

microui

User Manual



MICROEJ[®]

Reference:	TLT-XXX-MAN-microui-microui
Version:	14.5.5
Revision:	XXX

Confidentiality & Intellectual Property

All rights reserved. Information, technical data and tutorials contained in this document are confidential and proprietary under copyright Law of Industrial Smart Software Technology (IS2T S.A.) operating under the brand name MicroEJ®. Without written permission from IS2T S.A., *copying or sending parts of the document or the entire document by any means to third parties is not permitted*. Granted authorizations for using parts of the document or the entire document do not mean IS2T S.A. gives public full access rights.

The information contained herein is not warranted to be error-free. IS2T® and MicroEJ® and all relative logos are trademarks or registered trademarks of IS2T S.A. in France and other Countries.

Java™ is Sun Microsystems' trademark for a technology for developing application software and deploying it in cross-platform, networked environments. When it is used in this documentation without adding the ™ symbol, it includes implementations of the technology by companies other than Sun.

Java™, all Java-based marks and all related logos are trademarks or registered trademarks of Sun Microsystems Inc, in the United States and other Countries.

Other trademarks are proprietary of their authors.

Table of Contents

1. Data Structure Documentation	1
1.1. ui_rect_collection_t Struct Reference	1
1.1.1. Detailed Description	1
2. File Documentation	2
2.1. bsp/ui/inc/microui_event_decoder.h File Reference	2
2.1.1. Detailed Description	2
2.2. bsp/ui/inc/microui_heap.h File Reference	2
2.2.1. Detailed Description	2
2.3. bsp/ui/inc/ui_color.h File Reference	2
2.3.1. Detailed Description	4
2.3.2. Macro Definition Documentation	4
2.4. bsp/ui/inc/ui_configuration.h File Reference	6
2.4.1. Detailed Description	8
2.4.2. Macro Definition Documentation	8
2.5. bsp/ui/inc/ui_display_brs.h File Reference	11
2.5.1. Detailed Description	11
2.5.2. Function Documentation	12
2.6. bsp/ui/inc/ui_drawing_stub.h File Reference	13
2.6.1. Detailed Description	17
2.6.2. Function Documentation	17
2.7. bsp/ui/inc/ui_rect_util.h File Reference	49
2.7.1. Detailed Description	49
2.7.2. Function Documentation	49
2.8. bsp/ui/inc/ui_util.h File Reference	54
2.8.1. Detailed Description	54
2.9. bsp/ui/src/LLUI_DISPLAY_HEAP_impl.c File Reference	54
2.9.1. Detailed Description	54
2.10. bsp/ui/src/LLUI_INPUT_LOG_impl.c File Reference	54
2.10.1. Detailed Description	55
2.11. bsp/ui/src/microui_event_decoder.c File Reference	55
2.11.1. Detailed Description	55
Index	56

List of Tables

2.1.	4
2.2.	5
2.3.	5
2.4.	12
2.5.	18
2.6.	19
2.7.	19
2.8.	20
2.9.	21
2.10.	22
2.11.	22
2.12.	23
2.13.	24
2.14.	25
2.15.	25
2.16.	26
2.17.	27
2.18.	28
2.19.	28
2.20.	29
2.21.	30
2.22.	31
2.23.	32
2.24.	33
2.25.	33
2.26.	34
2.27.	35
2.28.	36
2.29.	36
2.30.	37
2.31.	38
2.32.	39
2.33.	40
2.34.	41
2.35.	42
2.36.	42
2.37.	43
2.38.	44
2.39.	44
2.40.	45
2.41.	46
2.42.	46
2.43.	47
2.44.	48

2.45.	48
2.46.	50
2.47.	50
2.48.	51

Chapter 1. Data Structure Documentation

1.1. ui_rect_collection_t Struct Reference

Exposes the ui_rect_collection_t type that handles unordered ui_rect_t collections.

```
#include <ui_rect_collection.h>
```

Data Fields

- ui_rect_t [UI_RECT_COLLECTION_MAX_LENGTH]
- size_t

1.1.1. Detailed Description

Exposes the ui_rect_collection_t type that handles unordered ui_rect_t collections.

A rectangle collection contains an array of rectangles and a pointer to the latest rectangle.

Definition at line 36 of file ui_rect_collection.h.

The documentation for this struct was generated from the following file:
bsp/ui/inc/ui_rect_collection.h

Chapter 2. File Documentation

2.1. bsp/ui/inc/microui_event_decoder.h File Reference

See microui_event_decoder.c.

```
#include "ui_configuration.h"
```

2.1.1. Detailed Description

See microui_event_decoder.c.

Author. MicroEJ Developer Team

Version. 14.5.5

Since. MicroEJ UI Pack 13.1.0

2.2. bsp/ui/inc/microui_heap.h File Reference

Provides some API to analyse the MicroUI image heap. The default allocator available in the Graphics Engine does not implement this file contrary to the allocator "LLUI_DISPLAY_HEAP_impl.c".

```
#include "ui_configuration.h"
```

2.2.1. Detailed Description

Provides some API to analyse the MicroUI image heap. The default allocator available in the Graphics Engine does not implement this file contrary to the allocator "LLUI_DISPLAY_HEAP_impl.c".

See also. The option UI_FEATURE_ALLOCATOR in ui_configuration.h

Author. MicroEJ Developer Team

Version. 14.5.5

Since. MicroEJ UI Pack 13.1.0

2.3. bsp/ui/inc/ui_color.h File Reference

Provides macros to manipulate the color channels of the ARGB8888 pixel format.

Macros

- `#define COLOR_ARGB8888_CHANNEL_SIZE 8`
Mask of a channel in ARGB8888 format.
- `#define COLOR_ARGB8888_CHANNEL_MASK ((1 << COLOR_ARGB8888_CHANNEL_SIZE) - 1)`
Mask of a channel in ARGB8888 format.
- `#define COLOR_CHANNEL_MASK(format, channel) ((1 << COLOR_CHANNEL_SIZE(format, channel)) - 1)`
Mask of a channel.
- `#define COLOR_CHANNEL_SIZE(format, channel) COLOR_ ## format ## _ ## channel ## _SIZE`
Size of a channel.
- `#define COLOR_CHANNEL_OFFSET(format, channel) COLOR_ ## format ## _ ## channel ## _OFFSET`
Offset of a channel.
- `#define COLOR_ARGB8888_ALPHA_SIZE COLOR_ARGB8888_CHANNEL_SIZE`
Alpha channel size for ARGB8888 format.
- `#define COLOR_ARGB8888_ALPHA_OFFSET 24`
Alpha channel offset for ARGB8888 format.
- `#define COLOR_ARGB8888_ALPHA_MASK COLOR_CHANNEL_MASK(ARGB8888, ALPHA)`
Alpha channel mask for ARGB8888 format.
- `#define COLOR_ARGB8888_RED_SIZE COLOR_ARGB8888_CHANNEL_SIZE`
Red channel size for ARGB8888 format.
- `#define COLOR_ARGB8888_RED_OFFSET 16`
Red channel offset for ARGB8888 format.
- `#define COLOR_ARGB8888_RED_MASK COLOR_CHANNEL_MASK(ARGB8888, RED)`
Red channel mask for ARGB8888 format.
- `#define COLOR_ARGB8888_GREEN_SIZE COLOR_ARGB8888_CHANNEL_SIZE`
Green channel size for ARGB8888 format.
- `#define COLOR_ARGB8888_GREEN_OFFSET 8`
Green channel offset for ARGB8888 format.
- `#define COLOR_ARGB8888_GREEN_MASK COLOR_CHANNEL_MASK(ARGB8888, GREEN)`

Green channel mask for ARGB8888 format.

- `#define COLOR_ARGB8888_BLUE_SIZE COLOR_ARGB8888_CHANNEL_SIZE`

Blue channel size for ARGB8888 format.

- `#define COLOR_ARGB8888_BLUE_OFFSET 0`

Blue channel offset for ARGB8888 format.

- `#define COLOR_ARGB8888_BLUE_MASK COLOR_CHANNEL_MASK(ARGB8888, BLUE)`

Blue channel mask for ARGB8888 format.

- `#define COLOR_GET_CHANNEL(color, format, channel)`

Gets a channel value from a color.

- `#define COLOR_SET_COLOR(alpha, red, green, blue, format)`

Sets a color.

2.3.1. Detailed Description

Provides macros to manipulate the color channels of the ARGB8888 pixel format.

Author. MicroEJ Developer Team

Version. 14.5.5

2.3.2. Macro Definition Documentation

2.3.2.1. COLOR_CHANNEL_MASK

```
#define COLOR_CHANNEL_MASK( format, channel) ((1 <<
COLOR_CHANNEL_SIZE(format, channel)) - 1)
```

Mask of a channel.

Parameters .

Table 2.1.

in	format	The color format
in	channel	The channel

Definition at line 43 of file ui_color.h.

2.3.2.2. COLOR_CHANNEL_OFFSET

```
#define COLOR_CHANNEL_OFFSET( format, channel) COLOR_ ## format ## _ ##  
channel ## _OFFSET
```

Offset of a channel.

Parameters .

Table 2.2.

in	format	The color format
in	channel	The channel

Definition at line 61 of file ui_color.h.

2.3.2.3. COLOR_CHANNEL_SIZE

```
#define COLOR_CHANNEL_SIZE( format, channel) COLOR_ ## format ## _ ##  
channel ## _SIZE
```

Size of a channel.

Parameters .

Table 2.3.

in	format	The color format
in	channel	The channel

Definition at line 52 of file ui_color.h.

2.3.2.4. COLOR_GET_CHANNEL

```
#define COLOR_GET_CHANNEL( color, format, channel)
```

```
(uint8_t)(  
    ((color) >> COLOR_CHANNEL_OFFSET(format, channel)) \  
    &COLOR_CHANNEL_MASK(format, channel)                \  
)
```

Gets a channel value from a color.

param color: The color from which to retrieve a channel param format: The color format (i.e. ARGB8888, etc...)

Definition at line 134 of file ui_color.h.

2.3.2.5. COLOR_SET_COLOR

```
#define COLOR_SET_COLOR( alpha, red, green, blue, format)
```

```

(uint32_t)(0
| (((alpha)&COLOR_ ## format ## _ALPHA_MASK) \
<< (COLOR_ ## format ## _ALPHA_OFFSET)) \
| (((red)&COLOR_ ## format ## _RED_MASK) \
<< (COLOR_ ## format ## _RED_OFFSET)) \
| (((green)&COLOR_ ## format ## _GREEN_MASK) \
<< (COLOR_ ## format ## _GREEN_OFFSET)) \
| (((blue)&COLOR_ ## format ## _BLUE_MASK) \
<< (COLOR_ ## format ## _BLUE_OFFSET)) \
)

```

Sets a color.

param alpha: The alpha channel value param red: The red channel value param green: The green channel value param blue: The blue channel value param format: The color format (i.e. ARGB8888, etc...)

Definition at line 149 of file ui_color.h.

