REFERENCE

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

Internet-of-Displays (IoD)

Arduino Libraries

Document Date: 8th November 2017

Document Revision: 1.0

Contents

1. Libraries Introduction

The GFX4d and GFX4dIoD9 libraries are provided by 4D Systems for use with gen4-IoD and IoD-09 product series.

The GFX4d library provides users access to the graphics, touch, and WiFi functionalities of gen4-IoD products. Similarly, the GFX4dIoD9 library provides access to graphics, touch, and WiFi functionalities of IoD-09 products. Note however that some functionalities might not be supported by a certain product, depending on its specifications. The IoD-09TH and IoD-09SM, for instance, are non-touch, so all touch-related functions are not applicable. For more information on the specifications of a product, refer to its datasheet.

The SOMOIoD library, on the other hand, is for controlling a **SOMO-II** or a **MOTG-MP3** interfaced to a gen4-IoD or an IoD-09 product through the pin GPIO16. It can also be used with any Arduino host with an available GPIO pin wired to the RX pin of the SOMO-II or MOTG-MP3. The SOMOIoD library does not need a serial port to operate.

Below is a list of functionalities supported by the GFX4d, GFX4dIoD9, and SOMOIoD libraries.

- Basic Graphics
- 4D Graphics Files (GCI and DAT files)
- Text Functions
- Touch Control
- Wi-Fi / Internet Download
- SOMO-II Control

The GFX4d and GFX4dIoD9 libraries are installed automatically to the Arduino library directory when Workshop4 IDE is installed. Please take note that Arduino IDE must be installed prior to the Workshop4 installation for this work.

Workshop4 is a Windows-only application so for those who are using a different operating system, the GFX4d library can be downloaded here. The GFX4dIoD9 library can be downloaded here. Lastly, the SOMOIoD library can be found here.

1.1. Include the Libraries

First, the correct Arduino library must be included, depending on the target product. For gen4-IoD products, include the GFX4d library like as shown below.

```
#include "GFX4d.h"
```

On the other hand, for IoD-09 products, include the GFX4dIoD9 library, like as shown below.

```
#include "GFX4dIoD9.h"
```

Finally, when using a **SOMO-II** or a **MOTG-MP3** product, include the SOMOIoD library as shown below.

```
#include "SOMOIoD.h"
```

To be able to use the Wi-Fi functionality with ease, include the recommended ESP8266WiFi library.

```
#include "ESP8266WiFi.h"
```

The library is automatically downloaded when you install the ESP8266 board package through the Boards Manager of the Arduino IDE. Please refer to the product datasheets for a more detailed discussion regarding this.

1.2. Create a GFX4d or GFX4dIoD9 Object

Once the correct library is included to the project, an instance of the library class can now be created. For the GFX4d library,

```
GFX4d gfx = GFX4d();
```

For the GFX4dIoD9 library, use the line below to instantiate an object.

```
GFX4dIoD9 gfx = GFX4dIoD9();
```

In the above example, the GFX4d or GFX4dIoD9 object is named **gfx**. This document will use the same object name in the examples.

1.3. Initialize the Display

The display is initialized during setup using the library function begin. Other library functions can also be included during setup. Here's an example of a setup function.

```
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);
}
```

1.4. Create a SOMOIoD Object

The SOMOIoD library allows the users to easily interface a SOMO-II or a MOTG-MP3 to a gen4-IoD or IoD-09 module. Once the SOMOIoD library is included to the project the user needs to create a SOMOIoD object.

```
SOMOIoD sound;
```

In this example, the SOMOIoD object is named **sound**. This document will use the same object name in the examples.

1.5. Initialize the Sound Module

The sound module is initialized during setup using the SOMOIoD function begin.

```
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);

  sound.begin(); // Initialize the Sound Module
}
```

2. Display Functions

These functions allows to set the displays mode of operation and check the properties of the screen.

- Orientation
  - Set Orientation
  - Get Orientation
- BacklightOn
- FillScreen
- Cls
- MoveTo
- getX
- getY
- getWidth
- getHeight
- Invert

2.1. Orientation

2.1.1. Set Orientation

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>Orientation (mode)</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td rowspan="2"><b>Arguments</b></td><td colspan="2"><b>mode</b></td></tr>
<tr><td><b>mode</b></td><td>Specifies the orientation</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Sets the orientation of the display the the <b>mode</b> specified.
<table>
<tr><th>Constant Definitions</th><th>Value</th></tr>
<tr><td>LANDSCAPE</td><td>0</td></tr>
<tr><td>LANDSCAPE_R</td><td>1</td></tr>
<tr><td>PORTRAIT</td><td>2</td></tr>
<tr><td>PORTRAIT_R</td><td>3</td></tr>
</table>
<b>Note</b>: The cursor position is not altered in any way by changing the orientation.</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><code>gfx.Orientation(PORTRAIT); // Sets Orientation to PORTRAIT</code></td></tr>
</table>

2.1.2. Get Orientation

Syntax	**Orientation ()**
Arguments	**none**
Returns	***int8_t* Orientation**
Description	Get the current display orientation

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

2.2. BacklightOn

Syntax	**BacklightOn (mode)**	
Arguments	**mode**	
	mode	Use `true` to turn **ON** and `false` to turn **OFF**
Returns	**none**	
Description	Turns the backlight **ON** if mode is `true` otherwise turns the backlight **OFF**	

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

2.3. FillScreen

Syntax	**FillScreen (colour)**	
Arguments	**colour**	
	colour	16 bit colour to fill the screen
Returns	**none**	
Description	Fills the screen with the specified colour.	
Example	`gfx.FillScreen(LIME); // Fills the screen with LIME`	

2.4. Cls

Syntax	**Cls () or Cls (colour)**	
Arguments	**colour**	
	colour	Specifies the colour to clear the screen with
Returns	**none**	
Description	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.	

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

2.5. MoveTo

Syntax	**MoveTo (x, y)**	
Arguments	**x, y**	
	x, y	Specifies the new cursor position
Returns	**none**	
Description	Moves the cursor to the specified position.	

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);`	

2.6. getX

Syntax	**getX ()**
Arguments	**none**
Returns	***int16_t*** **Cursor X Position**
Description	Returns the current X position of the cursor

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);`

2.7. getY

Syntax	**getY ()**
Arguments	**none**
Returns	***int16_t*** **Cursor Y Position**
Description	Returns the current Y position of the cursor

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);`

2.8. getWidth

Syntax	**getWidth ()**
Arguments	**none**
Returns	***int16_t* Display Width**
Description	Returns the width of the display in pixels

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

2.9. getHeight

Syntax	**getHeight ()**
Arguments	**none**
Returns	***int16_t* Display Height**
Description	Returns the height of the display in pixels

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

2.10. Invert

Syntax	**Invert (mode)**	
Arguments	**mode**	
	mode	Use `true` to **invert** display colours and `false` to display original
Returns	**none**	
Description	If mode is `true`, this will invert the colours displayed on the screen otherwise this will display original colours.	

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. Primitive Shapes

These functions allow easy generation of basic shapes.

- PutPixel
- Hline
- Vline
- Line
- Arc
- ArcFilled
- Circle
- CircleFilled
- Ellipse
- EllipseFilled
- Rectangle
- RectangleFilled
- RoundRect
- RoundRectFilled
- Triangle
- TriangleFilled

3.1. PutPixel

Syntax	**PutPixel (x, y, colour)**	
Arguments	**x, y, colour**	
	x, y	Specifies the position of the pixel
	colour	16 bit colour to be drawn to the specified position
Returns	**none**	
Description	Writes the pixel colour to the specified position	
Example	`gfx.PutPixel(5,10,RED); // Draws a RED pixel at (5,10)`	

3.2. Hline

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>Hline (x, y, width, colour)</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td rowspan="4"><b>Arguments</b></td><td colspan="2"><b>x, y, width, colour</b></td></tr>
<tr><td><b>x, y</b></td><td>Starting position of the line</td></tr>
<tr><td><b>width</b></td><td>Length in pixels of the horizontal line</td></tr>
<tr><td><b>colour</b></td><td>16 bit colour of the line</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Draws a horizontal line from point <b>(x, y)</b> with length equal to <b>width</b> using the specified <b>colour</b>. Direction is specified by the sign of <b>width</b>.
<table><tr><th>Sign</th><th>Drawing Direction</th></tr><tr><td>-</td><td>left</td></tr><tr><td>+</td><td>right</td></tr></table></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><code>gfx.Hline(5,10,100,RED);</code>
<code>// Draws a 100-pixel RED Hline from (5,10) to the right</code>

<code>gfx.Hline(5,10,-100,BLUE);</code>
<code>// Draws a 100-pixel BLUE Hline from (5,10) to the left</code></td></tr>
</table>

3.3. Vline

Syntax	**Vline (x, y, height, colour)**	
Arguments	**x, y, height, colour**	
	x, y	Starting position of the line
	height	Length in pixels of the vertical line
	colour	16 bit colour of the line
Returns	**none**	
Description	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**.	
Example	see below	

Sign	Drawing Direction
-	up
+	down

```
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
```

3.4. Line

Syntax	**Line (x, y, x1, y1, colour)**	
Arguments	**x, y, x1, y1, colour**	
	x, y	Starting position of the line
	x1,y1	Ending position of the line
	colour	16 bit colour of the line
Returns	**none**	
Description	Draws a line from point **(x,y)** to point **(x1,y1)** using the specified **colour**.	

