of the top side of the button
w	int16_t	Width of the button in pixels
h	int16_t	Height of the button in pixels
colourb	uint16_t	Button colour
btext	String	Button text
tfont	uint8_t const uint8_t * gfx4d_font	Button text font
tfontsize	int8_t	Font size multiplier
tfontsizeht	int8_t	
tcolour	uint16_t	Button text colour
compressed (optional)	bool	Only used when using font array (*const uint8_t **), specifies whether the font array is compressed or GCI formatted

Return: None (*void*)

Example

```
gfx.drawButton(1, 50, 50, 150, 50, GRAY, "START", FONT2, 1, 1, BLACK);
```

5.5. Slider

Draws a slider with diagonal at (x1, y1) and (x2, y2). The thumb will be drawn depending on the specified scale and value.

Syntax: `gfx.Slider(state, x1, y1, x2, y2, colourb, colourt, scale, value);`

Argument	Type	Description
state	uint8_t	State of the slider
x1	int16_t	Horizontal pixel position of the left side of the slider
y1	int16_t	Vertical pixel position of the top side of the slider
x2	int16_t	Horizontal pixel position of the right side of the slider
y2	int16_t	Vertical pixel position of the bottom side of the slider
colourb	uint16_t	Base colour
colourt	uint16_t	Thumb colour
scale	int16_t	Maximum value of the slider
value	int16_t	Value of the slider

Return: None (*void*)

Example

```
gfx.Slider(state, x1, y1, x2, y2, colourb, colourt, scale, value);
```

Note

x2 ad y2 should be greater than x1 and y1 respectively

5.6. Button

Draws a 3-dimensional Text Button at screen location defined by (x, y).

The font to be used can be system font, GCI font (opened using gfx.Open4dFont) or font array. When using font array, an optional parameter can be specified to indicate whether the format is compressed or GCI formatted.

Syntax: `gfx.Button(state, x, y, colorb, tcolor, tfont, tfontsizeh, tfontsize, btext [, compressed]);`

Argument	Type	Description
state	uint8_t	State of the button
x	int16_t	Horizontal pixel position of the left side of the button
y	int16_t	Vertical pixel position of the top side of the button
colorb	uint16_t	Button colour
tcolor	uint16_t	Text colour
tfont	uint8_t const uint8_t * gfx4d_font	Button text font
tfontsizeh	int8_t	Font height multiplier
tfontsize	int8_t	Font width multiplier
btext	String	Button text
compressed (optional)	bool	Only used when using font array (*const uint8_t **), specifies whether the font array is compressed or GCI formatted

Return: None (*void*)

Example

```
gfx.Button(state, x, y, colorb, tcolor, tfont, tfontsizeh, tfontsize, btext, compressed);
```

5.7. Buttonx

Draws a 3-dimensional Text Button at screen location defined by (x, y) parameters The user needs to specify a handler for the button that will be used by the functions:

- ButtonUp
- ButtonDown
- ButtonActive
- DeleteButton
- CheckButtons

The font to be used can be system font, GCI font (opened using gfx.Open4dFont) or font array. When using font array, an optional parameter can be specified to indicate whether the format is compressed or GCI formatted.

Syntax: `gfx.Buttonx(hndl, x, y, w, h, colorb, btext, tfont, tcolor [, compressed]);`

Argument	Type	Description
hndl	uint8_t	ID specified to identify the button
x	int16_t	Horizontal pixel position of the left side of the button
y	int16_t	Vertical pixel position of the top side of the button
w	int16_t	Width of the button in pixels
h	int16_t	Height of the button in pixels
colorb	uint16_t	Button colour
btext	String	Button text
tfont	uint8_t const uint8_t * gfx4d_font	Button text font
tcolor	uint16_t	Text colour
compressed (optional)	bool	Only used when using font array (*const uint8_t **), specifies whether the font array is compressed or GCI formatted

Return: None (*void*)

Example

```
gfx.Buttonx(hndl, x, y, w, h, colorb, btext, tfont, tcolor, compressed);
```

5.8. ButtonActive

Enable/disable the specified button.

Syntax: `gfx.ButtonActive(id, mode);`

Argument	Type	Description
id	uint8_t	ID specified using gfx.Buttonx to identify the button
mode	boolean	Indicates whether to activate (`true`) or deactivate (`false`) a buttonx

Return: None (*void*)

Example

```
gfx.ButtonActive(1, true);
```

5.9. DeleteButton

Delete previously created primitive button n and, optionally, redraw background in colour bc.

Syntax: `gfx.DeleteButton(hndl, bc);`

Argument	Type	Description
id	uint8_t	ID specified using gfx.Buttonx to identify the button
bc (optional)	uint16_t	Optional background color to redraw when deleting button

Return: None (*void*)

Example

```
gfx.DeleteButton(2);
gfx.DeleteButton(3, BLACK);
```

5.10. DeleteButtonBG

Delete previously created primitive button n and redraw section of background image bg.

Syntax: `gfx.DeleteButtonBG(id, bg);`

Argument	Type	Description
id	uint8_t	ID specified using gfx.Buttonx to identify the button
bg	uint16_t	Background image to redraw

Return: None (*void*)

Example

```
gfx.DeleteButton(3, iForm0);
```

6. Support Colour Functions

6.1. bevelColor

Returns a darker and lighter colour of the given colour (colorb) by a set 18 steps.

The darker color is stored in the high word of the return value while the lighter color is stored in the low word.

Syntax: `gfx.bevelColor(colorb);`

Argument	Type	Description
colorb	uint16_t	Color to process into darker and lighter shades

Return: Combined darkened and lightened color (*uint32_t*)

Example

```
uint32_t output = gfx.bevelColor(0x8787);
uint16_t darkColor = output >> 16;
uint16_t lightColor = output & 0xFFFF;
```

6.2. HighlightColors

Returns a darker and lighter colour of the given colour (colorh) by the given steps

The darker color is stored in the high word of the return value while the lighter color is stored in the low word.

Syntax: `gfx.HighlightColors(colorh, step);`

Argument	Type	Description
colorh	uint16_t	Color to process into darker and lighter shades
step	int	Number of steps to darken/lighten the original color

Return: Combined darkened and lightened color (*uint32_t*)

Example

```
uint32_t output = gfx.HighlightColors(0x8787, 15);
uint16_t darkColor = output >> 16;
uint16_t lightColor = output & 0xFFFF;
```

6.3. RGBto565

Converts red green and blue colour elements to RGB565 16bit colour value

Syntax: `gfx.RGBto565(rc, gc, bc);`

Argument	Type	Description
r	uint8_t	8-bit red component
g	uint8_t	8-bit green component
b	uint8_t	8-bit blue component

Return: 16-bit RGB565 color (*uint16_t*)

Example

```
uint16_t color = gfx.RGBto565(255, 0, 100);
```

6.4. RGBs2COL

Converts RGB888 to 16-bit RGB565 value

This function is provided for compatibility with gen4-IoD projects.

Syntax: `gfx.RGBs2COL(r, g, b);`

Argument	Type	Description
r	uint8_t	8-bit red component
g	uint8_t	8-bit green component
b	uint8_t	8-bit blue component

Return: 16-bit RGB565 color (*uint16_t*)

Example

```
uint16_t color = gfx.RGBs2COL(255, 0, 100);
```

7. Text Functions

7.1. write

The GFX4dESP32 class inherits Arduino's Print class by overriding this function.

This function can be used directly or simply use any print functions from the Print class such as `print`, `println` and `printf`

Syntax: `gfx.write(c);`

Argument	Type	Description
c	uint8_t	8-bit character to process/write to the display

Return: Number of characters written (*size_t*)

Example

```
void setup() {
    gfx.begin(); // Initialize the display
    gfx.Cls();
    gfx.ScrollEnable(true);
    gfx.BacklightOn(true);
    gfx.Orientation(PORTRAIT);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE); gfx.Font(2); gfx.TextSize(1);

    gfx.write('4'); gfx.write('D');
    gfx.println(" Systems");
}
```

7.2. print

The GFX4dESP32 class inherits Arduino's Print class by overriding write(c). This provides access to all functions from this parent class which is commonly used when using Serial.

Syntax: `gfx.print(value [, format]);`

Argument	Type	Description
value	*any*	The value to print
format (optional)	int	Specifies the number base (for intergral data types) or number of decimal places (for floating point types)

Return: Number of characters written (*size_t*)

Example

Autoformat

```
gfx.print(78);              // gives "78"
gfx.print(1.23456);         // gives "1.23"
gfx.print('N');             // gives "N"
gfx.print("Hello world."); // gives "Hello world."
```

Specify Format

```
gfx.print(78, BIN);         // gives "1001110"
gfx.print(78, OCT);         // gives "116"
gfx.print(78, DEC);         // gives "78"
gfx.print(78, HEX);         // gives "4E"
gfx.print(1.23456, 0);      // gives "1"
gfx.print(1.23456, 2);      // gives "1.23"
gfx.print(1.23456, 4);      // gives "1.2346"
```

Note

For more information, please refer to Arduino documentation.

7.3. println

The GFX4dESP32 class inherits Arduino's Print class by overriding write(c). This provides access to all functions from this parent class which is commonly used when using Serial.

The println function is very similar to print and simply adds carriage return '\r' and a line feed character '\n' at the end.

Syntax: `gfx.println(value [, format]);`

Argument	Type	Description
value	*any*	The value to print
format (optional)	int	Specifies the number base (for intergral data types) or number of decimal places (for floating point types)

Return: Number of characters written (*size_t*)

Example

Autoformat

```
gfx.println(78);              // gives "78\r\n"
gfx.println(1.23456);         // gives "1.23\r\n"
gfx.println('N');             // gives "N\r\n"
gfx.println("Hello world."); // gives "Hello world.\r\n"
```

Specify Format

```
gfx.println(78, BIN);         // gives "1001110\r\n"
gfx.println(78, OCT);         // gives "116\r\n"
gfx.println(78, DEC);         // gives "78\r\n"
gfx.println(78, HEX);         // gives "4E\r\n"
gfx.println(1.23456, 0);      // gives "1\r\n"
gfx.println(1.23456, 2);      // gives "1.23\r\n"
gfx.println(1.23456, 4);      // gives "1.2346\r\n"
```

Note

For more information, please refer to Arduino documentation.

7.4. printf

The GFX4dESP32 class inherits Arduino's Print class by overriding write(c). This provides access to all functions from this parent class which is commonly used when using Serial.

The printf function takes a format string as its first argument, followed by optional additional arguments that correspond to the placeholders in the format string. These placeholders are denoted by `%` followed by a format specifier, such as `%d` for integers, `%f` for floating-point numbers, `%s` for strings, and more.