2.4. bsp/ui/inc/ui_configuration.h File Reference

MicroEJ MicroUI library low level API: enable some features according to the hardware capabilities.

```

#include "veeport_configuration.h"
#include <stdio.h>

```

Macros

- #define UI_FEATURE_ALLOCATOR_GRAPHICS_ENGINE (0)

Value of "UI_FEATURE_ALLOCATOR" to use the default allocator available in the Graphics Engine (a best fit allocator that does not provide any functions to analyse its content).

- #define UI_FEATURE_ALLOCATOR_BESTFIT (1)

Value of "UI_FEATURE_ALLOCATOR" to use the allocator of the file LLUI_DISPLAY_HEAP_impl.c. This implementation uses a best fit allocator and features some additional APIs (see microui_heap.h) to analyse the allocation in the MicroUI image heap.

- #define UI_FEATURE_ALLOCATOR_CUSTOM (2)

Value of "UI_FEATURE_ALLOCATOR" to use a 3rd-party allocator. The implementation of the functions LLUI_DISPLAY_IMPL_imageHeap*() is required; otherwise the default allocator (see UI_FEATURE_ALLOCATOR_GRAPHICS_ENGINE) is used.

- #define UI_FEATURE_BRS_LEGACY (0)

Value of "UI_FEATURE_BRS". Defines the "legacy" strategy (equivalent to the one available in UI Packs 13.x) dedicated to the buffer policy SWAP (see MicroEJ documentation).

- #define UI_FEATURE_BRS_SINGLE (1)

Value of "UI_FEATURE_BRS". Defines the "single" strategy dedicated to the display with one buffer on the MCU side and one buffer on the display side (the front buffer is not mapped to the MCU address space). The refresh consists in transmitting the back buffer data to the front buffer. The content to transmit is a list of rectangles that have been modified since last flush or an unique rectangle that encapsulates all regions (see the option UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE).

- #define UI_FEATURE_BRS_PREDRAW (2)

Value of "UI_FEATURE_BRS". Defines the "predraw" strategy that consists in restoring the back buffer with the content of the previous drawings (the past) just before the very first drawing after a flush (and not just after the flush).

- #define UI_DEBUG_PRINT (void)printf

Standard "printf" indirection.

- #define UI_RECT_COLLECTION_MAX_LENGTH (8u)

Defines the number of rectangles that collections can contain.

- #define UI_FEATURE_ALLOCATOR (UI_FEATURE_ALLOCATOR_GRAPHICS_ENGINE)

Set this define to select the image heap allocator.

- #define UI_FEATURE_EVENT_DECODER (0)

Set this define to (1) to enable the MicroUI events logger. The call to LLUI_INPUT_dump() has no effect when the logger is disabled.

- #define UI_FEATURE_BRS (UI_FEATURE_BRS_PREDRAW)

Defines the display buffer refresh strategy (BRS) to use: one of the strategies above, the Graphics Engine's default refresh strategy or a BSP's custom refresh strategy.

- #define UI_FEATURE_BRS_DRAWING_BUFFER_COUNT (2u)

Defines the available number of drawing buffers; in other words, the number of buffers the Graphics Engine can use to draw into. According to the display BRS and the LCD connection, a drawing buffer can be alternatively a back buffer (the buffer currently used by the Graphics Engine), a front buffer (the buffer currently used by the LCD driver to map the LCD device), a transmission buffer (the buffer currently used by the LCD driver to transmit the data to the LCD device) or a free buffer (a buffer currently unused).

- #define UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE (0)

Defines the number of rectangles the strategy uses when calling LLUI_DISPLAY_IMPL_flush(). When set to 1, the strategy sends only one rectangle that includes all dirty regions (a rectangle that encapsulates all rectangles). Otherwise, the strategy sends all rectangles.

- #define UI_GC_SUPPORTED_FORMATS (1u)

Defines the number of supported destination formats. When not set or smaller than "2", the file `ui_drawing.c` considers only one destination format is available: the same format as the buffer of the display.

- `#define UI_FEATURE_IMAGE_CUSTOM_FORMATS (0)`

When set to 1, in addition to the standard image formats (ARGB8888, A8, etc), the VEE Port can support one or several custom formats.

- `#define UI_FEATURE_FONT_CUSTOM_FORMATS (0)`

When set to 1, in addition to the graphics engine's internal font format, the VEE Port can support one or several custom formats.

2.4.1. Detailed Description

MicroEJ MicroUI library low level API: enable some features according to the hardware capabilities.

Author. MicroEJ Developer Team

2.4.2. Macro Definition Documentation

2.4.2.1. UI_FEATURE_BRS

```
#define UI_FEATURE_BRS (UI_FEATURE_BRS_PREDRAW)
```

Defines the display buffer refresh strategy (BRS) to use: one of the strategies above, the Graphics Engine's default refresh strategy or a BSP's custom refresh strategy.

When not set, the BSP has to implement the `LLUI_DISPLAY_impl.h`'s API to define its custom display buffer refresh strategy. If no strategy is set, the default Graphics Engine's strategy is used that consists to only call `LLUI_DISPLAY_IMPL_flush()` (always full screen, no restore).

Definition at line 196 of file `ui_configuration.h`.

2.4.2.2. UI_FEATURE_BRS_DRAWING_BUFFER_COUNT

```
#define UI_FEATURE_BRS_DRAWING_BUFFER_COUNT (2u)
```

Defines the available number of drawing buffers; in other words, the number of buffers the Graphics Engine can use to draw into. According to the display BRS and the LCD connection, a drawing buffer can be alternatively a back buffer (the buffer currently used by the Graphics Engine), a front buffer (the buffer currently used by the LCD driver to map the LCD device), a transmission buffer (the buffer currently used by the LCD driver to transmit the data to the LCD device) or a free buffer (a buffer currently unused).

Warning: This counter only defines the buffers the Graphics Engine can use and not the buffer reserved to the LCD device (mapped or not on the MCU address space).

Definition at line 210 of file ui_configuration.h.

2.4.2.3. UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE

```
#define UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE (0)
```

Defines the number of rectangles the strategy uses when calling `LLUI_DISPLAY_IMPL_flush()`. When set to 1, the strategy sends only one rectangle that includes all dirty regions (a rectangle that encapsulates all rectangles). Otherwise, the strategy sends all rectangles.

When the implementation of `LLUI_DISPLAY_IMPL_flush()` only consists in swapping the back buffers, the rectangles list is useless.

When the implementation of `LLUI_DISPLAY_IMPL_flush()` consists in transmitting an unique portion of the back buffer (for instance: the back buffer is transmitted to the LCD through a DSI bus), the single rectangle mode is useful.

When the implementation of `LLUI_DISPLAY_IMPL_flush()` consists in transmitting several portions of the back buffer (for instance: the back buffer is transmitted to the LCD through a SPI bus), the rectangle list is useful.

By default, the rectangle list is given as-is (the option is not enabled).

See also. the strategies' comments to have more information on the use of this option.

Definition at line 231 of file ui_configuration.h.

2.4.2.4. UI_FEATURE_BRS_PREDRAW

```
#define UI_FEATURE_BRS_PREDRAW (2)
```

Value of "UI_FEATURE_BRS". Defines the "predraw" strategy that consists in restoring the back buffer with the content of the previous drawings (the past) just before the very first drawing after a flush (and not just after the flush).

This strategy requires the available number of back buffers (i.e. the number of buffers where the drawings can be performed) to be defined:

```
#define UI_FEATURE_BRS_DRAWING_BUFFER_COUNT (2u)
```

This strategy manages, by default, two or three back buffers. Its implementation can be easily modified to manage more back buffers. An error is thrown if the code is not modified and the number of back buffers is higher than three.

A warning is thrown if there is only one back buffer because this strategy is optimized for two or more back buffers. However, this strategy works with only one buffer but it is a little bit too complex for this use case.

Notes:

- The maximum number of rectangles given as parameter to the function `LLUI_DISPLAY_IMPL_flush()` is equal to `UI_RECT_COLLECTION_MAX_LENGTH`.
- This BRS takes in consideration the option `UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE`.

Definition at line 102 of file `ui_configuration.h`.

2.4.2.5. UI_FEATURE_BRS_SINGLE

```
#define UI_FEATURE_BRS_SINGLE (1)
```

Value of "UI_FEATURE_BRS". Defines the "single" strategy dedicated to the display with one buffer on the MCU side and one buffer on the display side (the front buffer is not mapped to the MCU address space). The refresh consists in transmitting the back buffer data to the front buffer. The content to transmit is a list of rectangles that have been modified since last flush or an unique rectangle that encapsulates all regions (see the option `UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE`).

Note: the maximum number of rectangles given as parameter to the function `LLUI_DISPLAY_IMPL_flush()` is equal to `UI_RECT_COLLECTION_MAX_LENGTH`.

Definition at line 78 of file `ui_configuration.h`.

2.4.2.6. UI_FEATURE_EVENT_DECODER

```
#define UI_FEATURE_EVENT_DECODER (0)
```

Set this define to (1) to enable the MicroUI events logger. The call to `LLUI_INPUT_dump()` has no effect when the logger is disabled.

By default the logger is not enabled.

Definition at line 137 of file `ui_configuration.h`.

2.4.2.7. UI_FEATURE_FONT_CUSTOM_FORMATS

```
#define UI_FEATURE_FONT_CUSTOM_FORMATS (0)
```

When set to 1, in addition to the graphics engine's internal font format, the VEE Port can support one or several custom formats.

See also. `ui_font_drawing.c` for more information.

Definition at line 258 of file `ui_configuration.h`.

2.4.2.8. UI_FEATURE_IMAGE_CUSTOM_FORMATS

```
#define UI_FEATURE_IMAGE_CUSTOM_FORMATS (0)
```

When set to 1, in addition to the standard image formats (ARGB8888, A8, etc), the VEE Port can support one or several custom formats.

See also. `ui_image_drawing.c` for more information.

Definition at line 249 of file `ui_configuration.h`.

2.4.2.9. UI_GC_SUPPORTED_FORMATS

```
#define UI_GC_SUPPORTED_FORMATS (1u)
```

Defines the number of supported destination formats. When not set or smaller than "2", the file `ui_drawing.c` considers only one destination format is available: the same format as the buffer of the display.

See also. `ui_drawing.c` for more information.

Definition at line 240 of file `ui_configuration.h`.

2.5. bsp/ui/inc/ui_display_brs.h File Reference

Provides some implementations of the `LLUI_DISPLAY_impl.h` functions relating to the display buffer strategy (BRS).

```
#include <LLUI_DISPLAY_impl.h>
#include "ui_util.h"
#include "ui_configuration.h"
#include "ui_trace.h"
```

Functions

- `DRAWING_Status UI_DISPLAY_BRS_restore (MICROUI_GraphicsContext *gc, MICROUI_Image *old_back_buffer, ui_rect_t *rect)`

Restores (copies) the given rectangular region from old back buffer to the graphics context's current buffer (the destination buffer can be retrieved thanks `LLUI_DISPLAY_getBufferAddress(&gc->image)`).