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

3.5. Circle

Syntax	**Circle (x, y, radius, colour)**	
Arguments	**x, y, radius, colour**	
	x, y	Center of the circle
	radius	Radius of the circle
	colour	16 bit colour of the circle
Returns	**none**	
Description	Draws a circle with the specified **radius** and **colour** with the center at **(x,y)**	

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

INTERNET-OF-DISPLAYS ARDUINO LIBRARIES

3.6. CircleFilled

Syntax	**CircleFilled (x, y, radius, colour)**	
Arguments	**x, y, radius, colour**	
	x, y	Center of the filled circle
	radius	Radius of the filled circle
	colour	16 bit colour of the filled circle
Returns	**none**	
Description	Draws a solid-coloured circle with the specified **radius** and **colour** with the center at **(x,y)**	

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

3.7. Ellipse

Syntax	**Ellipse (x, y, radx, rady, colour)**	
Arguments	**x, y, radx, rady, colour**	
	x, y	Center of the elllipse
	radx	Radius of the ellipse along the x-axis
	rady	Radius of the ellipse along the y-axis
	colour	16 bit colour of the elllipse
Returns	**none**	
Description	Draws an ellipse with the specified x radius (**radx**), y radius (**rady**), and **colour** with the center at **(x,y)**	

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)`	

3.8. EllipseFilled

Syntax	**EllipseFilled (x, y, radx, rady, colour)**	
Arguments	**x, y, radx, rady, colour**	
	x, y	Center of the filled elllipse
	radx	Radius of the filled ellipse along the x-axis
	rady	Radius of the filled ellipse along the y-axis
	colour	16 bit colour of the filled elllipse
Returns	**none**	
Description	Draws a solid coloured ellipse with the specified x radius (**radx**), y radius (**rady**), and **colour** with the center at **(x,y)**	

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)`	

3.9. Rectangle

Syntax	**Rectangle (x, y, x1, y1, colour)**	
Arguments	**x, y, x1, y1, colour**	
	x, y	Specifies an endpoint of one diagonal of the rectangle
	x1, y1	Specifies the other endpoint the same diagonal of the rectangle
	colour	16 bit colour of the rectangle
Returns	**none**	
Description	Draws a rectangle having a diagonal with endpoints at **(x, y)** and **(x1, y1)**.	

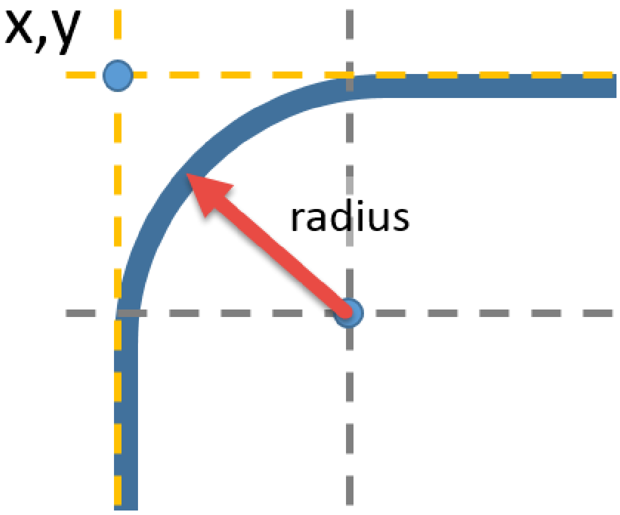
Example	`gfx.Rectangle(0,0,50,50,CYAN);`
	`// Draws a CYAN rectangle with:`
	`// a diagonal whose end points are (0,0) and (50,50)`	

3.10. RectangleFilled

Syntax	**RectangleFilled (x, y, x1, y1, colour)**	
Arguments	**x, y, x1, y1, colour**	
	x, y	Specifies an endpoint of one diagonal of the rectangle
	x1, y1	Specifies the other endpoint the same diagonal of the rectangle
	colour	16 bit colour of the rectangle
Returns	**none**	
Description	Draws a solid rectangle having a diagonal with endpoints at **(x, y)** and **(x1, y1)**.	

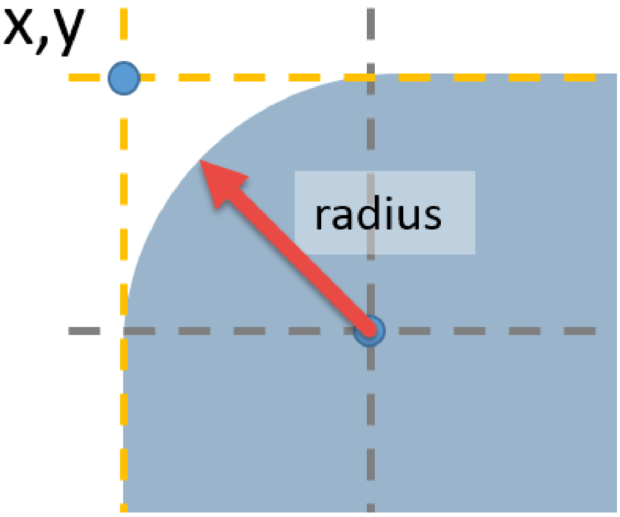
Example	`gfx.RectangleFilled(0,0,50,50,YELLOW);`
	`// Draws a YELLOW solid rectangle with:`
	`// a diagonal whose end points are (0,0) and (50,50)`	

3.11. RoundRect

Syntax	**RoundRect (x, y, x1, y1, radius, colour)**	
Arguments	**x, y, x1, y1, colour**	
	x, y	Specifies an endpoint of one diagonal of the round-cornered rectangle
	x1, y1	Specifies the other endpoint the same diagonal of the round-cornered rectangle
	radius	Specifies the corner radius. This is the distance in pixels extending from the corners of the **inner** rectangle.
	colour	16 bit colour of the rectangle
Returns	**none**	
Description	Draws a round-cornered rectangle having a diagonal with endpoints at **(x, y)** and **(x1, y1)** and with a corner radius of **radius**.  	

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`	

3.12. RoundRectFilled

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>RoundRectFilled (x, y, x1, y1, radius, colour)</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Arguments</b></td><td colspan="2"><b>x, y, x1, y1, colour</b></td></tr>
<tr><td></td><td><b>x, y</b></td><td>Specifies an endpoint of one diagonal of the round-cornered filled rectangle</td></tr>
<tr><td></td><td><b>x1, y1</b></td><td>Specifies the other endpoint the same diagonal of the round-cornered filled rectangle</td></tr>
<tr><td></td><td><b>radius</b></td><td>Specifies the corner radius. This is the distance in pixels extending from the corners of the <b>inner</b> rectangle.</td></tr>
<tr><td></td><td><b>colour</b></td><td>16 bit colour of the round-cornered filled rectangle</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Draws a solid round-cornered rectangle having a diagonal with endpoints at <b>(x, y)</b> and <b>(x1, y1)</b> and with a corner radius of <b>radius</b>.
 </td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><pre>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</pre></td></tr>
</table>

3.13. Triangle

Syntax	**Triangle (x, y, x1, y1, x2, y2, colour)**	
Arguments	**x, y, x1, y1, x2, y2, colour**	
	x, y	Specifies the first vertex of the triangle.
	x1, y1	Specifies the second vertex of the triangle.
	x2, y2	Specifies the third vertex of the triangle.
	colour	16 bit colour of the rectangle
Returns	**none**	
Description	Draws a triangle outline between vertices **(x,y), (x1,y1)**, and **(x2,y2)** using the specified **colour**.	

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

3.14. TriangleFilled

Syntax	**TriangleFilled (x, y, x1, y1, x2, y2, colour)**	
Arguments	**x, y, x1, y1, x2, y2, colour**	
	x, y	Specifies the first vertex of the triangle.
	x1, y1	Specifies the second vertex of the triangle.
	x2, y2	Specifies the third vertex of the triangle.
	colour	16 bit colour of the rectangle
Returns	**none**	
Description	Draws a solid triangle between vertices **(x,y), (x1,y1)**, and **(x2,y2)** using the specified **colour**.	