Syntax: `gfx.println(value [, format]);`

Argument	Type	Description
format	const char *	The base string format to use when printing
...	*any / multiple*	Multiple additional values that should be formatted according to the format string

Return: Number of characters written (*size_t*)

Example

```
gfx.printf("Slider Value: %d\r\n", sliderVal);
```

7.5. MoveTo

Moves the cursor to the new position specified by **(x, y)**.

Syntax: `gfx.MoveTo(x, y);`

Argument	Type	Description
x	int16_t	Horizontal pixel position of the print cursor
y	int16_t	Vertical pixel position of the print cursor

Return: None (*void*)

Example

```
gfx.MoveTo(50, 30);
int16_t CursorX = gfx.getX();
int16_t CursorY = gfx.getY();

// Get cursor X and Y positions then print their values
gfx.print("X-Position: "); gfx.println(CursorX);
gfx.print("Y-Position: "); gfx.println(CursorY);
```

7.6. getX

Returns the current position of the cursor in the X-axis

Syntax: `gfx.getX();`

Return: Current cursor position in X-axis (*int16_t*)

Example

```
gfx.MoveTo(50, 30);
int16_t CursorX = gfx.getX();

// Get cursor X position then print its value
gfx.print("X-Position: "); gfx.println(CursorX);
```

7.7. getY

Returns the current position of the cursor in the Y-axis

Syntax: `gfx.getY();`

Return: Current cursor position in Y-axis (*int16_t*)

Example

```
gfx.MoveTo(50, 30);
int16_t CursorY = gfx.getY();

// Get cursor Y position then print its value
gfx.print("Y-Position: "); gfx.println(CursorY);
```

7.8. Font

7.8.1. Get Font

Get the currently set font type or internal/system font.

Possible return values are:

Value	Description
2	System Font 2
1	System Font 1
0	Font file in uSD, opened using Open4dFont
-1	Font array in program space

Syntax: `gfx.Font();`

Return: Font type (*int8_t*)

Example

```
gfx.Font(FONT2); // Sets FONT2 as font to be used for printing text

int8_t fontID = gfx.Font(); // Get current font then print its value
gfx.print("Current Font: "); gfx.println(fontID);
```

7.8.2. Set System Font

Sets a system font to use for printing text.

Constants	Value
`FONT1`	1
`FONT2`	2

Syntax: `gfx.Font(f);`

Argument	Type	Description
f	uint8_t	System font to use when printing text

Return: None (*void*)

Example

```
gfx.Font(FONT2); // Sets FONT2 as font to be used for printing text
```

7.8.3. Set Font File

Sets a GCI/IFont file in the uSD as font to use for printing text.

Syntax: `gfx.Font(f);`

Argument	Type	Description
f	gfx4d_font	Font opened using Open4dFont

Return: None (*void*)

Example

```
gfx4d_font arial = gfx.Open4dFont("Arial.IFont");
gfx.Font(arial);
// Sets "Arial.IFont" as font to be used for printing text
```

7.8.4. Set Font Array

Sets a byte array to use as font for printing text.

Syntax: `gfx.Font(f [, compressed]);`

Argument	Type	Description
f	const uint8_t	Byte array containing font data in the GCI/IFont format of WS4's compressed font format
compressed (optional)	boolean	Specifies whether the data is compressed or following the original GCI/IFont format, default: `false`

Return: None (*void*)

Example

```
const uint8_t sample_data[BYTE_COUNT] = {
    // ... font data here
};

void setup() {
    gfx.begin(); // Initialize the display
    gfx.Cls();
    gfx.ScrollEnable(true);
    gfx.BacklightOn(true);
    gfx.Orientation(PORTRAIT);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE);
    gfx.Font(sample_data, true);
    gfx.TextSize(1);
}
```

7.9. TextSize

Sets the text width and height multiplier. Text will be printed magnified horizontally and vertically by this factor.

Syntax: `gfx.TextSize(s);`

Argument	Type	Description
s	uint8_t	Specifies the text width and height multiplier

Return: None (*void*)

Example

```
gfx.TextSize(1);
// Sets the current text width and height multiplier to 1
```

7.10. TextColor

Sets the text foreground and background colour for printing text.

If background colour is not specified, this function will treat it as transparent.

Syntax: `gfx.TextColor(c [, b]);`

Argument	Type	Description
c	uint16_t	Specifies the text foreground colour
b (optional)	uint16_t	Specifies the text background colour

Return: None (*void*)

Example

```
gfx.TextColor(WHITE);
// sets the text foreground colour to WHITE

gfx.TextColor(WHITE, BLACK);
// sets the text foreground colour to WHITE
// and the text background colour to BLACK
```

7.11. BGcolour

Set text background colour

Syntax: `gfx.BGcolour(c);`

Argument	Type	Description
c	uint16_t	Specifies the text background colour

Return: None (*void*)

Example

```
gfx.BGcolour(BLACK);
// sets the text background colour to BLACK
```

7.12. FGcolour

Set text foreground colour

Syntax: `gfx.FGcolour(c);`

Argument	Type	Description
c	uint16_t	Specifies the text foreground colour

Return: None (*void*)

Example

```
gfx.FGcolour(WHITE);
// sets the text foreground colour to WHITE
```

7.13. TextWrap

Text wrapping is **ENABLED** if mode is `true` otherwise text wrapping is **DISABLED**

Syntax: `gfx.TextWrap(mode);`

Argument	Type	Description
mode	boolean	Use `true` to **ENABLE** and `false` to **DISABLE**

Return: None (*void*)

Example

```
gfx.TextWrap(false);    // Disable text wrapping

gfx.TextWrap(true);     // Enable text wrapping
```

Note

The default mode is **ENABLED**.

7.14. FontHeight

Return height of current selected font without the multiplier

Syntax: `gfx.FontHeight();`

Return: Font height without multiplier (*int8_t*)

Example

```
gfx.Font(2);
int8_t fontHeight = gfx.FontHeight();
```

7.15. FontStyle

Select font style for Font1 characters.

Valid styles are:

- SOLID
- DOTMATRIXROUND
- DOTMATRIXLED
- DOTMATRIXSQUARE
- DOTMATRIXFADE

Syntax: `gfx.FontStyle(ctyp);`

Argument	Type	Description
ctyp	uint8_t	Specifies the style to use when using System Font 1

Return: None (*void*)

Example

```
gfx.Font(1);
gfx.FontStyle(DOTMATRIXROUND); // Sets FONT1 style to DOTMATRIXROUND
```

7.16. Opacity

Sets whether to draw background color when printing font characters or not.

Syntax: `gfx.Opacity(opacity);`

Argument	Type	Description
opacity	boolean	Enable (`true`) or disable (`false`) font background transparency

Return: None (*void*)

Example

```
// Enable font background transparency
gfx.Opacity(true);

// Disable font background transparency
gfx.Opacity(false);
```

7.17. UserCharacter

Draw user defined character using data array of size l at x, y, with foreground colour fc and background colour bc.

Syntax: `gfx.UserCharacter(data, l, x, y, fc, bc);`

Argument	Type	Description
data	uint32_t*	Array containing character data
l	uint8_t	Pixel size multiplier to use for displaying the character
x	int16_t	Horizontal pixel position of the character
y	int16_t	Vertical pixel position of the character
fc	uint16_t	16-bit RGB565 foreground color
bc	uint16_t	16-bit RGB565 background color

Return: None (*void*)

Example

```
gfx.UserCharacter(data, l, x, y, fc, bc);
```

7.18. UserCharacterBG

7.18.1. Option 1

Draw a user defined character with a specified (ui) gci image number being drawn as the character background

- Setting draw to false will draw backround only.
- Setting draw to true will draw background and character.

Syntax: `gfx.UserCharacterBG(ui, data, ucsize, ucx, ucy, colour, draw);`

Argument	Type	Description
ui	int8_t	Specifies the GCI index for the background
data	uint32_t *	Array containing character data
ucsize	uint8_t	Pixel size multiplier to use for displaying the character
ucx	int16_t	Horizontal pixel position of the character
ucy	int16_t	Vertical pixel position of the character
color	uint16_t	16-bit RGB565 foreground color
draw	boolean	Determines whether to draw the the character (`true`) or only the background (`false`)

Return: None (*void*)

Example

```
gfx.UserCharacterBG(ui, data, ucsize, ucx, ucy, colour, draw);
```

7.18.2. Option 2

Draw a user defined character with a downloaded gci image file being drawn as the character background. bgindex is set to 0 if a a single image exists in the gci image file or can be set to other images if they exist

- Setting draw to false will draw backround only.
- Setting draw to true will draw background and character.

Syntax: `gfx.UserCharacterBG(data, ucsize, ucx, ucy, colour, draw, bgindex);`

Argument	Type	Description
data	uint32_t *	Array containing character data
ucsize	uint8_t	Pixel size multiplier to use for displaying the character
ucx	int16_t	Horizontal pixel position of the character
ucy	int16_t	Vertical pixel position of the character
color	uint16_t	16-bit RGB565 foreground color
draw	boolean	Determines whether to draw the the character (`true`) or only the background (`false`)
bgindex	uint32_t	Specifies the GCI index for the background

Return: None (*void*)

Example

```
gfx.UserCharacterBG(data, ucsize, ucx, ucy, colour, draw, bgindex);
```

7.19. putstr

Prints string to the current cursor position

Syntax: `gfx.putstr(text);`

Argument	Type	Description
text	String const char *	Text to print at current cursor position

Return: None (*void*)

Example

```
gfx.putstr("4D Systems");
```

7.20. putstrXY

Prints string to the specified position

Syntax: `gfx.putstrXY(xpos, ypos, text);`

Argument	Type	Description
xpos	int	Horizontal pixel position
ypos	int	Vertical pixel position
text	String const char *	Text to print at specified position

Return: None (*void*)

Example

```
gfx.putstrXY(50, 100, "4D Systems");
```

7.21. putch

Prints a single character to the current cursor position

Syntax: `gfx.putch(chr);`

Argument	Type	Description
chr	uint8_t	Character to print at current cursor position

Return: None (*void*)

Example

```
gfx.putch('4');
gfx.putch('D');
```

7.22. putchXY

Prints a single character to the specified position

Syntax: `gfx.putchXY(xpos, ypos, chr);`

Argument	Type	Description
xpos	int	Horizontal pixel position
ypos	int	Vertical pixel position
chr	uint8_t	Character to print at specified position

Return: None (*void*)

Example

```
gfx.putch(50, 20, '4');
gfx.putch('D');
```

7.23. charWidth

Returns the width of the character in pixels with the font size multiplier

Syntax: `gfx.charWidth(ch);`

Argument	Type	Description
ch	uint8_t	Character to get the print width of

Return: Print width of the character (*int*)

Example

```
int width = gfx.charWidth('4') + gfx.charWidth('D');
```

7.24. charHeight

Returns the height of the character with the font size multiplier

Syntax: `gfx.charHeight(ch);`

Argument	Type	Description
ch	uint8_t	Character to get the print height of

Return: Print height of the character (*int*)