2.5.1. Detailed Description

Provides some implementations of the `LLUI_DISPLAY_impl.h` functions relating to the display buffer strategy (BRS).

A BRS

- implements the `LLUI_DISPLAY_impl.h`'s API `LLUI_DISPLAY_IMPL_refresh()` and `LLUI_DISPLAY_IMPL_newDrawingRegion()`.
- calls the `LLUI_DISPLAY_impl.h`'s API `LLUI_DISPLAY_IMPL_flush()` to let the display driver transmit / copy / flush back buffer data to the front buffer or swap back and front buffer (double or triple buffer management).

- ensures the coherence of the back buffer's content before and after a flush.

Several BRS are available in `ui_configuration.h` (the implementation has to select one of them).

Author. MicroEJ Developer Team

Version. 14.5.5

2.5.2. Function Documentation

2.5.2.1. UI_DISPLAY_BRS_restore()

```
DRAWING_Status UI_DISPLAY_BRS_restore (MICROUI_GraphicsContext * gc,  
MICROUI_Image * old_back_buffer, ui_rect_t * rect)
```

Restores (copies) the given rectangular region from old back buffer to the graphics context's current buffer (the destination buffer can be retrieved thanks `LLUI_DISPLAY_getBufferAddress(&gc->image)`).

When the copy is synchronous (immediate), the implementation has to return `DRAWING_DONE`. When the copy is asynchronous (performed by a DMA for instance), the the implementation has to return `DRAWING_RUNNING`. As the end of the asynchronous copy, the implementation has to call `LLUI_DISPLAY_notifyAsynchronousDrawingEnd()` to unlock the caller of this function.

The implementation of this function is optional; a weak function provides a simple implementation using `memcpy()`.

Parameters .

Table 2.4.

in	gc	the MicroUI GraphicsContext that targets the current back buffer
in	old_back_buffer	a MicroUI Image that symbolizes (targets) the old back buffer.
in	rect	the rectangular region to copy from old back buffer to new back buffer.

Returns. the drawing status.

Definition at line 33 of file `ui_display_brs.c`.

```

34      {
35          DRAWING_Status ret;
36          uint32_t bpp = LLUI_DISPLAY_getImageBPP(old_back_buffer);
37          uint32_t width = UI_RECT_get_width(rect);
38          uint32_t height = UI_RECT_get_height(rect);
39
40          (8u > bpp) {
41
42
43              ret = UI_DRAWING_copyImage(gc, old_back_buffer, rect->x1,
44              rect->y1, width, height, rect->x1, rect->y1);
45          } {
46              uint8_t *dst = LLUI_DISPLAY_getBufferAddress(&gc->image);
47              uint8_t *src =
48              LLUI_DISPLAY_getBufferAddress(old_back_buffer);
49              uint32_t stride = LLUI_DISPLAY_getStrideInBytes(&gc-
50              >image);
51
52              dst += rect->y1 * stride;
53              src += rect->y1 * stride;
54
55              ((0 == rect->x1) && (old_back_buffer->width == width)) {
56
57                  (void)memcpy(dst, src, stride * height);
58              } {
59
60                  dst += rect->x1 * bpp / 8u;
61                  src += rect->x1 * bpp / 8u;
62                  uint32_t size = width * bpp / 8u;
63
64                  (uint32_t y = 0; y < height; y++) {
65                      (void)memcpy(dst, src, size);
66                      dst += stride;
67                      src += stride;
68                  }
69              }
70
71              LLUI_DISPLAY_IMPL_notifyDrawingSoft(gc);
72
73              ret = DRAWING_DONE;
74          }
75      }
76      ret;
77 }

```

2.6. bsp/ui/inc/ui_drawing_stub.h File Reference

Implementation of all drawing functions of ui_drawing.h (MicroUI and Drawing libraries). These are stubbed implementations, there is no error but the drawings are not performed. Useful to stub a drawing on a custom destination (custom GraphicsContext format not supported by the Graphics Engine).

```
#include "ui_drawing.h"
```

Functions

- DRAWING_Status UI_DRAWING_STUB_writePixel (MICROUI_GraphicsContext *gc, jint x, jint y)
Stubbed implementation of writePixel. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawLine (MICROUI_GraphicsContext *gc, jint startX, jint startY, jint endX, jint endY)
Stubbed implementation of drawLine. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawHorizontalLine (MICROUI_GraphicsContext *gc, jint x1, jint x2, jint y)
Stubbed implementation of drawHorizontalLine. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawVerticalLine (MICROUI_GraphicsContext *gc, jint x, jint y1, jint y2)
Stubbed implementation of drawVerticalLine. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawRectangle (MICROUI_GraphicsContext *gc, jint x1, jint y1, jint x2, jint y2)
Stubbed implementation of drawRectangle. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_fillRectangle (MICROUI_GraphicsContext *gc, jint x1, jint y1, jint x2, jint y2)
Stubbed implementation of fillRectangle. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawRoundedRectangle (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jint cornerEllipseWidth, jint cornerEllipseHeight)
Stubbed implementation of drawRoundedRectangle. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_fillRoundedRectangle (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jint cornerEllipseWidth, jint cornerEllipseHeight)
Stubbed implementation of fillRoundedRectangle. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawCircleArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle)
Stubbed implementation of drawCircleArc. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_drawEllipseArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jfloat startAngle, jfloat arcAngle)
Stubbed implementation of drawEllipseArc. See ui_drawing.h.
- DRAWING_Status UI_DRAWING_STUB_fillCircleArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle)

Stubbed implementation of fillCircleArc. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_fillEllipseArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jfloat startAngle, jfloat arcAngle)

Stubbed implementation of fillEllipseArc. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawEllipse (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height)

Stubbed implementation of drawEllipse. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_fillEllipse (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height)

Stubbed implementation of fillEllipse. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawCircle (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter)

Stubbed implementation of drawCircle. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_fillCircle (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter)

Stubbed implementation of fillCircle. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawString (MICROUI_GraphicsContext *gc, jchar *chars, jint length, MICROUI_Font *font, jint x, jint y)

Stub implementation of drawString. See ui_drawing.h.

- jint UI_DRAWING_STUB_stringWidth (jchar *chars, jint length, MICROUI_Font *font)

Stub implementation of stringWidth. See ui_drawing.h.

- jint UI_DRAWING_STUB_initializeRenderableStringSNContext (jchar *chars, jint length, MICROUI_Font *font, MICROUI_RenderableString *renderableString)

Stub implementation of initializeRenderableStringSNContext. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawRenderableString (MICROUI_GraphicsContext *gc, jchar *chars, jint length, MICROUI_Font *font, jint width, MICROUI_RenderableString *renderableString, jint x, jint y)

Stub implementation of drawRenderableString. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawImage (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint regionX, jint regionY, jint width, jint height, jint x, jint y, jint alpha)

Stubbed implementation of drawImage. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_copyImage (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint regionX, jint regionY, jint width, jint height, jint x, jint y)

Stubbed implementation of copyImage. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawRegion (MICROUI_GraphicsContext *gc, jint regionX, jint regionY, jint width, jint height, jint x, jint y, jint alpha)

Stubbed implementation of drawRegion. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickFadedPoint (MICROUI_GraphicsContext *gc, jint x, jint y, jint thickness, jint fade)

Stubbed implementation of drawThickFadedPoint. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickFadedLine (MICROUI_GraphicsContext *gc, jint startX, jint startY, jint endX, jint endY, jint thickness, jint fade, DRAWING_Cap startCap, DRAWING_Cap endCap)

Stubbed implementation of drawThickFadedLine. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickFadedCircle (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jint thickness, jint fade)

Stubbed implementation of drawThickFadedCircle. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickFadedCircleArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle, jint thickness, jint fade, DRAWING_Cap start, DRAWING_Cap end)

Stubbed implementation of drawThickFadedCircleArc. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickFadedEllipse (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jint thickness, jint fade)

Stubbed implementation of drawThickFadedEllipse. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickLine (MICROUI_GraphicsContext *gc, jint startX, jint startY, jint endX, jint endY, jint thickness)

Stubbed implementation of drawThickLine. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickCircle (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jint thickness)

Stubbed implementation of drawThickCircle. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickEllipse (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height, jint thickness)

Stubbed implementation of drawThickEllipse. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawThickCircleArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle, jint thickness)

Stubbed implementation of drawThickCircleArc. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawFlippedImage (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint regionX, jint regionY, jint width, jint height, jint x, jint y, DRAWING_Flip transformation, jint alpha)

Stubbed implementation of drawFlippedImage. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawRotatedImageNearestNeighbor (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint x, jint y, jint rotationX, jint rotationY, jfloat angle, jint alpha)

Stubbed implementation of drawRotatedImageNearestNeighbor. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawRotatedImageBilinear (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint x, jint y, jint rotationX, jint rotationY, jfloat angle, jint alpha)

Stubbed implementation of drawRotatedImageBilinear. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawScaledImageNearestNeighbor (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint x, jint y, jfloat factorX, jfloat factorY, jint alpha)

Stubbed implementation of drawScaledImageNearestNeighbor. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawScaledImageBilinear (MICROUI_GraphicsContext *gc, MICROUI_Image *img, jint x, jint y, jfloat factorX, jfloat factorY, jint alpha)

Stubbed implementation of drawScaledImageBilinear. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawScaledStringBilinear (MICROUI_GraphicsContext *gc, jchar *chars, jint length, MICROUI_Font *font, jint x, jint y, jfloat xRatio, jfloat yRatio)

Stubbed implementation of drawScaledStringBilinear. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawScaledRenderableStringBilinear (MICROUI_GraphicsContext *gc, jchar *chars, jint length, MICROUI_Font *font, jint width, MICROUI_RenderableString *renderableString, jint x, jint y, jfloat xRatio, jfloat yRatio)

Stubbed implementation of drawScaledRenderableStringBilinear. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawCharWithRotationBilinear (MICROUI_GraphicsContext *gc, jchar c, MICROUI_Font *font, jint x, jint y, jint xRotation, jint yRotation, jfloat angle, jint alpha)

Stubbed implementation of drawCharWithRotationBilinear. See ui_drawing.h.

- DRAWING_Status UI_DRAWING_STUB_drawCharWithRotationNearestNeighbor (MICROUI_GraphicsContext *gc, jchar c, MICROUI_Font *font, jint x, jint y, jint xRotation, jint yRotation, jfloat angle, jint alpha)

Stubbed implementation of drawCharWithRotationNearestNeighbor. See ui_drawing.h.

2.6.1. Detailed Description

Implementation of all drawing functions of ui_drawing.h (MicroUI and Drawing libraries). These are stubbed implementations, there is no error but the drawings are not performed. Useful to stub a drawing on a custom destination (custom GraphicsContext format not supported by the Graphics Engine).

Author. MicroEJ Developer Team

Version. 14.5.5

2.6.2. Function Documentation

2.6.2.1. UI_DRAWING_STUB_copyImage()

```
DRAWING_Status UI_DRAWING_STUB_copyImage (MICROUI_GraphicsContext * gc,
MICROUI_Image * img, jint regionX, jint regionY, jint width, jint height,
jint x, jint y)
```

Stubbed implementation of copyImage. See ui_drawing.h.