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. Primitive Objects

These functions allows easy generation of primitive objects for basic user interface.

- Button
- Buttonx
- ButtonUp
- ButtonDown
- ButtonActive
- DeleteButton
- Panel
- PanelRecessed
- Slider

4.1. Button

Syntax	**Button (state, x, y, buttonColour, txtColour, fontID, txtWidth, txtHeight, text)**	
Arguments	**state, x, y, buttonColour, txtColour, fontID, txtWidth, txtHeight, text**	
	state	Specifies whether the button is pressed or raised
	x, y	Specifies the top left corner position of the button on the screen
	buttonColour	Button colour
	txtColour	Text Colour
	fontID	Specifies the Font ID. For more information, refer to this section.
	txtWidth	Specifies the width of the text. This value is the font width multiplier and minimum value must be 1
	txtHeight	Specifies the height of the text. This value is the font height multiplier and minimum value must be 1
	text	Specifies the text string. The text string must be within the range of printable ascii character set
Returns	**none**	
Description	Draws a 3-dimensional Text Button at screen location defined by **(x, y)** parameters (top left corner). The size of the button depends on the font, width, height and length of the text.	

Example	`gfx.Button(Pressed, 50, 50, RED, BLACK, 2, 1, 1, "TOGGLE");`
	`// Draws a "Pressed" RED button @(50,50)`
	`// Labelled "TOGGLE" (font color is BLACK)`	

Constant Definitions	Value
Released	0
Pressed	1

4.2. Buttonx

Syntax	**Buttonx (hndl, x, y, w, h, buttonColour, text, fontID, txtColour)**	
Arguments	**hndl, x, y, w, h, buttonColour, text, fontID, txtColour**	
	hndl	Specifies the handle for the button
	x, y	Specifies the top left corner position of the button on the screen
	w, h	Specifies the width and height of the button
	buttonColour	Button colour
	text	Specifies the text string. The text string must be within the range of printable ASCII character set
	fontID	Specifies the Font ID. For more information, refer to this section.
	txtColour	Text Colour
Returns	**none**	
Description	Draws a 3-dimensional Text Button at screen location defined by **(x, y)** parameters (top left corner). The user needs to specify a handler for the button that will be used by the functions: • ButtonUp • ButtonDown • ButtonActive • DeleteButton • CheckButtons **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	

Example	`gfx.Buttonx(BtnA, 50,50, 200,90, RED, "TOGGLE", 1, BLACK);`
	`// Draws a RED button with a handle BtnA @(50,50)`
	`// Labelled "TOGGLE" (font color is BLACK)`	

4.3. ButtonUp

Syntax	**ButtonUp (hndl)**	
Arguments	**hndl**	
	hndl	Specifies the selected button to display as a raised button
Returns	**none**	
Description	Displays the specified button as raised. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	
Example	`gfx.ButtonUp(BtnA); // Redraws BtnA as a Raised button`	

4.4. ButtonDown

Syntax	**ButtonDown (hndl)**	
Arguments	**hndl**	
	hndl	Specifies the selected button to display as a pressed button
Returns	**none**	
Description	Displays the specified button as pressed. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	
Example	`gfx.ButtonDown(BtnA); // Redraws BtnA as a Pressed button`	

4.5. ButtonActive

Syntax	**ButtonActive (hndl, mode)**	
Arguments	**hndl, mode**	
	hndl	Specifies the selected button to enable or disable
	mode	Use `true` to turn **ON** and `false` to turn **OFF**
Returns	**none**	
Description	Enable or Disable the specified button. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	

Example	`gfx.ButtonActive(BtnA, false); // Disable BtnA`
	`gfx.ButtonActive(BtnA, true);  // Enable BtnA`	

4.6. DeleteButton

Syntax	**DeleteButton (hndl) or DeleteButton (hndl, colour)**	
Arguments	**hndl, colour**	
	hndl	Specifies the handle of the button to be deleted
	colour	Specifies the colour to cover the button
Returns	**none**	
Description	Deleles the button specified by covering the button area with the specified **colour**. The handle for the button is removed making the button non-existent. **Note**: If no **colour** was specified, the button will be covered with its background colour. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	

Example	`gfx.DeleteButton(BtnA);`
	`// Delete the button and remove its handle`
	
	`gfx.DeleteButton(BtnA, BLUE);`
	`// Delete the button by covering it with BLUE and`
	`// remove its handle`	

4.7. CheckButtons

Syntax	**CheckButtons ()**
Arguments	**none**
Returns	***uint8_t*** **CheckButtons**
Description	Checks the status of the buttons. This function automatically displays the button as pressed or released button depending on the touch status. **Note:** Before using this function, it is required to enable touch. For more information, please refer to this section. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**

Example	`uint8_t btn;`
	`btn = gfx.CheckButtons(); // Check if a button was touched`
	`if (btn != -1) {`
	`  gfx.MoveTo(0,0);`
	`  gfx.print("Button ");`
	`  gfx.print(btn);`
	`  gfx.println(" was pressed.  ");`
	`}`

4.8. Panel

Syntax	**Panel (x, y, w, h, colour)**	
Arguments	**x, y, w, h, colour**	
	x, y	Specifies the top left corner position of the panel on the screen
	w, h	Specifies the width and height of the panel
	colour	16 bit colour of the panel
Returns	**none**	
Description	Draws a raised 3 dimensional rectangular panel at a screen location defined by **x, y** parameters (top left corner). The size of the panel is set with the **w** and **h** parameters. The colour is defined by **colour**.	

Example	`gfx.Panel(100,50,100,30,ORANGE);`
	`// Draws an ORANGE panel @(100,50) with:`
	`// width of 100 and height of 30`
	
	`//For smaller IoD products such as the IoD-09 series:`
	`gfx.Panel(0,0,50,30,ORANGE);`
	`// Draws an ORANGE panel @(0,0) with:`
	`// width of 50 and height of 30`	

4.9. PanelRecessed

Syntax	**PanelRecessed (x, y, w, h, colour)**	
Arguments	**x, y, w, h, colour**	
	x, y	Specifies the top left corner position of the panel on the screen
	w, h	Specifies the width and height of the panel
	colour	16 bit colour of the panel
Returns	**none**	
Description	Draws a recessed 3 dimensional rectangular panel at a screen location defined by **x, y** parameters (top left corner). The size of the panel is set with the **w** and **h** parameters. The colour is defined by **colour**.	
Example	See below	

Example

```
gfx.PanelRecessed(100,150,100,30,YELLOW);
// Draws a YELLOW recessed panel @(100,150) with:
// width of 100 and height of 30

//For smaller IoD products such as the IoD-09 series:
gfx.PanelRecessed(0,0,50,30,YELLOW);
// Draws a YELLOW recessed panel @(0,0) with:
// width of 50 and height of 30
```

4.10. Slider

Syntax	**Slider (mode, x, y, x1, y1, bgColour, thColour, scale, value)**	
Arguments	**mode, x, y, x1, y1, bgColour, thColour, scale, value**	
	mode	Specifies the type of slider to be displayed
	x, y	Top left corner position of the slider on the screen
	x1, y1	Bottom right corner position of the slider on the screen
	bgColour	Specifies a 16 bit colour for the background of the slider
	thColour	Specifies a 16 bit colour for the thumb of the slider
	scale	Sets the full scale range of the slider for the thumb
	value	Relative position of the thumb on the slider bar
Returns	**none**	
Description	Draws a slider with the top left corner at **(x,y)** and bottom right corner **(x1,y1)**. The thumb will be drawn depending on the specified **scale** and **value**.	

Constant Definitions	Value
SLIDER_RAISED	0
SLIDER_SUNKEN	1

Example

```
// Draws a SILVER raised slider
gfx.Slider(SLIDER_RAISED,50,50,150,100,SILVER,BLACK,10,5);

// Draws a GREEN sunken slider
gfx.Slider(SLIDER_SUNKEN,50,150,150,200,GREEN,BLACK,20,15);


//For smaller IoD products such as the IoD-09 series:
// Draws a SILVER raised slider
gfx.Slider(SLIDER_RAISED,0,0,50,30,SILVER,BLACK,10,5);

// Draws a GREEN sunken slider
gfx.Slider(SLIDER_SUNKEN,0,0,50,30,GREEN,BLACK,20,15);
```

5. Text Functions

This section contains functions allow setting and checking of text properties. This section also includes functions for displaying text on the screen.