Example

```
int height = gfx.charHeight('4');
```

7.25. strWidth

Returns the width of the string with the font size multiplier

Syntax: `gfx.strWidth(ts);`

Argument	Type	Description
ts	String const char *	Text to get the print width of

Return: Print width of the string (*int*)

Example

```
int width = gfx.strWidth("4D Systems");
```

8. Text Window Functions

8.1. TWprintln

Prints the value to the TextWindow followed by new line

Syntax: `gfx.TWprintln(value);`

Argument	Type	Description
value	String char * int8_t uint8_t int16_t uint16_t int32_t uint32_t int64_t uint64_t float	Value to print

Return: None (*void*)

Example

```
gfx.TWprint(4);
gfx.TWprintln("D Systems");
```

8.2. TWprint

Prints the value to the TextWindow

Syntax: `gfx.TWprint(value);`

Argument	Type	Description
value	String char * int8_t uint8_t int16_t uint16_t int32_t uint32_t int64_t uint64_t float	Value to print

Return: None (*void*)

Example

```
gfx.TWprint(4);
gfx.TWprintln("D Systems");
```

8.3. GetCommand

Retrieves the text entered in text window since previous carriage return sent

Syntax: `gfx.GetCommand();`

Return: Command entered in the text window (*string*)

Example

```
String cmd = gfx.GetCommand();
// Get the last entered command from the Text Window
```

8.4. TWtextcolor

Sets the print color when printing to the text window

Syntax: `gfx.TWtextcolor(twc);`

Argument	Type	Description
twc	uint16_t	Text foreground color

Return: None (*void*)

Example

```
String cmd = gfx.GetCommand();
// Get the last entered command from the Text Window
```

8.5. TWMoveTo

Sets the cursor position for printing in the text window

Syntax: `gfx.TWMoveTo(twcrx, twcry);`

Argument	Type	Description
twcrx	uint8_t	Horizontal cursor position
twcry	uint8_t	Vertical cursor position

Return: whether this is successful (`true`) or not (`false`) (*boolean*)

Example

```
result = gfx.TWMoveTo(10, 10);
```

8.6. TWprintAt

Print a string at the specified position in the text window

Syntax: `gfx.TWprintAt(pax, pay, istr);`

Argument	Type	Description
pax	uint8_t	Horizontal cursor position
pay	uint8_t	Vertical cursor position
istr	String	Text to print at specified position

Return: None (*void*)

Example

```
gfx.TWprintAt(10, 5, "4D Systems");
```

8.7. TWwrite

Write a character to Text Window.

Syntax: `gfx.TWwrite(c);`

Argument	Type	Description
c	char	Character to write to the text window

Return: None (*void*)

Example

```
gfx.TWwrite('4');
```

8.8. TWcursorOn

Enables cursor in Text Window

Syntax: `gfx.TWcursorOn(mode);`

Argument	Type	Description
mode	boolean	Specifies whether to enable (`true`) or disable (`false`) cursor

Return: None (*void*)

Example

```
gfx.TWcursorOn(true);
```

8.9. TWcls

Clear TextWindows of characters and reset cursor.

Syntax: `gfx.TWcls();`

Return: None (*void*)

Example

```
gfx.TWcls();
```

8.10. TWcolor

Sets the specified foreground colour (fcol) and background colour (bcol) as the colours of the text in the text window.

If background colour is not specified, this function will treat it as transparent.

Additionally, when gfx.TextWindowRestore is used, the text window background colour will match the background colour set by this function.

Syntax: `gfx.TWcolor(fcol, bcol);`

Argument	Type	Description
fcol	uint16_t	Specifies the text foreground color for text windows
bcol	uint16_t	Specifies the text background color for text windows

Return: None (*void*)

Example

```
gfx.Orientation(LANDSCAPE);
gfx.TextWindow(25, 25, 270, 190, BLACK, SILVER, BROWN);
// Creates a SILVER text window @(25,25) with:
// width of 190 and height of 270 pixels and BROWN frame
// The text printed in this text window is colour BLACK
gfx.TWprintln("1. 4D Systems");

gfx.TWcolor(BROWN);
// The text that will be printed next will be colour BROWN
gfx.TWprintln("2. 4D Systems");

gfx.TWcolor(LIME,GRAY);
// The text that will be printed next will be:
// colour LIME with GRAY background

gfx.TWprintln("3. 4D Systems");
```

8.11. TextWindowImage

Create Text window at **x**, **y**, with dimensions **w**, **h** and text colour **tcolour**, GCI image **img**, with optional frame colour **fcolour** to add frame.

Background image will be automatically replaced with image data as text changes

Syntax: `gfx.TextWindowImage(x, y, w, h, tc, img, fc);`

Argument	Type	Description
x	int16_t	Horizontal pixel position of the top left of the text window
y	int16_t	Vertical pixel position of the top left of the text window
w	int16_t	Width of the text window
h	int16_t	Height of the text window
tc	uint16_t	Text foreground colour
img	uint16_t	Specifies the GCI index to use as background
fc (optional)	uint16_t	Frame colour

Return: None (*void*)

Example

```
gfx.TextWindowImage(10, 20, 200, 100, BLACK, iForm3);
```

8.12. TextWindow

Creates a text window at x, y, with dimensions w, h, text colour tc, background colour bc, and frame in colour fc.

If no frame colour is specified, then no frame will not be rendered.

Syntax: `gfx.TextWindow(x, y, w, h, tc, bc [, fc]);`

Argument	Type	Description
x	int16_t	Horizontal pixel position of the top left of the text window
y	int16_t	Vertical pixel position of the top left of the text window
w	int16_t	Width of the text window
h	int16_t	Height of the text window
tc	uint16_t	Text foreground colour
bc	uint16_t	Text background colour
fc (optional)	uint16_t	Frame colour

Return: None (*void*)

Example

```
gfx.TextWindow(25,25, 190,270, BLACK, SILVER, DARKGRAY);
// Creates a SILVER text window @(25,25) with:
// width of 190 and height of 270 pixels
// and DARKGRAY frame
```

8.13. TWenable

Enables or Disables drawing to the text window

Syntax: `gfx.TWenable(mode);`

Argument	Type	Description
mode	boolean	Enables (`true`) or disables (`false`) writing to the text window

Return: None (*void*)

Example

```
gfx.TWenable(false);
```

8.14. TextWindowRestore

Restore a previously created text window and its contents.

Syntax: `gfx.TextWindowRestore();`

Return: None (*void*)

Example

```
gfx.TextWindow(25,25, 190,270, BLACK, SILVER, DARKGRAY);
// Creates a SILVER text window @(25,25) with:
// width of 190 and height of 270 pixels
// and DARKGRAY frame
// The text printed in this text window is colour BLACK
gfx.Cls();
delay(1000);
// Retrieve deleted text window
gfx.TextWindowRestore();
```

Note

Contents cleared using gfx.TWcls will not be restored

9. Scroll Functions

9.1. setScrollArea

9.1.1. By Specifing Top and Bottom

Set scroll window specified by the top and bottom pixel positions.

This assumes the horizontal area is the full width of the screen.

Syntax: `gfx.setScrollArea(y1, y2);`

Argument	Type	Description
y1	int	Top pixel position of the scroll area
y2	int	Bottom pixel position of the scroll area

Return: None (*void*)

Example

```
gfx.setScrollArea(10, 229);
```

9.1.2. By Specifing a Rectangle

Set scroll window specified by the rectangle with diagonal at (x1, y1) to (x2, y2)

Syntax: `gfx.setScrollArea(x1, y1, x2, y2);`

Argument	Type	Description
x1	int	Horizontal pixel position of the top-left pixel of the scroll area
y1	int	Vertical pixel position of the top-left pixel of the scroll area
x2	int	Horizontal pixel position of the bottom-right pixel of the scroll area
y2	int	Vertical pixel position of the bottom-right pixel of the scroll area

Return: None (*void*)

Example

```
gfx.setScrollArea(10, 0, 229, 319);
```

9.2. ScrollEnable

Enable or disable auto scrolling

Syntax: `gfx.ScrollEnable(mode);`

Argument	Type	Description
mode	boolean	Enables (`true`) or disables (`false`) scrolling

Return: None (*void*)

Example

```
gfx.Orientation(PORTRAIT);  // Sets Orientation to PORTRAIT
gfx.ScrollEnable(false);    // Disables Hardware Scrolling
gfx.ScrollEnable(true);     // Enables Hardware Scrolling
```

Note

Scrolling is disabled by default

9.3. Scroll

If scroll is enabled, this function scrolls the display by the specified number of pixels.

If SmoothScrollSpeed has been set, it will be scrolled pixel line by pixel line, delayed by the value set using gfx.SmoothScrollSpeed in milliseconds

Syntax: `gfx.Scroll(steps);`

Argument	Type	Description
steps	int	Specifies the number of pixels

Return: None (*void*)

Example

```
gfx.Scroll(10); // Scroll the screen by 10 pixels
```

9.4. setScrollBlankingColor

Set blanking line colour after scroll has moved. This can be used to match the current text background colour.

Syntax: `gfx.setScrollBlankingColor(colour);`

Argument	Type	Description
colour	uint16_t	Color to use for scroll blanking

Return: None (*void*)

Example

```
gfx.setScrollBlankingColor(WHITE);
```

9.5. SmoothScrollSpeed

Smoothens the scroll animation for the automatic scrolling that occurs when the text being printed is going outside of the display area.

Using zero will scroll height defined by character height in one step.

Syntax: `gfx.SmoothScrollSpeed(speed);`

Argument	Type	Description
speed	uint8_t	Speed to use for scrolling

Return: None (*void*)

Example

```
gfx.Orientation(PORTRAIT);  // Sets Orientation to PORTRAIT
gfx.SmoothScrollSpeed(7);   // Change Scroll Speed to 7
```

Note

Default delay is 5

9.6. setScrollDirection

Set scroll direction to the following:

- SCROLL_UP
- SCROLL_DOWN
- SCROLL_LEFT
- SCROLL_RIGHT

Syntax: `gfx.setScrollArea(scrDir);`

Argument	Type	Description
scrDir	uint8_t	Specifies the direction

Return: None (*void*)

Example

```
gfx.setScrollArea(SCROLL_UP);
```

10. 4D Graphics Functions

10.1. Open4dGFX

10.1.1. GCI/DAT File

Opens 4D DAT file for parsing and GCI file for read using filename without extension.