Parameters .

Table 2.5.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to copy.
in	regionX	the x coordinate of the upper-left corner of the region to copy.
in	regionY	the y coordinate of the upper-left corner of the region to copy.
in	width	the width of the region to copy.
in	height	the height of the region to copy.
in	x	the x coordinate of the top-left point in the destination.
in	y	the y coordinate of the top-left point in the destination.

Returns. the drawing status.

Definition at line 262 of file ui_drawing_stub.c.

```
263
    {
264     (void)gc;
265     (void)img;
266     (void)regionX;
267     (void)regionY;
268     (void)width;
269     (void)height;
270     (void)x;
271     (void)y;
272     not_implemented(gc);
273 }
```

2.6.2.2. UI_DRAWING_STUB_drawCharWithRotationBilinear()

```
DRAWING_Status UI_DRAWING_STUB_drawCharWithRotationBilinear
(MICROUI_GraphicsContext * gc, jchar c, MICROUI_Font * font, jint x,
jint y, jint xRotation, jint yRotation, jfloat angle, jint alpha)
```

Stubbed implementation of drawCharWithRotationBilinear. See ui_drawing.h.

Parameters .

Table 2.6.

in	gc	the targeted MicroUI GraphicsContext
in	c	a character
in	font	the MicroUI font to use
in	x	the left coordinate
in	y	the top coordinate
in	xRotation	the horizontal coordinate of the rotation center
in	yRotation	the vertical coordinate of the rotation center
in	angle	the rotation angle, in degrees
in	alpha	the alpha value to use

Returns. the drawing status.

Definition at line 504 of file ui_drawing_stub.c.

```

506
507     {
508         (void)gc;
509         (void)c;
510         (void)font;
511         (void)x;
512         (void)y;
513         (void)xRotation;
514         (void)yRotation;
515         (void)angle;
516         (void)alpha;
517         not_implemented(gc);
518     }

```

2.6.2.3. UI_DRAWING_STUB_drawCharWithRotationNearestNeighbor()

```

DRAWING_Status      UI_DRAWING_STUB_drawCharWithRotationNearestNeighbor
(MICROUI_GraphicsContext * gc, jchar c, MICROUI_Font * font, jint x,
jint y, jint xRotation, jint yRotation, jfloat angle, jint alpha)

```

Stubbed implementation of drawCharWithRotationNearestNeighbor. See ui_drawing.h.

Parameters .

Table 2.7.

in	gc	the targeted MicroUI GraphicsContext
in	c	a character
in	font	the MicroUI font to use

in	x	the left coordinate
in	y	the top coordinate
in	xRotation	the horizontal coordinate of the rotation center
in	yRotation	the vertical coordinate of the rotation center
in	angle	the rotation angle, in degrees
in	alpha	the alpha value to use

Returns. the drawing status.

Definition at line 520 of file ui_drawing_stub.c.

```

522                                     {
523     (void)gc;
524     (void)c;
525     (void)font;
526     (void)x;
527     (void)y;
528     (void)xRotation;
529     (void)yRotation;
530     (void)angle;
531     (void)alpha;
532     not_implemented(gc);
533 }
```

2.6.2.4. UI_DRAWING_STUB_drawCircle()

```
DRAWING_Status UI_DRAWING_STUB_drawCircle (MICROUI_GraphicsContext *
gc, jint x, jint y, jint diameter)
```

Stubbed implementation of drawCircle. See ui_drawing.h.

Parameters .

Table 2.8.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	diameter	the circle square size.

Returns. the drawing status.

Definition at line 193 of file ui_drawing_stub.c.

```

193                                     {
194     (void)gc;
195     (void)x;
196     (void)y;
197     (void)diameter;
198     not_implemented(gc);
199 }

```

2.6.2.5. UI_DRAWING_STUB_drawCircleArc()

DRAWING_Status UI_DRAWING_STUB_drawCircleArc (MICROUI_GraphicsContext *gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle)

Stubbed implementation of drawCircleArc. See ui_drawing.h.

Parameters .

Table 2.9.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	diameter	the diameter of the arc to draw.
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle

Returns. the drawing status.

Definition at line 123 of file ui_drawing_stub.c.

```

124                                     {
125     (void)gc;
126     (void)x;
127     (void)y;
128     (void)diameter;
129     (void)startAngle;
130     (void)arcAngle;
131     not_implemented(gc);
132 }

```

2.6.2.6. UI_DRAWING_STUB_drawEllipse()

DRAWING_Status UI_DRAWING_STUB_drawEllipse (MICROUI_GraphicsContext *gc, jint x, jint y, jint width, jint height)

Stubbed implementation of drawEllipse. See ui_drawing.h.

Parameters .

Table 2.10.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the ellipse width.
in	height	the ellipse height.

Returns. the drawing status.

Definition at line 173 of file ui_drawing_stub.c.

```

173                                     {
174     (void)gc;
175     (void)x;
176     (void)y;
177     (void)width;
178     (void)height;
179     not_implemented(gc);
180 }
```

2.6.2.7. UI_DRAWING_STUB_drawEllipseArc()

```
DRAWING_Status UI_DRAWING_STUB_drawEllipseArc (MICROUI_GraphicsContext
* gc, jint x, jint y, jint width, jint height, jfloat startAngle, jfloat
arcAngle)
```

Stubbed implementation of drawEllipseArc. See ui_drawing.h.

Parameters .

Table 2.11.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the rectangle width.
in	height	the rectangle height.
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle

Returns. the drawing status.

Definition at line 135 of file ui_drawing_stub.c.

```

136
137     {
138     (void)gc;
139     (void)x;
140     (void)y;
141     (void)width;
142     (void)height;
143     (void)startAngle;
144     (void)arcAngle;
145     not_implemented(gc);
146 }

```

2.6.2.8. UI_DRAWING_STUB_drawFlippedImage()

DRAWING_Status UI_DRAWING_STUB_drawFlippedImage
(MICROUI_GraphicsContext * gc, MICROUI_Image * img, jint regionX, jint
regionY, jint width, jint height, jint x, jint y, DRAWING_Flip trans-
formation, jint alpha)

Stubbed implementation of drawFlippedImage. See ui_drawing.h.

Parameters .

Table 2.12.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	regionX	the x coordinate of the upper-left corner of the region to draw.
in	regionY	the y coordinate of the upper-left corner of the region to draw.
in	width	the width of the region to copy.
in	height	the height of the region to copy.
in	x	the x coordinate of the top-left point in the destination.
in	y	the y coordinate of the top-left point in the destination.
in	transformation	the flip to apply.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 407 of file ui_drawing_stub.c.

```

409
410         {
411         (void)gc;
412         (void)img;
413         (void)regionX;
414         (void)regionY;
415         (void)width;
416         (void)height;
417         (void)x;
418         (void)y;
419         (void)transformation;
420         (void)alpha;
421         not_implemented(gc);
422     }

```

2.6.2.9. UI_DRAWING_STUB_drawHorizontalLine()

DRAWING_Status UI_DRAWING_STUB_drawHorizontalLine
(MICROUI_GraphicsContext * gc, jint x1, jint x2, jint y)

Stubbed implementation of drawHorizontalLine. See ui_drawing.h.

Parameters .

Table 2.13.

in	gc	the MicroUI GraphicsContext target.
in	x1	the first pixel line X coordinate.
in	x2	the last pixel line X coordinate.
in	y	the both pixels line Y coordinate.

Returns. the drawing status.

Definition at line 59 of file ui_drawing_stub.c.

```

59
60         {
61         (void)gc;
62         (void)x1;
63         (void)x2;
64         (void)y;
65         not_implemented(gc);
66     }

```

2.6.2.10. UI_DRAWING_STUB_drawImage()

DRAWING_Status UI_DRAWING_STUB_drawImage (MICROUI_GraphicsContext * gc,
MICROUI_Image * img, jint regionX, jint regionY, jint width, jint height,
jint x, jint y, jint alpha)

Stubbed implementation of drawImage. See ui_drawing.h.

Parameters .

Table 2.14.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	regionX	the x coordinate of the upper-left corner of the region to copy.
in	regionY	the y coordinate of the upper-left corner of the region to copy.
in	width	the width of the region to copy.
in	height	the height of the region to copy.
in	x	the x coordinate of the top-left point in the destination.
in	y	the y coordinate of the top-left point in the destination.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 247 of file ui_drawing_stub.c.

```

248
249         {
250     (void)gc;
251     (void)img;
252     (void)regionX;
253     (void)regionY;
254     (void)width;
255     (void)height;
256     (void)x;
257     (void)y;
258     (void)alpha;
259     not_implemented(gc);
260 }
```

2.6.2.11. UI_DRAWING_STUB_drawLine()

```
DRAWING_Status UI_DRAWING_STUB_drawLine (MICROUI_GraphicsContext * gc,
jint startX, jint startY, jint endX, jint endY)
```

Stubbed implementation of drawLine. See ui_drawing.h.

Parameters .

Table 2.15.

in	gc	the MicroUI GraphicsContext target.
in	startX	the first pixel line X coordinate.

in	startY	the first pixel line Y coordinate.
in	endX	the last pixel line X coordinate.
in	endY	the last pixel line Y coordinate.

Returns. the drawing status.

Definition at line 49 of file ui_drawing_stub.c.

```

49                                     {
50     (void)gc;
51     (void)startX;
52     (void)startY;
53     (void)endX;
54     (void)endY;
55     not_implemented(gc);
56 }
```

2.6.2.12. UI_DRAWING_STUB_drawRectangle()

DRAWING_Status UI_DRAWING_STUB_drawRectangle (MICROUI_GraphicsContext * gc, jint x1, jint y1, jint x2, jint y2)

Stubbed implementation of drawRectangle. See ui_drawing.h.

Parameters .

Table 2.16.

in	gc	the MicroUI GraphicsContext target.
in	x1	the top-left pixel X coordinate.
in	y1	the top-left pixel Y coordinate.
in	x2	the bottom-right pixel X coordinate.
in	y2	the bottom-right pixel Y coordinate.

Returns. the drawing status.

Definition at line 77 of file ui_drawing_stub.c.

```

77                                     {
78     (void)gc;
79     (void)x1;
80     (void)y1;
81     (void)x2;
82     (void)y2;
83     not_implemented(gc);
84 }
```

2.6.2.13. UI_DRAWING_STUB_drawRegion()

```
DRAWING_Status UI_DRAWING_STUB_drawRegion (MICROUI_GraphicsContext *
gc, jint regionX, jint regionY, jint width, jint height, jint x, jint
y, jint alpha)
```

Stubbed implementation of drawRegion. See ui_drawing.h.

Parameters .

