µGUI

Open Source GUI module for embedded systems

Reference Guide

Software version: v0.3
Document version: v0.3

www.embeddedlightning.com
(visit for latest updates)

Contents

1 Introduction

1.1 What is µGUI

µGUI is a free and open source graphic library for embedded systems. It is platform-independent and can be easily ported to almost any microcontroller system. As long as the display is capable of showing graphics, µGUI is not restricted to a certain display technology. Therefore, display technologies such as LCD, TFT, E-Paper, LED or OLED are supported. The whole module consists of two files: **ugui.c** and **ugui.h**.

1.2 µGUI Features

- µGUI supports any color, grayscale or monochrome display
- µGUI supports any display resolution
- µGUI supports multiple different displays
- µGUI supports any touch screen technology (e.g. AR, PCAP)
- µGUI supports windows and objects (e.g. button, textbox)
- 16 different fonts available
- integrated and free scalable system console
- basic geometric functions (e.g. line, circle, frame etc.)
- can be easily ported to almost any microcontroller system
- no risky dynamic memory allocation required

1.3 µGUI Requirements

µGUI is platform-independent, so there is no need to use a certain embedded system. In order to use µGUI, only two requirements are necessary:

- a C-function which is able to control pixels of the target display.
- integer types for the target platform have to be adjusted in **ugui.h**.

2 Windows

In addition to its 2D geometric functions, µGUI also supports windows. Each window can contain several objects like buttons, textboxes or images. The following section describes how to use windows.

Figure 1: µGUI window example

2.1 Window Management

Each window needs its own dedicated object buffer. In this buffer all window objects will be stored. Therefore, the size of this buffer defines the maximum object count of its associated window.

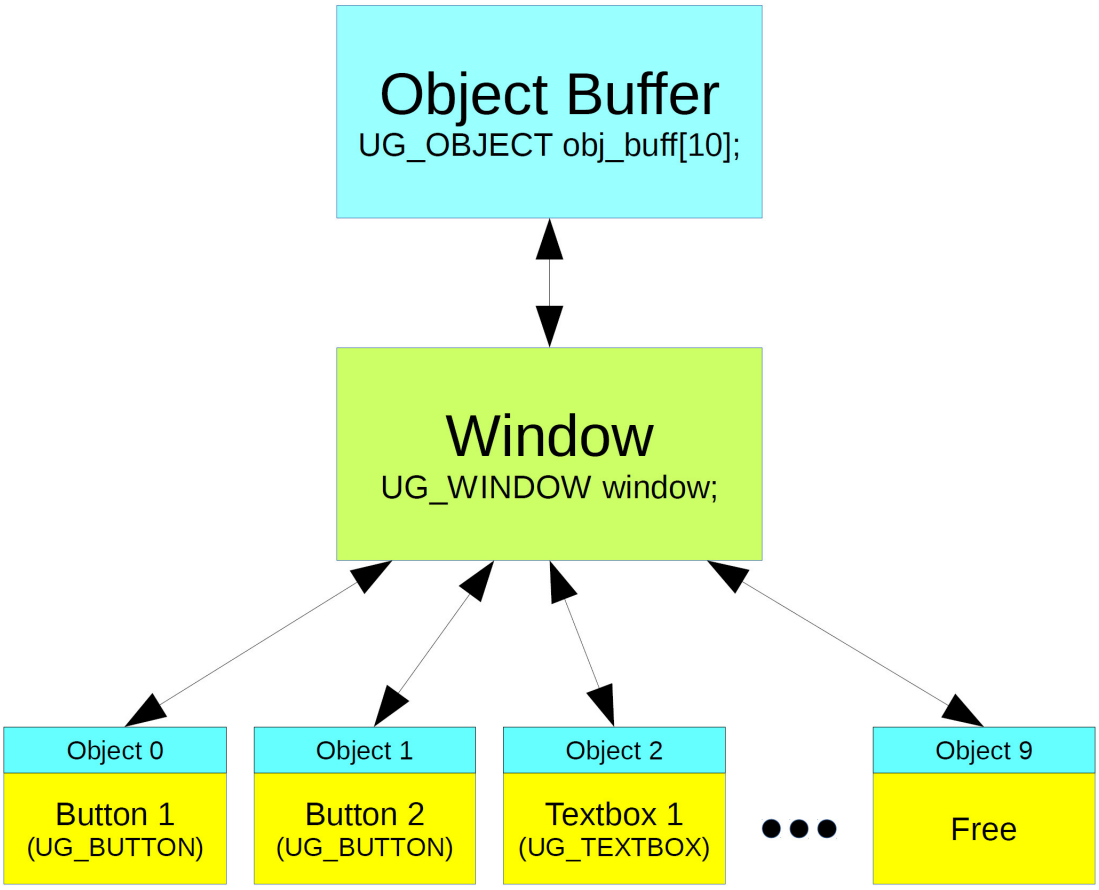


Figure 2: Object buffer mechanism

The link between a window and its object buffer will be established by calling the UG_WindowCreate() function. This function also connects a callback function to the window which will be called on incoming events like touches on a button. Once a window has been created, µGUI takes care of all window-related processes:

- drawing the window itself (including all objects)
- handling all object-related events
- processing incoming touch data
- calling the callback function which has been linked to the window

Since different objects support different features, not all object-specific data can be stored in the object buffer. Therefore, each object needs additionally its own container (e.g. UG_BUTTON button_1) to store special object-specific data. The linking process between a window and such a container takes place by calling the corresponding object create function (e.g. UG_ButtonCreate).

Example:

```
void window_1_callback( UG_MESSAGE* msg )
{
   //...
}

#define MAX_OBJECTS  10
int main( void )
{
   UG_WINDOW window_1;                          /* Window             */
   UG_BUTTON button_1;                          /* Button container   */
   UG_BUTTON button_2;                          /* Button container   */
   UG_BUTTON button_3;                          /* Button container   */
   UG_OBJECT obj_buff_wnd_1[MAX_OBJECTS];       /* Object buffer      */

   //...

   /* Create the window (link the object buffer and the callback function to the
      window) */
   UG_WindowCreate( &window_1, obj_buff_wnd_1, MAX_OBJECTS, window_1_callback );

   /* Create some buttons (link each object container to the window) */
   UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 10, 10, 110, 60 );
   UG_ButtonCreate( &window_1, &button_2, BTN_ID_1, 10, 80, 110, 130 );
   UG_ButtonCreate( &window_1, &button_3, BTN_ID_2, 10, 150, 110, 200 );

   /* Finally, show the window */
   UG_WindowShow( &window_1 );

   //...
}
```

2.2 Update Function

µGUI controls the refreshing process of each window including all daughter objects. This is all done by the function UG_Update(). Therefore, this function has to be called periodically either in an ISR or in background.

Note: If the user forgets to call this function, nothing will happen.

2.3 Callback Function

While beeing created, every window will be linked to a dedicated callback function. Once the window has detected an event (e.g. touch on a button), it will prepare a message which describes the event and pass it on to the callback function. The callback function receives this message and can then decide how to react to the event.

Example:

```
void window_1_callback( UG_MESSAGE* msg )
{
   if ( msg->type == MSG_TYPE_OBJECT )
   {
      if ( msg->id == OBJ_TYPE_BUTTON )
      {
         switch( msg->sub_id )
         {
            case BTN_ID_0:
            {
               /* Do something! */
               break;
            }
         }
      }
   }
}
```

3 Touch and Input devices

µGUI supports any touch technology (like analog resistive or projected capacitive) as long as it provides input data in X/Y format. It is very simple to connect a touch device to µGUI. The only thing the user needs to do is to call UG_TouchUpdate(). This function transfers the raw touch data to the µGUI touch processor which then will detect, validate and track all touch events.

Example:

```
TouchData = ReadTouchData();
if( TouchData->TouchDetected )
{
```

```
    UG_TouchUpdate( TouchData->X, TouchData->Y, TOUCH_STATE_PRESSED );
}
else
{
    UG_TouchUpdate(-1, -1, TOUCH_STATE_RELEASED );
}
```

Note: Mouses or joysticks are currently not supported since those input devices require a cursor which is not available at the moment.

4 Driver Support

µGUI can be enhanced by linking external driver modules to the core system. The following section describes how to use them and which drivers are supported.

4.1 Hardware Acceleration

Although µGUI only needs a user pset function in order to work properly, filling large areas on the target display could be very slow (especially on small microcontroller systems). Therefore, µGUI supports platform-specific hardware acceleration. By using this feature the user gives µGUI control of special hardware functions which are supported by the target platform. After an accelerator has been registered, the respective function will be done in hardware. Compared to a non-hardware accelerated function, a hardware accelerated one can be easily 100 times faster. Each user-provided hardware accelerator has to have a prototype which is identical to its non-accelerated counterpart except for its return type. If the hardware accelerator was able to handle the requested operation, it returns UG_RESULT_OK, otherwise UG_RESULT_FAIL. If µGUI detects that a hardware accelerator has returned UG_RESULT_FAIL, it will perform the operation again, but this time in software. All drivers can be registered and controlled by using the UG_DRIVER_x - functions. Please refer to the corresponding function section (7.2) for further information.

对于这个问题有一个简单的方案:所有的操作先对RAM操作,定时拷贝 RAM到屏幕,但也会遇到 RAM不够一个全屏的问题。

4.1.1 DRIVER_DRAW_LINE

The driver DRIVER_DRAW_LINE accelerates the function UG_DrawLine().
Its prototype has to be:

```
/* Hardware accelerator for UG_DrawLine */
UG_RESULT _HW_DrawLine( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c );
```

4.1.2 DRIVER_FILL_FRAME

The driver DRIVER_FILL_FRAME accelerates the function UG_FillFrame().
Its prototype has to be:

```
/* Hardware accelerator for UG_FillFrame */
UG_RESULT _HW_FillFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c )
```

5 Colors

5.1 Color Space

µGUI uses the color space ARGB8888:

D31-D24	D23-D16	D15-D8	D7-D0
Alpha Channel	Red	Green	Blue

The alpha channel is currently not supported, but will be implemented in future versions.

5.2 Predefined Colors

µGUI comes with the following predefined colors (RGB888)[1].

COLOR NAME	RGB VALUE (RGB888)
C_MAROON	0x800000
C_DARK_RED	0x8B0000
C_BROWN	0xA52A2A
C_FIREBRICK	0xB22222
C_CRIMSON	0xDC143C
C_RED	0xFF0000
C_TOMATO	0xFF6347
C_CORAL	0xFF7F50
C_INDIAN_RED	0xCD5C5C
C_LIGHT_CORAL	0xF08080
C_DARK_SALMON	0xE9967A
C_SALMON	0xFA8072
C_LIGHT_SALMON	0xFFA07A
C_ORANGE_RED	0xFF4500
C_DARK_ORANGE	0xFF8C00
C_ORANGE	0xFFA500
C_GOLD	0xFFD700
C_DARK_GOLDEN_ROD	0xB8860B
C_GOLDEN_ROD	0xDAA520
C_PALE_GOLDEN_ROD	0xEEE8AA
C_DARK_KHAKI	0xBDB76B
C_KHAKI	0xF0E68C
C_OLIVE	0x808000
C_YELLOW	0xFFFF00
C_YELLOW_GREEN	0x9ACD32
C_DARK_OLIVE_GREEN	0x556B2F
C_OLIVE_DRAB	0x6B8E23
C_LAWN_GREEN	0x7CFC00
C_CHART_REUSE	0x7FFF00

[1] Source: http://www.rapidtables.com/web/color/RGB_Color.htm

C_GREEN_YELLOW	0xADFF2F
C_DARK_GREEN	0x006400
C_GREEN	0x008000
C_FOREST_GREEN	0x228B22
C_LIME	0x00FF00
C_LIME_GREEN	0x32CD32
C_LIGHT_GREEN	0x90EE90
C_PALE_GREEN	0x98FB98
C_DARK_SEA_GREEN	0x8FBC8F
C_MEDIUM_SPRING_GREEN	0x00FA9A
C_SPRING_GREEN	0x00FF7F
C_SEA_GREEN	0x2E8B57
C_MEDIUM_AQUA_MARINE	0x66CDAA
C_MEDIUM_SEA_GREEN	0x3CB371
C_LIGHT_SEA_GREEN	0x20B2AA
C_DARK_SLATE_GRAY	0x2F4F4F
C_TEAL	0x008080
C_DARK_CYAN	0x008B8B
C_AQUA	0x00FFFF
C_CYAN	0x00FFFF
C_LIGHT_CYAN	0xE0FFFF
C_DARK_TURQUOISE	0x00CED1
C_TURQUOISE	0x40E0D0
C_MEDIUM_TURQUOISE	0x48D1CC
C_PALE_TURQUOISE	0xAFEEEE
C_AQUA_MARINE	0x7FFFD4
C_POWDER_BLUE	0xB0E0E6
C_CADET_BLUE	0x5F9EA0
C_STEEL_BLUE	0x4682B4
C_CORN_FLOWER_BLUE	0x6495ED
C_DEEP_SKY_BLUE	0x00BFFF
C_DODGER_BLUE	0x1E90FF
C_LIGHT_BLUE	0xADD8E6
C_SKY_BLUE	0x87CEEB
C_LIGHT_SKY_BLUE	0x87CEFA
C_MIDNIGHT_BLUE	0x191970
C_NAVY	0x000080
C_DARK_BLUE	0x00008B
C_MEDIUM_BLUE	0x0000CD
C_BLUE	0x0000FF
C_ROYAL_BLUE	0x4169E1
C_BLUE_VIOLET	0x8A2BE2
C_INDIGO	0x4B0082
C_DARK_SLATE_BLUE	0x483D8B
C_SLATE_BLUE	0x6A5ACD
C_MEDIUM_SLATE_BLUE	0x7B68EE
C_MEDIUM_PURPLE	0x9370DB

C_DARK_MAGENTA	0x8B008B
C_DARK_VIOLET	0x9400D3
C_DARK_ORCHID	0x9932CC
C_MEDIUM_ORCHID	0xBA55D3
C_PURPLE	0x800080
C_THISTLE	0xD8BFD8
C_PLUM	0xDDA0DD
C_VIOLET	0xEE82EE
C_MAGENTA	0xFF00FF
C_ORCHID	0xDA70D6
C_MEDIUM_VIOLET_RED	0xC71585
C_PALE_VIOLET_RED	0xDB7093
C_DEEP_PINK	0xFF1493
C_HOT_PINK	0xFF69B4
C_LIGHT_PINK	0xFFB6C1
C_PINK	0xFFC0CB
C_ANTIQUE_WHITE	0xFAEBD7
C_BEIGE	0xF5F5DC
C_BISQUE	0xFFE4C4
C_BLANCHED_ALMOND	0xFFEBCD
C_WHEAT	0xF5DEB3
C_CORN_SILK	0xFFF8DC
C_LEMON_CHIFFON	0xFFFACD
C_LIGHT_GOLDEN_ROD_YELLOW	0xFAFAD2
C_LIGHT_YELLOW	0xFFFFE0
C_SADDLE_BROWN	0x8B4513
C_SIENNA	0xA0522D
C_CHOCOLATE	0xD2691E
C_PERU	0xCD853F
C_SANDY_BROWN	0xF4A460
C_BURLY_WOOD	0xDEB887
C_TAN	0xD2B48C
C_ROSY_BROWN	0xBC8F8F
C_MOCCASIN	0xFFE4B5
C_NAVAJO_WHITE	0xFFDEAD
C_PEACH_PUFF	0xFFDAB9
C_MISTY_ROSE	0xFFE4E1
C_LAVENDER_BLUSH	0xFFF0F5
C_LINEN	0xFAF0E6
C_OLD_LACE	0xFDF5E6
C_PAPAYA_WHIP	0xFFEFD5
C_SEA_SHELL	0xFFF5EE
C_MINT_CREAM	0xF5FFFA
C_SLATE_GRAY	0x708090
C_LIGHT_SLATE_GRAY	0x778899
C_LIGHT_STEEL_BLUE	0xB0C4DE
C_LAVENDER	0xE6E6FA

C_FLORAL_WHITE	0xFFFAF0
C_ALICE_BLUE	0xF0F8FF
C_GHOST_WHITE	0xF8F8FF
C_HONEYDEW	0xF0FFF0
C_IVORY	0xFFFFF0
C_AZURE	0xF0FFFF
C_SNOW	0xFFFAFA
C_BLACK	0x000000
C_DIM_GRAY	0x696969
C_GRAY	0x808080
C_DARK_GRAY	0xA9A9A9
C_SILVER	0xC0C0C0
C_LIGHT_GRAY	0xD3D3D3
C_GAINSBORO	0xDCDCDC
C_WHITE_SMOKE	0xF5F5F5
C_WHITE	0xFFFFFF

6 Fonts

µGUI comes with the following fonts[2].

Note: A font has to be enabled in the config section of **ugui.h** before it can be used!