Syntax: `gfx.Open4dGFX(filename);`

Argument	Type	Description
filename	String	Specifies the filename of the GCI/DAT file without extension

Return: None (*void*)

Example

```
gfx.Open4dGFX("filename"); // Opens filename.dat and filename.gci
```

Note

Filename should have no extension. Both GCI and DAT file should share the same filename. Also, 4D Graphics files follow the 8.3 DOS format

10.1.2. GCI/DAT Arrays

Loads a 4D DAT array and GCI DATA array for reading 4D widgets

Syntax: `gfx.Open4dGFX(DATa, DATlen, GCIa, GCIlen);`

Argument	Type	Description
DATa	const uint8_t *	Specifies the array containing the DAT information
DATlen	uint32_t	Specifies the size of the DAT array
GCIa	const uint8_t *	Specifies the array containing the GCI information
GCIlen	uint32_t	Specifies the size of the DAT array

Return: None (*void*)

Example

```
gfx.Open4dGFX(dat_arr, dat_len, gci_arr, gci_len);
```

10.2. Close4dGFX

Closes the DAT and GCI files opened by gfx.Open4dGFX(filename)

Syntax: `gfx.Close4dGFX();`

Return: None (*void*)

Example

```
gfx.Close4dGFX();
```

10.3. Open4dFont

Opens a Workshop4 font (IFont/GCI format) for reading.

This returns a reference to the opened font and can be used with gfx.Font

Syntax: `gfx.Open4dFont(filename);`

Argument	Type	Description
filename	String	Complete filename of the font file in SD card

Return: Reference to the opened font (*gfx4d_font*)

Example

```
gfx4d_font customFont;

void setup() {
    gfx.begin(); // Initialize the display
    gfx.Cls();
    gfx.ScrollEnable(true);
    gfx.BacklightOn(true);
    gfx.Orientation(PORTRAIT);
    gfx.SmoothScrollSpeed(5);
    gfx.TextColor(WHITE); gfx.Font(2); gfx.TextSize(1);

    customFont = gfx.Open4dFont("custom.IFont");
}
```

10.4. DrawImageFile

Draw widget from GCI file

Syntax: `gfx.DrawImageFile(Fname);`

Argument	Type	Description
Fname	String	Filename of the GCI file in the SD card

Return: None (*void*)

Example

```
gfx.DrawImageFile("sample.gci");
```

10.5. DrawImageArray

Draw widget from GCI array

Syntax: `gfx.DrawImageArray(ImageArray);`

Argument	Type	Description
ImageArray	uint8_t * uint16_t *	Specifies the array containing the GCI data

Return: None (*void*)

Example

```
gfx.DrawImageArray(sample_gci_arr);
```

10.6. getImageValue

Returns the current value of a GCI widget

Syntax: `gfx.getImageValue(ui);`

Argument	Type	Description
ui	uint16_t	Index of GCI object

Return: Current value of the GCI widget (*int16_t*)

Example

```
uint16_t gauge2Val = gfx.getImageValue(iGauge2);
```

10.7. imageGetWord

Queries the current selected parameter of the GCI widget.

Parameters can be IMAGE_XPOS, IMAGE_YPOS, IMAGE_WIDTH or IMAGE_HEIGHT

Syntax: `gfx.imageGetWord(img, controlIndex);`

Argument	Type	Description
img	uint16_t	Index of GCI object
controlIndex	byte	Parameter to query

Return: Value of the selected parameter of the GCI widget (*int*)

Example

```
int gaugeWidth = gfx.imageGetWord(iGauge2, IMAGE_WIDTH);
```

10.8. imageSetWord

Sets the selected parameter(s) of the GCI widget.

Parameters can be IMAGE_XPOS, IMAGE_YPOS, or IMAGE_XYPOS. When using IMAGE_XYPOS, two values are required.

Syntax: `gfx.imageSetWord(img, controlIndex, val1 [, val2]);`

Argument	Type	Description
img	uint16_t	Index of GCI object
controlIndex	byte	Parameter to set
val1	int	Value to set the selected parameter to
val2 (optional)	int	Additional value when setting IMAGE_XYPOS

Return: None (*void*)

Example

```
gfx.imageSetWord(iGauge2, IMAGE_XPOS, 100);
```

10.9. getNumberofObjects

Get the number of GCI objects found in the GCI file opened with gfx.Open4dGFX

Syntax: `gfx.getNumberofObjects();`

Return: Number of GCI objects found (*uint16_t*)

Example

```
uint16_t gciCount = gfx.getNumberofObjects();
```

10.10. setGCIsystem

Sets the GCI system to uSD or from a data array

Valid options are:

- `GCI_SYSTEM_USD` - sets GCI system to uSD
- `GCI_SYSTEM_PROGMEM` - sets GCI system to data array

Syntax: `gfx.setGCIsystem(gs);`

Argument	Type	Description
gs	uint8_t	Indicates whether to set for uSD or progmem

Return: None (*void*)

Example

```
gfx.setGCIsystem(GCI_SYSTEM_USD);
```

10.11. getGCIsystem

Returns currently selected GCI system

Syntax: `gfx.getGCIsystem();`

Return: `GCI_SYSTEM_USD` or `GCI_SYSTEM_PROGMEM` (*uint8_t*)

Example

```
uint8_t gciSys = gfx.getGCIsystem();
```

10.12. UserImage

Draw a single frame GCI widget in the default location or as specified by (x, y)

Syntax: `gfx.UserImage(id [, x, y]);`

Argument	Type	Description
id	uint16_t	Index of the GCI object
x (optional)	int	Optional horizontal position to draw the image
y (optional)	int	Optional vertical position to draw the image

Return: None (*void*)

Example

```
gfx.UserImage(iImage1);
// Show iImage1

gfx.UserImage(iImage2,50,50);
// Show iImage2 at (50,50)
```

Note

The GCI and DAT files should have been previously opened with the function gfx.Open4dGFX functions

10.13. UserImages

Displays the frame of the GCI object specified by 'id'.

The following can be optionally specified:

- **Horizontal offset** - useful for Spectrum and Digits type widgets
- **Alternate X and Y positions** - useful for temporarily drawing widgets in alternate positions
- **Both horizontal offset and alternate X and Y positions** - useful for temporarily drawing spectrum and digits in alternate positions

Syntax: `gfx.UserImages(id, frame [, offset, x, y]);`

- `gfx.UserImages(id, frame);`
- `gfx.UserImages(id, frame, offset);`
- `gfx.UserImages(id, frame, newx, newy);`
- `gfx.UserImages(id, frame, offset, altx, alty);`

Argument	Type	Description
id	uint16_t	Index of the GCI object
frame	int16_t	Specifies the frame number of the selected GCI object
offset (optional)	int	Optional horizontal offset to draw the GCI object
x (optional)	int16_t	Optional horizontal position to draw the GCI object
y (optional)	int16_t	Optional vertical position to draw the GCI object

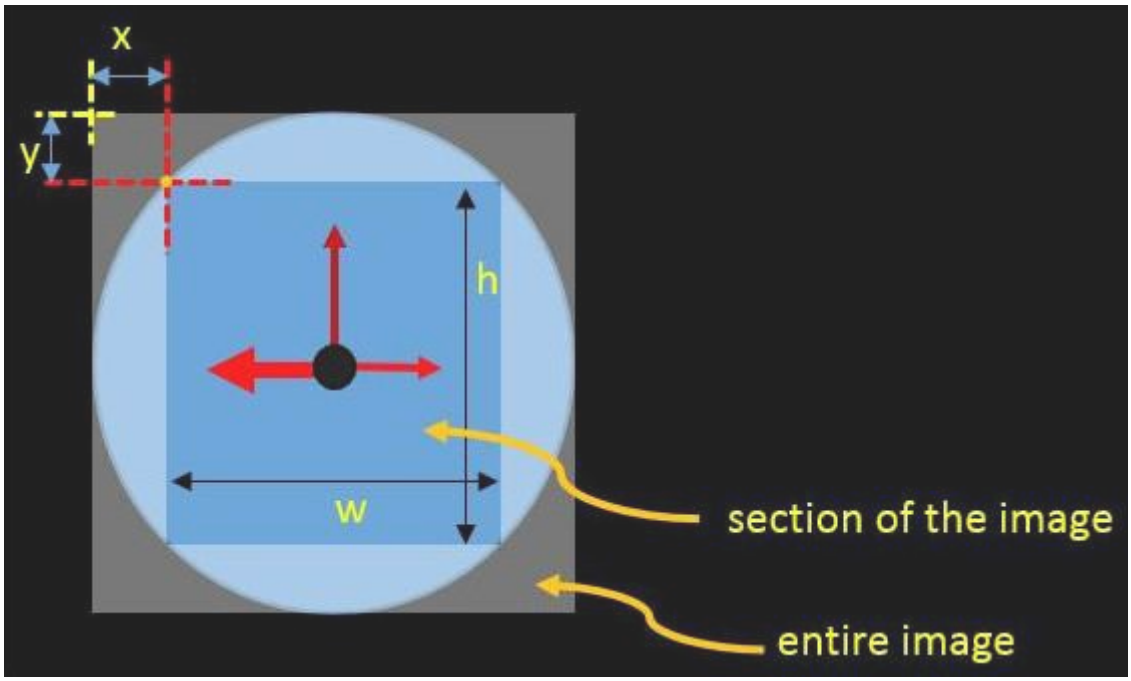
Return: None (*void*)

Note

The GCI and DAT files should have been previously opened with the function gfx.Open4dGFX functions

10.14. UserImageDR

Draws a section of the selected single frame GCI object (**id**). The section starts at **x** and **y** and has a width of **w** and height of **h**.