- Font
  - Set Font
  - Get Font
- TextSize
- TextColor
- TextWrap
- print
- println
- UserCharacter
- UserCharacterBG

5.1. Font

5.1.1. Set Font

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>Font (fontID)</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td rowspan="2"><b>Arguments</b></td><td colspan="2"><b>fontID</b></td></tr>
<tr><td><b>fontID</b></td><td>Specifies the font to use (FONT1 or FONT2)</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Sets the font to use for printing text.
<table>
<tr><th>Constant Definitions</th><th>Value</th></tr>
<tr><td>FONT1</td><td>1 (default)</td></tr>
<tr><td>FONT2</td><td>2</td></tr>
</table>
<b>Note:</b> Does nothing if fontID is not equal to FONT1 or FONT2</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><pre>gfx.Font(FONT2);
// Sets FONT2 as font to be used for printing text</pre></td></tr>
</table>

5.1.2. Get Font

<table>
<tr><td><b>Syntax</b></td><td><b>Font ()</b></td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Arguments</b></td><td><b>none</b></td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Returns</b></td><td><b><i>int8_t</i> Font</b></td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Description</b></td><td>Get the currently set text font</td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Example</b></td><td><pre>gfx.Font(FONT2);
// Sets FONT2 as font to be used for printing text

// Get current font then print its value
int8_t fontID = gfx.Font();
gfx.print("Current Font: "); gfx.println(fontID);</pre></td></tr>
</table>

5.2. TextSize

Syntax	**TextSize (multiplier)**	
Arguments	**multiplier**	
	multiplier	Specifies the text width and height multiplier
Returns	**none**	
Description	Sets the text width and height multiplier. Text will be printed magnified horizontally and vertically by this factor.	

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

5.3. TextColor

Syntax	**TextColor (fgColour) or TextColor (fgColour, bgColour)**	
Arguments	**fgColour, bgColour**	
	fgColour	Specifies the text foreground colour
	bgColour	Specifies the text background colour
Returns	**none**	
Description	Sets the text foreground and background colour for printing text. **Note**: If background colour is not specified, this function will treat it as transparent.	

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`	

5.4. TextWrap

Syntax	**TextWrap (mode)**	
Arguments	**mode**	
	mode	Use `true` to **ENABLE** and `false` to **DISABLE**
Returns	**none**	
Description	Text wrapping is **ENABLED** if mode is `true` otherwise text wrapping is **DISABLED** **Note:** The default mode is **ENABLED**.	

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

5.5. print

Syntax	**print (string)**	
Arguments	**string**	
	string	Specifies a string to print
Returns	**none**	
Description	Prints the specified string to the current cursor position	

Example	`gfx.MoveTo(50, 50);`
	`gfx.print("gen4-IoD");`	

5.6. println

Syntax	**println (string)**	
Arguments	**string**	
	string	Specifies a string to print
Returns	**none**	
Description	Prints the specified string to the current cursor position then moves the cursor position to the next line	

Example	`gfx.MoveTo(50, 50);`
	`gfx.println("gen4-IoD");`	

5.7. UserCharacter

Syntax	**UserCharacter (32bitArray, arraySize, x, y, fgColour, bgColour)**	
Arguments	**32bitArray, arraySize, x, y, fgColour, bgColour**	
	32bitArray	Specifies the array containing the character information
	arraySize	Specifies the size of the Array
	x, y	Specifies the top left coordinates
	fgColour	Specifies the character foreground colour
	bgColour	Specifies the character backgroun colour
Returns	**none**	
Description	User characters are ***W*** pixels wide and ***H*** pixels high. The user character function requires an array containing the **width** and ***height*** of the character followed by ***height*** x 32bit values	
Example		

```
uint32_t invader2a[20] = {

  24, 18,      // Character Width (Max: 32) and Height

  0x00000000, // ................................
  0x00060060, // .............11..........11.....
  0x000300C0, // ..............11........11......
  0x00618186, // .........11....11......11....11.
  0x0060C306, // .........11.....11....11.....11.
  0x0063FFC6, // .........11...111111111111...11.
  0x0067FFE6, // .........11..11111111111111..11.
  0x007E7E7E, // .........111111..111111..111111.
  0x007E7E7E, // .........111111..111111..111111.
  0x007FFFFE, // .........1111111111111111111111.
  0x003FFFFC, // ..........11111111111111111111..
  0x003FFFFC, // ..........11111111111111111111..
  0x001FFFF8, // ...........111111111111111111...
  0x00060060, // .............11..........11.....
  0x000C0030, // ............11............11....
  0x00180018, // ...........11..............11...
  0x0030000C, // ..........11................11..
  0x00000000  // ................................

};

for (int x = -10; x < 250; x++) {
  gfx.UserCharacter(invader2a, 20, x, 50, LIME, BLACK);
  delay(20);
}


//For smaller IoD products such as the IoD-09 series:

for (int y =-10; y < 180; y++) {
  gfx.UserCharacter(invader2a, 20, 50, y,LIME,BLACK);
  delay(20);
}
```

5.8. UserCharacterBG

Syntax	**UserCharacter (32bitArray, arraySize, x, y, fgColour, redrawBG, bgColour)**	
Arguments	**32bitArray, arraySize, x, y, fgColour, bgColour**	
	32bitArray	Specifies the array containing the character information
	arraySize	Specifies the size of the Array
	x, y	Specifies the top left coordinates
	fgColour	Specifies the character foreground colour
	redrawBG	Specifies whether the background image should be redrawn or not
	objectID	Specifies the background image (GCI object) to be restored
Returns	**none**	
Description	User characters are ***W*** pixels wide and ***H*** pixels high. The user character function requires an array containing the **width** and ***height*** of the character followed by ***height*** x 32bit values **Note**: This function does nothing is the character or a part of the character will be outside the display area.	

Example

```
uint32_t invader[20] = {

  24, 18,      // Character Width (Max: 32) and Height

  0x00000000, // ................................
  0x00060060, // .............11..........11.....
  0x000300C0, // ..............11........11......
  0x00618186, // .........11....11......11....11.
  0x0060C306, // .........11.....11....11.....11.
  0x0063FFC6, // .........11...111111111111...11.
  0x0067FFE6, // .........11..11111111111111..11.
  0x007E7E7E, // .........111111..111111..111111.
  0x007E7E7E, // .........111111..111111..111111.
  0x007FFFFE, // .........1111111111111111111111.
  0x003FFFFC, // ..........11111111111111111111..
  0x003FFFFC, // ..........11111111111111111111..
  0x001FFFF8, // ...........111111111111111111...
  0x00060060, // .............11..........11.....
  0x000C0030, // ............11............11....
  0x00180018, // ...........11..............11...
  0x0030000C, // ..........11................11..
  0x00000000  // ................................

};
gfx.PrintImageFile("Bkground.Gci");

for (int x = 0; x < 240-24; x++) {
  gfx.UserCharacterBG(invader, 20, x, 50, LIME, true, 0);
  delay(20);
}

//For smaller IoD products such as the IoD-09 series:
for (int x = 0; x < 150-24; x++) {
  gfx.UserCharacterBG(invader, 20, x,50,LIME,true, 0);
  delay(20);
}
```

6. Text Window Functions

This section contains functions that allows generation of a text window object and set its properties. Included as well are functions that allows printing of text inside the text window and clearing of text.

- TextWindow
- TextWindowRestore
- TWcolor
- TWwrite
- TWprint
- TWprintln
- TWcls

6.1. TextWindow

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>TextWindow (x, y, w, h, txtColour, bgColour) or</b>
<b>TextWindow (x, y, w, h, txtColour, bgColour, frameColour)</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="6"><b>Arguments</b></td><td colspan="2"><b>x, y, w, h, txtColour, bgColour, frameColour</b></td></tr>
<tr><td><b>x,y</b></td><td>Specifies the coordinates of the top-left corner of the text window</td></tr>
<tr><td><b>w,h</b></td><td>Specifies the width and height of the text window</td></tr>
<tr><td><b>txtColour</b></td><td>Specifies the text foreground colour</td></tr>
<tr><td><b>bgColour</b></td><td>Specifies the text background colour</td></tr>
<tr><td><b>frameColour</b></td><td>Specifies the frame colour</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Creates a text window at <b>x, y</b>, with dimensions <b>w, h</b>, text colour <b>txtColour</b>, background colour <b>bgColour,</b> and frame in colour <b>frameColour</b>.