6.1 FONT_4X6

Figure 3: FONT_4X6

6.2 FONT_5X8

Figure 4: FONT_5X8

[2]Source: Benedikt K. http://www.mikrocontroller.net/user/show/benedikt

6.3 FONT_5X12

Figure 5: FONT_5X12

6.4 FONT_6X8

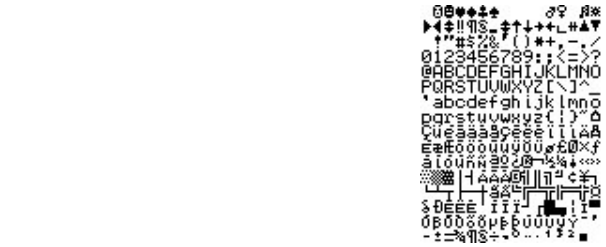


Figure 6: FONT_6X8

6.5 FONT_6X10

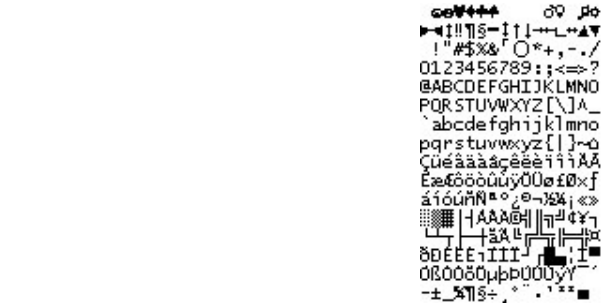


Figure 7: FONT_6X10

6.6 FONT_7X12

Figure 8: FONT_7X12

6.7 FONT_8X8

Figure 9: FONT_8X8

6.8 FONT_8X12

Figure 10: FONT_8X12

6.9 FONT_8X14

Figure 11: FONT_8X14

6.10 FONT_10X16

Figure 12: FONT_10X16

6.11 FONT_12X16

Figure 13: FONT_12X16

6.12 FONT_12X20

Figure 14: FONT_12X20

6.13 FONT_16X26

Figure 15: FONT_16X26

6.14 FONT_22X36

Figure 16: FONT_22X36

6.15 FONT_24X40

Figure 17: FONT_24X40

6.16 FONT_32X53

Figure 18: FONT_32X53

7 Functions

7.1 Classic Functions

7.1.1 UG_Init()

This function initializes the GUI module. Furthermore it links the user pset function to the µGUI core.

Prototype:

```
UG_S16 UG_Init( UG_GUI* g, void (*p)(UG_S16,UG_S16,UG_COLOR), UG_S16 x, UG_S16 y );
```

Parameters:

UG_GUI* g	Pointer to the GUI structure
void (*p)	Function pointer to the user pset-function
UG_S16 x	X-Dimension (= X-Resolution) of the display
UG_S16 y	Y-Dimension (= Y-Resolution) of the display

Example:

```
void UserPixelSetFunction( UG_S16 x, UG_S16 y,UG_COLOR c )
{
   // Your code....
}

UG_GUI gui; // Global GUI structure

int main( void )
{
   UG_Init(&gui, UserPixelSetFunction, 320, 240);
   //...
   //...
}
```

7.1.2 UG_SelectGUI()

With this function you can switch between different GUIs / displays.

Prototype:

```
UG_S16 UG_SelectGUI( UG_GUI* g );
```

Parameters:

UG_GUI* g Pointer to the GUI structure

Example:

```
UG_GUI gui_oled; // Global GUI structure (OLED)
UG_GUI gui_tft; // Global GUI structure (TFT)

int main( void )
{
   UG_Init(&gui_oled, OLEDPixelSetFunction, 128, 64);
   UG_Init(&gui_tft, TFTPixelSetFunction, 480, 272);
   UG_SelectGUI( &gui_oled );
   //...
   UG_SelectGUI( &gui_tft );
   //...
}
```

7.1.3 UG_FontSelect()

With this function you can select a font.
The following fonts are available:

FONT_4X6
FONT_5X8
FONT_5X12
FONT_6X8
FONT_6X10
FONT_7X12
FONT_8X8
FONT_8X12
FONT_8X14
FONT_10X16
FONT_12X16
FONT_12X20
FONT_16X26
FONT_22X36
FONT_24X40
FONT_32X53

Note:
A font has to be enabled in the config section of **ugui.h** before it can be used!

Prototype:

```
void UG_FontSelect( const UG_FONT* font )
```

Parameters:

const UG_FONT* font Pointer to the font

Example:

```
int main( void )
{
  //...
  UG_FontSelect( &FONT_8X8 );
  //...
}
```

7.1.4 UG_FillScreen()

Fills the whole screen with the selected color.

Prototype:

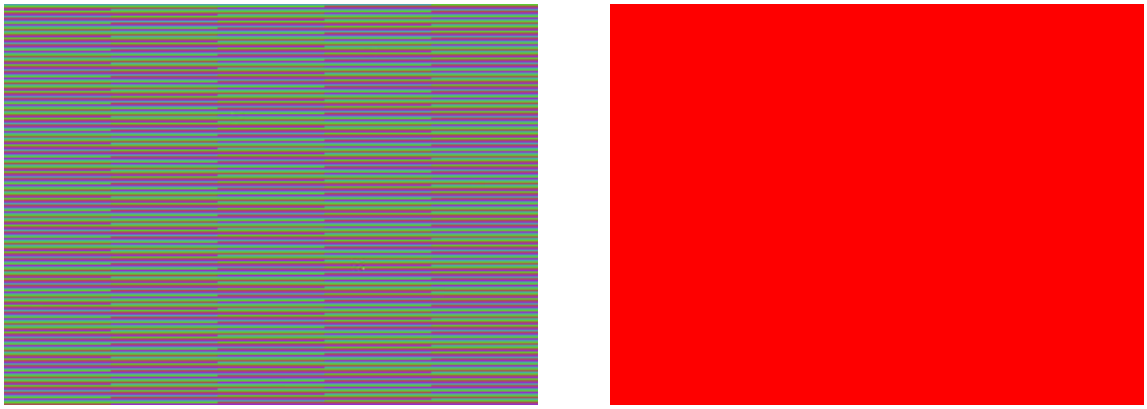
```
void UG_FillScreen( UG_COLOR c );
```

Parameters:

UG_COLOR c Color

Example:

```
int main( void )
{
  //...
  UG_FillScreen( C_RED );
  //...
}
```



(a) Before (b) After

Figure 19: UG_FillScreen() example

7.1.5 UG_FillFrame()

Fills a rectangular area with a selected color.

Prototype:

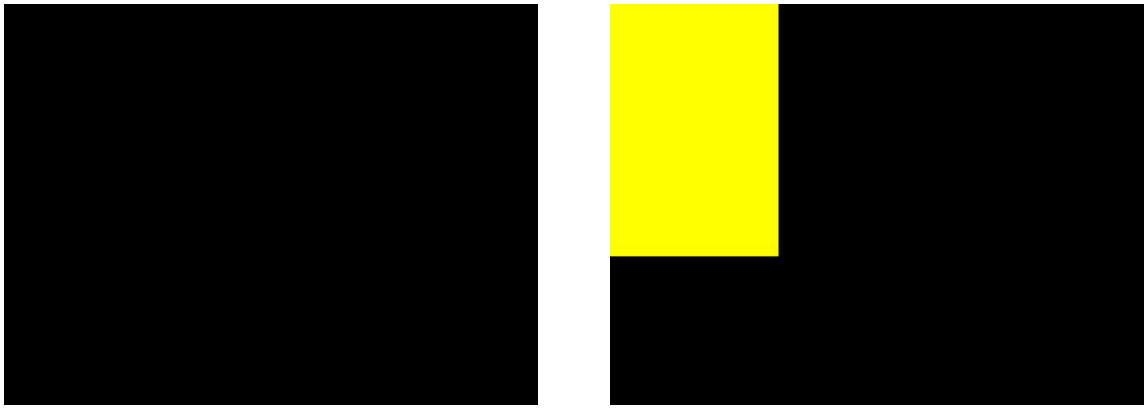
```
void UG_FillFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the frame
UG_S16 y1	Y start position of the frame
UG_S16 x2	X end position of the frame
UG_S16 y2	Y end position of the frame
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_FillFrame(0, 0,100, 150, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 20: UG_FillFrame() example

7.1.6 UG_FillRoundFrame()

Fills a rectangular area with a selected color. The rectangular area has rounded corners.

Prototype:

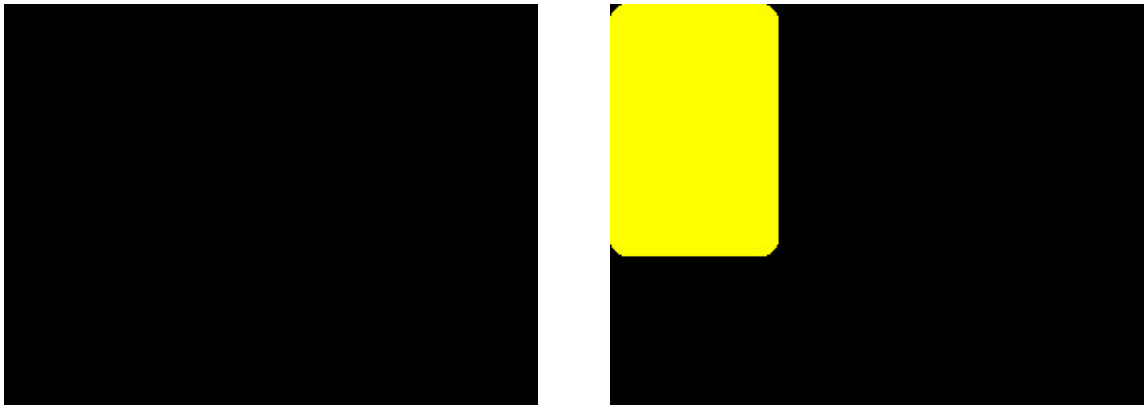
```
void UG_FillRoundFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_S16 r,
   UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the frame
UG_S16 y1	Y start position of the frame
UG_S16 x2	X end position of the frame
UG_S16 y2	Y end position of the frame
UG_S16 r	Corner radius
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_FillRoundFrame(0, 0, 100, 150, 10, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 21: UG_FillRoundFrame() example

7.1.7 UG_DrawMesh()

Draws a rectangular mesh with a selected color.

Prototype:

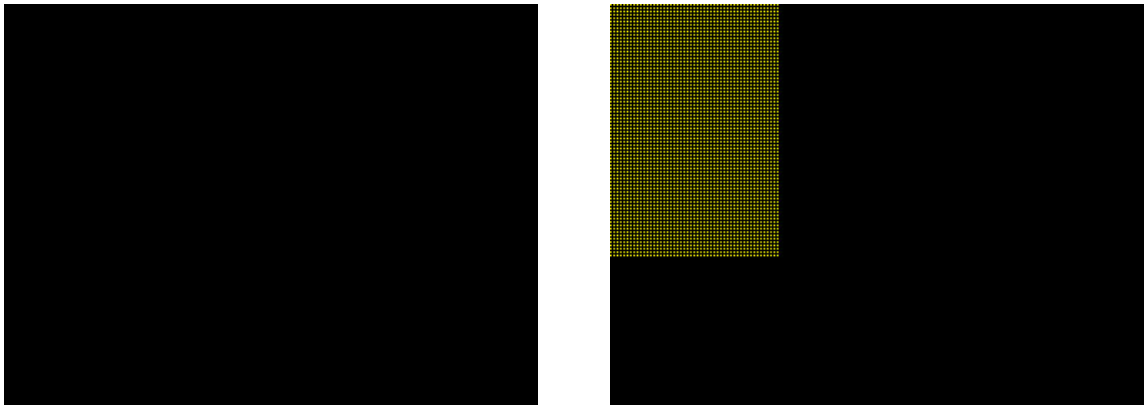
```
void UG_DrawMesh( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the mesh
UG_S16 y1	Y start position of the mesh
UG_S16 x2	X end position of the mesh
UG_S16 y2	Y end position of the mesh
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_DrawMesh(0, 0,100, 150, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 22: UG_DrawMesh() example

7.1.8 UG_DrawFrame()

Draws a frame with a selected color.
Prototype:

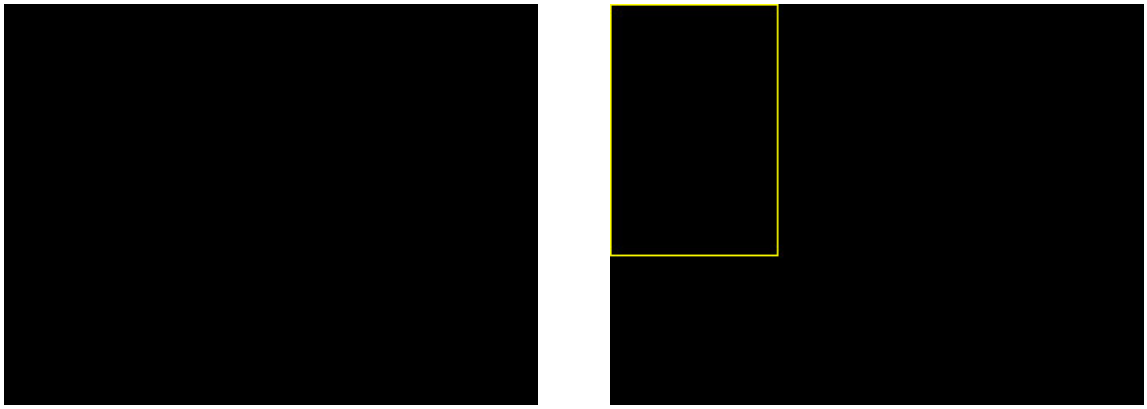
```
void UG_DrawFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the frame
UG_S16 y1	Y start position of the frame
UG_S16 x2	X end position of the frame
UG_S16 y2	Y end position of the frame
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_DrawFrame(0, 0,100, 150, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 23: UG_DrawFrame() example

7.1.9 UG_DrawRoundFrame()

Draws a frame with a selected color. The frame has rounded corners.
Prototype:

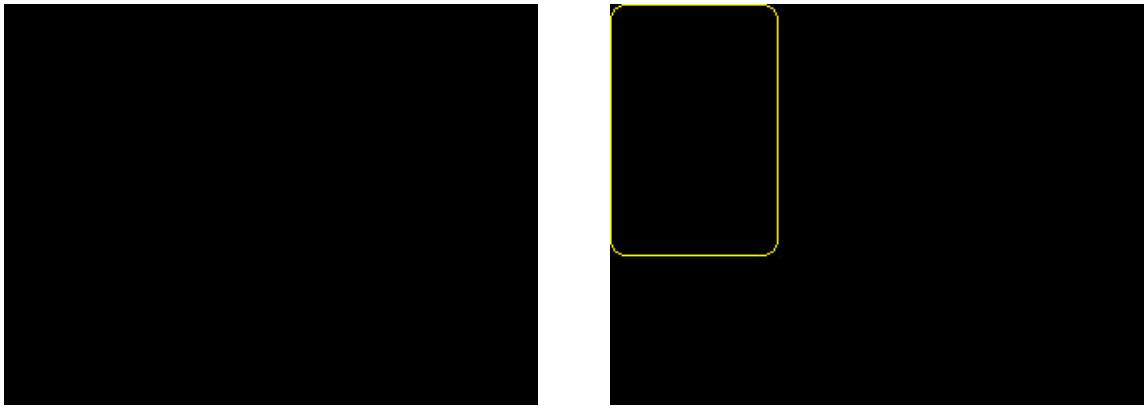
```
void UG_DrawRoundFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_S16 r,
   UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the frame
UG_S16 y1	Y start position of the frame
UG_S16 x2	X end position of the frame
UG_S16 y2	Y end position of the frame
UG_S16 r	Corner radius
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_DrawRoundFrame(0, 0, 100, 150, 10, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 24: UG_DrawRoundFrame() example

7.1.10 UG_DrawPixel()

Draws a pixel with a selected color.

Prototype:

```
void UG_DrawPixel( UG_S16 x0, UG_S16 y0, UG_COLOR c );
```

Parameters:

UG_S16 x0	X position of the pixel
UG_S16 y0	Y position of the pixel
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_DrawPixel(20, 70, C_GREEN);
   //...
}
```

(a) Before (b) After

Figure 25: UG_DrawPixel() example

7.1.11 UG_DrawCircle()

Draws a circle with a selected color and radius.