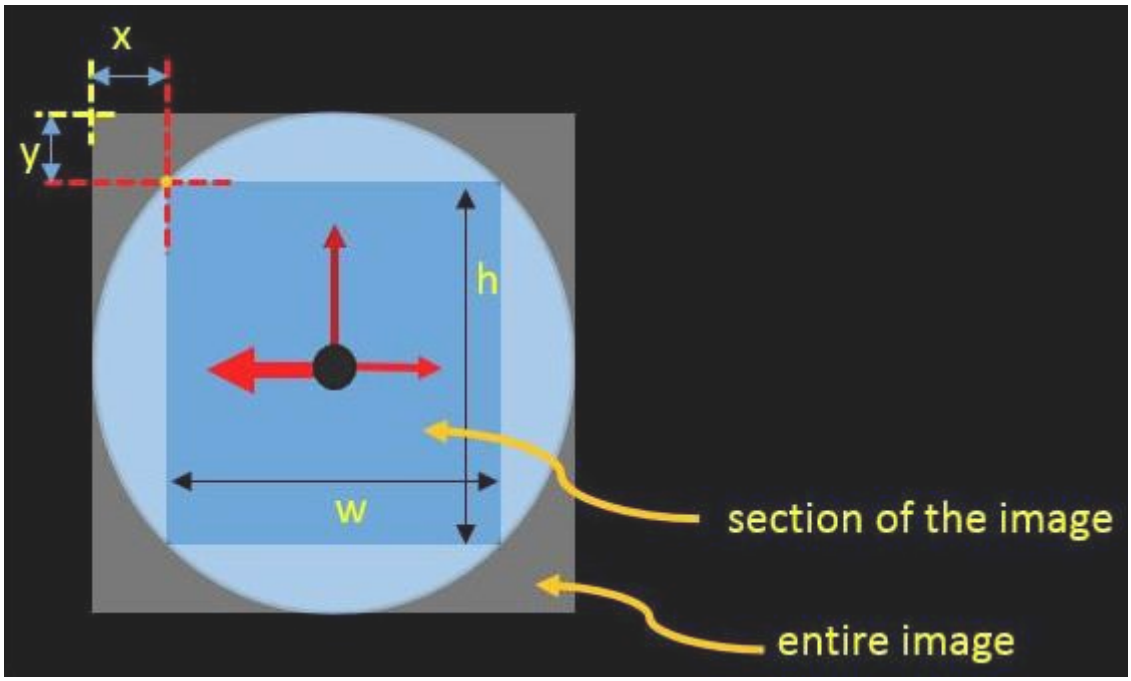
Syntax: `gfx.UserImageDR(id, x, y, w, h, altx, alty);`

Argument	Type	Description
id	uint16_t	Index of the GCI object
frame	int	Specifies the frame number of the selected GCI object
x	int16_t	Specifies the horizontal position of the top left pixel of the section relattve to the top left pixel of the image
y	int16_t	Specifies the vertical position of the top left pixel of the section relattve to the top left pixel of the image
w	int16_t	Specifies the width of the section
h	int16_t	Specifies the height of the section
altx	int16_t	Specifies the alternate horizontal position for the GCI object
alty	int16_t	Specifies the alternate vertical position for the GCI object

Return: None (*void*)

10.15. UserImagesDR

Draws a section of the **frame** of the selected GCI object (**id**). The section starts at **x** and **y** and has a width of **w** and height of **h**.



Syntax: `gfx.UserImagesDR(id, frame, x, y, w, h);`

Argument	Type	Description
id	uint16_t	Index of the GCI object
frame	int	Specifies the frame number of the selected GCI object
x	int16_t	Specifies the horizontal position of the top left pixel of the section relattve to the top left pixel of the image
y	int16_t	Specifies the vertical position of the top left pixel of the section relattve to the top left pixel of the image
w	int16_t	Specifies the width of the section
h	int16_t	Specifies the height of the section

Return: None (*void*)

Note

The GCI and DAT files should have been previously opened with the function gfx.Open4dGFX functions

10.16. UserImageDRcache

Draw section of a single frame GCI widget while utilizing cache

Syntax: `gfx.UserImageDRcache(id, x, y, w, h, altx, alty);`

Argument	Type	Description
id	uint16_t	Index of the GCI object
frame	int	Specifies the frame number of the selected GCI object
x	int16_t	Specifies the horizontal position of the top left pixel of the section relattve to the top left pixel of the image
y	int16_t	Specifies the vertical position of the top left pixel of the section relattve to the top left pixel of the image
w	int16_t	Specifies the width of the section
h	int16_t	Specifies the height of the section
altx	int16_t	Specifies the alternate horizontal position for the GCI object
alty	int16_t	Specifies the alternate vertical position for the GCI object

Return: None (*void*)

10.17. setCacheSize

Sets the cache size to use with gfx.UserImageDRcache

Syntax: `gfx.setCacheSize(cs);`

Argument	Type	Description
cs	uint32_t	Size of cache to use with gfx.UserImageDRcache

Return: None (*void*)

Table

```
gfx.setCacheSize(2048000);
```

10.18. LedDigitsDisplaySigned

Displays a signed or unsigned numeric value using graphics from Led or Custom Digits.

This routine only works with all numbers and must be generated with project option 'Allow -ve Led and Custom Digits and leading blanks on Custom Digits' is enabled. If not, use gfx.LedDigitsDisplay instead

Position can be overriden by specifying x and y values.

Each of the Leddigits objects and Customdigits objects is composed of 2 GCI objects. A Leddigits object at index 1 si composed of GCI objects named iLeddigits1 and iiLeddigits1. The first one being a single frame containing the whole digits area as seen in Workshop4's WYSIWYG. The other GCI object is composed of multiple frames containing the digits 0-9, a blank space and a negative sign depending on the setting enabled in the project.

It is ideal to simply let Workshop4 generate this code using the Paste Code functionality.

Syntax: `gfx.LedDigitsDisplaySigned(newval, index, Digits, MinDigits, WidthDigit, LeadingBlanks [, x, y]);`

Argument	Type	Description
newval	int64_t	Value to display using the Digits widget
index	uint16_t	Specifies which LedDigits object to update
Digits	int16_t	Maximum number of digits in the widget
MinDigits	int16_t	Minimum number of digits in the object
WidthDigit	int16_t	Width of each digit image
LeadingBlanks	int16_t	Specifies whether to display leading blanks or not
x (optional)	int16_t	Optional horizontal position to draw the GCI object
y (optional)	int16_t	Optional vertical position to draw the GCI object

Return: None (*void*)

10.19. LedDigitsDisplay

Displays a signed or unsigned numeric value using graphics from Led or Custom Digits.

This routine only works with all numbers and must be generated with project option 'Allow -ve Led and Custom Digits and leading blanks on Custom Digits' is disabled. If not, use gfx.LedDigitsDisplaySigned instead

Position can be overriden by specifying x and y values.

Each of the Leddigits objects and Customdigits objects is composed of 2 GCI objects. A Leddigits object at index 1 is composed of GCI objects named iLeddigits1 and iiLeddigits1. The first one being a single frame containing the whole digits area as seen in Workshop4's WYSIWYG. The other GCI object is composed of multiple frames containing the digits 0-9, a blank space and a negative sign depending on the setting enabled in the project.

It is ideal to simply let Workshop4 generate this code using the Paste Code functionality.

Syntax: `gfx.LedDigitsDisplay(newval, index, Digits, MinDigits, WidthDigit, LeadingBlanks [, x, y]);`

Argument	Type	Description
newval	int64_t	Value to display using the Digits widget
index	uint16_t	Specifies which LedDigits object to update
Digits	int16_t	Maximum number of digits in the widget
MinDigits	int16_t	Minimum number of digits in the object
WidthDigit	int16_t	Width of each digit image
LeadingBlanks	int16_t	Specifies whether to display leading blanks or not
x (optional)	int16_t	Optional horizontal position to draw the GCI object
y (optional)	int16_t	Optional vertical position to draw the GCI object

Return: None (*void*)

Example

```
gfx.LedDigitsDisplay(50, iLeddigits1, 4, 3, 20, false);
// Writes the value 50 to the iLedDigits1 object

gfx.LedDigitsDisplay(50, iLeddigits1, 4, 3, 20, false, 5, 50);
// Writes the value 50 to the iiLeddigits2 object.
// The object will then be shown at (5,50)
```

10.20. PrintImage

Print image from GCI file at 'index' at current cursor position.

Syntax: `gfx.PrintImage(index);`

Argument	Type	Description
index	uint16_t	Specifies the GCI object to print

Return: None (*void*)

Table

```
gfx.MoveTo(50, 50);
gfx.PrintImage(iImage0);
```

10.21. PrintImageFile

Prints the first frame of the first object from the specified GCI file at the current cursor position

Syntax: `gfx.PrintImageFile(filename);`

Argument	Type	Description
filename	String	Specifies the GCI file to print an image from

Return: None (*void*)

Example

```
gfx.MoveTo(50, 50);
gfx.PrintImageFile("filename.GCI");
// Prints the 1st frame of the 1st object from filename.GCI
```

Note

Unlike the function gfx.Open4dGFX, this function requires the extension of the file

10.22. UserImageHide

Hides the GCI widget using the specified colour or BLACK if no colour is specified. Use -1 for hndl to apply to all GCI widgets.

For touch capable display modules, this also disables touch.

Syntax: `gfx.UserImageHide(hndl [, colour]);`

Argument	Type	Description
hndl	int	Specifies the widget or all widgets (-1) to hide
colour (optional)	uint16_t	Optional background colour to use to hide the widget

Return: None (*void*)

Example

```
gfx.UserImageHide(iImage1);         // Hide iImage1 using default BLACK colour
gfx.UserImageHide(iImage2, WHITE);  // Hide iImage2 using colour WHITE
```

10.23. UserImageHideBG

Hides the GCI widget using the specified background image objBG. Use -1 for hndl to apply to all GCI widgets.

For touch capable display modules, this also disables touch.

Syntax: `gfx.UserImageHideBG(hndl, objBG);`

Argument	Type	Description
hndl	int	Specifies the widget or all widgets (-1) to hide
objBG	int	Specifies the index of GCI background to use to hide the widget

Return: None (*void*)

Example

```
gfx.UserImageHide(iImage2, iForm1);  // Hide iImage2 using background image iForm1
```

10.24. getLastPointerPos

Get the x or y value of the change position of 2 & 3 image widgets

Syntax: `gfx.getLastPointerPos();`

Return: Value of the widget, x value if horizontal or y value if vertical (*int16_t*)

11. Wi-Fi Functions

11.1. DownloadFile

11.1.1. Full Address

Downloads the file from the specified web address and save it with the specified filename.

An optional certificate can be used.

Syntax: `gfx.DownloadFile(WebAddr, Fname [, sha1]);`

Argument	Type	Description
WebAddr	String	Full URL address of the file
Fname	String	Filename to store the file with
sha1 (optional)	const char *	Optional certificate to use

Return: None (*void*)

Example

```
gfx.DownloadFile("example.com/sample.txt", "sample.txt");
```

Note

It is advisable to follow the 8.3 DOS format

11.1.2. Address and Port

Downloads the file from the specified web address and port and save it with the specified filename.

An optional certificate can be used.

Syntax: `gfx.DownloadFile(Address, port, hfile, Fname [, certUsed, sha1]);`

Argument	Type	Description
Address	String	Server address hosting the file
port	uint16_t	Port number used by the server to host the file
hfile	String	File path of file to download from the server
Fname	String	Filename to save the file with
certUsed	bool	Optionally indicate to use a certificate
sha1	const char *	Optional certificate to use

Return: None (*void*)

Example

```
gfx.DownloadFile("example.com", 80, "/sample.txt", "sample.txt");
```

Note

It is advisable to follow the 8.3 DOS format

11.2. CheckDL

Check if the last download performed was successful

Syntax: `gfx.CheckDL();`

Return: `true` if successful otherwise `false` (*boolean*)

Example

```
gfx.DownloadFile("example.com/sample.txt", "sample.txt");
if (gfx.CheckDL()) {
    gfx.println("Download successful");
} else {
    gfx.println("Download failed");
}
```

12. Sprite Functions

12.1. SetMaxNumberSprites

Sets the maximum number of sprites in the sprites list

Syntax: `gfx.SetMaxNumberSprites(count);`

Argument	Type	Description
count	uint16_t	Specifies the new max number of sprites

Return: None (*void*)

Example

```
gfx.SetMaxNumberSprites(100);
```

12.2. SpriteAreaSet

Sets the area of the screen that sprites will be displayed in.