<b>Note:</b> If no <b>frameColour</b> is specified, then no frame will not be rendered.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><pre>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.TextWindow(25,25, 190,270, BLACK, SILVER);
// Creates a SILVER text window @(25,25) with:
// width of 190 and height of 270 pixels
// The text printed in this text window is colour BLACK

// For smaller IoD products such as the IoD-09 series:
 gfx.TextWindow(0,0,50,30,BLACK,SILVER,DARKGRAY);
// Creates a SILVER text window @(0,0) with:
// width of 50 and height of 30 pixels
// and DARKGRAY frame
// The text printed in this text window is colour BLACK

gfx.TextWindow(0,0,20,40,BLACK,SILVER);
// Creates a SILVER text window @(0,0) with:
// width of 20 and height of 40 pixels
// The text printed in this text window is colour BLACK</pre></td></tr>
</table>

6.2. TextWindowRestore

Syntax	**TextWindowRestore ()**
Arguments	**none**
Returns	**none**
Description	Restore a previously created text window and its contents. **Note**: Contents cleared using `gfx.TWcls` will not be restored.
Example	See code below

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();


// For smaller IoD products such as the IoD-09 series:

gfx.TextWindow(0,0,50,30,BLACK,SILVER,DARKGRAY);
// Creates a SILVER text window @(0,0) with:
// width of 50 and height of 30 pixels
// and DARKGRAY frame
// The text printed in this text window is colour BLACK

gfx.Cls();
delay(1000);

// Retrieve deleted text window
gfx.TextWindowRestore();
```

6.3. TWcolor

Syntax	**TWcolor (fgColour) or** **TWcolor (fgColour, bgColour)**	
Arguments	**fgColour, bgColour**	
	fgColour	Specifies the colour of the text printed inside the text window
	bgColour	Specifies the background colour of the text window
Returns	**none**	
Description	Sets the specified foreground colour (**fgColour**) and background colour (**bgColour**) as the colours of the text in the text window. **Note**: 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.	

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. gen4-IoD");`
	
	`gfx.TWcolor(BROWN);`
	`// The text that will be printed next will be colour BROWN`
	
	`gfx.TWprintln("2. gen4-IoD");`
	
	`gfx.TWcolor(LIME,GRAY);`
	`// The text that will be printed next will be:`
	`// colour LIME with GRAY background`
	
	`gfx.TWprintln("3. gen4-IoD");`
	
	`// For smaller IoD products such as the IoD-09 series:`
	`gfx.Orientation(PORTRAIT);`
	`gfx.TextWindow(0,0,70,110,BLACK,SILVER,BROWN);`
	`// Creates a SILVER text window @(0,0) with:`
	`// width of 70 and height of 110 pixels and BROWN frame`
	`// The text printed in this text window is colour BLACK`
	`gfx.TWprintln("1. IoD-09");`
	
	`gfx.TWcolor(BROWN);`
	`// The text that will be printed next will be colour BROWN`
	
	`gfx.TWprintln("2. IoD-09");`
	
	`gfx.TWcolor(LIME,GRAY);`
	`// The text that will be printed next will be:`
	`// colour LIME with GRAY background`
	
	`gfx.TWprintln("3. IoD-09");`	

6.4. TWwrite

Syntax	**TWwrite (character)**	
Arguments	**character**	
	character	Specifies a single character to write on the text window
Returns	**none**	
Description	Write a single character to the text window	
Example	`gfx.TWwrite('4');`	

6.5. TWprint

Syntax	**TWprint (string)**	
Arguments	**string**	
	string	Specifies a string to print on the text window
Returns	**none**	
Description	Write a string to the text window	
Example	`gfx.TWprint("gen4-IoD");`	

6.6. TWprintln

Syntax	**TWprintln (string)**	
Arguments	**string**	
	string	Specifies a string to print on the text window
Returns	**none**	
Description	Write a string to the text window then move the text window cursor to a new line.	
Example	`gfx.TWprintln("gen4-IoD");`	

6.7. TWcls

Syntax	**TWcls ()**
Arguments	**none**
Returns	**none**
Description	Clears the contents of text window area. **Note**: Text windows contents cleared this way can not be retrieved using `gfx.TextWindowRestore`
Example	`gfx.TWcls();`

6.8. GetCommand

Syntax	**GetCommand ()**
Arguments	**none**
Returns	***String*** **Text/Command**
Description	Retrieves the text entered in text window since previous carriage return

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

7. Scroll Functions

These functions are used to perform a scrolling animation and to set parameters for scrolling effect for the display.

- ScrollEnable
- SmoothScrollSpeed
- Scroll
- getScrollOffset

Note: These functions are only available when in PORTRAIT orientation

7.1. ScrollEnable

Syntax	**ScrollEnable (mode)**	
Arguments	**mode**	
	mode	Use `true` to **enable** and `false` to **disable**
Returns	**none**	
Description	Enables hardware scrolling if **mode** is `true` otherwise disables it **Note**: This is disabled by default.	

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

7.2. SmoothScrollSpeed

Syntax	**SmoothScrollSpeed (delay)**	
Arguments	**delay**	
	delay	Specifies a short delay for scrolling
Returns	**none**	
Description	Smoothens the scroll animation for the automatic scrolling that occurs when the text being printed is going outside of the display area. **Note**: Default delay is 5	

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

7.3. Scroll

Syntax	**Scroll (pixels)**	
Arguments	**pixels**	
	pixels	Specifies the number of pixels
Returns	**none**	
Description	If scroll is enabled, this function scrolls the display by the specified number of pixels.	
Example	`gfx.Scroll(10); // Scroll the screen by 10 pixels`	

7.4. getScrollOffset

Syntax	**getScrollOffset ()**
Arguments	**none**
Returns	***int16_t*** **Scroll Offset**
Description	Returns the scroll offset from the last `gfx.Scroll` command

Example	`gfx.Scroll(20);`
	`int16_t scrollOffset = gfx.getScrollOffset();`
	
	`// Get scroll offset then print its value`
	`gfx.print("Scroll Offset: "); gfx.println(scrollOffset);`

8. 4D Graphics Functions

This section contains advanced graphics functions that utilizes 4D Graphics files.

- CheckSD
- Open4dGFX
- Userimage
- UserImageDR
- Userimages
- UserImagesDR
- PrintImage
- PrintImageFile
- LedDigitsDisplay
- LedDigitsDisplaySigned

Note: It is advisable to use Workshop4 IDE for its WYSIWYG environment when using these functions but with sufficient knowledge on 4D Graphics files, these can still be used with Arduino IDE.

8.1. CheckSD

Syntax	**CheckSD ()**
Arguments	**none**
Returns	***boolean* SD Card Status**
Description	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`.

Example	`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`

8.2. Open4dGFX

Syntax	**Open4dGFX (file4d)**	
Arguments	**file4d**	
	file4d	Specifies the filename of the 4D Graphics file (DAT and GCI files)
Returns	**none**	
Description	Opens 4D Graphics files. The DAT file is opened for parsing while the GCI file is opened for reading. **Note**: file4d should have no extension. Both GCI and DAT file should share the same filename. Also, 4D Graphics files follow the 8.3 DOS format	

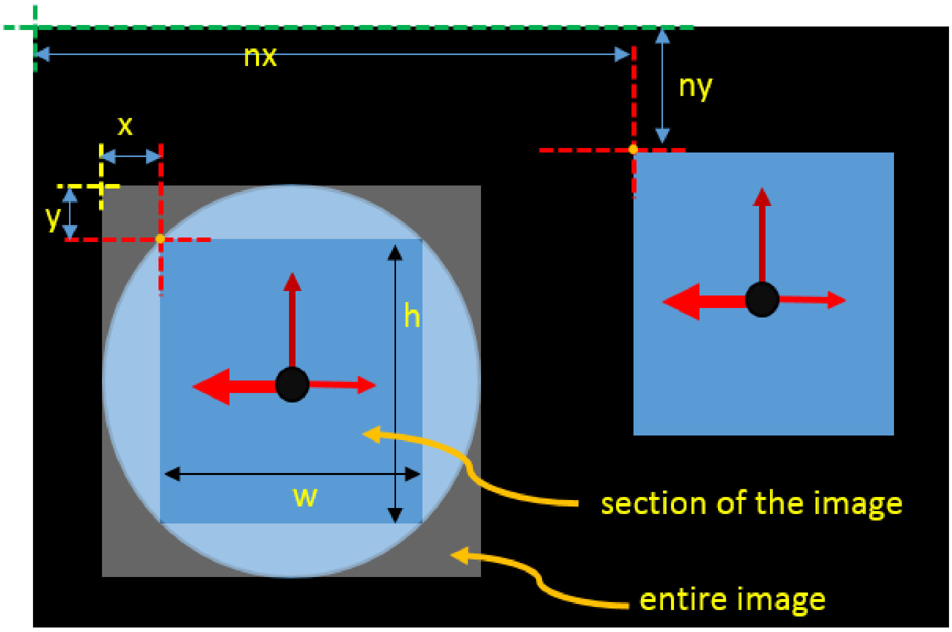
Example	`gfx.Open4dGFX("filename");`
	`// Opens filename.dat and filename.gci`	

8.3. UserImage

Syntax	**UserImage (objectID) or** **UserImage (objectID, frame, nx, ny)**	
Arguments	**objectID**	
	objectID	Specifies the object ID
Returns	**none**	
Description	**UserImage (objectID)** displays the target GCI object **objectID** at its set position determined by the 4D DAT file. **UserImage (objectID, frame, nx, ny)** displays the target GCI object **objectID** at **nx,ny**. These functions are normally used when displaying single-frame objects such as an image or a static text. When used with multiple-frame objects, they dislay the first frame. **Note**: The GCI and DAT files should have been previously opened with the function `gfx.Open4dGFX`	

Example	`gfx.UserImage(iImage1);`
	`// Show iImage1`
	
	`gfx.UserImage(iImage1,50,50);`
	`// Show iImage1 at (50,50)`	

8.4. UserImageDR

Syntax	**UserImageDR (objectID, x, y, w, h, nx, ny)**	
Arguments	**objectID, x, y, w, h, nx, ny**	
	objectID	Specifies the object ID
	x, y	Specifies the top left position of the section of the image to be drawn. This is relative to the position of the entire image.
	w, h	Specifies the width and height of the section of the image to be drawn
	nx, ny	Specifies the top left position at which the partial image will be drawn. This is relative to the origin (0,0).
Returns	**none**	
Description	Draws a section of image **objectID** at new co-ordinates **nx, ny**. The section starts at **x** and **y** and has a width of **w** and height of **h**. **Note**: The GCI and DAT files should have been previously opened with the function `gfx.Open4dGFX`  nx ny x y h w section of the image entire image	

Example	`gfx.UserImageDR(iImage1, 10, 5, 50, 50, 15, 10);`
	`// Partially draw iImage1 at (15,10)`
	`// The part drawn starts at (10,5) and`
	`// has a width and height of 50 pixels`	

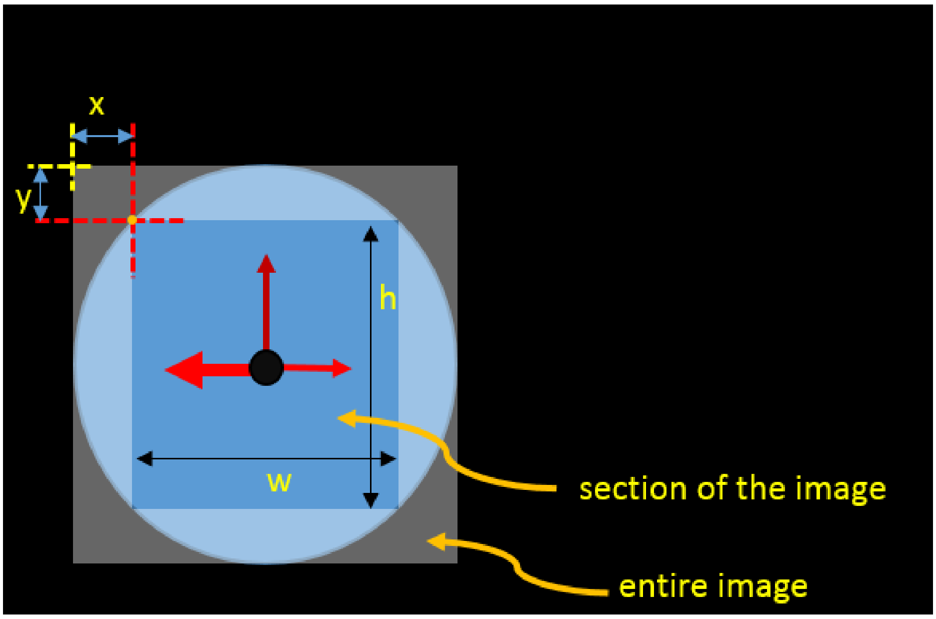
8.5. UserImages

Syntax	**UserImages (objectID, frame) or** **UserImages (objectID, frame, xOffset)** **UserImages (objectID, frame, nx, ny)**	
Arguments	**objectID, frame, nx, ny**	
	objectID	Specifies the object ID
	frame	Specifies the frame number of the target Userimage
	xOffset	Specifies the offset of the position of the image along the x-axis
	nx, ny	Specifies the new position of the image
Returns	**none**	
Description	Displays frame **frame** of the target GCI object **objectID**. When using **UserImages (objectID, frame)**, the frame is displayed at its set position determined by the 4D DAT file. When using **UserImages (objectID, frame, xOffset)**, the frame is displayed with the x position offset by **xOffset.** When using **UserImages (objectID, frame, nx, ny)**, the frame is displayed at **(nx,ny)**. These functions are used when displaying multiple-frame objects such as a slider or a gauge. **Note**: The GCI and DAT files should have been previously opened with the function `gfx.Open4dGFX`.	

Examples	`gfx.UserImages(iUserimage1, 10);`
	`// Show frame 10 of iUserimage1.`
	`// The position is taken from the DAT file.`
	
	`gfx.UserImages(iUserimage1, 10, 5);`
	`// Show frame 10 of iUserimage1.`
	`// The position is taken from the DAT file,`
	`// and the x-position is offset by 5 pixels`
	
	`gfx.UserImages(iUserimage1, 10, 50, 50);`
	`// Show frame 10 of iUserimage1 at (50,50)`	

8.6. UserImagesDR

Syntax	**UserimagesDR (objectID, frame, x, y, w, h)**	
Arguments	**objectID, frame, x, y, w, h**	
	objectID	Specifies the object ID
	frame	Specifies the frame of the user image
	x, y	Specifies the top left position of the section of the image to be drawn. This is relative to the position of the entire image.
	w, h	Specifies the width and height of the part of the image to be drawn.
Returns	**none**	
Description	Draws a section of frame **frame** of image **objectID**. The section starts at **x** and **y** and has a width of **w** and height of **h** **Note**: The GCI and DAT files should have been previously opened with the function `gfx.Open4dGFX`.	



Example