Table 2.17.

in	gc	the MicroUI GraphicsContext source and target.
in	regionX	the x coordinate of the upper-left corner of the region to copy.
in	regionY	the y coordinate of the upper-left corner of the region to copy.
in	width	the width of the region to copy.
in	height	the height of the region to copy.
in	x	the x coordinate of the top-left point in the destination.
in	y	the y coordinate of the top-left point in the destination.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 276 of file ui_drawing_stub.c.

```
277
    {
278     (void)gc;
279     (void)regionX;
280     (void)regionY;
281     (void)width;
282     (void)height;
283     (void)x;
284     (void)y;
285     (void)alpha;
286     not_implemented(gc);
287 }
```

2.6.2.14. UI_DRAWING_STUB_drawRenderableString()

```
DRAWING_Status UI_DRAWING_STUB_drawRenderableString
(MICROUI_GraphicsContext * gc, jchar * chars, jint length, MICROUI_Font
* font, jint width, MICROUI_RenderableString * renderableString, jint
x, jint y)
```

Stub implementation of drawRenderableString. See ui_drawing.h.

Parameters .

Table 2.18.

in	gc	the targeted MicroUI GraphicsContext
in	chars	the renderable string of characters
in	length	the number of characters
in	font	the MicroUI Font to use
in	width	the renderable string width in pixels
in	render-ableString	the renderable string
in	x	the left coordinate
in	y	the top coordinate

Returns. the drawing status.

Definition at line 238 of file ui_drawing_stub.c.

```

240                                     {
241     (void)width;
242     (void)renderableString;
243     UI_FONT_DRAWING_drawString(gc, chars, length, font, x, y);
244 }
```

2.6.2.15. UI_DRAWING_STUB_drawRotatedImageBilinear()

```

DRAWING_Status      UI_DRAWING_STUB_drawRotatedImageBilinear
(MICROUI_GraphicsContext * gc, MICROUI_Image * img, jint x, jint y, jint
rotationX, jint rotationY, jfloat angle, jint alpha)
```

Stubbed implementation of drawRotatedImageBilinear. See ui_drawing.h.

Parameters .

Table 2.19.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	x	the x coordinate of the image reference anchor top-left point.
in	y	the y coordinate of the image reference anchor top-left point.
in	rotationX	the x coordinate of the rotation center.

in	rotationY	the y coordinate of the rotation center.
in	angle	the rotation angle.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 439 of file ui_drawing_stub.c.

```

440
441                                     {
442     (void)gc;
443     (void)img;
444     (void)x;
445     (void)y;
446     (void)rotationX;
447     (void)rotationY;
448     (void)angle;
449     (void)alpha;
450     not_implemented(gc);
451 }
```

2.6.2.16. UI_DRAWING_STUB_drawRotatedImageNearestNeighbor()

DRAWING_Status UI_DRAWING_STUB_drawRotatedImageNearestNeighbor
(MICROUI_GraphicsContext * gc, MICROUI_Image * img, jint x, jint y, jint
rotationX, jint rotationY, jfloat angle, jint alpha)

Stubbed implementation of drawRotatedImageNearestNeighbor. See ui_drawing.h.

Parameters .

Table 2.20.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	x	the x coordinate of the image reference anchor top-left point.
in	y	the y coordinate of the image reference anchor top-left point.
in	rotationX	the x coordinate of the rotation center.
in	rotationY	the y coordinate of the rotation center.
in	angle	the rotation angle.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 424 of file ui_drawing_stub.c.

```

426     {
427     (void)gc;
428     (void)img;
429     (void)x;
430     (void)y;
431     (void)rotationX;
432     (void)rotationY;
433     (void)angle;
434     (void)alpha;
435     not_implemented(gc);
436 }

```

2.6.2.17. UI_DRAWING_STUB_drawRoundedRectangle()

DRAWING_Status UI_DRAWING_STUB_drawRoundedRectangle
(MICROUI_GraphicsContext * gc, jint x, jint y, jint width, jint height,
jint cornerEllipseWidth, jint cornerEllipseHeight)

Stubbed implementation of drawRoundedRectangle. See ui_drawing.h.

Parameters .

Table 2.21.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the rectangle width.
in	height	the rectangle height.
in	cornerEl- lipseWidth	the horizontal diameter of the arc at the corners.
in	cornerEllipse- Height	the vertical diameter of the arc at the corners.

Returns. the drawing status.

Definition at line 97 of file ui_drawing_stub.c.

```

98                                     {
99     (void)gc;
100     (void)x;
101     (void)y;
102     (void)width;
103     (void)height;
104     (void)cornerEllipseWidth;
105     (void)cornerEllipseHeight;
106     not_implemented(gc);
107 }

```

2.6.2.18. UI_DRAWING_STUB_drawScaledImageBilinear()

```

DRAWING_Status      UI_DRAWING_STUB_drawScaledImageBilinear
(MICROUI_GraphicsContext * gc, MICROUI_Image * img, jint x, jint y,
jfloat factorX, jfloat factorY, jint alpha)

```

Stubbed implementation of drawScaledImageBilinear. See ui_drawing.h.

Parameters .

Table 2.22.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	x	the x coordinate of the image reference anchor top-left point.
in	y	the y coordinate of the image reference anchor top-left point.
in	factorX	scaling X factor.
in	factorY	scaling Y factor.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 466 of file ui_drawing_stub.c.

```

467                                     {
468     (void)gc;
469     (void)img;
470     (void)x;
471     (void)y;
472     (void)factorX;
473     (void)factorY;
474     (void)alpha;
475     not_implemented(gc);
476 }

```

2.6.2.19. UI_DRAWING_STUB_drawScaledImageNearestNeighbor()

```
DRAWING_Status      UI_DRAWING_STUB_drawScaledImageNearestNeighbor
(MICROUI_GraphicsContext * gc, MICROUI_Image * img, jint x, jint y,
jfloat factorX, jfloat factorY, jint alpha)
```

Stubbed implementation of drawScaledImageNearestNeighbor. See ui_drawing.h.

Parameters .

Table 2.23.

in	gc	the MicroUI GraphicsContext target.
in	img	the MicroUI Image to draw.
in	x	the x coordinate of the image reference anchor top-left point.
in	y	the y coordinate of the image reference anchor top-left point.
in	factorX	scaling X factor.
in	factorY	scaling Y factor.
in	alpha	the opacity level to apply to the region.

Returns. the drawing status.

Definition at line 453 of file ui_drawing_stub.c.

```
454                                     {
455     (void)gc;
456     (void)img;
457     (void)x;
458     (void)y;
459     (void)factorX;
460     (void)factorY;
461     (void)alpha;
462     not_implemented(gc);
463 }
```

2.6.2.20. UI_DRAWING_STUB_drawScaledRenderableStringBilinear()

```
DRAWING_Status      UI_DRAWING_STUB_drawScaledRenderableStringBilinear
(MICROUI_GraphicsContext * gc, jchar * chars, jint length, MICROUI_Font
* font, jint width, MICROUI_RenderableString * renderableString, jint
x, jint y, jfloat xRatio, jfloat yRatio)
```

Stubbed implementation of drawScaledRenderableStringBilinear. See ui_drawing.h.

Parameters .

Table 2.24.

in	gc	the targeted MicroUI GraphicsContext
in	chars	a string of characters
in	length	the number of characters
in	font	the MicroUI Font to use
in	width	the renderable string width in pixels
in	render-ableString	the renderable string
in	x	the left coordinate
in	y	the top coordinate
in	xRatio	the horizontal scaling ratio (1.0f meaning no scaling)
in	yRatio	the vertical scaling ratio (1.0f meaning no scaling)

Returns. the drawing status.

Definition at line 494 of file ui_drawing_stub.c.

```

497                                     {
498     (void)width;
499     (void)renderableString;
500     UI_FONT_DRAWING_drawScaledStringBilinear(gc, chars, length,
501     font, x, y, xRatio, yRatio);
501 }
```

2.6.2.21. UI_DRAWING_STUB_drawScaledStringBilinear()

```

DRAWING_Status      UI_DRAWING_STUB_drawScaledStringBilinear
(MICROUI_GraphicsContext * gc, jchar * chars, jint length, MICROUI_Font
* font, jint x, jint y, jfloat xRatio, jfloat yRatio)
```

Stubbed implementation of drawScaledStringBilinear. See ui_drawing.h.

Parameters .

Table 2.25.

in	gc	the targeted MicroUI GraphicsContext
in	chars	a string of characters
in	length	the number of characters

in	font	the MicroUI Font to use
in	x	the left coordinate
in	y	the top coordinate
in	xRatio	the horizontal scaling ratio (1.0f meaning no scaling)
in	yRatio	the vertical scaling ratio (1.0f meaning no scaling)

Returns. the drawing status.

Definition at line 479 of file ui_drawing_stub.c.

```

481
482 {
483     (void)gc;
484     (void)chars;
485     (void)length;
486     (void)font;
487     (void)x;
488     (void)y;
489     (void)xRatio;
490     (void)yRatio;
491     not_implemented(gc);
492 }
```

2.6.2.22. UI_DRAWING_STUB_drawString()

```
DRAWING_Status UI_DRAWING_STUB_drawString (MICROUI_GraphicsContext *
gc, jchar * chars, jint length, MICROUI_Font * font, jint x, jint y)
```

Stub implementation of drawString. See ui_drawing.h.

Parameters .

Table 2.26.

in	gc	the targeted MicroUI GraphicsContext.
in	chars	a string of characters
in	length	the number of characters
in	font	the MicroUI Font to use.
in	x	the left coordinate
in	y	the top coordinate

Returns. the drawing status.

Definition at line 211 of file ui_drawing_stub.c.

```

212                                     {
213     (void)gc;
214     (void)chars;
215     (void)length;
216     (void)font;
217     (void)x;
218     (void)y;
219     not_implemented(gc);
220 }

```

2.6.2.23. UI_DRAWING_STUB_drawThickCircle()

DRAWING_Status UI_DRAWING_STUB_drawThickCircle (MICROUI_GraphicsContext * gc, jint x, jint y, jint diameter, jint thickness)

Stubbed implementation of drawThickCircle. See ui_drawing.h.

Parameters .

Table 2.27.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the square where the circle is drawn
in	y	the y coordinate of the upper-left corner of the square where the circle is drawn
in	diameter	the diameter of the circle to draw
in	thickness	the circle thickness.

Returns. the drawing status.

Definition at line 371 of file ui_drawing_stub.c.

```

372                                     {
373     (void)gc;
374     (void)x;
375     (void)y;
376     (void)diameter;
377     (void)thickness;
378     not_implemented(gc);
379 }

```

2.6.2.24. UI_DRAWING_STUB_drawThickCircleArc()

DRAWING_Status UI_DRAWING_STUB_drawThickCircleArc (MICROUI_GraphicsContext * gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle, jint thickness)

Stubbed implementation of drawThickCircleArc. See ui_drawing.h.

Parameters .

Table 2.28.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the square where the arc is drawn
in	y	the y coordinate of the upper-left corner of the square where the arc is drawn
in	diameter	the diameter of the circle to draw
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle
in	thickness	the arc thickness.

Returns. the drawing status.

