

microui

User Manual



MICROEJ[®]

Reference:	TLT-XXX-MAN-microui-microui
Version:	14.5.2
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_heap.h File Reference	2
2.1.1. Detailed Description	2
2.2. bsp/ui/inc/ui_configuration.h File Reference	2
2.2.1. Detailed Description	4
2.2.2. Macro Definition Documentation	4
2.3. bsp/ui/src/LLUI_DISPLAY_HEAP_impl.c File Reference	7
2.3.1. Detailed Description	7
2.4. bsp/ui/src/LLUI_INPUT_LOG_impl.c File Reference	7
2.4.1. Detailed Description	8
2.5. bsp/ui/src/microui_event_decoder.c File Reference	8
2.5.1. Detailed Description	8
Index	9

Chapter 1. Data Structure Documentation

1.1. ui_rect_collection_t Struct Reference

Data Fields

- ui_rect_t [UI_RECT_COLLECTION_MAX_LENGTH]
- size_t

1.1.1. Detailed Description

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_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"
```

2.1.1. 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. 14.5.2

Since. MicroEJ UI Pack 13.1.0

2.2. 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 (8u)`

- `#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.

2.2.1. Detailed Description

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

Author. MicroEJ Developer Team

2.2.2. Macro Definition Documentation

2.2.2.1. 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`.

2.2.2.2. 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.

2.2.2.3. 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.

2.2.2.4. 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`.

2.2.2.5. `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`.

2.2.2.6. `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`.

2.2.2.7. `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`.

2.2.2.8. `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`.

2.2.2.9. 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`.

2.3. 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"
```

2.3.1. 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. 14.5.2

Since. MicroEJ UI Pack 13.1.0

2.4. 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"
```

2.4.1. 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. 14.5.2

Since. MicroEJ UI Pack 13.1.0

2.5. 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"
```

2.5.1. 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. 14.5.2

Since. MicroEJ UI Pack 13.1.0

Index

B

bsp/ui/inc/microui_heap.h, 2
bsp/ui/inc/ui_configuration.h, 2
bsp/ui/src/LLUI_DISPLAY_HEAP_impl.c, 7
bsp/ui/src/LLUI_INPUT_LOG_impl.c, 7
bsp/ui/src/microui_event_decoder.c, 8

U

ui_configuration.h
 UI_FEATURE_BRS, 4
 UI_FEATURE_BRS_DRAWING_BUFFER_COUNT, 4
 UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE, 5
 UI_FEATURE_BRS_PREDRAW, 5
 UI_FEATURE_BRS_SINGLE, 6
 UI_FEATURE_EVENT_DECODER, 6
 UI_FEATURE_FONT_CUSTOM_FORMATS, 6
 UI_FEATURE_IMAGE_CUSTOM_FORMATS, 6
 UI_GC_SUPPORTED_FORMATS, 7
UI_FEATURE_BRS
 ui_configuration.h, 4
UI_FEATURE_BRS_DRAWING_BUFFER_COUNT
 ui_configuration.h, 4
UI_FEATURE_BRS_FLUSH_SINGLE_RECTANGLE
 ui_configuration.h, 5
UI_FEATURE_BRS_PREDRAW
 ui_configuration.h, 5
UI_FEATURE_BRS_SINGLE
 ui_configuration.h, 6
UI_FEATURE_EVENT_DECODER
 ui_configuration.h, 6
UI_FEATURE_FONT_CUSTOM_FORMATS
 ui_configuration.h, 6
UI_FEATURE_IMAGE_CUSTOM_FORMATS
 ui_configuration.h, 6
UI_GC_SUPPORTED_FORMATS
 ui_configuration.h, 7
ui_rect_collection_t, 1