```
gfx.UserImagesDR(iUserimage1, 4, 10, 5, 50, 50);
// Partially draw frame 4 of iUserimage1
// The part drawn starts at (10,5) (relative to the position
// of the entire image) and has a width and height
// of 50 pixels
```

8.7. PrintImage

Syntax	**PrintImage (objectOffset)**	
Arguments	**objectOffset**	
	objectOffset	Specifies the offset of the GCI object to be printed
Returns	**none**	
Description	Prints the object specified by **objectOffset** from GCI file with its top left corner at the current cursor position	

Example	`gfx.MoveTo(50, 50);`
	`gfx.PrintImage(0x81EC00);`
	`// Prints image found at offset 0x81EC00`
	`// with its top left corner @(50,50)`	

8.8. PrintImageFile

Syntax	**PrintImageFile (filename)**	
Arguments	**filename**	
	filename	Specifies the GCI file containing the image to be printed
Returns	**none**	
Description	Prints the first frame of the first object from the specified GCI file at the current cursor position **Note**: Unlike the function `gfx.Open4dGFX`, this function requires the extension of the file	

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

8.9. LedDigitsDisplay

Syntax	**LedDigitsDisplay (value, index, maxDigits, minDigits, widthDigit, leadingBlanks) or** **LedDigitsDisplay (value, index, maxDigits, minDigits, widthDigit, leadingBlanks, x, y)**	
Arguments	**value, index, maxDigits, minDigits, widthDigit, leadingBlanks**	
	value	New value to display on the LED digits display
	index	Specifies which LedDigits object to modify
	digits	Maximum number of digits in the object
	minDigits	Minimum number of digits in the object. See note in the description for more information.
	widthDigit	Width of each digit image
	leadingBlanks	Specifies whether to display leading blanks or not
	x, y	Specifies the position at which the entire object will be displayed
Returns	**none**	
Description	This function handles displaying unsigned values to the Leddigits object and Customdigits object of a Workshop4 gen4-IoD or IoD-09 project. 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.	

Example	`gfx.LedDigitsDisplay(50, iiLeddigits1, 4, 3, 20, false);`
	`// Writes the value 50 to the iLedDigits1 object`
	
	`int ix = iiLeddigits1;`
	`gfx.LedDigitsDisplay(50, ix, 4, 3, 20, false, 5, 50);`
	`// Writes the value 50 to the iLeddigits1 object.`
	`// The object will then be shown at (5,50)`	

8.10. LedDigitsDisplaySigned

Syntax	**LedDigitsDisplaySigned (value, index, maxDigits, minDigits, widthDigit, leadingBlanks)** **or** **LedDigitsDisplaySigned (value, index, maxDigits, minDigits, widthDigit, leadingBlanks, x, y)**	
Arguments	**value, index, maxDigits, minDigits, widthDigit, leadingBlanks**	
	value	New value to display on the LED digits display
	index	Specifies which LedDigits object to modify
	digits	Maximum number of digits in the object
	minDigits	Minimum number of digits in the object. See note in the description for more information.
	widthDigit	Width of each digit image
	leadingBlanks	Specifies whether to display leading blanks or not
	x,y	Specifies the position at which the entire object will be displayed
Returns	**none**	
Description	This function handles displaying signed values to the Leddigits object and Customdigits object of a Workshop4 gen4-IoD or IoD-09 project. 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.	

Example	`int ix = iiLeddigits1;`
	`gfx.LedDigitsDisplaySigned(-50, ix, 4, 3, 20, false);`
	`// Writes the value -50 to the iLeddigits1 object`
	
	`gfx.LedDigitsDisplaySigned(50, ix, 4, 3, 20, false, 5, 50);`
	`// Writes the value 50 to the iLeddigits1 object.`
	`// The object will then be shown at (5,50)`	

9. Touch Functions

This section discusses about touch functions. These includes functions for checking the properties of touch as well as for evaluating the current touch action. This section does not apply to non-touch gen4-IoD and non-touch IoD-09 products.

- touch_Set
- touch_Update
- touch_Get
- touch_GetPen
- touch_GetX
- touch_GetY
- imageTouchEnable
- imageTouched
- XYposToDegree

9.1. touch_Set

Syntax	**touch_Set (mode)**	
Arguments	**mode**	
	mode	Use `true` to **enable** and `false` to **disable** touch
Returns	**none**	
Description	Enables/Disables touch functionality. Constant Definitions / Value: `TOUCH_ENABLE` — `true` `TOUCH_DISABLE` — `false` (default) **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**	
Example	`gfx.touch_Set(TOUCH_ENABLE); // Enable Touch`	

Constant Definitions	Value
TOUCH_ENABLE	true
TOUCH_DISABLE	false (default)

9.2. touch_Update

Syntax	**touch_Update ()**
Arguments	**none**
Returns	***Boolean* New Update**
Description	Updates the value of touch parameters which can be retrieved by the following functions. This function will return `true` if there is a new update. Otherwise, this function will return `false`. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**

Example	`if (gfx.touch_Update()) { // Update touch parameter values`
	`    // Evaluate touch if successful`
	`}`

Functions	Touch Parameter
gfx.touch_GetPen	Pen Value
gfx.touch_GetX	X Coordinate of Touch
gfx.touch_GetY	Y Coordinate of Touch
gfx.imageTouched	Object ID of Touched Image

9.3. touch_GetPen

Syntax	**touch_GetPen ()**
Arguments	**none**
Returns	***uint8_t*** **Touch Status**
Description	This function returns the pen/touch status from the last `gfx.touch_Update` execution.

Constant	Value	Meaning
NOTOUCH	0	No touch detected.
TOUCH_PRESSED	1	The touch panel is pressed.
TOUCH_RELEASED	2	The touch panel has been released.

This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.

Example