Definition at line 394 of file ui_drawing_stub.c.

```

395                                     {
396     (void)gc;
397     (void)x;
398     (void)y;
399     (void)diameter;
400     (void)startAngle;
401     (void)arcAngle;
402     (void)thickness;
403     not_implemented(gc);
404 }
```

2.6.2.25. UI_DRAWING_STUB_drawThickEllipse()

DRAWING_Status UI_DRAWING_STUB_drawThickEllipse
(MICROUI_GraphicsContext * gc, jint x, jint y, jint width, jint height,
jint thickness)

Stubbed implementation of drawThickEllipse. See ui_drawing.h.

Parameters .

Table 2.29.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the rectangle where the ellipse is drawn

in	y	the y coordinate of the upper-left corner of the rectangle where the ellipse is drawn
in	width	the width of the ellipse to draw
in	height	the height of the ellipse to draw
in	thickness	the ellipse thickness.

Returns. the drawing status.

Definition at line 382 of file ui_drawing_stub.c.

```

383                                     {
384     (void)gc;
385     (void)x;
386     (void)y;
387     (void)width;
388     (void)height;
389     (void)thickness;
390     not_implemented(gc);
391 }
```

2.6.2.26. UI_DRAWING_STUB_drawThickFadedCircle()

DRAWING_Status UI_DRAWING_STUB_drawThickFadedCircle
(MICROUI_GraphicsContext * gc, jint x, jint y, jint diameter, jint thickness, jint fade)

Stubbed implementation of drawThickFadedCircle. See ui_drawing.h.

Parameters .

Table 2.30.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the square where the circle is drawn
in	y	the y coordinate of the upper-left corner of the square where the circle is drawn
in	diameter	the diameter of the circle to draw
in	thickness	the circle thickness.
in	fade	the fade to apply.

Returns. the drawing status.

Definition at line 317 of file ui_drawing_stub.c.

```

318
319     {
320     (void)gc;
321     (void)x;
322     (void)y;
323     (void)diameter;
324     (void)thickness;
325     (void)fade;
326     not_implemented(gc);
327 }

```

2.6.2.27. UI_DRAWING_STUB_drawThickFadedCircleArc()

DRAWING_Status UI_DRAWING_STUB_drawThickFadedCircleArc(MICROUI_GraphicsContext * gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle, jint thickness, jint fade, DRAWING_Cap start, DRAWING_Cap end)

Stubbed implementation of drawThickFadedCircleArc. See ui_drawing.h.

Parameters .

Table 2.31.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the square where the arc is drawn
in	y	the y coordinate of the upper-left corner of the square where the arc is drawn
in	diameter	the diameter of the circle to draw
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle
in	thickness	the arc thickness.
in	fade	the fade to apply.
in	start	cap representation of start of shape
in	end	cap representation of end of shape

Returns. the drawing status.

Definition at line 329 of file ui_drawing_stub.c.

```

331
332         {
333         (void)gc;
334         (void)x;
335         (void)y;
336         (void)diameter;
337         (void)startAngle;
338         (void)arcAngle;
339         (void)thickness;
340         (void)fade;
341         (void)start;
342         (void)end;
343         not_implemented(gc);
344     }

```

2.6.2.28. UI_DRAWING_STUB_drawThickFadedEllipse()

DRAWING_Status UI_DRAWING_STUB_drawThickFadedEllipse
(MICROUI_GraphicsContext * gc, jint x, jint y, jint width, jint height,
jint thickness, jint fade)

Stubbed implementation of drawThickFadedEllipse. See ui_drawing.h.

Parameters .

Table 2.32.

in	gc	the MicroUI GraphicsContext target.
in	x	the x coordinate of the upper-left corner of the rectangle where the ellipse is drawn
in	y	the y coordinate of the upper-left corner of the rectangle where the ellipse is drawn
in	width	the width of the ellipse to draw
in	height	the height of the ellipse to draw
in	thickness	the ellipse thickness.
in	fade	the fade to apply.

Returns. the drawing status.

Definition at line 346 of file ui_drawing_stub.c.

```

347
348         (void)gc;
349         (void)x;
350         (void)y;
351         (void)width;
352         (void)height;
353         (void)thickness;
354         (void)fade;
355         not_implemented(gc);
356     }

```

2.6.2.29. UI_DRAWING_STUB_drawThickFadedLine()

```

DRAWING_Status          UI_DRAWING_STUB_drawThickFadedLine
(MICROUI_GraphicsContext * gc, jint startX, jint startY, jint endX,
jint endY, jint thickness, jint fade, DRAWING_Cap startCap, DRAWING_Cap
endCap)

```

Stubbed implementation of drawThickFadedLine. See ui_drawing.h.

Parameters .

Table 2.33.

in	gc	the MicroUI GraphicsContext target.
in	startX	the x coordinate of the start of the line
in	startY	the y coordinate of the start of the line
in	endX	the x coordinate of the end of the line
in	endY	the y coordinate of the end of the line
in	thickness	the line thickness.
in	fade	the fade to apply.
in	startCap	cap representation of start of shape
in	endCap	cap representation of end of shape

Returns. the drawing status.

Definition at line 301 of file ui_drawing_stub.c.

```

303
304 {
305     (void)gc;
306     (void)startX;
307     (void)startY;
308     (void)endX;
309     (void)endY;
310     (void)thickness;
311     (void)fade;
312     (void)startCap;
313     (void)endCap;
314     not_implemented(gc);
315 }

```

2.6.2.30. UI_DRAWING_STUB_drawThickFadedPoint()

DRAWING_Status UI_DRAWING_STUB_drawThickFadedPoint
(MICROUI_GraphicsContext * gc, jint x, jint y, jint thickness, jint fade)

Stubbed implementation of drawThickFadedPoint. See ui_drawing.h.

Parameters .

Table 2.34.

in	gc	the MicroUI GraphicsContext target.
in	x	the point X coordinate.
in	y	the point Y coordinate.
in	thickness	the point thickness.
in	fade	the fade to apply.

Returns. the drawing status.

Definition at line 290 of file ui_drawing_stub.c.

```

291
292     (void)gc;
293     (void)x;
294     (void)y;
295     (void)thickness;
296     (void)fade;
297     not_implemented(gc);
298 }

```

2.6.2.31. UI_DRAWING_STUB_drawThickLine()

DRAWING_Status UI_DRAWING_STUB_drawThickLine (MICROUI_GraphicsContext *
gc, jint startX, jint startY, jint endX, jint endY, jint thickness)

Stubbed implementation of drawThickLine. See ui_drawing.h.

Parameters .

Table 2.35.

in	gc	the MicroUI GraphicsContext target.
in	startX	the x coordinate of the start of the line
in	startY	the y coordinate of the start of the line
in	endX	the x coordinate of the end of the line
in	endY	the y coordinate of the end of the line
in	thickness	the line thickness.

Returns. the drawing status.

Definition at line 359 of file ui_drawing_stub.c.

```

360
    {
361     (void)gc;
362     (void)startX;
363     (void)startY;
364     (void)endX;
365     (void)endY;
366     (void)thickness;
367     not_implemented(gc);
368 }

```

2.6.2.32. UI_DRAWING_STUB_drawVerticalLine()

DRAWING_Status UI_DRAWING_STUB_drawVerticalLine
(MICROUI_GraphicsContext * gc, jint x, jint y1, jint y2)

Stubbed implementation of drawVerticalLine. See ui_drawing.h.

Parameters .

Table 2.36.

in	gc	the MicroUI GraphicsContext target.
in	x	the both pixels line X coordinate.
in	y1	the first pixel line Y coordinate.
in	y2	the last pixel line Y coordinate.

Returns. the drawing status.

Definition at line 68 of file ui_drawing_stub.c.

```

68                                     {
69     (void)gc;
70     (void)x;
71     (void)y1;
72     (void)y2;
73     not_implemented(gc);
74 }
```

2.6.2.33. UI_DRAWING_STUB_fillCircle()

```
DRAWING_Status UI_DRAWING_STUB_fillCircle (MICROUI_GraphicsContext *
gc, jint x, jint y, jint diameter)
```

Stubbed implementation of fillCircle. See ui_drawing.h.

Parameters .

Table 2.37.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	diameter	the circle square size.

Returns. the drawing status.

Definition at line 202 of file ui_drawing_stub.c.

```

202                                     {
203     (void)gc;
204     (void)x;
205     (void)y;
206     (void)diameter;
207     not_implemented(gc);
208 }
```

2.6.2.34. UI_DRAWING_STUB_fillCircleArc()

```
DRAWING_Status UI_DRAWING_STUB_fillCircleArc (MICROUI_GraphicsContext *
gc, jint x, jint y, jint diameter, jfloat startAngle, jfloat arcAngle)
```

Stubbed implementation of fillCircleArc. See ui_drawing.h.

Parameters .

Table 2.38.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	diameter	the diameter of the arc to draw.
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle

Returns. the drawing status.

Definition at line 148 of file ui_drawing_stub.c.

```

149
    {
150     (void)gc;
151     (void)x;
152     (void)y;
153     (void)diameter;
154     (void)startAngle;
155     (void)arcAngle;
156     not_implemented(gc);
157 }
```

2.6.2.35. UI_DRAWING_STUB_fillEllipse()

```
DRAWING_Status UI_DRAWING_STUB_fillEllipse (MICROUI_GraphicsContext *
gc, jint x, jint y, jint width, jint height)
```

Stubbed implementation of fillEllipse. See ui_drawing.h.

Parameters .

Table 2.39.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the ellipse width.
in	height	the ellipse height.

Returns. the drawing status.

Definition at line 183 of file ui_drawing_stub.c.

```

183                                     {
184     (void)gc;
185     (void)x;
186     (void)y;
187     (void)width;
188     (void)height;
189     not_implemented(gc);
190 }

```

2.6.2.36. UI_DRAWING_STUB_fillEllipseArc()

DRAWING_Status UI_DRAWING_STUB_fillEllipseArc (MICROUI_GraphicsContext * gc, jint x, jint y, jint width, jint height, jfloat startAngle, jfloat arcAngle)

Stubbed implementation of fillEllipseArc. See ui_drawing.h.

Parameters .

Table 2.40.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the rectangle width.
in	height	the rectangle height.
in	startAngle	the beginning angle of the arc to draw
in	arcAngle	the angular extent of the arc from startAngle

Returns. the drawing status.

Definition at line 160 of file ui_drawing_stub.c.

```

161     {
162     (void)gc;
163     (void)x;
164     (void)y;
165     (void)width;
166     (void)height;
167     (void)startAngle;
168     (void)arcAngle;
169     not_implemented(gc);
170 }

```

2.6.2.37. UI_DRAWING_STUB_fillRectangle()

DRAWING_Status UI_DRAWING_STUB_fillRectangle (MICROUI_GraphicsContext * gc, jint x1, jint y1, jint x2, jint y2)

Stubbed implementation of fillRectangle. See ui_drawing.h.

Parameters .

Table 2.41.

in	gc	the MicroUI GraphicsContext target.
in	x1	the top-left pixel X coordinate.
in	y1	the top-left pixel Y coordinate.
in	x2	the bottom-right pixel X coordinate.
in	y2	the bottom-right pixel Y coordinate.