Prototype:

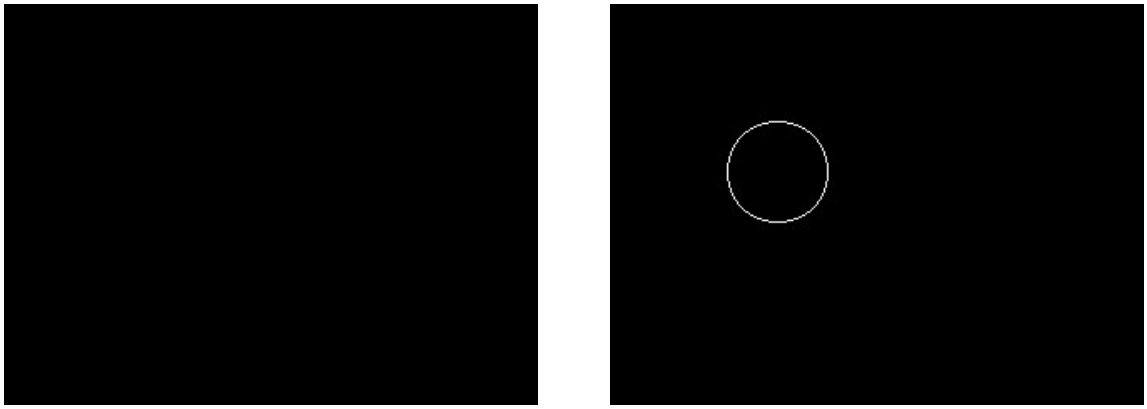
```
void UG_DrawCircle( UG_S16 x0, UG_S16 y0, UG_S16 r, UG_COLOR c );
```

Parameters:

UG_S16 x0	X center position of the circle
UG_S16 y0	Y center position of the circle
UG_S16 r	Radius of the circle
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_DrawCircle(100, 100, 30, C_WHITE);
   //...
}
```



(a) Before (b) After

Figure 26: UG_DrawCircle() example

7.1.12 UG_FillCircle()

Fills a circle with a selected color.

Prototype:

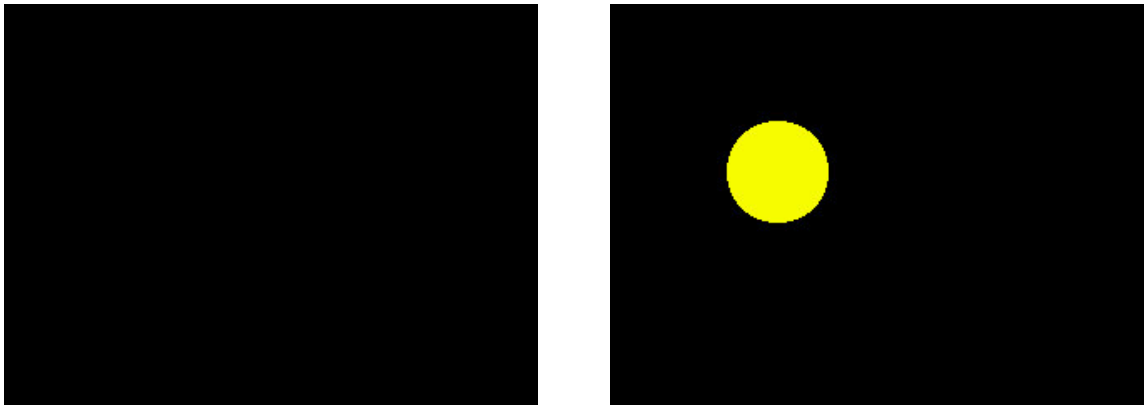
```
void UG_FillCircle( UG_S16 x0, UG_S16 y0, UG_S16 r, UG_COLOR c );
```

Parameters:

UG_S16 x0	X center position of the circle
UG_S16 y0	Y center position of the circle
UG_S16 r	Radius of the circle
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_FillCircle(100, 100, 30, C_YELLOW);
   //...
}
```



(a) Before (b) After

Figure 27: UG_FillCircle() example

7.1.13 UG_DrawArc()

Draws an arc with a selected color.

Prototype:

```
void UG_DrawArc( UG_S16 x0, UG_S16 y0, UG_S16 r, UG_U8 s, UG_COLOR c );
```

Parameters:

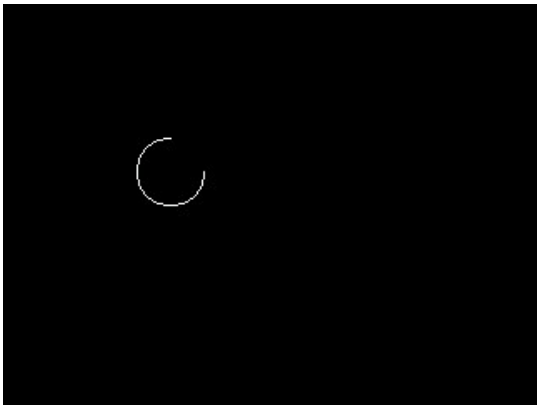
UG_S16 x0	X center position of the arc
UG_S16 y0	Y center position of the arc
UG_S16 r	Radius of the arc
UG_U8 s	Selected sectors
UG_COLOR c	Color

Example:

```
int main( void )
{
   //...
   UG_U16 sec;
   UG_U8 j,tog;
   while(1) /* Guess what it does :) */
   {
      for( sec = 1; sec != 0x100; sec<<=1 )
      {
         j++;
         if ( j >=9 )
         {
            j = 0;
            tog = !tog;
         }
```

```
            if ( tog )
            {
               UG_DrawArc( 100, 200, 20, sec, C_BLACK );
            }
            else
            {
               UG_DrawArc( 100, 200, 20, sec, C_WHITE );
            }

            /* Some delay */
            delay_ms(60);
         }
      }
      //...
}
```

(a) Before



(b) After

Figure 28: UG_DrawArc() example

7.1.14 UG_DrawLine()

Draws a line between two points.
Prototype:

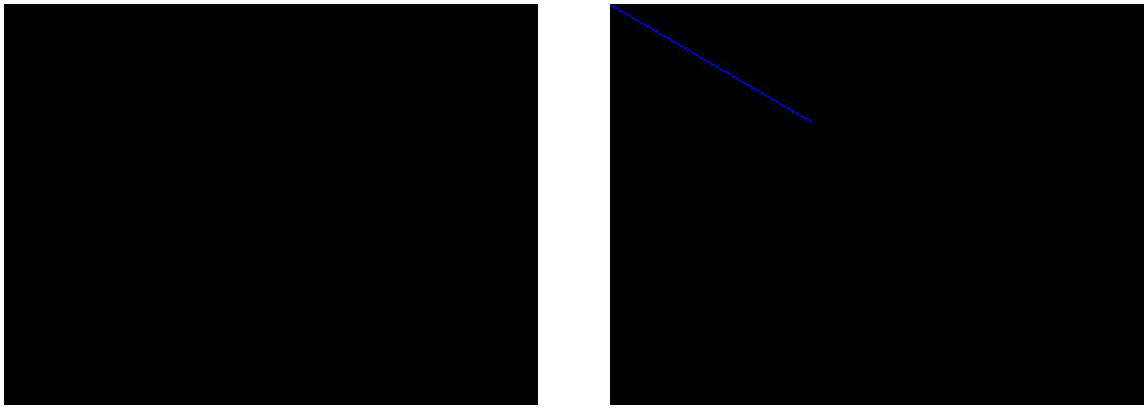
```
void UG_DrawLine( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c );
```

Parameters:

UG_S16 x1	X start position of the line
UG_S16 y1	Y start position of the line
UG_S16 x2	X end position of the line
UG_S16 y2	Y end position of the line
UG_COLOR c	Color

Example:

```
int main( void )
{
    //...
    UG_DrawLine(0, 0, 120, 70, C_BLUE);
    //...
}
```



(a) Before (b) After

Figure 29: UG_DrawLine() example

7.1.15 UG_PutString()

Draws a string.

Prototype:

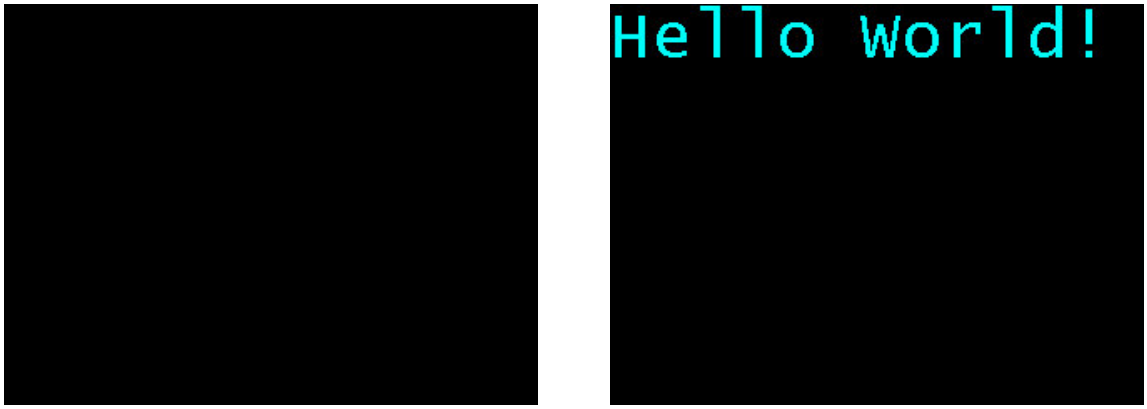
```
void UG_PutString( UG_S16 x, UG_S16 y, char* str );
```

Parameters:

UG_S16 x	X start position of the string
UG_S16 y	Y start position of the string
char* str	Pointer to the string

Example:

```
int main( void )
{
    //...
    UG_FontSelect( &FONT_24X40 );
    UG_SetBackcolor( C_BLACK );
    UG_SetForecolor( C_CYAN );
    UG_PutString ( 0, 0, "Hello World!" );
    //...
}
```



(a) Before (b) After

Figure 30: UG_PutString() example

7.1.16 UG_PutChar()

Draws a single char.

Prototype:

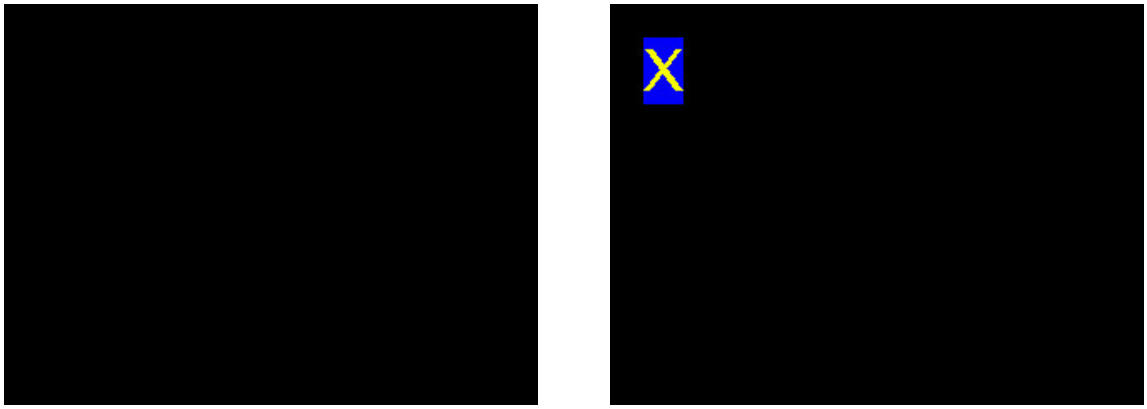
```
void UG_PutChar( char chr, UG_S16 x, UG_S16 y, UG_COLOR fc, UG_COLOR bc );
```

Parameters:

char chr	Char
UG_S16 x	X start position of the char
UG_S16 y	Y start position of the char
UG_COLOR fc	Fore color of the char
UG_COLOR bc	Backcolor of the char

Example:

```
int main( void )
{
   //...
   UG_FontSelect( &FONT_24X40 );
   UG_PutChar( 'X', 20, 20, C_YELLOW, C_BLUE );
   //...
}
```



(a) Before (b) After

Figure 31: UG_PutChar() example

7.1.17 UG_ConsolePutString()

Adds a string to the console.

Prototype:

```
void UG_ConsolePutString( char* str );
```

Parameters:

char* str Pointer to the string

Example:

```
int main( void )
{
  //...
  UG_FontSelect( &FONT_12X16 );
  UG_ConsoleSetBackcolor( C_BLACK );
  UG_ConsoleSetForecolor( C_WHITE );
  UG_ConsolePutString( "System initialized!\n" );
  //...
  UG_ConsolePutString( "SD-Card mounted!\n" );
  //...
  UG_ConsolePutString( "BLDC: " );
  //...
  UG_ConsoleSetForecolor( C_GREEN );
  UG_ConsolePutString( "Trapezoidal mode\n" );
  //...
  UG_ConsoleSetForecolor( C_WHITE );
  UG_ConsolePutString( "CPU core voltage:" );
  UG_ConsoleSetForecolor( C_RED );
```

```
    UG_ConsolePutString( " >1.9V!\n" );
    //...
}
```

(a) Before (b) After

Figure 32: UG_ConsolePutString() example

7.1.18 UG_ConsoleSetArea()

Defines the active console area.

Prototype:

```
void UG_ConsoleSetArea( UG_S16 xs, UG_S16 ys, UG_S16 xe, UG_S16 ye );
```

Parameters:

UG_S16 xs	X start position of the console
UG_S16 ys	Y start position of the console
UG_S16 xe	X end position of the console
UG_S16 ye	Y end position of the console

Example:

```
int main( void )
{
    //...
    UG_ConsoleSetArea(0, 0, 200, 200 );
    //...
}
```

7.1.19 UG_ConsoleSetForecolor()

Defines the fore color of the console.

Prototype:

```
void UG_ConsoleSetForecolor( UG_COLOR c );
```

Parameters:

UG_COLOR c Fore color

Example:

```
int main( void )
{
   //...
   UG_ConsoleSetForecolor( C_YELLOW );
   //...
}
```

7.1.20 UG_ConsoleSetBackcolor()

Defines the back color of the console.

Prototype:

```
void UG_ConsoleSetBackcolor( UG_COLOR c );
```

Parameters:

UG_COLOR c Backcolor

Example:

```
int main( void )
{
   //...
   UG_ConsoleSetBackcolor( C_BLUE );
   //...
}
```

7.1.21 UG_SetForecolor()

Defines the fore color of the string.

Prototype:

```
void UG_SetForecolor( UG_COLOR c );
```

Parameters:

UG_COLOR c Fore color

Example:

```
int main( void )
{
   //...
   UG_SetForecolor( C_YELLOW );
   //...
}
```

7.1.22 UG_SetBackcolor()

Defines the back color of the string.

Prototype:

```
void UG_SetBackcolor( UG_COLOR c );
```

Parameters:

UG_COLOR c Backcolor

Example:

```
int main( void )
{
   //...
   UG_SetBackcolor( C_BLUE );
   //...
}
```

7.1.23 UG_GetXDim()

Returns the X-Dimension of the display.

Prototype:

```
UG_S16 UG_GetXDim( void );
```

Returns:

UG_S16 X-Dimension

Example:

```
int main( void )
{
   //...
   val = UG_GetXDim( );
   //...
}
```

7.1.24 UG_GetYDim()

Returns the Y-Dimension of the display.

Prototype:

```
UG_S16 UG_GetYDim( void );
```

Returns:

UG_S16 Y-Dimension

Example:

```
int main( void )
{
   //...
   val = UG_GetYDim( );
   //...
}
```

7.1.25 UG_FontSetHSpace()

Defines the horizontal space between each char.

Prototype:

```
void UG_FontSetHSpace( UG_U16 s );
```

Parameters:

UG_U16 s Horizontal space

Example:

```
int main( void )
{
   //...
   UG_FontSetHSpace( 4 );
   //...
}
```

7.1.26 UG_FontSetVSpace()

Defines the vertical space between each char.

Prototype:

```
void UG_FontSetVSpace( UG_U16 s );
```

Parameters:

UG_U16 s Vertical space

Example:

```
int main( void )
{
   //...
   UG_FontSetVSpace( 4 );
   //...
}
```

7.2 Driver Functions

7.2.1 UG_DriverRegister()

Registers a driver.

Prototype:

```
void UG_DriverRegister( UG_U8 type, void* driver );
```

Parameters:

UG_U8 type	Driver type
void* driver	Pointer to the driver function

Returns:

void none

Example:

```
/* Hardware accelerator for UG_DrawLine (Platform: STM32F4x9) */
UG_RESULT _HW_DrawLine( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c )
{
  DMA2D_InitTypeDef DMA2D_InitStruct;

  RCC_AHB1PeriphResetCmd(RCC_AHB1Periph_DMA2D, ENABLE);
  RCC_AHB1PeriphResetCmd(RCC_AHB1Periph_DMA2D, DISABLE);
  DMA2D_InitStruct.DMA2D_Mode = DMA2D_R2M;
  DMA2D_InitStruct.DMA2D_CMode = DMA2D_RGB565;
  /* Convert UG_COLOR to RGB565 */
  DMA2D_InitStruct.DMA2D_OutputBlue = (c>>3) & 0x1F;
  DMA2D_InitStruct.DMA2D_OutputGreen = (c>>10) & 0x3F;
  DMA2D_InitStruct.DMA2D_OutputRed = (c>>19) & 0x1F;
  DMA2D_InitStruct.DMA2D_OutputAlpha = 0x0F;

  /* horizontal line */
  if ( y1 == y2 )
  {
    DMA2D_InitStruct.DMA2D_OutputOffset = 0;
    DMA2D_InitStruct.DMA2D_NumberOfLine = 1;
    DMA2D_InitStruct.DMA2D_PixelPerLine = x2-x1+1;
  }
  /* vertical line */
  else if ( x1 == x2 )
  {
    DMA2D_InitStruct.DMA2D_OutputOffset = LCD_PIXEL_WIDTH - 1;
    DMA2D_InitStruct.DMA2D_NumberOfLine = y2-y1+1;
    DMA2D_InitStruct.DMA2D_PixelPerLine = 1;
  }
  else
  {
    return UG_RESULT_FAIL;
```

```
    }

    if ( ltdc_work_layer == LAYER_1 )
    {
        DMA2D_InitStruct.DMA2D_OutputMemoryAdd = SDRAM_BANK_ADDR + LAYER_1_OFFSET +
            2*(LCD_PIXEL_WIDTH * y1 + x1);
    }
    else
    {
        DMA2D_InitStruct.DMA2D_OutputMemoryAdd = SDRAM_BANK_ADDR + LAYER_2_OFFSET +
            2*(LCD_PIXEL_WIDTH * y1 + x1);
    }
    DMA2D_Init(&DMA2D_InitStruct);
    DMA2D_StartTransfer();
    while(DMA2D_GetFlagStatus(DMA2D_FLAG_TC) == RESET){};
    return UG_RESULT_OK;
}

/* Hardware accelerator for UG_FillFrame (Platform: STM32F4x9) */
UG_RESULT _HW_FillFrame( UG_S16 x1, UG_S16 y1, UG_S16 x2, UG_S16 y2, UG_COLOR c )
{
    DMA2D_InitTypeDef       DMA2D_InitStruct;

    DMA2D_DeInit();
    DMA2D_InitStruct.DMA2D_Mode = DMA2D_R2M;
    DMA2D_InitStruct.DMA2D_CMode = DMA2D_RGB565;
    /* Convert UG_COLOR to RGB565 */
    DMA2D_InitStruct.DMA2D_OutputBlue = (c>>3) & 0x1F;
    DMA2D_InitStruct.DMA2D_OutputGreen = (c>>10) & 0x3F;
    DMA2D_InitStruct.DMA2D_OutputRed = (c>>19) & 0x1F;
    DMA2D_InitStruct.DMA2D_OutputAlpha = 0x0F;
    DMA2D_InitStruct.DMA2D_OutputOffset = (LCD_PIXEL_WIDTH - (x2-x1+1));
    DMA2D_InitStruct.DMA2D_NumberOfLine = y2-y1+1;
    DMA2D_InitStruct.DMA2D_PixelPerLine = x2-x1+1;
    if ( ltdc_work_layer == LAYER_1 )
    {
        DMA2D_InitStruct.DMA2D_OutputMemoryAdd = SDRAM_BANK_ADDR + LAYER_1_OFFSET +
            2*(LCD_PIXEL_WIDTH * y1 + x1);
    }
    else
    {
        DMA2D_InitStruct.DMA2D_OutputMemoryAdd = SDRAM_BANK_ADDR + LAYER_2_OFFSET +
            2*(LCD_PIXEL_WIDTH * y1 + x1);
    }
    DMA2D_Init(&DMA2D_InitStruct);

    DMA2D_StartTransfer();
    while(DMA2D_GetFlagStatus(DMA2D_FLAG_TC) == RESET){}
    return UG_RESULT_OK;
}


int main( void )
{
    //...
     /* Unleash the performance of the STM32 */
```

```
    UG_DriverRegister( DRIVER_DRAW_LINE, (void*)_HW_DrawLine );
    UG_DriverRegister( DRIVER_FILL_FRAME, (void*)_HW_FillFrame );
    //...
}
```

7.2.2 UG_DriverEnable()

Enables a driver.

