

μGUI

Open Source GUI module
for embedded systems

Reference Guide

Software version: v0.2
Document version: v0.2

www.embeddedlightning.com
(visit for latest updates)

©Achim Döbler

Contents

1	Introduction	3
1.1	What is µGUI	3
1.2	µGUI Features	3
1.3	µGUI Requirements	3
2	Colors	3
2.1	Color Space	3
2.2	Predefined Colors	4
3	Fonts	7
3.1	FONT_4X6	7
3.2	FONT_5X8	7
3.3	FONT_5X12	7
3.4	FONT_6X8	8
3.5	FONT_6X10	8
3.6	FONT_7X12	8
3.7	FONT_8X8	8
3.8	FONT_8X12	9
3.9	FONT_8X14	9
3.10	FONT_10X16	9
3.11	FONT_12X16	10
3.12	FONT_12X20	10
3.13	FONT_16X26	11
3.14	FONT_22X36	12
3.15	FONT_24X40	13
3.16	FONT_32X53	14
4	Functions	15
4.1	UG_Init()	15
4.2	UG_SelectGUI()	15
4.3	UG_FontSelect()	16
4.4	UG_FillScreen()	17
4.5	UG_FillFrame()	17
4.6	UG_FillRoundFrame()	18
4.7	UG_DrawMesh()	19
4.8	UG_DrawFrame()	19
4.9	UG_DrawRoundFrame()	20
4.10	UG_DrawPixel()	20
4.11	UG_DrawCircle()	21
4.12	UG_FillCircle()	21
4.13	UG_DrawArc()	22
4.14	UG_DrawLine()	23
4.15	UG_PutString()	23
4.16	UG_PutChar()	24
4.17	UG_ConsolePutString()	25

4.18	UG_ConsoleSetArea()	25
4.19	UG_ConsoleSetForecolor()	26
4.20	UG_ConsoleSetBackcolor()	26
4.21	UG_SetForecolor()	27
4.22	UG_SetBackcolor()	27
4.23	UG_GetXDim()	28
4.24	UG_GetYDim()	28
4.25	UG_FontSetHSpace()	29
4.26	UG_FontSetVSpace()	29
5	Revision history	30
5.1	Software	30
5.2	Document	30

1 Introduction

1.1 What is μ GUI

μ GUI is a free and open source graphic library for embedded systems. It is platform-independent and can be easily ported to almost every microcontroller system. As long as the display is capable of showing graphics, μ GUI 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 μ GUI Features

- μ GUI supports any color, grayscale or monochrome display
- μ GUI supports any display resolution
- μ GUI supports multiple different displays
- 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

1.3 μ GUI Requirements

μ GUI is platform-independent, so there is no need to use a certain embedded system. In order to use μ GUI, 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 Colors

2.1 Color Space

μ GUI 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.

2.2 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
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

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

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_ANTIQUUE_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

3.11 FONT_12X16

[illegible]

3.12 FONT_12X20

[illegible]

3.13 FONT_16X26

[illegible]

3.14 FONT_22X36

☺☹♥♦♣♠ ♂♀ 🎵⚙
 ▶◀↕↗↘ⓂⓈ-↕↗↘→←↔↻▲▼
 ! " # \$ % & ' () * + , - . /
 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
 @ A B C D E F G H I J K L M N O
 P Q R S T U V W X Y Z [\] ^ _
 ` a b c d e f g h i j k l m n o
 p q r s t u v w x y z { | } ~ □
 Ç ü é â ä à å ç ê ë è ì í î ï Ä Å
 Ë æ Æ ô ö ò û ü ÿ Ö Ü ø £ Ø × f
 á í ó ú ñ Ñ ª ° ¿ ® ¬ ½ ¼ ¡ « »
 ▨ ▩ | − Á Â Ã Ä Å Æ Ç È É Ê Ë
 Ì Í Î Ï Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ
 à á â ã ä å æ ç è é ê ë ì í î ï ð
 ñ ò ó ô õ ö ÷ , ° ´ · ¹ º ³ ² ■