Syntax: `gfx.SpriteAreaSet(x, y, x1, y1);`

Argument	Type	Description
x	uint16_t	Horizontal pixel position of the top-left corner of the sprite area
x	uint16_t	Vertical pixel position of the top-left corner of the sprite area
x1	uint16_t	Horizontal pixel position of the bottom-right corner of the sprite area
y1	uint16_t	Vertical pixel position of the bottom-right corner of the sprite area

Return: None (*void*)

Example

```
gfx.SpriteAreaSet(0, 0, 239, 319);
```

12.3. SetNumberSprites

Sets number of sprites to use

Syntax: `gfx.SetNumberSprites(count);`

Argument	Type	Description
count	uint16_t	Number of sprites to use

Return: None (*void*)

Example

```
gfx.SetNumberSprites(20);
```

12.4. SpriteInit

Initialize the sprites data array

Syntax: `gfx.SpriteInit(data, len);`

Argument	Type	Description
data	uint16_t *	Pointer to the sprite data array
len	size_t	Size of sprite data

Return: `true` if successful otherwise `false` (*boolean*)

Example

```
gfx.SpriteInit(spriteData, sizeof(spriteData));
```

12.5. GetNumberSprites

Return the total number of initialized sprites

Syntax: `gfx.GetNumberSprites();`

Return: Total number of initialized sprites (*int*)

Example

```
initSprites = gfx.GetNumberSprites();
```

12.6. SpriteAdd

Add a sprite into a specified position in the sprite list.

Syntax: `gfx.SpriteAdd(pos, num, x, y, data);`

Argument	Type	Description
pos	int	Position in the sprite list
num	int	Number of the sprite in the sprite data
x	int	Horizontal start position of the sprite
y	int	Vertical start position of the sprite
data	uint16_t *	Sprite data array

Return: `true` if successful otherwise `false` (*boolean*)

12.7. SetSprite

Sets the position of sprite.

Syntax: `gfx.SetSprite(num, active, x, y, bscolor, sdata);`

Argument	Type	Description
num	int	Number of the sprite in the sprite data
active	bool	Specifies whether the sprite is active (`true`) or not (`false)
x	int	Horizontal new position of the sprite
y	int	Vertical new position of the sprite
bscolor	uint16_t	Background color to use to clear previous sprite position
data	uint16_t	Sprite data array

Return: None (*void*)

12.8. SpriteEnable

Enables or disables chosen sprite

Syntax: `gfx.SpriteEnable(snum, mode);`

Argument	Type	Description
snum	int	Number of the sprite in the sprite data
mode	bool	Enables (`true`) or disables (`false`) sprite

Return: None (*void*)

12.9. UpdateSprites

Updates all sprites in the previously set sprite area

Syntax: `gfx.UpdateSprites(bscolor, sdata);`

Argument	Type	Description
bscolor	uint16_t	Background colour to use for clearing old sprite position
sdata	uint16_t *	Sprite data array

Return: None (*void*)

12.10. SpriteUpdate

Update all sprites in the selected area

Syntax: `gfx.SpriteUpdate(tsx, tsy, tsx1, tsy1, bscolor, sdata);`

Argument	Type	Description
tsx	int16_t	Horizontal pixel position of the top-left corner of the selected area
tsy	int16_t	Vertical pixel position of the top-left corner of the selected area
tsx1	int16_t	Horizontal pixel position of the bottom-right corner of the selected area
tsy1	int16_t	Vertical pixel position of the bottom-right corner of the selected area
bscolor	uint16_t	Background colour to use for clearing old sprite position
sdata	uint16_t *	Sprite data array

Return: None (*void*)

12.11. SpriteGetPixel

Returns the colour of the pixel of the chosen sprite at x, y position

Syntax: `gfx.SpriteGetPixel(snum, xo, yo, tcolor, sdata);`

Argument	Type	Description
snum	int	Number of the sprite in the sprite data
xo	int	Horizontal position of selected pixel
yo	int	Vertical position of selected pixel
tcolor	uint16_t	Colour to test against as transparent
sdata	uint16_t *	Sprite data array

Return: Pixel colour of the chosen sprite at x, y position (*uint16_t*)

12.12. SpriteGetPalette

Return the 16-bit colour in the palette array position specified by pnumber

Syntax: `gfx.SpriteGetPalette(pnumber);`

Argument	Type	Description
pnumber	int	Palette array position

Return: 16-bit colour in the palette array position specified (*uint16_t*)

12.13. GetSpritesAt

Queries the sprites at the current x,y position to a created slist array

Syntax: `gfx.GetSpritesAt(xo, yo, tcolour, slist, sdata);`

Argument	Type	Description
xo	int	Horizontal position of selected pixel
yo	int	Vertical position of selected pixel
tcolour	uint16_t	Colour to test against as transparent
slist	int *	Array to store the list of sprites found
sdata	uint16_t *	Sprite data array

Return: Number of sprites found (*int*)

12.14. GetSprite

Returns the selected sprite parameter.

Parameter	Description
`SPRITE_X`	x position of sprite
`SPRITE_Y`	y position of sprite
`SPRITE_W`	width of sprite
`SPRITE_H`	height of sprite
`SPRITE_ACTIVE`	1 if sprite enabled
`SPRITE_COLLIDE1`	number of first collided sprite
`SPRITE_COLLIDE2`	number of second collided sprite

Syntax: `gfx.GetSprite(snum, param);`

Argument	Type	Description
snum	int	Number of the sprite in the sprite data
param	int	Sprite parameter to query

Return: Value of the queried parameter (*int*)

12.15. GetSpriteImageNum

Returns the position of the chosen sprite in the sprite list

Syntax: `gfx.GetSpriteImageNum(snum);`

Argument	Type	Description
snum	int	Number of the sprite in the sprite data

Return: Position of the chosen sprite in the sprite list (*int16_t*)

12.16. SpriteSetPalette

Set the colour in the 4 bit pallette specified by the colour number (pnumber) with 16bit colour (plcolour)

Syntax: `gfx.SpriteSetPalette(pnumber, plcolour);`

Argument	Type	Description
pnumber	int	Colour number
plcolour	uint16_t	Colour value

Return: None (*void*)

13. GRAM Functions

13.1. SetGRAM

Define pixel write area in the current framebuffer

Syntax: `gfx.SetGRAM(x1, y1, x2, y2);`

Argument	Type	Description
x1	int16_t	Horizontal position of the top-left pixel of the GRAM
y1	int16_t	Vertical position of the top-left pixel of the GRAM
x2	int16_t	Horizontal position of the bottom-right pixel of the GRAM
y2	int16_t	Vertical position of the bottom-right pixel of the GRAM

Return: None (*void*)

Example

```
gfx.SetGRAM(0, 0, 239, 319);
```

13.2. setAddrWindow

Sets the GRAM window as specified by the top-left corner (x, y) and dimensions w and h

Syntax: `gfx.setAddrWindow(x, y, w, h);`

Argument	Type	Description
x	int16_t	Horizontal position of the top-left pixel of the GRAM
y	int16_t	Vertical position of the top-left pixel of the GRAM
w	int16_t	Width of the GRAM window
h	int16_t	Height of the GRAM window

Return: None (*void*)

Example

```
gfx.setAddrWindow(0, 0, 240, 320);
```

13.3. WrGRAMs

Write an array of pixels with the specified length len from an array to the selected framebuffer. The array can be 8-bit, 16-bit or 32-bit.

WrGRAMs should be used intead of pushColors if image data is likely to exceed display boundaries or if transparency is used

Syntax: `gfx.WrGRAMs(data, len);`

Argument	Type	Description
data	uint8_t *uint16_t* uint32_t *	Pointer to array of pixels
len	uint16_t	Length of data in array

Return: None (*void*)

Example

```
gfx.WrGRAMs(color_data, color_data_len);
```

Note

32bit data arrays can used for IoD compatibility

13.4. WrGRAM

Write a single 16-bit RGB565 colour to GRAM window to the selected framebuffer

Syntax: `gfx.WrGRAM(colour);`

Argument	Type	Description
colour	uint16_t	16-bit colour to write to the GRAM window

Return: None (*void*)

Example

```
gfx.WrGRAM(0xFFE0);
```

13.5. pushColors

Write an array of pixels with the specified length len from an array to the selected framebuffer. The array can be 8-bit, 16-bit or 32-bit.

This function operates faster than WrGRAMs due to no boundary or transparency checks.

Care must be taken to ensure image area does not exceed the displays boundaries as an error will occur.

Syntax: `gfx.pushColors(color_data, len);`

Argument	Type	Description
data	uint8_t *uint16_t* uint32_t *	Pointer to array of pixels
len	uint16_t	Length of data in array

Return: None (*void*)

Example

```
gfx.pushColors(color_data, color_data_len);
```

Note

32-bit data arrays can used for IoD compatibility

13.6. StartWrite

Sets the start write condition

Syntax: `gfx.StartWrite();`

Return: `true` if successful otherwise `false` (*boolean*)

Example

```
bool res = gfx.StartWrite();

// Use draw functions as required
// and when done, run EndWrite

if (res) gfx.EndWrite();
```

13.7. EndWrite

Unsets the start write condition allowing all changes to the framebuffer to update the display

Syntax: `gfx.EndWrite();`

Return: None (*void*)

Example

```
bool res = gfx.StartWrite();

// Use draw functions as required
// and when done, run EndWrite

if (res) gfx.EndWrite();
```

13.8. ReadPixel

Read a single pixel from the framebuffer

Syntax: `gfx.ReadPixel(x, y);`

Argument	Type	Description
x	uint16_t	Horizontal position of the pixel
y	uint16_t	Vertical position of the pixel

Return: Pixel colour at specified x, y position (*uint16_t*)

Example

```
pixelColour = gfx.ReadPixel(50, 20);
```

13.9. ReadLine

Read a line of pixels starting from (x, y) and store it in the specified 16-bit data array

Syntax: `gfx.ReadLine(x, y, w, data);`

Argument	Type	Description
x	int16_t	Horizontal position of the starting pixel
y	int16_t	Vertical position of the starting pixel
w	int16_t	Number of pixels to read
data	uint16_t *	Array to store the colour data

Return: Number of pixels read (*uint16_t*)

Example

```
uint16_t line_data[100];
gfx.ReadLine(40, 50, 100, line_data);
```

13.10. WriteLine

Writes a line of pixels from the specified 16-bit array starting from (x, y)

Syntax: `gfx.WriteLine(x, y, w, data);`

Argument	Type	Description
x	int16_t	Horizontal position of the starting pixel
y	int16_t	Vertical position of the starting pixel
w	int16_t	Number of pixels to write
data	uint16_t *	Array containing the colour data to write

Return: None (*void*)

Example

```
gfx.WriteLine(40, 50, 100, colour_data);
```

13.11. DrawToframebuffer

Set the frame buffer for drawing functions. Once set, all drawing functions will be sent to specified framebuffer.