Prototype:

```
void UG_DriverEnable( UG_U8 type );
```

Parameters:

UG_U8 type	Driver type

Returns:

void	none

Example:

```
int main( void )
{
    //...
    UG_DriverEnable( DRIVER_DRAW_LINE );
    //...
}
```

7.2.3 UG_DriverDisable()

Disables a driver.

Prototype:

```
void UG_DriverDisable( UG_U8 type );
```

Parameters:

UG_U8 type	Driver type

Returns:

void	none

Example:

```
int main( void )
{
   //...
   UG_DriverDisable( DRIVER_DRAW_LINE );
   //...
}
```

7.3 Window Functions

7.3.1 UG_WindowCreate()

Creates a window.

Prototype:

```
UG_RESULT UG_WindowCreate( UG_WINDOW* wnd, UG_OBJECT* objlst, UG_U8 objcnt,
void (*cb)( UG_MESSAGE* ) );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_OBJECT* objlst	Pointer to the object list
UG_U8 objcnt	Maximal amount of objects (=elements of objlst)
void (*cb)(UG_MESSAGE*)	Pointer to the callback function of the window

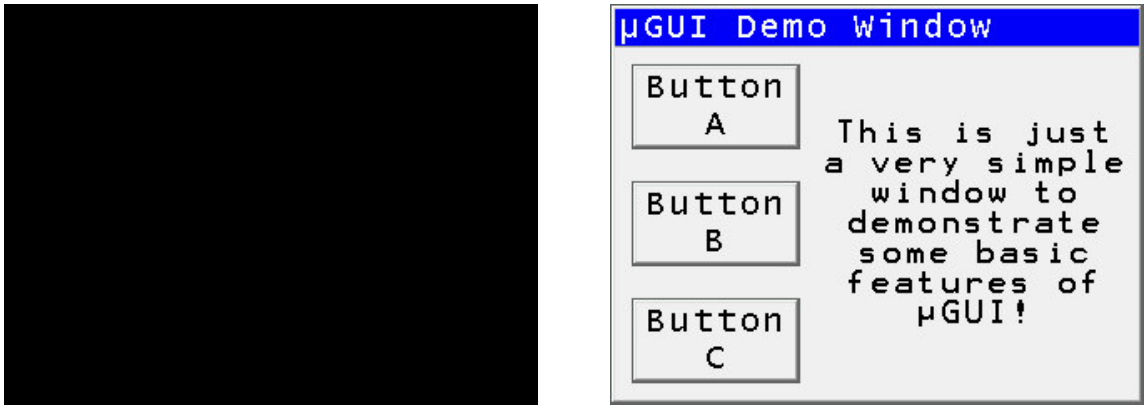
Returns:

UG_RESULT	Result of the function

Example:

```
void window_1_callback( UG_MESSAGE* msg )
{
   if ( msg->type == MSG_TYPE_OBJECT )
   {
      if ( msg->id == OBJ_TYPE_BUTTON )
      {
         switch( msg->sub_id )
         {
            case BTN_ID_0:
            {
               //...
               break;
            }
            case BTN_ID_1:
            {
               //...
               break;
            }
            case BTN_ID_2:
```

```
            {
               //...
               break;
            }
         }
      }
   }
}

#define MAX_OBJECTS  10
int main( void )
{
   UG_WINDOW window_1;
   UG_BUTTON button_1;
   UG_BUTTON button_2;
   UG_BUTTON button_3;
   UG_TEXTBOX textbox_1;
   UG_OBJECT obj_buff_wnd_1[MAX_OBJECTS];

   //...

   /* Create the window */
   UG_WindowCreate( &window_1, obj_buff_wnd_1, MAX_OBJECTS, window_1_callback );

   /* Modify the window title */
   UG_WindowSetTitleText( &window_1, "uGUI Demo Window" );
   UG_WindowSetTitleTextFont( &window_1, &FONT_12X20 );

   /* Create some buttons */
   UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 10, 10, 110, 60 );
   UG_ButtonCreate( &window_1, &button_2, BTN_ID_1, 10, 80, 110, 130 );
   UG_ButtonCreate( &window_1, &button_3, BTN_ID_2, 10, 150, 110, 200 );

   /* Label the buttons */
   UG_ButtonSetFont( &window_1, BTN_ID_0, &FONT_12X20 );
   UG_ButtonSetText( &window_1, BTN_ID_0, "Button\nA" );
   UG_ButtonSetFont( &window_1, BTN_ID_1, &FONT_12X20 );
   UG_ButtonSetText( &window_1, BTN_ID_1, "Button\nB" );
   UG_ButtonSetFont( &window_1, BTN_ID_2, &FONT_12X20 );
   UG_ButtonSetText( &window_1, BTN_ID_2, "Button\nC" );

   /* Create a Textbox */
   UG_TextboxCreate( &window_1, &textbox_1, TXB_ID_0, 120, 10, 310, 200 );
   UG_TextboxSetFont( &window_1, TXB_ID_0, &FONT_12X16 );
   UG_TextboxSetText( &window_1, TXB_ID_0,
   "This is just\na very simple\nwindow to\ndemonstrate\nsome basic\nfeatures of
      uGUI!" );
   UG_TextboxSetForeColor( &window_1, TXB_ID_0, C_BLACK );
   UG_TextboxSetAlignment( &window_1, TXB_ID_0, ALIGN_CENTER );

   /* Finally, show the window */
   UG_WindowShow( &window_1 );
   //...
}
```



(a) Before (b) After

Figure 33: UG_WindowCreate() example

7.3.2 UG_WindowDelete()

Deletes a window.

Prototype:

```
UG_RESULT UG_WindowDelete( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowDelete( &window_1 );
   //...
}
```

7.3.3 UG_WindowShow()

Shows a window.

Prototype:

```
UG_RESULT UG_WindowShow( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    UG_WindowShow( &window_1 );
    //...
}
```

7.3.4 UG_WindowHide()

Hides a window.

Prototype:

```
UG_RESULT UG_WindowHide( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    UG_WindowHide( &window_1 );
    //...
}
```

7.3.5 UG_WindowResize()

Changes the size of a window.

Note: All objects which don't fit into the window after resizing it will be hidden.

Prototype:

```
UG_RESULT UG_WindowResize( UG_WINDOW* wnd, UG_S16 xs, UG_S16 ys, UG_S16 xe, UG_S16
    ye );
```

Parameters:

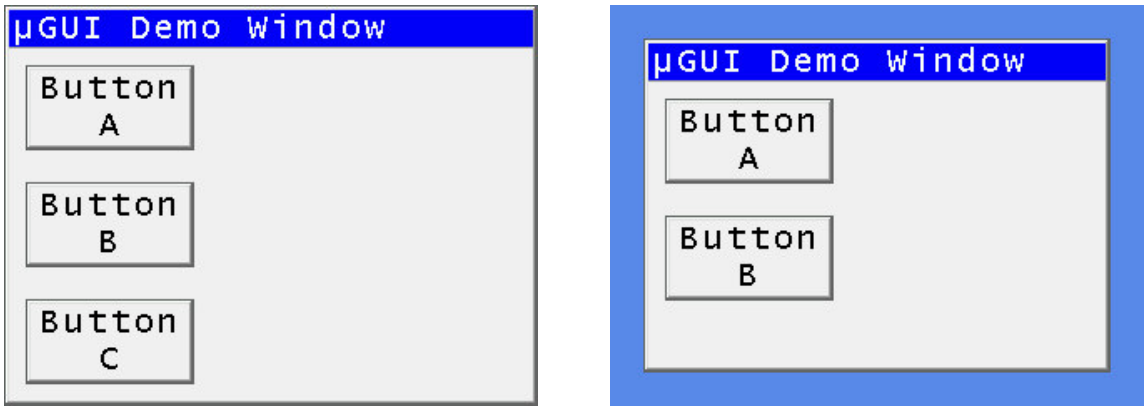
UG_WINDOW* wnd	Pointer to the window
UG_S16 xs	X start position of the window
UG_S16 ys	Y start position of the window
UG_S16 xe	X end position of the window
UG_S16 ye	Y end position of the window

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
   UG_WindowResize( &window_1, 20, 20, 319-20, 239-20 );
   //...
}
```



(a) Before (b) After

Figure 34: UG_WindowResize() example

7.3.6 UG_WindowAlert()

Swaps fore- and back color of the window title.

Prototype:

```
UG_RESULT UG_WindowAlert( UG_WINDOW* wnd );
```

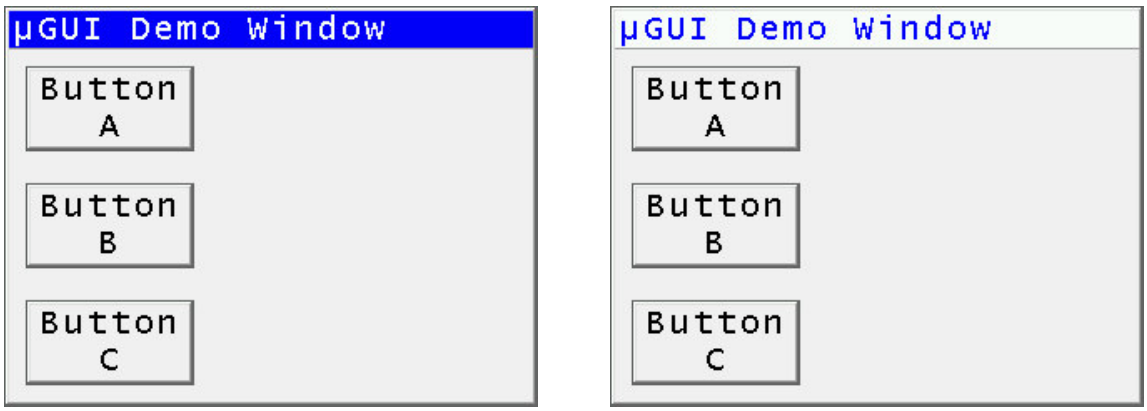
Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowAlert( &window_1 );
   //...
}
```



(a) Before (b) After

Figure 35: UG_WindowAlert() example

7.3.7 UG_WindowSetForeColor()

Changes the fore color of the window. The fore color of a window is the default fore color of all objects.

Prototype:

```
UG_RESULT UG_WindowSetForeColor( UG_WINDOW* wnd, UG_COLOR fc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_COLOR fc	New fore color

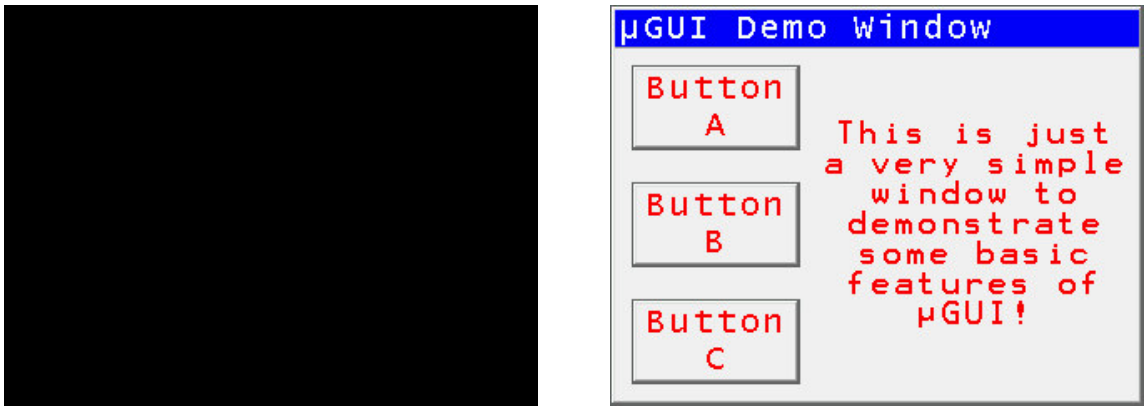
Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
  UG_WINDOW window_1;
  UG_BUTTON button_1;
  UG_BUTTON button_2;
  UG_BUTTON button_3;
  UG_TEXTBOX textbox_1;
  UG_OBJECT obj_buff_wnd_1[MAX_OBJECTS];

  //...

  /* Create the window */
  UG_WindowCreate( &window_1, obj_buff_wnd_1, MAX_OBJECTS, window_1_callback );

  /* Modify the window title */
  UG_WindowSetTitleText( &window_1, "uGUI Demo Window" );
  UG_WindowSetTitleTextFont( &window_1, &FONT_12X20 );

  /* Change the window fore color (before creating the objects) */
  UG_WindowSetForeColor( &window_1, C_RED );

  /* Create some buttons */
  UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 10, 10, 110, 60 );
  UG_ButtonCreate( &window_1, &button_2, BTN_ID_1, 10, 80, 110, 130 );
  UG_ButtonCreate( &window_1, &button_3, BTN_ID_2, 10, 150, 110, 200 );

  /* Label the buttons */
  UG_ButtonSetFont( &window_1, BTN_ID_0, &FONT_12X20 );
  UG_ButtonSetText( &window_1, BTN_ID_0, "Button\nA" );
  UG_ButtonSetFont( &window_1, BTN_ID_1, &FONT_12X20 );
  UG_ButtonSetText( &window_1, BTN_ID_1, "Button\nB" );
  UG_ButtonSetFont( &window_1, BTN_ID_2, &FONT_12X20 );
  UG_ButtonSetText( &window_1, BTN_ID_2, "Button\nC" );

  /* Create a Textbox */
  UG_TextboxCreate( &window_1, &textbox_1, TXB_ID_0, 120, 10, 310, 200 );
  UG_TextboxSetFont( &window_1, TXB_ID_0, &FONT_12X16 );
  UG_TextboxSetText( &window_1, TXB_ID_0,
  "This is just\na very simple\nwindow to\ndemonstrate\nsome basic\nfeatures of
     uGUI!" );
  UG_TextboxSetForeColor( &window_1, TXB_ID_0, C_BLACK );
```

```
    UG_TextboxSetAlignment( &window_1, TXB_ID_0, ALIGN_CENTER );

    /* Finally, show the window */
    UG_WindowShow( &window_1 );
    //...
}
```



(a) Before (b) After

Figure 36: UG_WindowSetForeColor() example

7.3.8 UG_WindowSetBackColor()

Changes the back color of the window.

Prototype:

```
UG_RESULT UG_WindowSetBackColor( UG_WINDOW* wnd, UG_COLOR bc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_COLOR bc	New back color

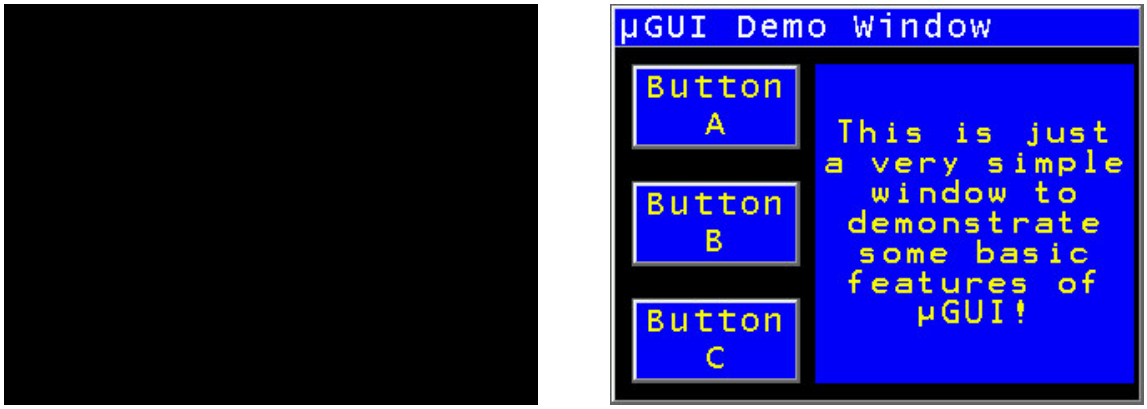
Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_BUTTON button_1;
   UG_BUTTON button_2;
   UG_BUTTON button_3;
   UG_TEXTBOX textbox_1;
```

```
    UG_OBJECT obj_buff_wnd_1[MAX_OBJECTS];

    //...