3.15 FONT_24X40

☺☹♥♦♣♠ ♂♀ 🎵⚙
 ▶◀↕!!¶§−↑↑↓→←↲↻▲▼
 !"#\$%&'()*+,-./
 0123456789:;<=>?
 @ABCDEFGHIJKLMNO
 PQRSTUVWXYZ[\]^_
 `abcdefghijklmnopqrstuvwxyz{|}~□
 ÇüéâäåàçêëèìíîÏÄÅ
 ÆœôöòûüÿÖÜøƒØ×ƒ
 áíóúñÑªº¿®¬½¼¡«»
 ░▒▓▔▕▖▗▘▙▚▛▜▝▞▟■□▢▣▤▥▦▧▨▩
 |−ÁÂÃÄÅ Æ Ç È É Ê Ë Ì Í Î Ï Ñ
 Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã
 ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷
 −±¼¶§÷, ° ´ · ¹ º ³ ² ■

3.16 FONT_32X53

☺☻♥♦♣♠ ♂♀ 🎵⚙️
 ⏮⏭↕!!¶§-↕↑↓→←↲↻▲▼
 !"#\$%&'()*+,-./
 0123456789:;=<=>?
 @ABCDEFGHIJKLMNO
 PQRSTUVWXYZ[\]^_
 `abcdefghijklmnopqrstuvwxyz{|}~□
 ÇüéâäàåçêëèìíîÏÄÅ
 ÆæŒœôöòûüÿÖÜø£ø×ƒ
 áíóúñÑªº¿®¬½¼¡«»
 ░▒▓▔▕▖▗▘▙▚▛▜▝▞▟■□▢▣▤▥▦▧▨▩
 | Á Â Ã Ä Å © ª « ¬ ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾
 ¨ ð ð É Ê Ë Ì Í Î Ï Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã
 ô õ ö ÷ , ° ´ · ¹ º ³ ² ■

4 Functions

4.1 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);
    // ...
    // ...
}
```

4.2 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 );
    //...
}
```

4.3 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( UG_U16 font );
```

Parameters:

UG_U16 font Font

Example:

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

4.4 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 );
    //...
}
```

4.5 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_BLUE);
    //...
}
```

4.6 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_BLUE);
    //...
}
```

4.7 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_BLUE);
    // ...
}
```

4.8 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_BLUE);
    // ...
}

```

4.9 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_BLUE);
    // ...
}

```

4.10 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(0, 0, C_GREEN);
    // ...
}
```

4.11 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);
    // ...
}
```

4.12 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);
    //...
}
```

4.13 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);
    }
}
// ...
}

```

4.14 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);
    // ...
}

```

4.15 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_PutString(0, 0, "Hello World!");  
    //...  
}
```

4.16 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 Forecolor of the char
UG_COLOR bc Backcolor of the char

Example:

```
int main( void )  
{  
    //...  
    UG_PutChar( 'A', 0, 0, C_BLUE, C_YELLOW );  
    //...  
}
```

4.17 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_ConsolePutString("System initialized!\n");
    //...
    UG_ConsolePutString("SD-Card mounted!\n");
    //...
}
```

4.18 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 );
}
```

```
} //...
```

4.19 UG_ConsoleSetForecolor()

Defines the forecolor of the console.

Prototype:

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

Parameters:

UG_COLOR c Forecolor

Example:

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

4.20 UG_ConsoleSetBackcolor()

Defines the backcolor of the console.

Prototype:

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

Parameters:

UG_COLOR c Backcolor

Example:

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

```
} //...
```

4.21 UG_SetForecolor()

Defines the forecolor of the string.

Prototype:

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

Parameters:

UG_COLOR c Forecolor

Example:

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

4.22 UG_SetBackcolor()

Defines the backcolor of the string.

Prototype:

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

Parameters:

UG_COLOR c Backcolor

Example:

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

4.23 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( );
    // ...
}
```

4.24 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( );
    // ...
}
```

4.25 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 );
    // ...
}
```

4.26 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 );
    // ...
}
```

5 Revision history

5.1 Software

Software Version	Date	Description
v0.2	Oct 20, 2014	<ul style="list-style-type: none">■ Function <code>UG_DrawRoundFrame()</code> added.■ Function <code>UG_FillRoundFrame()</code> added.■ Function <code>UG_DrawArc()</code> added.■ Fixed some minor bugs.
v0.1	Oct 11, 2014	<ul style="list-style-type: none">■ First release.

5.2 Document

Document Version	Date	Description
v0.2	Oct 20, 2014	<ul style="list-style-type: none">■ 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	<ul style="list-style-type: none">■ First release.