```
int touchStatus;

gfx.touch_Set(TOUCH_ENABLE); // Enable Touch

if (gfx.touch_Update()) { // Update touch parameter values

   // Get Pen/Touch Status
   touchStatus = gfx.touch_GetPen();
   if (touchStatus == NOTOUCH) {
      // Do something here
   }

   else if (touchStatus == TOUCH_PRESSED) {
      // Do something here
   }

   else if (touchStatus == TOUCH_RELEASED) {
      // Do something here
   }
}
```

9.4. touch_GetX

Syntax	**touch_GetX ()**
Arguments	**none**
Returns	***uint16_t*** **X Coordinate Touched Position**
Description	This function returns the X coordinate of the position touched on the screen from the last `gfx.touch_Update` execution. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**

Example	`int touchXpos;`
	`gfx.touch_Set(TOUCH_ENABLE); // Enable Touch`
	`if (gfx.touch_Update()) { // Update touch parameter values`
	`    // Get X Coordinate of touch position`
	`    touchXpos = gfx.touch_GetX();`
	`}`

9.5. touch_GetY

Syntax	**touch_GetY ()**
Arguments	**none**
Returns	***uint16_t* Y Coordinate Touched Position**
Description	This function returns the X coordinate of the position touched on the screen from the last `gfx.touch_Update` execution. **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**
Example	

```
int touchYpos;

gfx.touch_Set(TOUCH_ENABLE); // Enable Touch

if (gfx.touch_Update()) { // Update touch parameter values

   // Get Y Coordinate of touch position
   touchYpos = gfx.touch_GetY();

}
```

9.6. imageTouchEnable

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>imageTouchEnable (objectID, mode)</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="3"><b>Arguments</b></td><td colspan="2"><b>objectID, mode</b></td></tr>
<tr><td><b>objectID</b></td><td>Specifies the target GCI object</td></tr>
<tr><td><b>mode</b></td><td>Use true to enable touch for the object and false to disable</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Enable or disables touch for the specified object using mode as true or false respectively
<table>
<tr><th>Constant Definitions</th><th>Value</th></tr>
<tr><td>TOUCH_ENABLE</td><td>true</td></tr>
<tr><td>TOUCH_DISABLE</td><td>false (default)</td></tr>
</table>
<b>This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><pre>gfx.imageTouchEnable(iWinbutton1, true); // Enable Button 1
gfx.imageTouchEnable(iWinbutton2, true); // Enable Button 2</pre></td></tr>
</table>

9.7. imageTouched

Syntax	**imageTouched ()**
Arguments	**none**
Returns	***uint8_t*** **Touched Image**
Description	Returns the object ID of the last touched GCI object from the last `gfx.touch_Update` **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**

Example

```
gfx.touch_Set(TOUCH_ENABLE); // Enable Touch

if (gfx.touch_Update()) { // Update touch parameter values

   if (gfx.touch_GetPen() == TOUCH_PRESSED) {
     switch(gfx.imageTouched()){
       case iWinbutton1:
           gfx.println("Button 1 was touched");
         break;
       case iWinbutton2:
           gfx.println("Button 2 was touched");
         break;
     }
   }

}
```

9.8. XYposToDegree

Syntax	**XYposToDegree (xOffset, yOffset)**
Arguments	**none**
Returns	***int16_t*** **degrees**
Description	This function returns the angular equivalent of the offset of x and y position from the center of the object **This function does not apply to non-touch gen4-IoD and non-touch IoD-09 products.**
Example	

```
int touchXpos, touchYpos, deg;

if (gfx.touch_Update()) { // Update touch parameter values

   // Get X Coordinate of touch position
   touchXpos    = gfx.touch_GetX();
   // Get Y Coordinate of touch position
   touchYpos    = gfx.touch_GetY();

   deg = gfx.XYposToDegree(x-242,y-70); // OffsetX, OffsetY

   if (deg < 45)
       // anything in the first 'dead zone' is minvalue
       deg = 0 ;
   else if (deg > 315)
       // anything in the last 'dead zone' is maxvalue
       deg = 270 ;
   else
       deg -= 45 ; // offset by -baseangle
   }

   // convert degrees to position
   posit = degrees * 100 / 270 ;
   gfx.UserImages(iKnob1, posit);

}
```

10. Wi-Fi Functions

These functions allows the users to download and use files from the internet or local network.

- DownloadFile
- PrintImageWifi

10.1. DownloadFile

Syntax	**DownloadFile (Addr, Fname)** **DownloadFile (Addr, port, hFile, Fname)**	
Arguments	**Addr, port, hFile, Fname**	
	Addr	Specifies the web address or local server hosting the file
	port	Specifies the port number to use when accessing the file from the local server
	hFile	Specifies the filename of the file to download
	Fname	Specifies the filename to used when saving the file to the uSD Card
Returns	**none**	
Description	**Mode 1: Addr, Fname** Downloads the file from the specified web address and save it with the specified filename. **Mode 2: Addr, port, hFile, Fname** Downloads the file from the local server through the specified port and save it with the specified filename. **Note**: It is advisable to follow the 8.3 DOS format	

Examples	`String i;`
	`i = "http://www.4dsystems.com.au/downloads/RAW/conectd.gci";`
	`gfx.DownloadFile(i, "conectd.gci");`
	
	`String localServer = "http://192.168.0.35";`
	`gfx.DownloadFile(localServer, 9969, "space.gci","space.gci");`
	`// Download the file "space.gci" from a local server`
	`// The file "space.gci" is then created on the uSD card.`	

10.2. PrintImageWifi

Syntax	**PrintImageWifi (Addr) or** **PrintImageWifi (Addr, port, hFile)**	
Arguments	**Addr, port, hFile**	
	Addr	Specifies the URL of the GCI file or the local server hosting the file
	port	Specifies the port to be used when accessing the local server
	hFile	Specifies the file from the local server
Returns	**none**	
Description	Prints the file at the current cursor position **Mode 1: Addr** Prints the file from the specified web address at the current cursor position. **Mode 2: Addr, port, hFile** Access the local server through the specified port and print the specified file at the current cursor position.	

Example	`gfx.MoveTo(50, 50);`
	`String i;`
	`i="http://www.4dsystems.com.au/downloads/RAW/conectd.gci";`
	`gfx.PrintImageWifi(i);`
	`// If the display module is connected to the internet,`
	`// Display the image from the web`
	
	`gfx.PrintImageWifi("http://192.168.0.35",9969,"space.gci");`
	`// Print the image inside the file "space.gci"`
	`// from a local server`	

11. GRAM Functions

These functions allow direct display access for fast blitting operations:

- setGRAM
- WrGRAM
- WrGRAM16
- WrGRAMs
- WrGRAMs16

11.1. setGRAM

Syntax	**setGRAM (x0, y0, x1, y1)**	
Arguments	**x0, y0, x1, y1**	
	x0, y0	Specifies the top left of GRAM window
	x1, y1	Specifies the bottom right of GRAM window
Returns	**none**	
Description	Prepares the GRAM area for access	
Example		