    /* Create the window */
    UG_WindowCreate( &window_1, obj_buff_wnd_1, MAX_OBJECTS, window_1_callback );

    /* Modify the window title */
    UG_WindowSetTitleText( &window_1, "uGUI Demo Window" );
    UG_WindowSetTitleTextFont( &window_1, &FONT_12X20 );

    /* Change the window fore- and back color (before creating the objects) */
    UG_WindowSetForeColor( &window_1, C_YELLOW );
    UG_WindowSetBackColor( &window_1, C_BLUE );

    /* Create some buttons */
    UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 10, 10, 110, 60 );
    UG_ButtonCreate( &window_1, &button_2, BTN_ID_1, 10, 80, 110, 130 );
    UG_ButtonCreate( &window_1, &button_3, BTN_ID_2, 10, 150, 110, 200 );

    /* Label the buttons */
    UG_ButtonSetFont( &window_1, BTN_ID_0, &FONT_12X20 );
    UG_ButtonSetText( &window_1, BTN_ID_0, "Button\nA" );
    UG_ButtonSetFont( &window_1, BTN_ID_1, &FONT_12X20 );
    UG_ButtonSetText( &window_1, BTN_ID_1, "Button\nB" );
    UG_ButtonSetFont( &window_1, BTN_ID_2, &FONT_12X20 );
    UG_ButtonSetText( &window_1, BTN_ID_2, "Button\nC" );

    /* Create a Textbox */
    UG_TextboxCreate( &window_1, &textbox_1, TXB_ID_0, 120, 10, 310, 200 );
    UG_TextboxSetFont( &window_1, TXB_ID_0, &FONT_12X16 );
    UG_TextboxSetText( &window_1, TXB_ID_0,"This is just\na very simple\nwindow to\
       ndemonstrate\nsome basic\nfeatures of\nuGUI!" );
    UG_TextboxSetAlignment( &window_1, TXB_ID_0, ALIGN_CENTER );

    /* Finally, show the window */
    UG_WindowShow( &window_1 );

    /* Change the window back color (after creating the objects) */
    UG_WindowSetBackColor( &window_1, C_BLACK );

    //...
}
```



(a) Before (b) After

Figure 37: UG_WindowSetBackColor() example

7.3.9 UG_WindowSetTitleTextColor()

Changes the text color of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleTextColor( UG_WINDOW* wnd, UG_COLOR c );
```

Parameters:

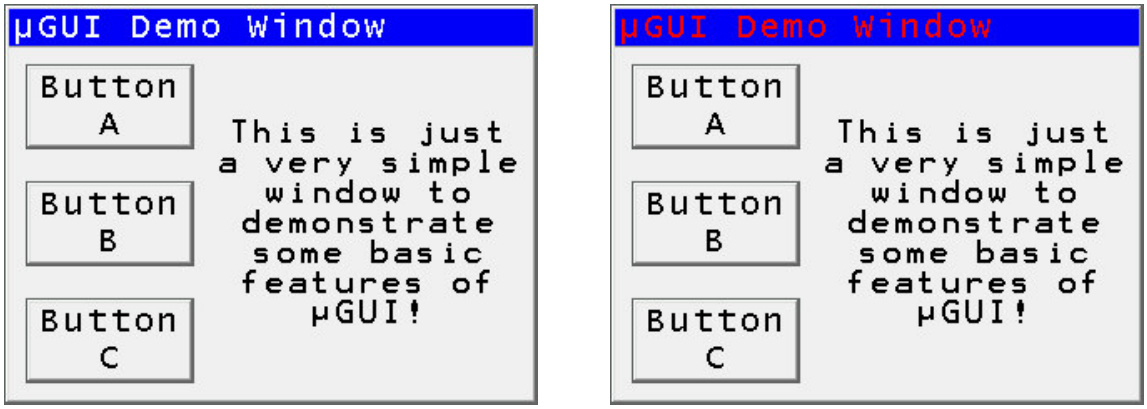
UG_WINDOW* wnd	Pointer to the window
UG_COLOR c	New text color of the window title

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleTextColor( &window_1, C_RED );
   //...
}
```



(a) Before (b) After

Figure 38: UG_WindowSetTitleTextColor() example

7.3.10 UG_WindowSetTitleColor()

Changes the color of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleColor( UG_WINDOW* wnd, UG_COLOR c );
```

Parameters:

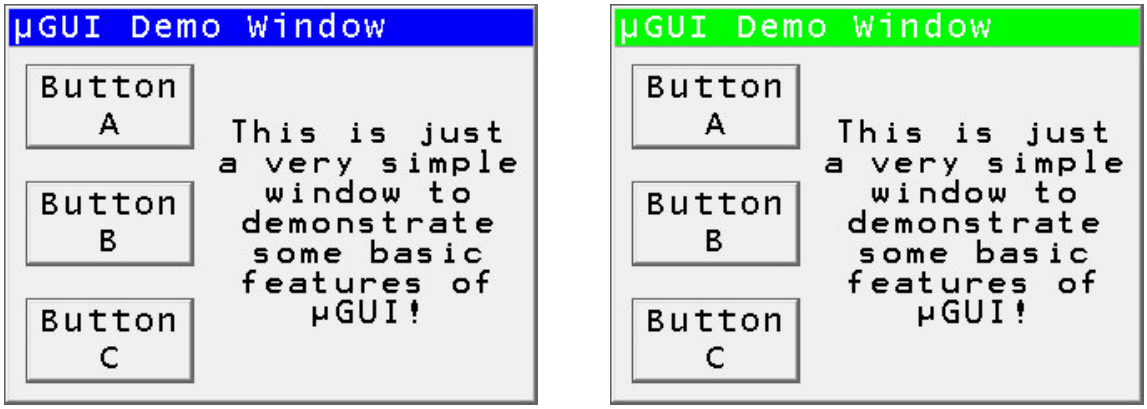
UG_WINDOW* wnd	Pointer to the window
UG_COLOR c	New color of the window title

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleColor( &window_1, C_GREEN );
   //...
}
```



(a) Before (b) After

Figure 39: UG_WindowSetTitleColor() example

7.3.11 UG_WindowSetTitleInactiveTextColor()

Changes the inactive text color of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleInactiveTextColor( UG_WINDOW* wnd, UG_COLOR c );
```

Parameters:

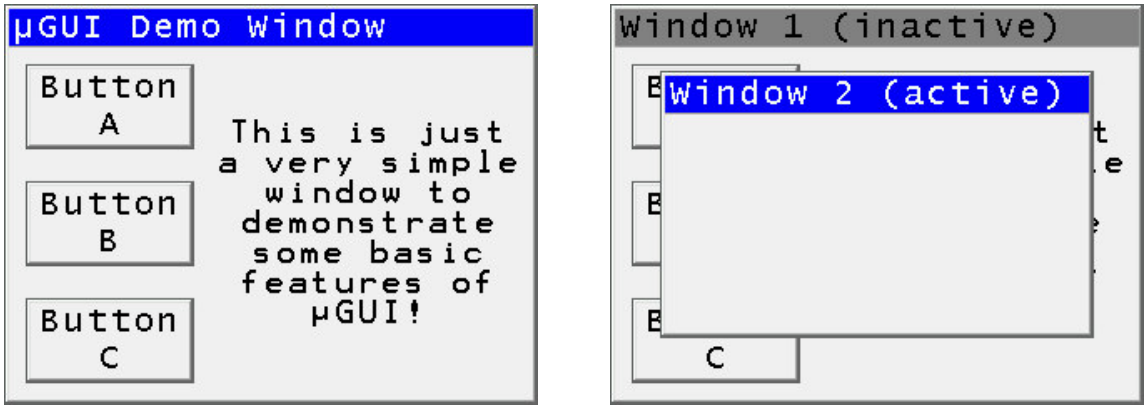
UG_WINDOW* wnd	Pointer to the window
UG_COLOR c	New inactive text color of the window title

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleInactiveTextColor( &window_1, C_BLACK );
   UG_WindowShow( &window_2 );
   //...
}
```



(a) Before (b) After

Figure 40: UG_WindowSetTitleInactiveTextColor() example

7.3.12 UG_WindowSetTitleInactiveColor()

Changes the inactive color of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleInactiveColor( UG_WINDOW* wnd, UG_COLOR c );
```

Parameters:

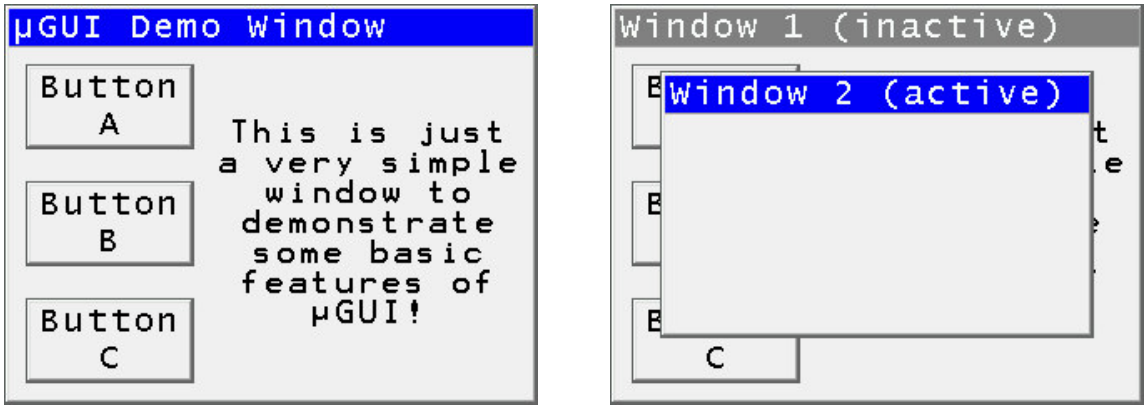
UG_WINDOW* wnd	Pointer to the window
UG_COLOR c	New inactive color of the window title

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleInactiveColor( &window_1, C_GRAY );
   UG_WindowShow( &window_2 );
   //...
}
```



(a) Before (b) After

Figure 41: UG_WindowSetTitleInactiveColor() example

7.3.13 UG_WindowSetTitleText()

Changes the text of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleText( UG_WINDOW* wnd, char* str );
```

Parameters:

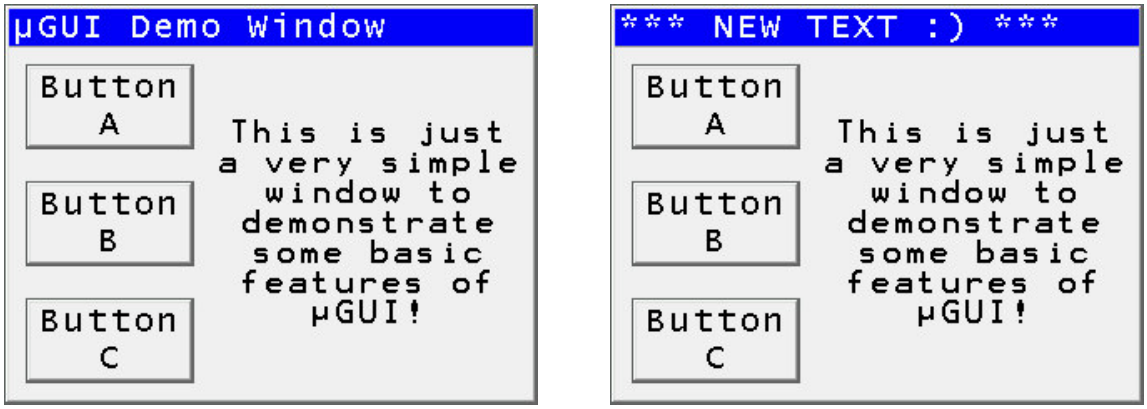
UG_WINDOW* wnd	Pointer to the window
char* str	Pointer to the title text

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    UG_WindowSetTitleText( &window_1, "*** NEW TEXT :) ***" );
    //...
}
```



(a) Before (b) After

Figure 42: UG_WindowSetTitleText() example

7.3.14 UG_WindowSetTitleTextFont()

Changes the text font of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleTextFont( UG_WINDOW* wnd, const UG_FONT* font );
```

Parameters:

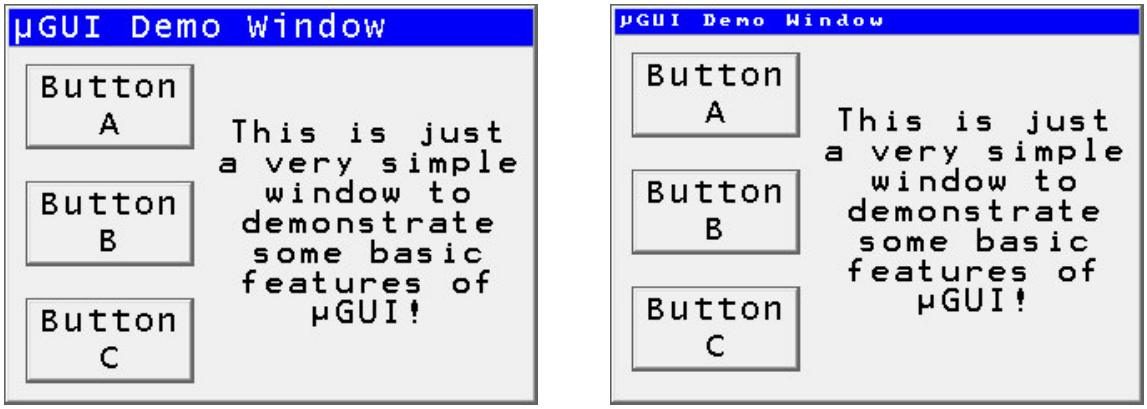
UG_WINDOW* wnd	Pointer to the window
const UG_FONT* font	Pointer to the title text font

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleTextFont( &window_1, &FONT_8X8 );
   //...
}
```



(a) Before (b) After

Figure 43: UG_WindowSetTitleTextFont() example

7.3.15 UG_WindowSetTitleTextHSpace()

Changes the horizontal space between the characters in the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleTextHSpace( UG_WINDOW* wnd, UG_S8 hs );
```

Parameters:

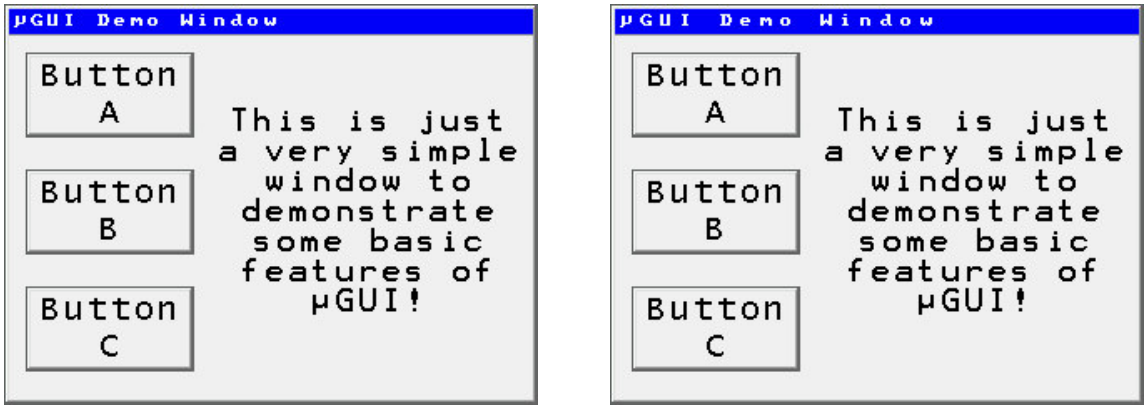
UG_WINDOW* wnd	Pointer to the window
UG_S8 hs	Horizontal space

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleTextHSpace( &window_1, 4 );
   //...
}
```



(a) Before (b) After

Figure 44: UG_WindowSetTitleTextHSpace() example

7.3.16 UG_WindowSetTitleTextVSpace()

Changes the vertical space between the characters in the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleTextVSpace( UG_WINDOW* wnd, UG_S8 vs );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_S8 vs	Vertical space

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleTextVSpace( &window_1, 3 );
   //...
}
```

7.3.17 UG_WindowSetTitleTextAlignment()

Changes the alignment of the text in the window title.

The following alignments are available:

ALIGN_BOTTOM_RIGHT
ALIGN_BOTTOM_CENTER
ALIGN_BOTTOM_LEFT
ALIGN_CENTER_RIGHT
ALIGN_CENTER
ALIGN_CENTER_LEFT
ALIGN_TOP_RIGHT
ALIGN_TOP_CENTER
ALIGN_TOP_LEFT

Prototype:

```
UG_RESULT UG_WindowSetTitleTextAlignment( UG_WINDOW* wnd, UG_U8 align );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_U8 align	Text alignment

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleTextAlignment( &window_1, ALIGN_CENTER );
   //...
}
```

7.3.18 UG_WindowSetTitleHeight()

Changes the height of the window title.

