microui

User Manual

Confidentiality & Intellectual Property

Table of Contents

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.   14.1.1

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.

Macros

- #define UI_CONFIGURATION_VERSION (1)

  Compatibility sanity check value. This define value is checked in the implementation to validate that the version of this configuration is compatible with the implementation.

- #define UI_FEATURE_ALLOCATOR_BESTFIT (1u)

  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_BRS_LEGACY (0u)

  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 (1u)

  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 (2u)

  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_BRS (UI_FEATURE_BRS_PREDRAW)

  Uncomment this define to use the allocator "BESTFIT".

- #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_GC_SUPPORTED_FORMATS (1u)

  Defines the number of rectangles the strategy uses when calling LLUI_DISPLAY_IMPL_flush(). When set, the strategy sends only one rectangle that includes all dirty regions (a rectangle that encapsulates all rectangles). When not set, the strategy sends all rectangles.

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_CONFIGURATION_VERSION

```
#define UI_CONFIGURATION_VERSION (1)
```

Compatibility sanity check value. This define value is checked in the implementation to validate that the version of this configuration is compatible with the implementation.

This value must not be changed by the user of this C module. This value must be incremented by the implementor of this C module when a configuration define is added, deleted or modified.

Definition at line 32 of file ui_configuration.h.

2.2.2.2. UI_FEATURE_BRS

`#define UI_FEATURE_BRS (UI_FEATURE_BRS_PREDRAW)`

Uncomment this define to use the allocator "BESTFIT".

When unset, the Graphics Engine uses its internal allocator (a best fit allocator that does not provide any functions to analyse its content).

The default graphics engine's allocator can be replaced by a 3rd-party allocator by implementing the functions LLUI_DISPLAY_IMPL_imageHeap*().

When defined, the logger is enabled. The call to LLUI_INPUT_dump() has no effect when the logger is disabled.

By default the logger is not enabled.

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 called LLUI_DISPLAY_IMPL_flush()(always full screen, no restore).

Definition at line 190 of file ui_configuration.h.

2.2.2.3. 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 or not on the MCU address space).

Definition at line 205 of file ui_configuration.h.

2.2.2.4. UI_FEATURE_BRS_PREDRAW

`#define UI_FEATURE_BRS_PREDRAW (2u)`

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 90 of file ui_configuration.h.

2.2.2.5. UI_FEATURE_BRS_SINGLE

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

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 65 of file ui_configuration.h.

2.2.2.6. UI_GC_SUPPORTED_FORMATS

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

Defines the number of rectangles the strategy uses when calling LLUI_DISPLAY_IMPL_flush(). When set, the strategy sends only one rectangle that includes all dirty regions (a rectangle that encapsulates all rectangles). When not set, 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.
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 236 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.   14.1.1

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.   14.1.1

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.   14.1.1

Since.   MicroEJ UI Pack 13.1.0

Index