Returns. the drawing status.

Definition at line 87 of file ui_drawing_stub.c.

```

87                                     {
88     (void)gc;
89     (void)x1;
90     (void)y1;
91     (void)x2;
92     (void)y2;
93     not_implemented(gc);
94 }
```

2.6.2.38. UI_DRAWING_STUB_fillRoundedRectangle()

DRAWING_Status UI_DRAWING_STUB_fillRoundedRectangle (MICROUI_GraphicsContext * gc, jint x, jint y, jint width, jint height, jint cornerEllipseWidth, jint cornerEllipseHeight)

Stubbed implementation of fillRoundedRectangle. See ui_drawing.h.

Parameters .

Table 2.42.

in	gc	the MicroUI GraphicsContext target.
in	x	the top-left pixel X coordinate.
in	y	the top-left pixel Y coordinate.
in	width	the rectangle width.
in	height	the rectangle height.

in	cornerEllip- lipseWidth	the horizontal diameter of the arc at the corners.
in	cornerEllipse- Height	the vertical diameter of the arc at the corners.

Returns. the drawing status.

Definition at line 110 of file ui_drawing_stub.c.

```

111                                     {
112     (void)gc;
113     (void)x;
114     (void)y;
115     (void)width;
116     (void)height;
117     (void)cornerEllipseWidth;
118     (void)cornerEllipseHeight;
119     not_implemented(gc);
120 }
```

2.6.2.39. UI_DRAWING_STUB_initializeRenderableStringSNIContext()

```

jint  UI_DRAWING_STUB_initializeRenderableStringSNIContext  (jchar *
chars, jint length, MICROUI_Font * font, MICROUI_RenderableString * ren-
derableString)
```

Stub implementation of initializeRenderableStringSNIContext. See ui_drawing.h.

Parameters .

Table 2.43.

in	chars	the renderable string of characters
in	length	the number of characters
in	font	the MicroUI Font to use
in,out	render- ableString	the SNI context of the renderable string

Returns. the renderable string width in pixels

Definition at line 231 of file ui_drawing_stub.c.

```

232                                     {
233     (void)renderableString;
234     UI_FONT_DRAWING_stringWidth(chars, length, font);
235 }
```

2.6.2.40. UI_DRAWING_STUB_stringWidth()

```
jint  UI_DRAWING_STUB_stringWidth  (jchar  *  chars,  jint  length,
MICROUI_Font * font)
```

Stub implementation of stringWidth. See ui_drawing.h.

Parameters .

Table 2.44.

in	chars	a string of characters
in	length	the number of characters
in	font	the MicroUI Font used

Returns. the width in pixels

Definition at line 223 of file ui_drawing_stub.c.

```
223
    {
224     (void)chars;
225     (void)length;
226     (void)font;
227     0;
228 }
```

2.6.2.41. UI_DRAWING_STUB_writePixel()

```
DRAWING_Status  UI_DRAWING_STUB_writePixel  (MICROUI_GraphicsContext  *
gc, jint x, jint y)
```

Stubbed implementation of writePixel. See ui_drawing.h.

Parameters .

Table 2.45.

in	gc	the MicroUI GraphicsContext target.
in	x	the pixel X coordinate.
in	y	the pixel Y coordinate.

Returns. the drawing status.

Definition at line 41 of file ui_drawing_stub.c.

```
41
    {
42    (void)gc;
43    (void)x;
44    (void)y;
45    not_implemented(gc);
46 }
```

2.7. bsp/ui/inc/ui_rect_util.h File Reference

Utility functions to handle rectangles.

```
#include <stddef.h>
#include "ui_rect.h"
#include "ui_util.h"
```

Macros

- `#define UI_RECT_EMPTY { 1u, 1u, 0u, 0u }`

The initializer of an empty rectangle: its end (x2,y2) is before its start (x1,y1).

Functions

- `ui_rect_t UI_RECT_get_minimum_bounding_rect (const ui_rect_t rects[], const size_t count)`
Computes the minimum bounding rectangle of an array of rectangles.
- `uint32_t UI_RECT_union (ui_rect_t output[3], const ui_rect_t *first, const ui_rect_t *second)`
Computes the union of two rectangles.
- `uint32_t UI_RECT_subtract (ui_rect_t output[4], const ui_rect_t *first, const ui_rect_t *second)`
Computes the difference between two rectangles.

2.7.1. Detailed Description

Utility functions to handle rectangles.

See also. `ui_rect_t`

Author. MicroEJ Developer Team

Version. 14.5.5

2.7.2. Function Documentation

2.7.2.1. UI_RECT_get_minimum_bounding_rect()

```
ui_rect_t UI_RECT_get_minimum_bounding_rect (const ui_rect_t rects[],
const size_t count)
```

Computes the minimum bounding rectangle of an array of rectangles.

The behavior is undefined if the array is empty.

Parameters .

Table 2.46.

in	rects	the array of rectangles
in	count	the length of the array

Returns. the minimum bounding rectangle

Definition at line 32 of file ui_rect_util.c.

```

32
33         {
34     assert(count > 0u);
35     ui_rect_t result = rects[0];
36     ( i = 1; i < count; i++) {
37         result =
38     UI_RECT_get_minimum_bounding_rect_two_rects(&result, &rects[i]);
39     }
40     result;
41 }
```

2.7.2.2. UI_RECT_subtract()

```
uint32_t UI_RECT_subtract (ui_rect_t output[4], const ui_rect_t * first,
const ui_rect_t * second)
```

Computes the difference between two rectangles.

Parameters .

Table 2.47.

out	output	the rectangles defining the difference between the parameters
in	first	the original rectangle
in	second	the rectangle that will be removed from the original one

Returns. the number of resulting rectangles. Some of them may be empty.

Definition at line 89 of file ui_rect_util.c.

```

89
90         {
91         uint32_t ret;
92         (UI_RECT_contains_rect(first, second)) {
93             output[0] = UI_RECT_new_xyxy(first->x1, first->y1, first-
94             >x2, second->y1 - 1);
95             output[1] = UI_RECT_new_xyxy(first->x1, second->y1, second-
96             >x1 - 1, second->y2);
97             output[2] = UI_RECT_new_xyxy(second->x2 + 1, second->y1,
98             first->x2, second->y2);
99             output[3] = UI_RECT_new_xyxy(first->x1, second->y2 + 1,
100             first->x2, first->y2);
101             ret = 4u;
102         } (UI_RECT_contains_rect(second, first)) {
103             UI_RECT_mark_empty(&output[0]);
104             UI_RECT_mark_empty(&output[1]);
105             UI_RECT_mark_empty(&output[2]);
106             UI_RECT_mark_empty(&output[3]);
107             ret = 0u;
108         } (!UI_RECT_intersects_rect(first, second)) {
109             output[0] = *first;
110             UI_RECT_mark_empty(&output[1]);
111             UI_RECT_mark_empty(&output[2]);
112             UI_RECT_mark_empty(&output[3]);
113             ret = 1u;
114         } {
115             output[0] = UI_RECT_new_xyxy(first->x1, first->y1, first-
116             >x2, second->y1 - 1);
117             output[1] = UI_RECT_new_xyxy(first->x1, MAX(first->y1,
118             second->y1), second->x1 - 1, MIN(first->y2, second->y2));
119             output[2] = UI_RECT_new_xyxy(second->x2 + 1, MAX(first-
120             >y1, second->y1), first->x2, MIN(first->y2, second->y2));
121             output[3] = UI_RECT_new_xyxy(first->x1, second->y2 + 1,
122             first->x2, first->y2);
123             ret = 4u;
124         }
125         ret;
126     }

```

2.7.2.3. UI_RECT_union()

```
uint32_t UI_RECT_union (ui_rect_t output[3], const ui_rect_t * first,
const ui_rect_t * second)
```

Computes the union of two rectangles.

Parameters .

Table 2.48.

out	output	rectangles defining the union of the parameters
-----	--------	---

in	first	the first rectangle
in	second	the second rectangle

Returns. the number of resulting rectangles. Some of them may be empty.

Definition at line 42 of file ui_rect_util.c.

```

42                                     {
43     uint32_t ret;
44     (UI_RECT_contains_rect(first, second)) {
45         output[0] = *first;
46         UI_RECT_mark_empty(&output[1]);
47         UI_RECT_mark_empty(&output[2]);
48         ret = 1u;
49     } (UI_RECT_contains_rect(second, first)) {
50         output[0] = *second;
51         UI_RECT_mark_empty(&output[1]);
52         UI_RECT_mark_empty(&output[2]);
53         ret = 1u;
54     } (!UI_RECT_intersects_rect(first, second)) {
55         output[0] = *first;
56         output[1] = *second;
57         UI_RECT_mark_empty(&output[2]);
58         ret = 2u;
59     } {
60         ui_rect_t *top_above = (first->y1 < second->y1) ? first :
        second;
61         ui_rect_t *top_below = (first->y1 < second->y1) ? second :
        first;
62         ui_rect_t *bottom_below = (first->y2 > second->y2) ?
        first : second;
63         ui_rect_t *bottom_above = (first->y2 > second->y2) ?
        second : first;
64
65         uint32_t count = 0u;
66
67         output[count] = UI_RECT_new_xyxy(top_above->x1, top_above->
        y1, top_above->x2, top_below->y1 - 1);
68         (!UI_RECT_is_empty(output)) {
69             count++;
70         }
71
72         output[count] = UI_RECT_new_xyxy(MIN(first->x1, second->
        x1), top_below->y1, MAX(first->x2, second->x2),
73                                     bottom_above->y2);
74         (!UI_RECT_is_empty(output)) {
75             count++;
76         }
77
78         output[count] = UI_RECT_new_xyxy(bottom_below->x1,
        bottom_above->y2 + 1, bottom_below->x2, bottom_below->y2);
79         (!UI_RECT_is_empty(output)) {
80             count++;
81         }
82
83         ret = count;
84     }
85     ret;
86 }

```

2.8. bsp/ui/inc/ui_util.h File Reference

Utility functions.

Macros

- `#define MIN(a, b) ((a) < (b) ? (a) : (b))`
Gets the lowest value.
- `#define MAX(a, b) ((a) > (b) ? (a) : (b))`
Gets the highest value.

2.8.1. Detailed Description

Utility functions.

Author. MicroEJ Developer Team

Version. 14.5.5

2.9. bsp/ui/src/LLUI_DISPLAY_HEAP_impl.c File Reference

This MicroUI images heap allocator replaces the default allocator embedded in the MicroUI Graphics Engine. It is using a best fit allocator and provides some additional APIs to retrieve the heap information: total space, free space, number of blocks allocated.

```
#include "ui_configuration.h"
```

2.9.1. Detailed Description

This MicroUI images heap allocator replaces the default allocator embedded in the MicroUI Graphics Engine. It is using a best fit allocator and provides some additional APIs to retrieve the heap information: total space, free space, number of blocks allocated.