Prototype:

```
UG_RESULT UG_WindowSetTitleHeight( UG_WINDOW* wnd, UG_U8 height );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_U8 height	Title height

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetTitleHeight( &window_1, 25 );
   //...
}
```

7.3.19 UG_WindowSetXStart()

Changes the x start position of the window.

Prototype:

```
UG_RESULT UG_WindowSetXStart( UG_WINDOW* wnd, UG_S16 xs );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_S16 xs	X start position of the window

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetXStart( &window_1, 40 );
   //...
}
```

7.3.20 UG_WindowSetYStart()

Changes the y start position of the window.

Prototype:

```
UG_RESULT UG_WindowSetYStart( UG_WINDOW* wnd, UG_S16 ys );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_S16 ys	Y start position of the window

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetYStart( &window_1, 40 );
   //...
}
```

7.3.21 UG_WindowSetXEnd()

Changes the x end position of the window.

Prototype:

```
UG_RESULT UG_WindowSetXEnd( UG_WINDOW* wnd, UG_S16 xe );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_S16 xe	X end position of the window

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetXEnd( &window_1, 200 );
   //...
}
```

7.3.22 UG_WindowSetYEnd()

Changes the y end position of the window.

Prototype:

```
UG_RESULT UG_WindowSetYEnd( UG_WINDOW* wnd, UG_S16 ye );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_S16 ye	Y end position of the window

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetYEnd( &window_1, 150 );
   //...
}
```

7.3.23 UG_WindowSetStyle()

Changes the style of the window.

The following styles are available:

WND_STYLE_2D WND_STYLE_3D WND_STYLE_HIDE_TITLE WND_STYLE_SHOW_TITLE

Prototype:

```
UG_RESULT UG_WindowSetStyle( UG_WINDOW* wnd, UG_U8 style );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_U8 style	Window style

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowSetStyle( &window_1, WND_STYLE_3D | WND_STYLE_HIDE_TITLE );
   //...
}
```

7.3.24 UG_WindowGetForeColor()

Returns the fore color of the window.

Prototype:

```
UG_COLOR UG_WindowGetForeColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Fore color of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetForeColor( &window_1 );
   //...
}
```

7.3.25 UG_WindowGetBackColor()

Returns the back color of the window.

Prototype:

```
UG_COLOR UG_WindowGetBackColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Back color of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetBackColor( &window_1 );
   //...
}
```

7.3.26 UG_WindowGetTitleTextColor()

Returns the text color of the window title.

Prototype:

```
UG_COLOR UG_WindowGetTitleTextColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Text color of the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetTitleTextColor( &window_1 );
   //...
}
```

7.3.27 UG_WindowGetTitleColor()

Returns the title color of the window.

Prototype:

```
UG_COLOR UG_WindowGetTitleColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Title color of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetTitleColor( &window_1 );
   //...
}
```

7.3.28 UG_WindowGetTitleInactiveTextColor()

Returns the inactive text color of the window title.

Prototype:

```
UG_COLOR UG_WindowGetTitleInactiveTextColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Inactive text color of the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetTitleInactiveTextColor( &window_1 );
   //...
}
```

7.3.29 UG_WindowGetTitleInactiveColor()

Returns the inactive color of the window title.

Prototype:

```
UG_COLOR UG_WindowGetTitleInactiveColor( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_COLOR Inactive color of the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    color = UG_WindowGetTitleInactiveColor( &window_1 );
   //...
}
```

7.3.30 UG_WindowGetTitleText()

Returns a pointer to the title text of the window.

Prototype:

```
char* UG_WindowGetTitleText( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
char* Pointer to the title text of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
```

```
    str = (char*)UG_WindowGetTitleText( &window_1 );
    //...
}
```

7.3.31 UG_WindowGetTitleTextFont()

Returns a pointer to the font of the title text of the window.

Prototype:

```
UG_FONT* UG_WindowGetTitleTextFont( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_FONT* Pointer to the font of the title text of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    font = (UG_FONT*)UG_WindowGetTitleTextFont( &window_1 );
    //...
}
```

7.3.32 UG_WindowGetTitleTextHSpace()

Returns the horizontal space between the characters in the window title.

Prototype:

```
UG_S8 UG_WindowGetTitleTextHSpace( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S8 Horizontal space between the characters in the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   UG_S8 space;
   //...
    space = UG_WindowGetTitleTextHSpace( &window_1 );
   //...
}
```

7.3.33 UG_WindowGetTitleTextVSpace()

Returns the vertical space between the characters in the window title.

Prototype:

```
UG_S8 UG_WindowGetTitleTextVSpace( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S8 Vertical space between the characters in the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   UG_S8 space;
   //...
    space = UG_WindowGetTitleTextVSpace( &window_1 );
   //...
}
```

7.3.34 UG_WindowGetTitleTextAlignment()

Returns the alignment of the text in the window title.

Prototype:

```
UG_U8 UG_WindowGetTitleTextAlignment( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_U8 Alignment of the text in the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    align = UG_WindowGetTitleTextAlignment( &window_1 );
   //...
}
```

7.3.35 UG_WindowGetTitleHeight()

Returns the height of the window title.

Prototype:

```
UG_U8 UG_WindowGetTitleHeight( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_U8 Height of the window title

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    height = UG_WindowGetTitleHeight( &window_1 );
   //...
}
```

7.3.36 UG_WindowGetXStart()

Returns the x start position of the window.

Prototype:

```
UG_S16 UG_WindowGetXStart( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 X start position of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    pos = UG_WindowGetXStart( &window_1 );
   //...
}
```

7.3.37 UG_WindowGetYStart()

Returns the y start position of the window.

Prototype:

```
UG_S16 UG_WindowGetYStart( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 Y start position of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    pos = UG_WindowGetYStart( &window_1 );
   //...
}
```

7.3.38 UG_WindowGetXEnd()

Returns the x end position of the window.

Prototype:

```
UG_S16 UG_WindowGetXEnd( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 X end position of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    pos = UG_WindowGetXEnd( &window_1 );
   //...
}
```

7.3.39 UG_WindowGetYEnd()

Returns the y end position of the window.

Prototype:

```
UG_S16 UG_WindowGetYEnd( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 Y end position of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
```

```
    pos = UG_WindowGetYEnd( &window_1 );
    //...
}
```

7.3.40 UG_WindowGetStyle()

Returns the style of the window.

Prototype:

```
UG_U8 UG_WindowGetStyle( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window

Returns:

UG_U8	Style of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    style = UG_WindowGetStyle( &window_1 );
    //...
}
```

7.3.41 UG_WindowGetArea()

Writes the area of the window to a variable.

Prototype:

```
UG_RESULT UG_WindowGetArea( UG_WINDOW* wnd, UG_AREA* a );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window
UG_AREA* a	Pointer to the area destination variable

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    UG_WindowGetArea( &window_1, &area );
   //...
}
```

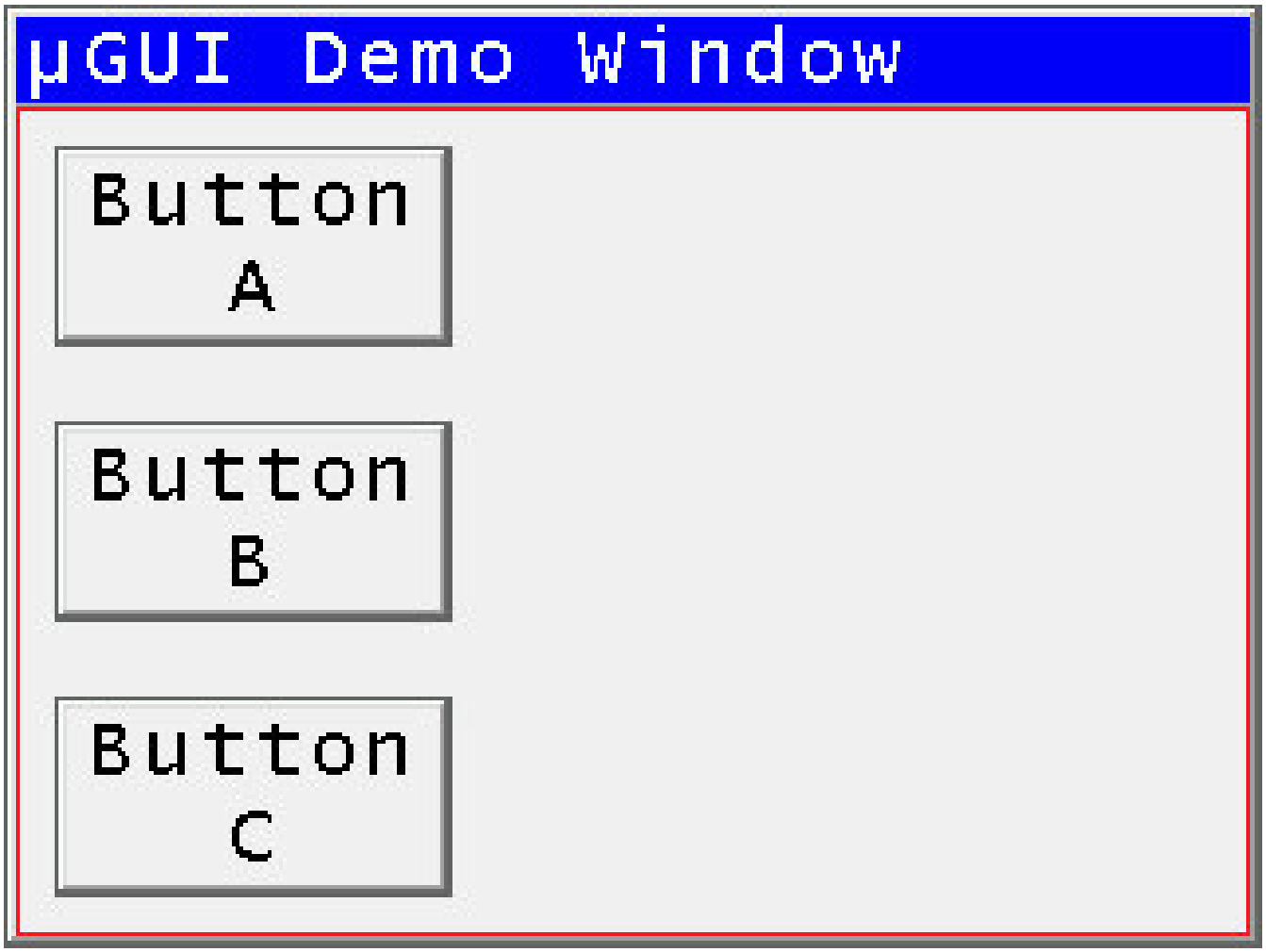


Figure 45: UG_WindowGetArea() example

7.3.42 UG_WindowGetInnerWidth()

Returns the inner width of the window.

Prototype:

```
UG_S16 UG_WindowGetInnerWidth( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:

UG_S16 Inner width of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    width = UG_WindowGetInnerWidth( &window_1 );
   //...
}
```

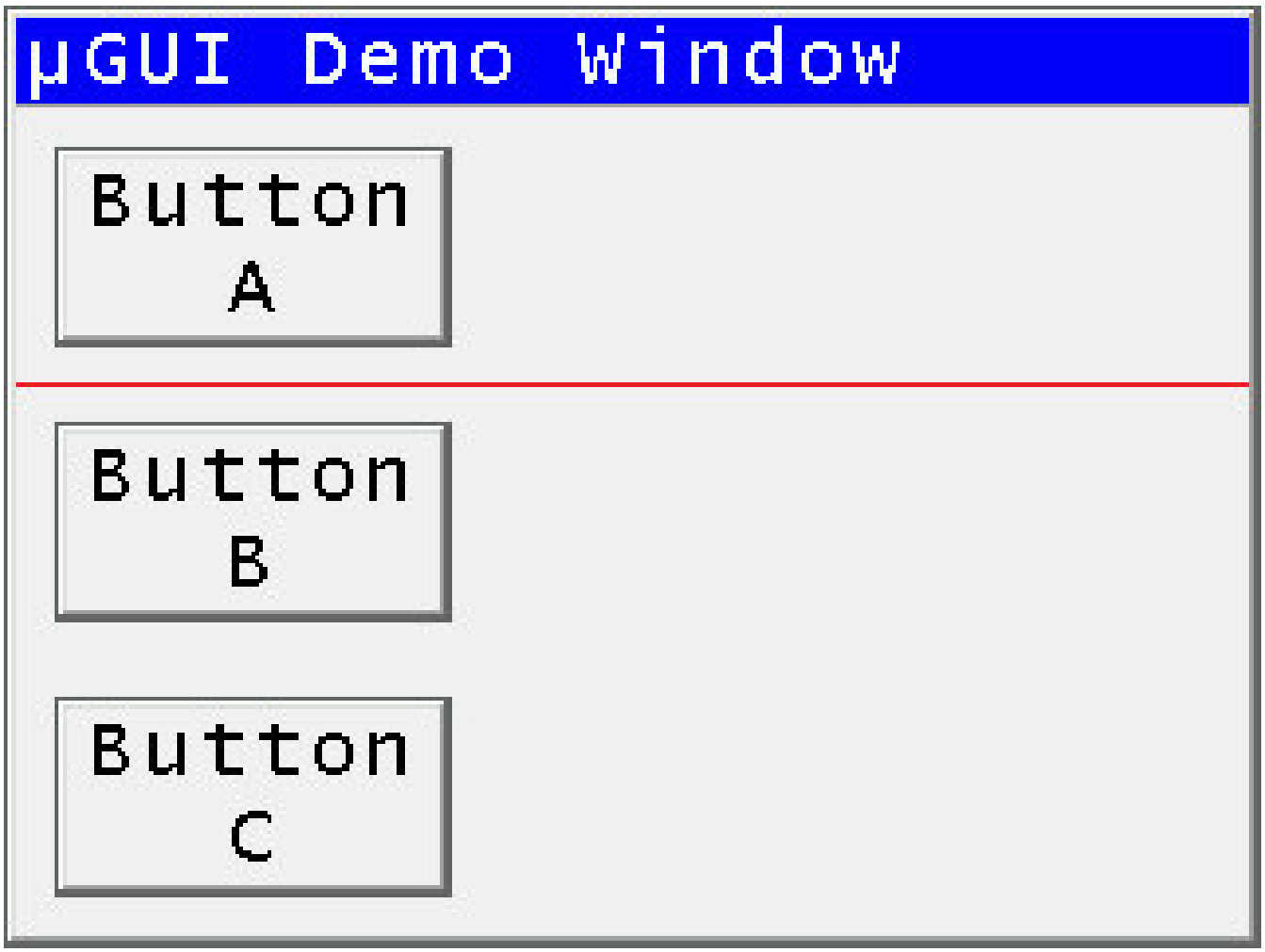


Figure 46: UG_WindowGetInnerWidth() example

7.3.43 UG_WindowGetOuterWidth()

Returns the outer width of the window.

Prototype:

```
UG_S16 UG_WindowGetOuterWidth( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 Outer width of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    width = UG_WindowGetOuterWidth( &window_1 );
   //...
}
```

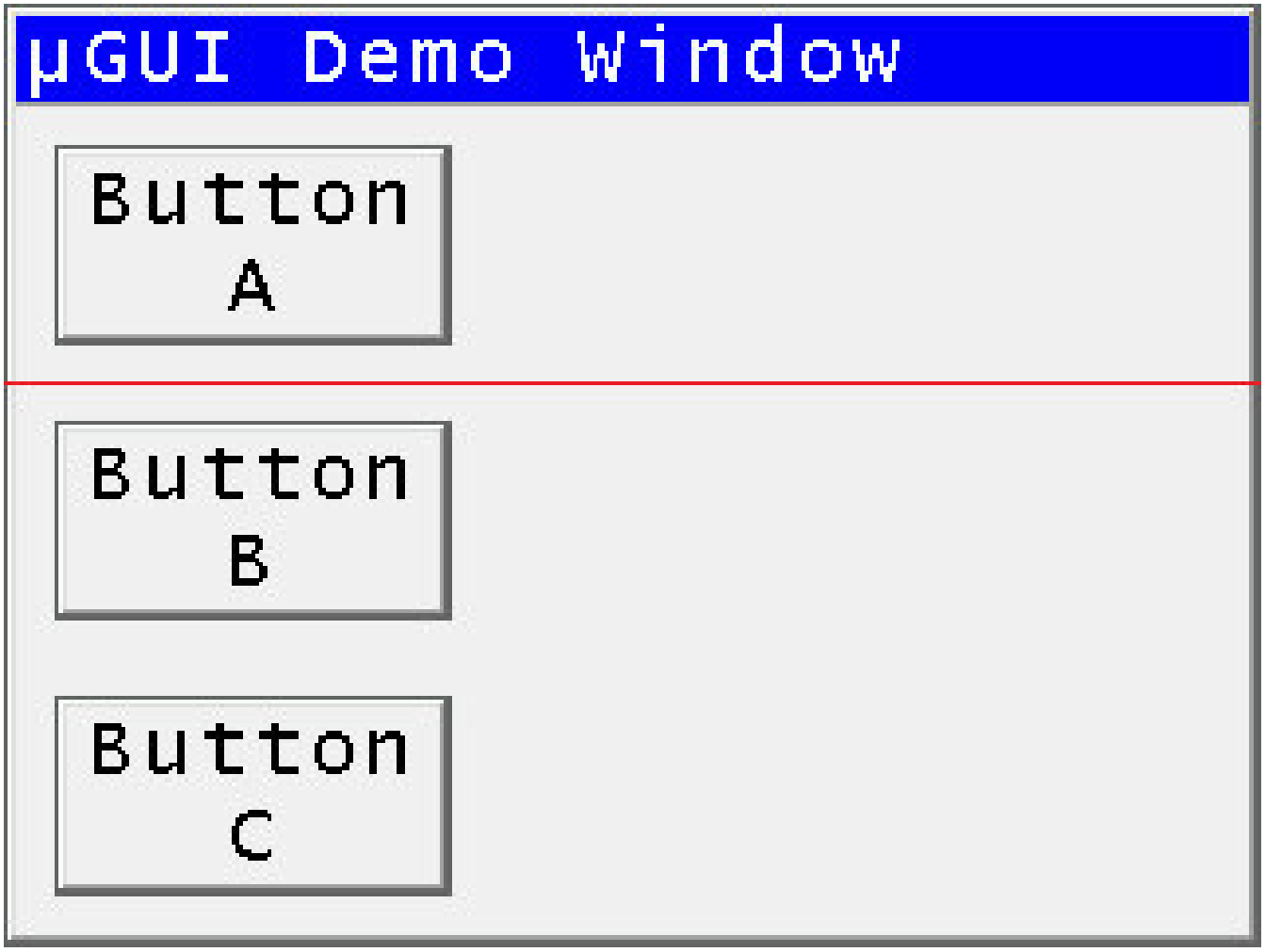


Figure 47: UG_WindowGetOuterWidth() example

7.3.44 UG_WindowGetInnerHeight()

Returns the inner height of the window.