```
gfx.setGRAM(101, 101, 200, 200);
// Sets a 20 by 20 display area as GRAM

for (int i = 0; i < 200 ; i++) {
  int color = rand();
  for (int j = 0; j < 200 ; j++) {
    gfx.WrGRAM16(color);
  }
}


// For smaller IoD products such as the IoD-09 series:
gfx.setGRAM(1,1,50,50);
// Sets a 50 by 50 display area as GRAM

for (int i = 0; i < 50 ; i++) {
  int color =rand();
  for (int j = 0; j < 50 ; j++) {
    gfx.WrGRAM16(color);
  }
}
```

11.2. WrGRAM

Syntax	**WrGRAM (colours)**	
Arguments	**colours**	
	colours	32 bit value containing two 16 bit colour values
Returns	**none**	
Description	Writes two 16 bit colours from a 32 bit value to the current pixel position **Note**: The position is moved by two pixels.	
Example		

```
gfx.Cls(YELLOW); // Clear the screen with YELLOW

gfx.setGRAM(101, 101, 200, 200);

for (int i = 0; i < 200 ; i++) {
  for (int j = 0; j < 100 ; j++) {
    gfx.WrGRAM(BLACK << 16 | WHITE);
  }
} // Create 200 vertical lines of BLACK and WHITE on GRAM


// For smaller IoD products such as the IoD-09 series:
gfx.Cls(YELLOW);// Clear the screen with YELLOW

gfx.setGRAM(1,1,50,50);

for (int i = 0; i < 50 ; i++) {
  for (int j = 0; j < 25 ; j++) {
    gfx.WrGRAM(BLACK<< 16 |WHITE);
  }
}// Create vertical lines of BLACK and WHITE on GRAM
```

11.3. WrGRAM16

<table>
<tr><td><b>Syntax</b></td><td colspan="2"><b>WrGRAM16 (colour)</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="2"><b>Arguments</b></td><td colspan="2"><b>colour</b></td></tr>
<tr><td><b>colour</b></td><td>16 bit colour value</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Returns</b></td><td colspan="2"><b>none</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">Writes a 16 bit colour to the current pixel position

<b>Note</b>: The position is moved by one pixel.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Example</b></td><td colspan="2"><pre>
gfx.setGRAM(101, 101, 200, 200);

for (int i = 0; i < 200 ; i++) {
  int color = rand();
  for (int j = 0; j < 200 ; j++) {
    gfx.WrGRAM16(color);
  }
} // Create 200 horizontal lines w/ random colors on GRAM

// For smaller IoD products such as the IoD-09 series:
gfx.setGRAM(1,1,50,50);

for (int i = 0; i < 50 ; i++) {
  int color =rand();
  for (int j = 0; j < 50 ; j++) {
    gfx.WrGRAM16(color);
  }
}// Create 50 horizontal lines w/ random colors on GRAM
</pre></td></tr>
</table>

11.4. WrGRAMs

Syntax	**WrGRAMs (coloursArray, length)**	
Arguments	**coloursArray, length**	
	coloursArray	Pointer to a 32 bit data array
	length	Length of 32 bit data to write to GRAM
Returns	**none**	
Description	Writes a number **(2 * length)** of 16 bit colours from a 32 bit data array to the current cursor position **Note**: The position is moved by **(2 * length)** pixels.	
Example	See code below	

```
uint32_t data[5] = {
  WHITE << 16 | RED,
  GREEN << 16 | YELLOW,
  BROWN << 16 | LIME,
  BLACK << 16 | ORANGE,
  CYAN  << 16 | MAGENTA
};

gfx.setGRAM(101, 101, 200, 200);

for (int i = 0; i < 200 ; i++) {
  for (int j = 0; j < 20 ; j++) {
    gfx.WrGRAMs(data, 5);
    // Writes colours from 32bit array
  }
}


// For smaller IoD products such as the IoD-09 series:
gfx.setGRAM(1,1,50,50);

for (int i = 0; i < 50 ; i++) {
  for (int j = 0; j < 5 ; j++) {
    gfx.WrGRAMs(data, 5);
// Writes colours from 32bit array
  }
}
```

11.5. WrGRAMs16

Syntax	**WrGRAMs16 (colourArray, length)**	
Arguments	**colourArray, length**	
	colourArray	Pointer to a 16 bit data array
	length	Length of 16 bit data to write to GRAM
Returns	**none**	
Description	Writes a number **(length)** of 16 bit colours from a 16 bit data array to the current cursor position **Note**: The position is moved by **(length)** pixels.	

Example

```
uint16_t data[10] = {
  WHITE, RED, GREEN, YELLOW, BROWN,
  LIME, BLACK, ORANGE, CYAN, MAGENTA
};

gfx.setGRAM(101, 101, 200, 200);

for (int i = 0; i < 200 ; i++) {
  for (int j = 0; j < 20 ; j++) {
    gfx.WrGRAMs16(data, 10);
    // Writes colours from 16 bit array
  }
}

// For smaller IoD products such as the IoD-09 series:
gfx.setGRAM(1,1,50,50);

for (int i = 0; i < 50 ; i++) {
  for (int j = 0; j < 5 ; j++) {
    gfx.WrGRAMs16(data, 10);
// Writes colours from 16 bit array
  }
}
```

12. Sound Module Functions

The following are functions from the SOMOIoD library.

- Command
- LastCommand

12.1. Command

Syntax	**Command (cmd) or** **Command (cmd, value1) or** **Command (cmd, value1, value2)**	
Arguments	**cmd, value1, value2**	
	cmd	Specifies the action/command for the sound module
	value1, value2	Specifies the value(s) to be sent with the command being used
Returns	**none**	

Description

Sends a command for the sound module to execute. For a detailed discussion of the commands that can be used with SOMO-II and MOTG-MP3, please refer to their corresponding datasheets.

Command	First Value	Second Value
PLAY	---	---
STOP	---	---
PREVIOUS	---	---
NEXT	---	---
SOURCE_SD	---	---
SOURCE_USB	---	---
EQ_BASS	---	---
VOLUMEMAX	---	---
VOLUMEMIN	---	---
VOLUMEUP	---	---
VOLUMEDOWN	---	---
CONTINUOUS	---	---
RANDOM	---	---
PAUSE	---	---
EQ_NORMAL	---	---
EQ_POP	---	---
EQ_ROCK	---	---
EQ_JAZZ	---	---
EQ_CLASSIC	---	---
REPEAT	---	---
SINGLE	---	---
SLEEP	---	---
RESET	---	---
SPECIFY_TRACK	Track Number	---
VOLUME	Volume (0-30)	---
REPEAT_A_TRACK	Track Number	---
FOLDER_TRACK	Folder Number	Track Number

Note: The commands specified in the table able may not be totally the same as discussed with the datasheets.

Example

```
sound.Command(PLAY);   // Play the first track
delay(10000);          // Let the track play for 10s

sound.Command(PAUSE); // Pause
delay(2000);           // Wait for 2s

sound.Command(PLAY);   // Resume Playing
```

12.2. LastCommand

Syntax	**LastCommand ()**
Arguments	**none**
Returns	***uint8_t*** **Last Command**
Description	This function returns the last command sent to the sound module using the `sound.Command` function.

Example	`sound.Command(PLAY);   // Play the first track`
	`delay(10000);          // Let the track play for 10s`
	
	`sound.Command(PAUSE); // Pause`
	`delay(2000);           // Wait for 2s`
	
	`sound.Command(PLAY);   // Resume Playing`
	
	`int lastCommand = sound.LastCommand();`
	`// Get Last Command Sent`

13. Revision History

Revision No.	Description	Revision Date
1.0	Initial document release	08/01/2017

14. Legal Notice

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.

Disclaimer of Warranties & Limitation of Liability

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.

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.

15. Contact Information

For Technical Support: www.4dsystems.com.au/support

For Sales Support: sales@4dsystems.com.au

Website: www.4dsystems.com.au