If frame buffer 0 is set (default) all drawing functions will appear immediately on the display.

Syntax: `gfx.DrawToframebuffer(fbnum);`

Argument	Type	Description
fbnum	uint8_t	Specifies the framebuffer number (0 to 4)

Return: None (*void*)

Example

```
gfx.DrawToframebuffer(1);
```

13.12. DrawFrameBuffer

Flush the whole specified framebuffer

Syntax: `gfx.DrawFrameBuffer(fbnum);`

Argument	Type	Description
fbnum	uint8_t	Specifies the framebuffer number (0 to 4)

Return: None (*void*)

Example

```
gfx.DrawFrameBuffer(1);
```

13.13. DrawFrameBufferArea

13.13.1. Using GCI Object Info

Flush the area occupied by GCI widget specified by id from the selected framebuffer fbnum

Syntax: `gfx.DrawFrameBufferArea(fbnum, ui);`

Argument	Type	Description
fbnum	uint8_t	Specifies the framebuffer number (0 to 4)
ui	int16_t	Specifies the GCI widget to get position and size

Return: None (*void*)

Example

```
gfx.DrawFrameBufferArea(1, iGauge1);
```

13.13.2. Using Custom Area

Flush the rectangular area specified by the diagonal (x1, y1), (x2, y2) from the selected framebuffer fbnum

Syntax: `gfx.DrawFrameBufferArea(fbnum, x1, y1, x2, y2);`

Argument	Type	Description
fbnum	uint8_t	Specifies the framebuffer number (0 to 4)
x1	uint16_t	Horizontal position of top-left pixel of the area
y1	uint16_t	Vertical position of top-left pixel of the area
x2	uint16_t	Horizontal position of bottom-right pixel of the area
y2	uint16_t	Vertical position of bottom-right pixel of the area

Return: None (*void*)

Example

```
gfx.DrawFrameBufferArea(1, 0, 0, 99, 199);
```

13.14. MergeFrameBuffers

Merge 2 or 3 frame buffers and send to specified frame buffer.

Syntax: `gfx.MergeFrameBuffers(fbto, fbfrom1, fbfrom2[, fbfrom3], transColor [, transColor1]);`

Argument	Type	Description
fbto	uint8_t	Output framebuffer
fbfrom1	uint8_t	Base framebuffer for the second framebuffer to be merged to
fbfrom2	uint8_t	Second framebuffer to merge to the first
fbfrom3 (optional)	uint8_t	Third framebuffer to merge to the second and first
transColor	uint16_t	16-bit colour from second framebuffer to treat as transparent
transColor1 (optional)	uint16_t	16-bit colour from third framebuffer to treat as transparent

Return: None (*void*)

Example

Merge 2 Framebuffers

```
gfx.MergeFrameBuffers(0, 1, 2, BLACK); // Merge Frame Buffers 1 and 2 to 0
```

Merge 3 Framebuffers

```
gfx.MergeFrameBuffers(0, 1, 2, 3, BLACK, WHITE); // Merge Frame Buffers 1, 2 and 3 to 0
```

Note

Using this function without first writing to a frame buffer will cause issue

13.15. WriteToFrameBuffer

Writes 8bit or 16bit colour data array directly to current frame buffer at given offset. Needs a gfx.FlushArea(x1, x2, y1, y2, -1); after write to display new data.

Syntax: `gfx.WriteToFrameBuffer(offset, data, len);`

Argument	Type	Description
offset	uint32_t	Offset indicating the position of the first write
data	uint16_t *	Colour data array to write
len	uint32_t	Length/Size of data to write to the frame buffer

Return: None (*void*)

Example

```
gfx.WriteToFrameBuffer(0, colour_data, 100);
```

13.16. FlushArea

Flushes display area specified.

Syntax: `gfx.FlushArea(x1, x2, y1, y2, xpos);`

Argument	Type	Description
x1	int	Horizontal position of top-left pixel of the area
x2	int	Horizontal position of bottom-right pixel of the area
y1	int	Vertical position of top-left pixel of the area
y2	int	Vertical position of bottom-right pixel of the area
xpos	int	-1 for normal use, otherwise refreshes one pixel at xpos position

Return: None (*void*)

Example

```
gfx.FlushArea(10, 100, 10, 100, -1);
```

13.17. drawBitmap

Draws a bitmap using the data array at the given co-ordinates

Syntax: `gfx.drawBitmap(x1, y1, x2, y2, data);`

Argument	Type	Description
x1	int	Horizontal position of top-left pixel of the area
y1	int	Vertical position of top-left pixel of the area
x2	int	Horizontal position of bottom-right pixel of the area
y2	int	Vertical position of bottom-right pixel of the area
data	uint16_t *	Pointer to array containing the colour data

Return: None (*void*)

Example

```
gfx.drawBitmap(0, 0, 100, 200, bmp_data);
```

14. GPIO Functions

14.1. PinMode

Sets the operation mode of the specified pin

Syntax: `gfx.PinMode(pin, mode);`

Argument	Type	Description
pin	byte	Specifies the pin number/name
mode	bool	Specifies whether to use the pin as INPUT or OUTPUT

Return: None (*void*)

Example

```
gfx.PinMode(5, OUTPUT);
```

14.2. DigitalWrite

Sets the specified pin to either HIGH or LOW. The pin must first be set as OUTPUT using gfx.PinMode

Syntax: `gfx.DigitalWrite(pin, state);`

Argument	Type	Description
pin	byte	Specifies the pin number/name
state	bool	Specifies whether to set the pin to HIGH or LOW

Return: None (*void*)

Example

```
gfx.PinMode(5, OUTPUT);
gfx.DigitalWrite(5, HIGH);
```

14.3. DigitalRead

Reads the current state of the specified pin

Syntax: `gfx.DigitalRead(pin);`

Argument	Type	Description
pin	byte	Specifies the pin number/name

Return: Whether the pin is HIGH (1) or LOW (0) (*int*)

Example

```
gfx.PinMode(5, INPUT);
pinState = gfx.DigitalRead(5);
```

15. Touch Functions

The functions included in this section are only applicable for touch enabled display modules.

15.1. touch_Set

Enables or disable touch functionality

Syntax: `gfx.touch_Set(mode);`

Argument	Type	Description
mode	uint8_t	Specifies whether to enable (TOUCH_ENABLE) or disable (TOUCH_DISABLE) touch

Return: None (*void*)

Example

```
touch_Set(TOUCH_DISABLE);
```

Note

This function is only available for capacitive and resistive touch displays

15.2. touch_Update

Checks whether a touch event is detected.

If a touch event has occurred, pen, xpos, ypos and images touched will be updated

Syntax: `gfx.touch_Update();`

Return: `true` if successful otherwise `false` (*boolean*)

Example

```
gfx.Cls();
gfx.println("Draw something, exit by pressing center square");

gfx.RectangleFilled(midx0, midy0, midx1, midy1, RED);

while (true) {

    gfx.touch_Update();
    tx = gfx.touch_GetX();
    ty = gfx.touch_GetY();
    TOUCHED = gfx.touch_GetPen();

    // If the screen gets touched
    if (TOUCHED == TOUCH_PRESSED) {

        if (tx >= midx0 && tx <= midx1 && ty >= midy0 && ty <= midy1) {
            waitForRelease();
            break;
        }

        // First touch of the screen, draw a pixel where it was touched
        if (firstTouch) {
            gfx.PutPixel(tx, ty, BLUE);
            firstTouch = false;
        }
        // Screen previously touched and pixel drawn, so draw a line from first touch to this
location
        else {
            gfx.Line(gfx.touch_GetLastX(), gfx.touch_GetLastY(), tx, ty, BLUE);
        }
    }

    // If the screen touch is released after previously touching it, draw a circle to mark the
release point
    if (TOUCHED == NOTOUCH || TOUCHED == TOUCH_RELEASED) {
        gfx.CircleFilled(tx, ty, 3, RED);
        firstTouch = true;
    }
};
```

Note

This function is only available for capacitive and resistive touch displays

15.3. touch_GetPen

Returns the latest pen status after the last gfx.touch_Update

Syntax: `gfx.touch_GetPen();`

Return: Pen Status (*int16_t*)

Example

See gfx.touch_Update example

Note

This function is only available for capacitive and resistive touch displays.

15.4. touch_GetX

Returns the latest touch X position after the last gfx.touch_Update

Syntax: `gfx.touch_GetX();`

Return: Horizontal pixel position of last touch event (*int16_t*)

Example

See gfx.touch_Update example

Note

This function is only available for capacitive and resistive touch displays

15.5. touch_GetY

Returns the latest touch Y position after the last gfx.touch_Update

Syntax: `gfx.touch_GetY();`

Return: Vertical pixel position of last touch event (*int16_t*)

Example

See gfx.touch_Update example

Note

This function is only available for capacitive and resistive touch displays

15.6. touch_GetLastX

Returns the previous touch X position prior to the last gfx.touch_Update

Syntax: `gfx.touch_GetLastX();`

Return: Horizontal pixel position of touch event prior to the last (*int16_t*)

Example

See gfx.touch_Update example

Note

This function is only available for capacitive and resistive touch displays

15.7. touch_GetLastY

Returns the previous touch Y position prior to the last gfx.touch_Update

Syntax: `gfx.touch_GetLastY();`

Return: Vertical pixel position of touch event prior to the last (*int16_t*)

Example

See gfx.touch_Update example

Note

This function is only available for capacitive and resistive touch displays

15.8. imageTouchEnable

Enable or disable touch for the specified GCI widget Optionally, a special type of input widget can be specified:

- TOGGLE4STATES
- SLIDER3IMAGE
- GAUGE2IMAGE
- MOMENTARY
- TOGGLE

Syntax: `gfx.imageTouchEnable(gcinum, en [, type]);`

Argument	Type	Description
gcinum	int	Specifies the target widget or -1 for all widgets
en	boolean	Enable (`true`) or disable (`false`)
type (optional)	int	Optional type of widget

Example

```
gfx.imageTouchEnable(iWinbutton1, true, MOMENTARY);
gfx.imageTouchEnable(iWinbutton2, true, TOGGLE);
```

Return: None (*void*)

Note

1. This function is only available for capacitive and resistive touch displays
2. It is advisable to use Workshop4 to generate necessary touch related code by using the **Paste Code** option
3. When using Workshop4 buttons, Momentary options Yes, No and On can be selected. Momentary Both can also be selected but will behave exactly the same as Momentary Yes.