Prototype:

```
UG_S16 UG_WindowGetInnerHeight( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:
UG_S16 Inner height of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    height = UG_WindowGetInnerHeight( &window_1 );
   //...
}
```

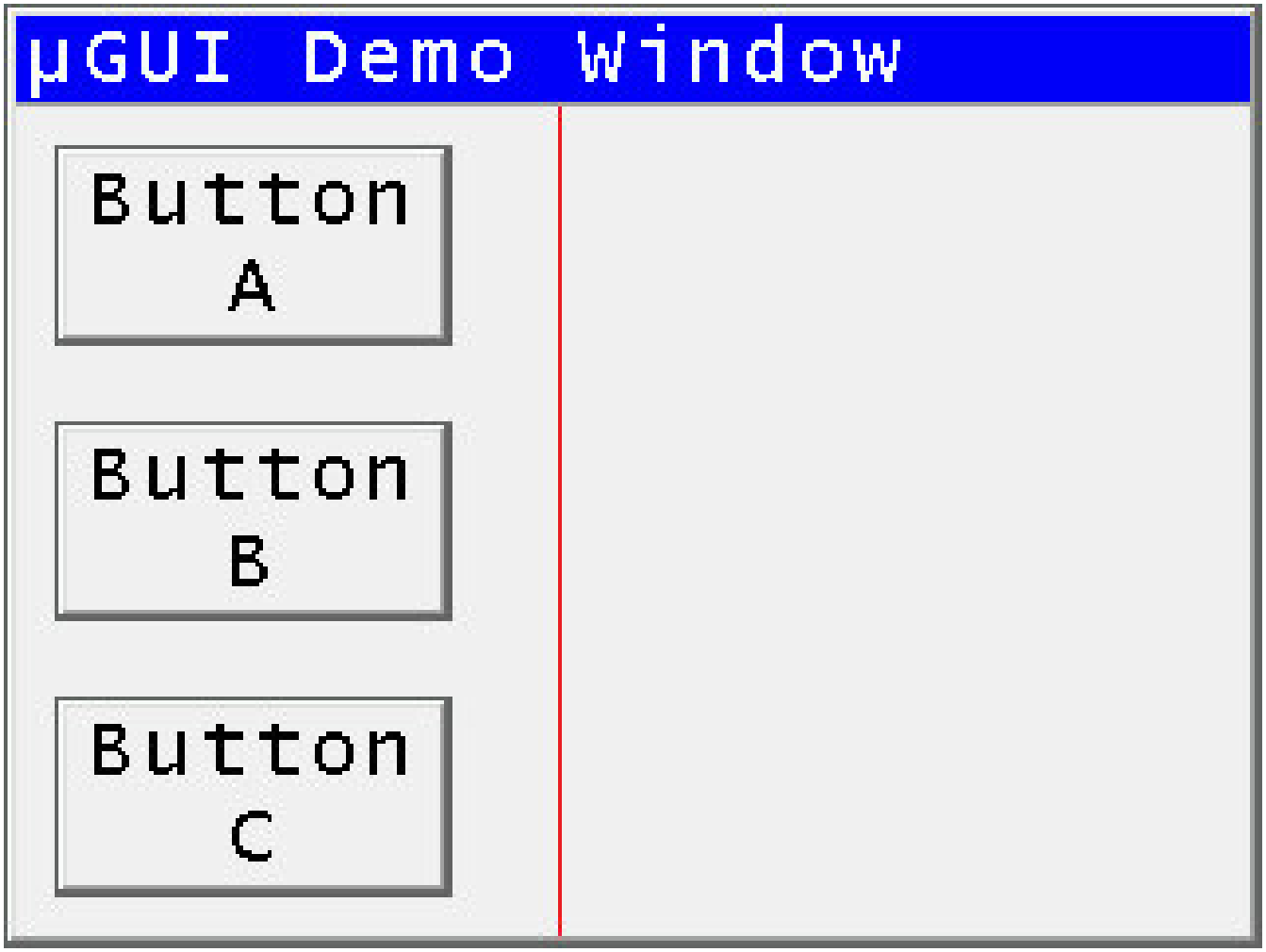


Figure 48: UG_WindowGetInnerHeight() example

7.3.45 UG_WindowGetOuterHeight()

Returns the outer height of the window.

Prototype:

```
UG_S16 UG_WindowGetOuterHeight( UG_WINDOW* wnd );
```

Parameters:

UG_WINDOW* wnd Pointer to the window

Returns:

UG_S16 Outer height of the window

Example:

```
int main( void )
{
    UG_WINDOW window_1;
   //...
    height = UG_WindowGetOuterHeight( &window_1 );
   //...
}
```

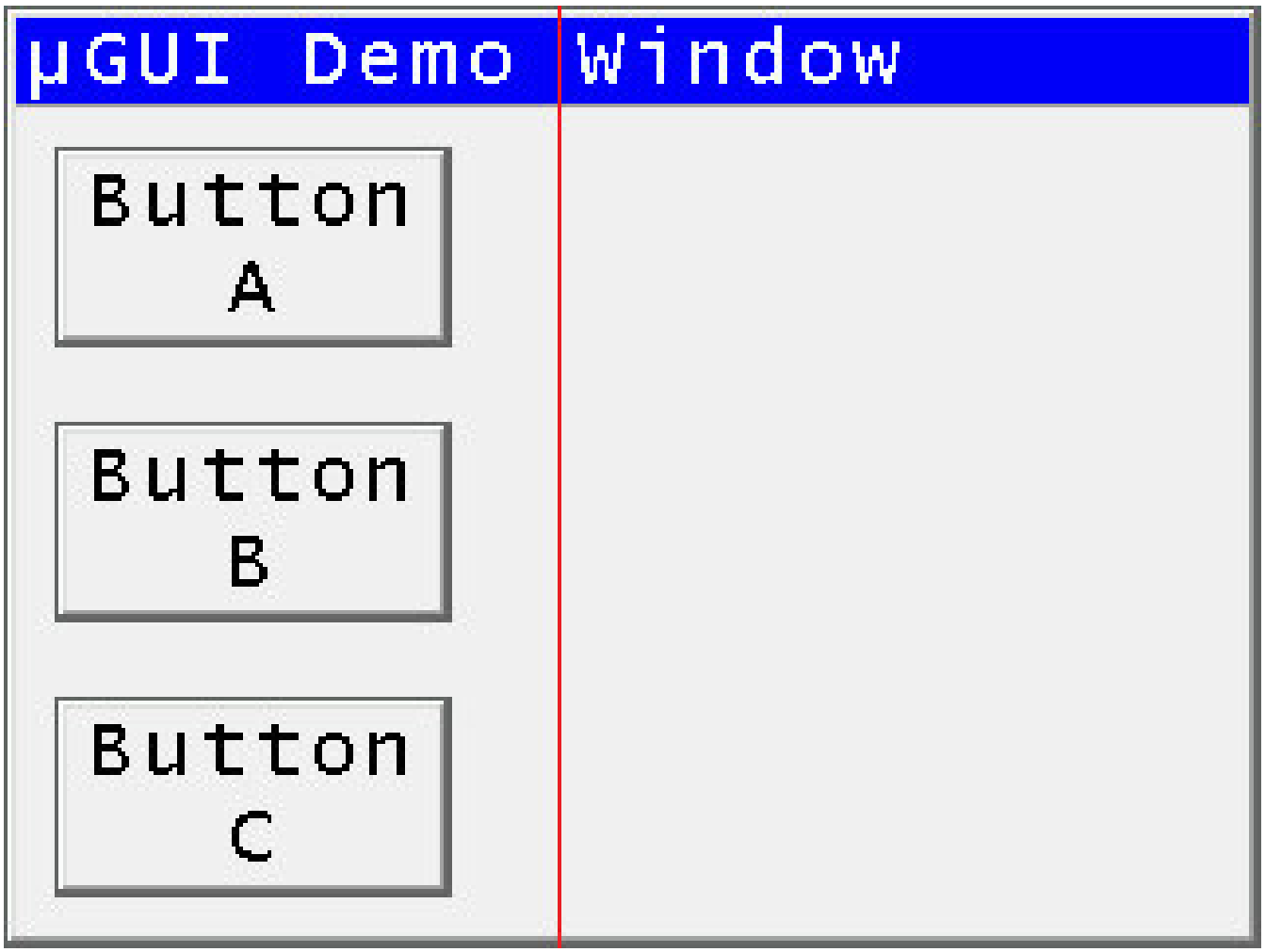


Figure 49: UG_WindowGetOuterHeight() example

7.4 Button Functions

7.4.1 UG_ButtonCreate()

Creates a button.

Prototype:

```
UG_RESULT UG_ButtonCreate( UG_WINDOW* wnd, UG_BUTTON* btn, UG_U8 id, UG_S16 xs,
    UG_S16 ys, UG_S16 xe, UG_S16 ye );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_BUTTON* btn	Pointer to the button
UG_U8 id	Button ID
UG_S16 xs	X start position of the button
UG_S16 ys	Y start position of the button
UG_S16 xe	X end position of the button
UG_S16 ye	Y end position of the button

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_BUTTON button_1;
   //...
   UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 10, 10, 110, 60 );
   //...
}
```

7.4.2 UG_ButtonDelete()

Deletes a button.

Prototype:

```
UG_RESULT UG_ButtonDelete( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonDelete( &window_1, BTN_ID_0 );
   //...
}
```

7.4.3 UG_ButtonShow()

Shows a button.

Prototype:

```
UG_RESULT UG_ButtonShow( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonShow( &window_1, BTN_ID_0 );
   //...
}
```

7.4.4 UG_ButtonHide()

Hides a button.

Prototype:

```
UG_RESULT UG_ButtonHide( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonHide( &window_1, BTN_ID_0 );
   //...
}
```

7.4.5 UG_ButtonSetForeColor()

Changes the fore color of the button.

Prototype:

```
UG_RESULT UG_ButtonSetForeColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR fc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
UG_COLOR fc	Fore color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetForeColor( &window_1, BTN_ID_0, C_BLACK );
   //...
}
```

7.4.6 UG_ButtonSetBackColor()

Changes the back color of the button.

Prototype:

```
UG_RESULT UG_ButtonSetBackColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR bc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
UG_COLOR bc	Back color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetBackColor( &window_1, BTN_ID_0, C_GRAY );
   //...
}
```

7.4.7 UG_ButtonSetAlternateForeColor()

Changes the alternate fore color of the button.

Prototype:

```
UG_RESULT UG_ButtonSetAlternateForeColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR afc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
UG_COLOR afc	Alternate fore color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetAlternateForeColor( &window_1, BTN_ID_0, C_BLUE );
   //...
}
```

7.4.8 UG_ButtonSetAlternateBackColor()

Changes the alternate back color of the button.

Prototype:

```
UG_RESULT UG_ButtonSetAlternateBackColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR abc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
UG_COLOR abc	Alternate back color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetAlternateBackColor( &window_1, BTN_ID_0, C_YELLOW );
   //...
}
```

7.4.9 UG_ButtonSetText()

Changes the button text.

Prototype:

```
UG_RESULT UG_ButtonSetText( UG_WINDOW* wnd, UG_U8 id, char* str );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
char* str	Pointer to the button text

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetText( &window_1, BTN_ID_0, "Button\n:)" );
   //...
}
```

7.4.10 UG_ButtonSetFont()

Changes the button font.

Prototype:

```
UG_RESULT UG_ButtonSetFont( UG_WINDOW* wnd, UG_U8 id, const UG_FONT* font );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
const UG_FONT* font	Pointer to the button font

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetFont( &window_1, BTN_ID_0, &FONT_12X20 );
   //...
}
```

7.4.11 UG_ButtonSetStyle()

Changes the button style.

The following styles are available:
BTN_STYLE_2D
BTN_STYLE_3D
BTN_STYLE_TOGGLE_COLORS
BTN_STYLE_USE_ALTERNATE_COLORS

Prototype:

```
UG_RESULT UG_ButtonSetStyle( UG_WINDOW* wnd, UG_U8 id, UG_U8 style );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID
UG_U8 style	Button style

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_ButtonSetStyle( &window_1, BTN_ID_0, BTN_STYLE_3D | BTN_STYLE_TOGGLE_COLORS )
       ;
   //...
}
```

7.4.12 UG_ButtonGetForeColor()

Returns the fore color of the button.

Prototype:

```
UG_COLOR UG_ButtonGetForeColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:
UG_COLOR Fore color of the button

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR fc;
   //...
   fc = UG_ButtonGetForeColor( &window_1, BTN_ID_0 );
   //...
}
```

7.4.13 UG_ButtonGetBackColor()

Returns the back color of the button.

Prototype:

```
UG_COLOR UG_ButtonGetBackColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:
UG_COLOR Back color of the button

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR bc;
   //...
   bc = UG_ButtonGetBackColor( &window_1, BTN_ID_0 );
   //...
}
```

7.4.14 UG_ButtonGetAlternateForeColor()

Returns the alternate fore color of the button.

Prototype:

```
UG_COLOR UG_ButtonGetAlternateForeColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

UG_COLOR	Alternate fore color of the button

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR afc;
   //...
   afc = UG_ButtonGetAlternateForeColor( &window_1, BTN_ID_0 );
   //...
}
```

7.4.15 UG_ButtonGetAlternateBackColor()

Returns the alternate back color of the button.

Prototype:

```
UG_COLOR UG_ButtonGetAlternateBackColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

UG_COLOR	Alternate back color of the button

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR abc;
   //...
   abc = UG_ButtonGetAlternateBackColor( &window_1, BTN_ID_0 );
   //...
}
```

7.4.16 UG_ButtonGetText()

Returns a pointer to the button text.

Prototype:

```
char* UG_ButtonGetText( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

char*	Pointer to the button text

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   char* str;
   //...
   str = UG_ButtonGetText( &window_1, BTN_ID_0 );
   //...
}
```

7.4.17 UG_ButtonGetFont()

Returns a pointer to the button font.

Prototype:

```
UG_FONT* UG_ButtonGetFont( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

UG_FONT*	Pointer to the button font

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_FONT* font;
   //...
   font = UG_ButtonGetFont( &window_1, BTN_ID_0 );
   //...
}
```

7.4.18 UG_ButtonGetStyle()

Returns the button style.

Prototype:

```
UG_U8 UG_ButtonGetStyle( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the button
UG_U8 id	Button ID

Returns:

UG_U8	Button style

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_U8 style;
   //...
   style = UG_ButtonGetStyle( &window_1, BTN_ID_0 );
   //...
}
```

7.5 Textbox Functions

7.5.1 UG_TextboxCreate()

Creates a textbox.

Prototype:

```
UG_RESULT UG_TextboxCreate( UG_WINDOW* wnd, UG_TEXTBOX* txb, UG_U8 id, UG_S16 xs,
    UG_S16 ys, UG_S16 xe, UG_S16 ye );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_TEXTBOX* txb	Pointer to the textbox
UG_U8 id	Textbox ID
UG_S16 xs	X start position of the textbox
UG_S16 ys	Y start position of the textbox
UG_S16 xe	X end position of the textbox
UG_S16 ye	Y end position of the textbox

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_TEXTBOX textbox_1;
   //...
   UG_TextboxCreate( &window_1, &textbox_1, TXB_ID_0, 10, 10, 100, 100 );
   //...
}
```

7.5.2 UG_TextboxDelete()

Deletes a textbox.

Prototype:

```
UG_RESULT UG_TextboxDelete( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxDelete( &window_1, TXB_ID_0 );
   //...
}
```

7.5.3 UG_TextboxShow()

Shows a textbox.

Prototype:

```
UG_RESULT UG_TextboxShow( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxShow( &window_1, TXB_ID_0 );
   //...
}
```

7.5.4 UG_TextboxHide()

Hides a textbox.