See also. LLUI_DISPLAY_impl.h file comment

Author. MicroEJ Developer Team

Version. 14.5.5

Since. MicroEJ UI Pack 13.1.0

2.10. bsp/ui/src/LLUI_INPUT_LOG_impl.c File Reference

This MicroUI FIFO (queue) logger replaces the default logger embedded in the MicroUI Input Engine. For each queue event, it stores the event's data size. This allows to be able to decode the event when `LLUI_INPUT_dump()` is called.

```
#include "ui_configuration.h"
```

2.10.1. Detailed Description

This MicroUI FIFO (queue) logger replaces the default logger embedded in the MicroUI Input Engine. For each queue event, it stores the event's data size. This allows to be able to decode the event when `LLUI_INPUT_dump()` is called.

This logger does not interpret the event: it just recognizes the event's first element and event's data. When an event is detected, the logger calls `microui_event_decoder.h` functions.

See also. `LLUI_INPUT_impl.h` file comment

Author. MicroEJ Developer Team

Version. 14.5.5

Since. MicroEJ UI Pack 13.1.0

2.11. bsp/ui/src/microui_event_decoder.c File Reference

This MicroUI Events decoder describes the events to the standard output stream.

```
#include "ui_configuration.h"
```

2.11.1. Detailed Description

This MicroUI Events decoder describes the events to the standard output stream.

See also. `LLUI_INPUT_LOG_impl.c` file comment

Author. MicroEJ Developer Team

Version. 14.5.5

Since. MicroEJ UI Pack 13.1.0

Index

B

bsp/ui/inc/microui_event_decoder.h, 2
bsp/ui/inc/microui_heap.h, 2
bsp/ui/inc/ui_color.h, 2
bsp/ui/inc/ui_configuration.h, 6
bsp/ui/inc/ui_display_brs.h, 11
bsp/ui/inc/ui_drawing_stub.h, 13
bsp/ui/inc/ui_rect_util.h, 49
bsp/ui/inc/ui_util.h, 54
bsp/ui/src/LLUI_DISPLAY_HEAP_impl.c, 54
bsp/ui/src/LLUI_INPUT_LOG_impl.c, 54
bsp/ui/src/microui_event_decoder.c, 55

C

COLOR_CHANNEL_MASK
 ui_color.h, 4
COLOR_CHANNEL_OFFSET
 ui_color.h, 4
COLOR_CHANNEL_SIZE
 ui_color.h, 5
COLOR_GET_CHANNEL
 ui_color.h, 5
COLOR_SET_COLOR
 ui_color.h, 5

U

ui_color.h
 COLOR_CHANNEL_MASK, 4
 COLOR_CHANNEL_OFFSET, 4
 COLOR_CHANNEL_SIZE, 5
 COLOR_GET_CHANNEL, 5
 COLOR_SET_COLOR, 5
ui_configuration.h
 UI_FEATURE_BRS, 8
 UI_FEATURE_BRS_DRAWING_BUFFER_COUNT, 8
 UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE, 9
 UI_FEATURE_BRS_PREDRAW, 9
 UI_FEATURE_BRS_SINGLE, 10
 UI_FEATURE_EVENT_DECODER, 10
 UI_FEATURE_FONT_CUSTOM_FORMATS, 10
 UI_FEATURE_IMAGE_CUSTOM_FORMATS, 10
 UI_GC_SUPPORTED_FORMATS, 11
ui_display_brs.h
 UI_DISPLAY_BRS_restore, 12
UI_DISPLAY_BRS_restore
 ui_display_brs.h, 12
ui_drawing_stub.h
 UI_DRAWING_STUB_copyImage, 18
 UI_DRAWING_STUB_drawCharWithRotationBilinear, 18
 UI_DRAWING_STUB_drawCharWithRotationNearestNeighbor, 19
 UI_DRAWING_STUB_drawCircle, 20
 UI_DRAWING_STUB_drawCircleArc, 21
 UI_DRAWING_STUB_drawEllipse, 21
 UI_DRAWING_STUB_drawEllipseArc, 22
 UI_DRAWING_STUB_drawFlippedImage, 23
 UI_DRAWING_STUB_drawHorizontalLine, 24
 UI_DRAWING_STUB_drawImage, 24
 UI_DRAWING_STUB_drawLine, 25
 UI_DRAWING_STUB_drawRectangle, 26
 UI_DRAWING_STUB_drawRegion, 27
 UI_DRAWING_STUB_drawRenderableString, 27
 UI_DRAWING_STUB_drawRotatedImageBilinear, 28
 UI_DRAWING_STUB_drawRotatedImageNearestNeighbor, 29
 UI_DRAWING_STUB_drawRoundedRectangle, 30
 UI_DRAWING_STUB_drawScaledImageBilinear, 31
 UI_DRAWING_STUB_drawScaledImageNearestNeighbor, 32
 UI_DRAWING_STUB_drawScaledRenderableStringBilinear, 32
 UI_DRAWING_STUB_drawScaledStringBilinear, 33
 UI_DRAWING_STUB_drawString, 34
 UI_DRAWING_STUB_drawThickCircle, 35
 UI_DRAWING_STUB_drawThickCircleArc, 35
 UI_DRAWING_STUB_drawThickEllipse, 36
 UI_DRAWING_STUB_drawThickFadedCircle, 37
 UI_DRAWING_STUB_drawThickFadedCircleArc, 38
 UI_DRAWING_STUB_drawThickFadedEllipse, 39
 UI_DRAWING_STUB_drawThickFadedLine, 40
 UI_DRAWING_STUB_drawThickFadedPoint, 41
 UI_DRAWING_STUB_drawThickLine, 41
 UI_DRAWING_STUB_drawVerticalLine, 42
 UI_DRAWING_STUB_fillCircle, 43
 UI_DRAWING_STUB_fillCircleArc, 43
 UI_DRAWING_STUB_fillEllipse, 44
 UI_DRAWING_STUB_fillEllipseArc, 45
 UI_DRAWING_STUB_fillRectangle, 46
 UI_DRAWING_STUB_fillRoundedRectangle, 46
 UI_DRAWING_STUB_initializeRenderableStringSNContext, 47
 UI_DRAWING_STUB_stringWidth, 48
 UI_DRAWING_STUB_writePixel, 48
UI_DRAWING_STUB_copyImage
 ui_drawing_stub.h, 17
UI_DRAWING_STUB_drawCharWithRotationBilinear
 ui_drawing_stub.h, 18
UI_DRAWING_STUB_drawCharWithRotationNearestNeighbor
 ui_drawing_stub.h, 19
UI_DRAWING_STUB_drawCircle
 ui_drawing_stub.h, 20
UI_DRAWING_STUB_drawCircleArc
 ui_drawing_stub.h, 21
UI_DRAWING_STUB_drawEllipse
 ui_drawing_stub.h, 21
UI_DRAWING_STUB_drawEllipseArc
 ui_drawing_stub.h, 22
UI_DRAWING_STUB_drawFlippedImage

ui_drawing_stub.h, 23
UI_DRAWING_STUB_drawHorizontalLine
ui_drawing_stub.h, 24
UI_DRAWING_STUB_drawImage
ui_drawing_stub.h, 24
UI_DRAWING_STUB_drawLine
ui_drawing_stub.h, 25
UI_DRAWING_STUB_drawRectangle
ui_drawing_stub.h, 26
UI_DRAWING_STUB_drawRegion
ui_drawing_stub.h, 27
UI_DRAWING_STUB_drawRenderableString
ui_drawing_stub.h, 27
UI_DRAWING_STUB_drawRotatedImageBilinear
ui_drawing_stub.h, 28
UI_DRAWING_STUB_drawRotatedImageNearestNeighbor
ui_drawing_stub.h, 29
UI_DRAWING_STUB_drawRoundedRectangle
ui_drawing_stub.h, 30
UI_DRAWING_STUB_drawScaledImageBilinear
ui_drawing_stub.h, 31
UI_DRAWING_STUB_drawScaledImageNearestNeighbor
ui_drawing_stub.h, 32
UI_DRAWING_STUB_drawScaledRenderableStringBilinear
ui_drawing_stub.h, 32
UI_DRAWING_STUB_drawScaledStringBilinear
ui_drawing_stub.h, 33
UI_DRAWING_STUB_drawString
ui_drawing_stub.h, 34
UI_DRAWING_STUB_drawThickCircle
ui_drawing_stub.h, 35
UI_DRAWING_STUB_drawThickCircleArc
ui_drawing_stub.h, 35
UI_DRAWING_STUB_drawThickEllipse
ui_drawing_stub.h, 36
UI_DRAWING_STUB_drawThickFadedCircle
ui_drawing_stub.h, 37
UI_DRAWING_STUB_drawThickFadedCircleArc
ui_drawing_stub.h, 38
UI_DRAWING_STUB_drawThickFadedEllipse
ui_drawing_stub.h, 39
UI_DRAWING_STUB_drawThickFadedLine
ui_drawing_stub.h, 40
UI_DRAWING_STUB_drawThickFadedPoint
ui_drawing_stub.h, 41
UI_DRAWING_STUB_drawThickLine
ui_drawing_stub.h, 41
UI_DRAWING_STUB_drawVerticalLine
ui_drawing_stub.h, 42
UI_DRAWING_STUB_fillCircle
ui_drawing_stub.h, 43
UI_DRAWING_STUB_fillCircleArc
ui_drawing_stub.h, 43
UI_DRAWING_STUB_fillEllipse
ui_drawing_stub.h, 44
UI_DRAWING_STUB_fillEllipseArc
ui_drawing_stub.h, 45
UI_DRAWING_STUB_fillRectangle
ui_drawing_stub.h, 45
UI_DRAWING_STUB_fillRoundedRectangle
ui_drawing_stub.h, 46
UI_DRAWING_STUB_initializeRenderableStringSNIContext
ui_drawing_stub.h, 47
UI_DRAWING_STUB_stringWidth
ui_drawing_stub.h, 48
UI_DRAWING_STUB_writePixel
ui_drawing_stub.h, 48
UI_FEATURE_BRS
ui_configuration.h, 8
UI_FEATURE_BRS_DRAWING_BUFFER_COUNT
ui_configuration.h, 8
UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE
ui_configuration.h, 9
UI_FEATURE_BRS_PREDRAW
ui_configuration.h, 9
UI_FEATURE_BRS_SINGLE
ui_configuration.h, 10
UI_FEATURE_EVENT_DECODER
ui_configuration.h, 10
UI_FEATURE_FONT_CUSTOM_FORMATS
ui_configuration.h, 10
UI_FEATURE_IMAGE_CUSTOM_FORMATS
ui_configuration.h, 10
UI_GC_SUPPORTED_FORMATS
ui_configuration.h, 11
ui_rect_collection_t, 1
UI_RECT_get_minimum_bounding_rect
ui_rect_util.h, 50
UI_RECT_subtract
ui_rect_util.h, 50
UI_RECT_union
ui_rect_util.h, 51
ui_rect_util.h
UI_RECT_get_minimum_bounding_rect, 50
UI_RECT_subtract, 50
UI_RECT_union, 51