15.9. imageTouchEnableRange

Enable or disable touch for the specified range of GCI widget. Optionally, a special type of input widget can be specified:

- TOGGLE4STATES
- SLIDER3IMAGE
- GAUGE2IMAGE
- MOMENTARY
- TOGGLE
- KEYPAD

Syntax: `gfx.imageTouchEnableRange(gcinumFrom, gcinumTo, en [, type]);`

Argument	Type	Description
gcinumFrom	int	Specifies the target widget or -1 for all widgets
gcinumTo	int	Specifies the target widget or -1 for all widgets
en	boolean	Enable (`true`) or disable (`false`)
type (optional)	int	Optional type of widget

Return: None (*void*)

Example

```
gfx.imageTouchEnableRange(iKeyboard1_0, iKeyboard1_59, true, KEYPAD);
```

Note

1. This function is only available for capacitive and resistive touch displays
2. It is advisable to use Workshop4 to generate necessary touch related code by using the **Paste Code** option

15.10. ImageTouchedAuto

Evaluate touch events for the input widgets automatically. The correct widget type should be set using gfx.imageTouchEnable

Syntax: `gfx.ImageTouchedAuto();`

Return: Returns the widget pressed or -1 if no touch (*int16_t*)

Example

```
imgTouched = gfx.ImageTouchedAuto();
```

Note

1. This function is only available for capacitive and resistive touch displays
2. It is advisable to use Workshop4 to generate necessary touch related code by using the **Paste Code** option

15.11. imageTouched

Returns the touched widget after the last gfx.touch_Update or return -1 if no touch.

Syntax: `gfx.imageTouched();`

Return: Returns the touched widget (*int*)

Example

```
imgTouched = gfx.imageTouched();
```

Note

This function is only available for capacitive and resistive touch displays

15.12. imageAutoSlider

Calculates the value of the slider based on the last user input position

Syntax: `gfx.imageAutoSlider(ui, axis, uiv, ming, maxg);`

Argument	Type	Description
ui	uint16_t	Specifies the GCI slider
axis	uint16_t	Indicates whether the slider is a HORIZONTAL_SLIDER or VERTICAL SLIDER
uiv	uint16_t	Touch position based on the orientation for the slider - X position for horizontal - Y position for vertical
ming	uint16_t	Touch offset near the minimum position
maxg	uint16_t	Touch offset near the maximum position

Return: Value of the Slider (*int16_t*)

Example

```
sliderVal = gfx.imageAutoSlider(iSlider1, HORIZONTAL_SLIDER, gfx.touch_GetX(), 8, 8);
```

Note

This function is only available for capacitive and resistive touch displays

15.13. imageAutoKnob

Calculates the value of the knob based on the last user input position

Syntax: `gfx.imageAutoKnob(hndl, uix, uiy, minarc, maxarc, ming, maxg);`

Argument	Type	Description
hndl	int	Specifies the GCI knob
uix	int	Horizontal touch position
uiy	int	Vertical touch position
minarc	int	Angle position of minimum value
maxarc	int	Angle position of maximum value
ming	int	Value at minimum position
maxg	int	Value at maximum position

Return: Value of the Knob (*int16_t*)

Example

```
sliderVal = gfx.imageAutoKnob(iKnob1, gfx.touch_GetX(), gfx.touch_GetY(), 30, 330, 0, 100);
```

Note

This function is only available for capacitive and resistive touch displays

15.14. CheckButtons

Evaluate the button touches and return the buttonx touched

Syntax: `gfx.CheckButtons();`

Return: ID of the ButtonX that was touched (*uint8_t*)

Example

```
btnxPressed = gfx.CheckButtons();
```

Note

This function is only available for capacitive and resistive touch displays

15.15. GetSliderValue

Calculate the value of slider based on touch position

Syntax: `gfx.GetSliderValue(ui, axis, uiv, ming, maxg);`

Argument	Type	Description
ui	uint16_t	Specifies the GCI slider
axis	uint16_t	Indicates whether the slider is a HORIZONTAL_SLIDER or VERTICAL SLIDER
uiv	uint16_t	Touch position based on the orientation for the slider - X position for horizontal - Y position for vertical
ming	uint16_t	Touch offset near the minimum position
maxg	uint16_t	Touch offset near the maximum position

Return: Value of the slider (*uint16_t*)

Example

```
sliderVal = gfx.GetSliderValue(iSlider1, HORIZONTAL_SLIDER, gfx.touch_GetX(), 8, 8);
```

Note

This function is only available for capacitive and resistive touch displays

15.16. DecodeKeypad

Decodes the key pressed in the keyboard

Syntax: `gfx.DecodeKeypad(kpad, kpress, kbks, kbck);`

Argument	Type	Description
kpad	int	Base keyboard widget
kpress	int	Keyboard button pressed
kbks	byte *	Keyboard information generated by Workshop4
kbck	int8_t *	Keyboard information generated by Workshop4

Return: Value of the pressed key (*int*)

Note

1. This function is only available for capacitive and resistive touch displays.
2. It is highly advisable to use Workshop4 when using this function.

15.17. ResetKeypad

Resets the state of the keypad

Syntax: `gfx.ResetKeypad();`

Return: None (*void*)

> **Note**
>
> This function is only available for capacitive and resistive touch displays

15.18. KeypadStatus

Checks whether the keypad state is on SHIFT, CAPSLOCK or CTRL or not.

Syntax: `gfx.KeypadStatus(keyType);`

Argument	Type	Description
keyType	int	Specify the status to check

Return: `true` if successful otherwise `false` (*boolean*)

Example

```
isShifted = gfx.KeypadStatus(SHIFT);
isCapsLocked = gfx.KeypadStatus(CAPSLOCK);
isCtrled = gfx.KeypadStatus(CTRL);
```

Note

This function is only available for capacitive and resistive touch displays

15.19. SpriteTouched

Returns the id of the sprite touched

Syntax: `gfx.SpriteTouched();`

Return: ID of the touched sprite (*int*)

Example

```
int spriteTouched = gfx.SpriteTouched();
```

Note

This function is only available for capacitive and resistive touch displays

15.20. touchCalibration

Starts touch calibration for resistive touch displays

Syntax: `gfx.touchCalibration();`

Return: None (*void*)

Example

```
if (!calibrated) {
    gfx.touchCalibration();
    calibrated = true;
}
```

Note

This is only available for resistive touch displays

16. RTC Functions

The functions included in this section are only applicable for gen4-ESP32 variants with on-board RTC. This includes display modules sized

16.1. RTCinit

Initializes the RTC module

Syntax: `gfx.RTCinit();`

Return: None (*void*)

Example

```
gfx.RTCinit();
```

Note

This is only available for display modules with built-in RTC.

16.2. RTCstartClock

Starts the RTC clock

Syntax: `gfx.RTCstartClock();`

Return: None (*void*)

Example

```
gfx.RTCstartClock();
```

Note

This is only available for display modules with built-in RTC.

16.3. RTCstopClock

Stops the RTC clock

Syntax: `gfx.RTCstopClock();`

Return: None (*void*)

Example

```
gfx.RTCstopClock();
```

Note

This is only available for display modules with built-in RTC.

16.4. RTCcheckClockIntegrity

Checks for clock integrity

Syntax: `gfx.RTCcheckClockIntegrity();`

Return: Clock integrity (*bool*)

Example

```
rtcOk = gfx.RTCcheckClockIntegrity();
```

Note

This is only available for display modules with built-in RTC.

16.5. RTCsetYear

Sets the year value for the RTC

Syntax: `gfx.RTCsetYear(year);`

Argument	Type	Description
year	uint16_t	Specify the new year value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetYear(2023);
```

Note

This is only available for display modules with built-in RTC.

16.6. RTCsetMonth

Sets the month value for the RTC from 1 to 12

Syntax: `gfx.RTCsetMonth(month);`

Argument	Type	Description
month	uint8_t	Specify the new month value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetMonth(9);
```

Note

This is only available for display modules with built-in RTC.

16.7. RTCsetDay

Sets the day of month value for the RTC from 1 to 31

Syntax: `gfx.RTCsetDay(day);`

Argument	Type	Description
day	uint8_t	Specify the new day value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetDay(5);
```

Note

This is only available for display modules with built-in RTC.

16.8. RTCsetHour

Sets the hour value for the RTC from 0 to 23

Syntax: `gfx.RTCsetHour(hour);`

Argument	Type	Description
hour	uint8_t	Specify the new hour value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetHour(14);
```

Note

This is only available for display modules with built-in RTC.

16.9. RTCsetMinute

Sets the minutes value for the RTC from 0 to 59

Syntax: `gfx.RTCsetMinute(minute);`

Argument	Type	Description
minute	uint8_t	Specify the new minute value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetMinute(42);
```

Note

This is only available for display modules with built-in RTC.

16.10. RTCsetSecond

Sets the seconds value for the RTC from 0 to 59

Syntax: `gfx.RTCsetSecond(second);`

Argument	Type	Description
second	uint8_t	Specify the new second value to set the RTC to

Return: None (*void*)

Example

```
gfx.RTCsetSecond(57);
```

Note

This is only available for display modules with built-in RTC.

16.11. RTCgetTime

Queries the current time

Syntax: `gfx.RTCgetTime();`

Return: Current time in Time struct (*Time*)

Example

```
Time curTime = gfx.RTCgetTime();
```

Note

This is only available for display modules with built-in RTC.

16.12. RTCformatDateTime

Formats the time and return a pointer to the string.

The following are the accepted styles:

- `RTC_TIMEFORMAT_HM`
- `RTC_TIMEFORMAT_HMS`
- `RTC_TIMEFORMAT_YYYY_MM_DD`
- `RTC_TIMEFORMAT_MM_DD_YYYY`
- `RTC_TIMEFORMAT_DD_MM_YYYY`
- `RTC_TIMEFORMAT_YYYY_MM_DD_H_M_S`
- `RTC_TIMEFORMAT_DD_MM_YYYY_H_M_S`

If an invalid format is specified, this will default to RTC_TIMEFORMAT_HM

Syntax: `gfx.RTCformatDateTime(style);`

Argument	Type	Description
style	uint8_t	Specify the style to format the current time to

Return: Character pointer to the date/time string (*const char **)

Example

```
const char * curTimeStr = gfx.RTCformatDateTime(RTC_TIMEFORMAT_HMS);
```

Note

This is only available for display modules with built-in RTC.

17. Legal Notice

17.1. Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission. 4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems. 4D Systems reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

All trademarks belong to their respective owners and are recognised and acknowledged.

17.2. Disclaimer of Warranties & Limitations of Liabilities

4D Systems makes no warranty, either expressed or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Systems range of products, however the quality may vary.
In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.
4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail - safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.