Prototype:

```
UG_RESULT UG_TextboxHide( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxHide( &window_1, TXB_ID_0 );
   //...
}
```

7.5.5 UG_TextboxSetForeColor()

Changes the fore color of the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetForeColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR fc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
UG_COLOR fc	Fore color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxSetForeColor( &window_1, TXB_ID_0, C_BLACK );
   //...
}
```

7.5.6 UG_TextboxSetBackColor()

Changes the back color of the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetBackColor( UG_WINDOW* wnd, UG_U8 id, UG_COLOR bc );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
UG_COLOR bc	Back color

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxSetBackColor( &window_1, TXB_ID_0, C_YELLOW );
   //...
}
```

7.5.7 UG_TextboxSetText()

Changes the text of the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetText( UG_WINDOW* wnd, UG_U8 id, char* str );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
char* str	Pointer to the text

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
```

```
    UG_TextboxSetText( &window_1, TXB_ID_0, "Hello World!" );
    //...
}
```

7.5.8 UG_TextboxSetFont()

Changes the font of the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetFont( UG_WINDOW* wnd, UG_U8 id, const UG_FONT* font );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
const UG_FONT* font	Pointer to the font

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
    UG_WINDOW window_1;
    //...
    UG_TextboxSetFont( &window_1, TXB_ID_0, &FONT_12X20 );
    //...
}
```

7.5.9 UG_TextboxSetHSpace()

Changes the horizontal space between the characters in the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetHSpace( UG_WINDOW* wnd, UG_U8 id, UG_S8 hs );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
UG_S8 hs	Horizontal space

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxSetHSpace( &window_1, TXB_ID_0, 4 );
   //...
}
```

7.5.10 UG_TextboxSetVSpace()

Changes the vertical space between the characters in the textbox.

Prototype:

```
UG_RESULT UG_TextboxSetVSpace( UG_WINDOW* wnd, UG_U8 id, UG_S8 vs );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
UG_S8 vs	Vertical space

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   //...
   UG_TextboxSetVSpace( &window_1, TXB_ID_0, 4 );
   //...
}
```

7.5.11 UG_TextboxSetAlignment()

Changes the alignment of the text in the textbox.

The following alignments are available:

ALIGN_BOTTOM_RIGHT
ALIGN_BOTTOM_CENTER
ALIGN_BOTTOM_LEFT
ALIGN_CENTER_RIGHT
ALIGN_CENTER
ALIGN_CENTER_LEFT
ALIGN_TOP_RIGHT
ALIGN_TOP_CENTER
ALIGN_TOP_LEFT

Prototype:

```
UG_RESULT UG_TextboxSetAlignment( UG_WINDOW* wnd, UG_U8 id, UG_U8 align );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID
UG_U8 align	Alignment

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
  UG_WINDOW window_1;
  //...
  UG_TextboxSetAlignment( &window_1, TXB_ID_0, ALIGN_CENTER );
  //...
}
```

7.5.12 UG_TextboxGetForeColor()

Returns the fore color of the textbox.

Prototype:

```
UG_COLOR UG_TextboxGetForeColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:
UG_COLOR Fore color

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR fc;
   //...
   fc = UG_TextboxGetForeColor( &window_1, TXB_ID_0 );
   //...
}
```

7.5.13 UG_TextboxGetBackColor()

Returns the back color of the textbox.

Prototype:

```
UG_COLOR UG_TextboxGetBackColor( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:
UG_COLOR Back color

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_COLOR bc;
   //...
   bc = UG_TextboxGetBackColor( &window_1, TXB_ID_0 );
   //...
}
```

7.5.14 UG_TextboxGetText()

Returns a pointer to the textbox text.

Prototype:

```
char* UG_TextboxGetText( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

char* Pointer to the textbox text

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   char* str;
   //...
   str = UG_TextboxGetText( &window_1, TXB_ID_0 );
   //...
}
```

7.5.15 UG_TextboxGetFont()

Returns a pointer to the textbox font.

Prototype:

```
UG_FONT* UG_TextboxGetFont( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_FONT* Pointer to the textbox font

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_FONT* font;
   //...
   font = UG_TextboxGetFont( &window_1, TXB_ID_0 );
   //...
}
```

7.5.16 UG_TextboxGetHSpace()

Returns the horizontal space between the characters in the textbox.

Prototype:

```
UG_S8 UG_TextboxGetHSpace( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_S8	Horizontal space

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_S8 space;
   //...
   space = UG_TextboxGetHSpace( &window_1, TXB_ID_0 );
   //...
}
```

7.5.17 UG_TextboxGetVSpace()

Returns the vertical space between the characters in the textbox.

Prototype:

```
UG_S8 UG_TextboxGetVSpace( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_S8	Vertical space

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_S8 space;
   //...
   space = UG_TextboxGetVSpace( &window_1, TXB_ID_0 );
   //...
}
```

7.5.18 UG_TextboxGetAlignment()

Returns the alignment of the textbox text.

Prototype:

```
UG_U8 UG_TextboxGetAlignment( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the textbox
UG_U8 id	Textbox ID

Returns:

UG_U8	Alignment

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_U8 align;
   //...
   align = UG_TextboxGetAlignment( &window_1, TXB_ID_0 );
   //...
}
```

7.6 Image Functions

7.6.1 UG_ImageCreate()

Creates an image.

Prototype:

```
UG_RESULT UG_ImageCreate( UG_WINDOW* wnd, UG_IMAGE* img, UG_U8 id, UG_S16 xs,
    UG_S16 ys, UG_S16 xe, UG_S16 ye );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the image
UG_TEXTBOX* txb	Pointer to the image
UG_U8 id	Image ID
UG_S16 xs	X start position of the image
UG_S16 ys	Y start position of the image
UG_S16 xe	X end position of the image
UG_S16 ye	Y end position of the image

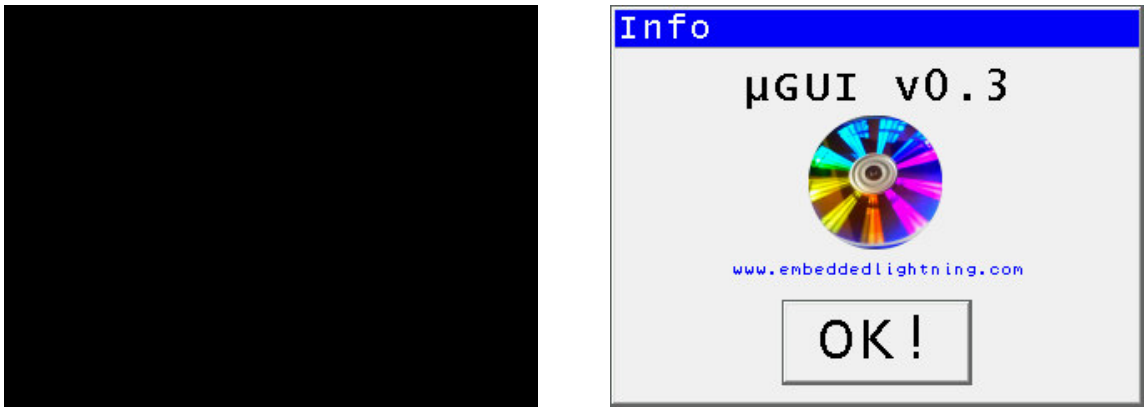
Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   UG_TEXTBOX textbox_1;
   UG_TEXTBOX textbox_2;
   UG_BUTTON button_1;
   UG_IMAGE image_1;

   const UG_U16 logo_bmp[] = {...};

   const UG_BMP logo =
   {
      (void*)logo_bmp,
      80,
      80,
      BMP_BPP_16,
      BMP_RGB565
   };

   //...
   /* Create the window */
   UG_WindowCreate( &window_1, obj_buff_wnd_1, MAX_OBJECTS, window_1_callback );
   UG_WindowSetTitleText( &window_1, "Info" );
   UG_WindowSetTitleTextFont( &window_1, &FONT_12X20 );

   /* Create the button */
   UG_ButtonCreate( &window_1, &button_1, BTN_ID_0, 100, 150,
      UG_WindowGetInnerWidth( &window_1 )-100, 200 );
   UG_ButtonSetFont( &window_1, BTN_ID_0, &FONT_22X36 );
   UG_ButtonSetText( &window_1, BTN_ID_0, "OK!" );

   /* Create textbox 1 */
   UG_TextboxCreate( &window_1, &textbox_1, TXB_ID_0, 10, 10,
      UG_WindowGetInnerWidth( &window_1 )-10, 40 );
```

```
    UG_TextboxSetFont( &window_1, TXB_ID_0, &FONT_16X26 );
    UG_TextboxSetText( &window_1, TXB_ID_0, "uGUI v0.3" );
    UG_TextboxSetAlignment( &window_1, TXB_ID_0, ALIGN_TOP_CENTER );

    /* Create textbox 2 */
    UG_TextboxCreate(&window_1, &textbox_2, TXB_ID_1, 10, 125,
        UG_WindowGetInnerWidth( &window_1 )-10, 135 );
    UG_TextboxSetFont(&window_1, TXB_ID_1, &FONT_6X8 );
    UG_TextboxSetText(&window_1, TXB_ID_1, "www.embeddedlightning.com" );
    UG_TextboxSetAlignment(&window_1, TXB_ID_1, ALIGN_BOTTOM_CENTER );
    UG_TextboxSetForeColor( &window_1, TXB_ID_1, C_BLUE );
    UG_TextboxSetHSpace( &window_1, TXB_ID_1, 1 );

    /* Create the image */
    UG_ImageCreate( &window_1, &image_1, IMG_ID_0, (UG_WindowGetInnerWidth( &
        window_1 )>>1) - (logo.width>>1), 40, 0, 0 );
    UG_ImageSetBMP( &window_1, IMG_ID_0, &logo );

    UG_WindowShow( &window_1 );
    //...
}
```



(a) Before (b) After

Figure 50: UG_ImageCreate() example

7.6.2 UG_ImageDelete()

Deletes an image.

Prototype:

```
UG_RESULT UG_ImageDelete( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the image
UG_U8 id	Image ID

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;

   //...
   UG_ImageDelete( &window_1, IMG_ID_0 );
   //...
}
```

7.6.3 UG_ImageShow()

Shows an image.

Prototype:

```
UG_RESULT UG_ImageShow( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the image
UG_U8 id	Image ID

Returns:
UG_RESULT Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;

   //...
   UG_ImageShow( &window_1, IMG_ID_0 );
   //...
}
```

7.6.4 UG_ImageHide()

Hides an image.

Prototype:

```
UG_RESULT UG_ImageHide( UG_WINDOW* wnd, UG_U8 id );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the image
UG_U8 id	Image ID

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;

   //...
   UG_ImageHide( &window_1, IMG_ID_0 );
   //...
}
```

7.6.5 UG_ImageSetBMP()

Links an µGUI bitmap (UG_BMP) to the image object.

Prototype:

```
UG_RESULT UG_ImageSetBMP( UG_WINDOW* wnd, UG_U8 id, const UG_BMP* bmp );
```

Parameters:

UG_WINDOW* wnd	Pointer to the window which contains the image
UG_U8 id	Image ID
const UG_BMP* bmp	Pointer to the µGUI bitmap

Returns:

UG_RESULT	Result of the function

Example:

```
int main( void )
{
   UG_WINDOW window_1;
   const UG_U16 logo_bmp[] = {...};
   const UG_BMP logo =
   {
```

```
        (void*)logo_bmp,
        80,
        80,
        BMP_BPP_16,
        BMP_RGB565
    };

    //...
    UG_ImageSetBMP( &window_1, IMG_ID_0, &logo );
    //...
}
```

List of Figures

8 Revision history

8.1 Software

Software Version	Date	Description
v0.3	Mar 18, 2015	▪ Support for hardware acceleration (driver) added. ▪ Support for external fonts added (Font system changed). ▪ Color C_GREEN changed to 0x00FF00. ▪ Function UG_DriverRegister() added. ▪ Function UG_DriverEnable() added. ▪ Function UG_DriverDisable() added. ▪ Function UG_WindowCreate() added. ▪ Function UG_WindowDelete() added. ▪ Function UG_WindowShow() added. ▪ Function UG_WindowHide() added. ▪ Function UG_WindowResize() added. ▪ Function UG_WindowAlert() added. ▪ Function UG_WindowSetForeColor() added. ▪ Function UG_WindowSetBackColor() added. ▪ Function UG_WindowSetTitleTextColor() added. ▪ Function UG_WindowSetTitleColor() added. ▪ Function UG_WindowSetTitleInactiveTextColor() added. ▪ Function UG_WindowSetTitleInactiveColor() added. ▪ Function UG_WindowSetTitleText() added. ▪ Function UG_WindowSetTitleTextFont() added. ▪ Function UG_WindowSetTitleTextHSpace() added. ▪ Function UG_WindowSetTitleTextVSpace() added. ▪ Function UG_WindowSetTitleTextAlignment() added. ▪ Function UG_WindowSetTitleHeight() added. ▪ Function UG_WindowSetXStart() added. ▪ Function UG_WindowSetYStart() added. ▪ Function UG_WindowSetXEnd() added. ▪ Function UG_WindowSetYEnd() added. ▪ Function UG_WindowSetStyle() added. ▪ Function UG_WindowGetForeColor() added. ▪ Function UG_WindowGetBackColor() added. ▪ Function UG_WindowGetTitleTextColor() added. ▪ Function UG_WindowGetTitleColor() added. ▪ Function UG_WindowGetTitleInactiveTextColor() added. ▪ Function UG_WindowGetTitleInactiveColor() added. ▪ Function UG_WindowGetTitleText() added. ▪ Function UG_WindowGetTitleTextFont() added. ▪ Function UG_WindowGetTitleTextHSpace() added. ▪ Function UG_WindowGetTitleTextVSpace() added. ▪ Function UG_WindowGetTitleTextAlignment() added. ▪ Function UG_WindowGetTitleHeight() added.

		▪ Function UG_WindowGetXStart() added. ▪ Function UG_WindowGetYStart() added. ▪ Function UG_WindowGetXEnd() added. ▪ Function UG_WindowGetYEnd() added. ▪ Function UG_WindowGetStyle() added. ▪ Function UG_WindowGetArea() added. ▪ Function UG_WindowGetInnerWidth() added. ▪ Function UG_WindowGetOuterWidth() added. ▪ Function UG_WindowGetInnerHeight() added. ▪ Function UG_WindowGetOuterHeight() added. ▪ Function UG_ButtonCreate() added. ▪ Function UG_ButtonDelete() added. ▪ Function UG_ButtonShow() added. ▪ Function UG_ButtonHide() added. ▪ Function UG_ButtonSetForeColor() added. ▪ Function UG_ButtonSetBackColor() added. ▪ Function UG_ButtonSetAlternateForeColor() added. ▪ Function UG_ButtonSetAlternateBackColor() added. ▪ Function UG_ButtonSetText() added. ▪ Function UG_ButtonSetFont() added. ▪ Function UG_ButtonSetStyle() added. ▪ Function UG_ButtonGetForeColor() added. ▪ Function UG_ButtonGetBackColor() added. ▪ Function UG_ButtonGetAlternateForeColor() added. ▪ Function UG_ButtonGetAlternateBackColor() added. ▪ Function UG_ButtonGetText() added. ▪ Function UG_ButtonGetFont() added. ▪ Function UG_ButtonGetStyle() added. ▪ Function UG_TextboxCreate() added. ▪ Function UG_TextboxDelete() added. ▪ Function UG_TextboxShow() added. ▪ Function UG_TextboxHide() added. ▪ Function UG_TextboxSetForeColor() added. ▪ Function UG_TextboxSetBackColor() added. ▪ Function UG_TextboxSetText() added. ▪ Function UG_TextboxSetFont() added. ▪ Function UG_TextboxSetHSpace() added. ▪ Function UG_TextboxSetVSpace() added. ▪ Function UG_TextboxSetAlignment() added. ▪ Function UG_TextboxGetForeColor() added. ▪ Function UG_TextboxGetBackColor() added. ▪ Function UG_TextboxGetText() added. ▪ Function UG_TextboxGetFont() added. ▪ Function UG_TextboxGetHSpace() added. ▪ Function UG_TextboxGetVSpace() added. ▪ Function UG_TextboxGetAlignment() added. ▪ Function UG_ImageCreate() added.

		▪ Function UG_ImageDelete() added. ▪ Function UG_ImageShow() added. ▪ Function UG_ImageHide() added. ▪ Function UG_ImageSetBMP() added. ▪ Function UG_WaitForUpdate() added. ▪ Function UG_Update() added. ▪ Function UG_DrawBMP() added. ▪ Function UG_TouchUpdate() added. ▪ Fixed some minor bugs.
v0.2	Oct 20, 2014	▪ Function UG_DrawRoundFrame() added. ▪ Function UG_FillRoundFrame() added. ▪ Function UG_DrawArc() added. ▪ Fixed some minor bugs.
v0.1	Oct 11, 2014	▪ First release.

8.2 Document

Document Version	Date	Description
v0.3	Mar 18, 2015	▪ µGUI v0.3 features added (see software history for details). ▪ Window section added. ▪ Touch section added. ▪ Driver section added.
v0.2	Oct 20, 2014	▪ µGUI v0.2 features added (see software history for details). ▪ Color space added. ▪ Revisions section added. ▪ Documentation generated by LaTeX.
v0.1	Oct 11, 2014	▪ First release.