

Buffered Serial

1.0

Generated by Doxygen 1.8.18

1 Buffered Serial	1
1.1 Features:	1
1.2 Considerations:	1
1.3 Getting Started	1
1.3.1 UART Error handling in buffered_serial.c	1
1.3.2 Configure IDLE interrupt in stm32f1xx_it.c	2
1.3.3 Initializing library and getting serial descriptor in main.c file	2
1.3.4 Writing a string	2
1.3.5 Writing a string as a line	2
1.3.6 Reading a line	2
1.3.7 Writing a character	2
1.3.8 Configure serials quantity and size of the buffers	2
2 Module Index	3
2.1 Modules	3
3 Data Structure Index	5
3.1 Data Structures	5
4 File Index	7
4.1 File List	7
5 Module Documentation	9
5.1 Serial buffers size and quantity	9
5.1.1 Detailed Description	9
5.2 Error handling	10
5.2.1 Detailed Description	10
5.2.2 Variable Documentation	10
5.2.2.1 buffered_serial_error_code	10
5.3 Serial buffers	11
5.3.1 Detailed Description	11
6 Data Structure Documentation	13
6.1 buffered_serial_serial_descriptor Struct Reference	13
6.1.1 Detailed Description	13
6.1.2 Field Documentation	13
6.1.2.1 rx_buffer_data_finish	13
6.1.2.2 rx_buffer_data_start	13
7 File Documentation	15
7.1 buffered_serial.c File Reference	15
7.1.1 Detailed Description	15
7.1.2 Function Documentation	16
7.1.2.1 buffered_serial_available()	16

7.1.2.2 buffered_serial_get_huart_serial_descriptor()	16
7.1.2.3 buffered_serial_init()	16
7.1.2.4 buffered_serial_print_character()	18
7.1.2.5 buffered_serial_print_line()	18
7.1.2.6 buffered_serial_print_string()	18
7.1.2.7 buffered_serial_read_line()	19
7.1.2.8 buffered_serial_update_rx_buffer_data()	19
7.2 buffered_serial.h File Reference	20
7.2.1 Detailed Description	21
7.2.2 Function Documentation	21
7.2.2.1 buffered_serial_available()	21
7.2.2.2 buffered_serial_get_huart_serial_descriptor()	21
7.2.2.3 buffered_serial_init()	22
7.2.2.4 buffered_serial_print_character()	22
7.2.2.5 buffered_serial_print_line()	22
7.2.2.6 buffered_serial_print_string()	23
7.2.2.7 buffered_serial_read_line()	23
7.2.2.8 buffered_serial_update_rx_buffer_data()	24
Index	25

Chapter 1

Buffered Serial

1.1 Features:

- Developed for the STM32F103.
- Serial communication with DMA in circular mode and IDLE interrupt.
- Configurable quantity of serials and size of rx and tx buffers.
- Simple communication with print string, print character, print line and read line functions.
- STM32CubeIDE project configuration guide.
- Error handling with buffered_serial_error_code.
- UART Error handling

1.2 Considerations:

- BUFFERED_SERIAL_SERIALS_QUANTITY must be configured to correspond the quantity of huart configured, by default is one.
- BUFFERED_SERIAL_BUFFERS_SIZE at most can be maximum value of uint16_t, since buffered_serial_↔ available is this type.
- Buffers can hold at most BUFFERED_SERIAL_BUFFERS_SIZE - 1 data, because when rx_buffer_data_↔ start and rx_buffer_data_finish pointers are equals it can be 0 data or maximum data but the library interpret as 0 data.

1.3 Getting Started

1.3.1 UART Error handling in buffered_serial.c

```
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
    HAL_UART_Receive_DMA(huart, buffered_serial_get_huart_serial_descriptor(huart) -> rx_buffer, BUFFERED_SERIAL_BUFFERS_SIZE);
}
```

1.3.2 Configure IDLE interrupt in stm32f1xx_it.c

Configure project as described in file project_configuration.pdf in root folder. IDLE interrupt must be configured for all huart interrupt handlers.

```
void USART1_IRQHandler(void)
{
    HAL_UART_IRQHandler(&huart1);
    buffered_serial_update_rx_buffer_data(&huart1);
}
```

1.3.3 Initializing library and getting serial descriptor in main.c file

```
MX_GPIO_Init();
MX_DMA_Init();
MX_USART1_UART_Init();
UART_HandleTypeDef *huarts[] = {&huart1};
buffered_serial_init(huarts);
buffered_serial_serial_descriptor *serial1 = buffered_serial_get_huart_serial_descriptor(&huart1);
```

1.3.4 Writing a string

```
uint8_t test[40] = "2A6V7W5NL5ZZC6AYE84NKZ6MVFMZ5DZSYD9TM3\r\n";
static_strings_string_descriptor *string_descriptor = static_strings_save(test);
buffered_serial_print_string(test, string_descriptor);
static_strings_deallocate(string_descriptor);
```

DON'T FORGET TO DEALLOCATE STRING AFTER USING.

1.3.5 Writing a string as a line

```
static_strings_string_descriptor *print_line_test = static_strings_save((uint8_t *) "this is not a line");
buffered_serial_print_line(serial1, print_line_test);
static_strings_deallocate(print_line_test);
```

DON'T FORGET TO DEALLOCATE STRING AFTER USING.

1.3.6 Reading a line

```
if(buffered_serial_available(serial1) > 0){
    uint16_t available = buffered_serial_available(serial1);
    static_strings_string_descriptor *string_descriptor = buffered_serial_read_line(serial1);
    if(string_descriptor != NULL){
        buffered_serial_print_string(serial1, string_descriptor);
        static_strings_deallocate(string_descriptor);
    }
    else{
        handle_error(buffered_serial_error_code);
    }
}
```

DON'T FORGET TO DEALLOCATE STRING AFTER USING.

1.3.7 Writing a character

```
uint8_t character = 'A';
buffered_serial_print_character(serial1, character);
```

1.3.8 Configure serials quantity and size of the buffers

Just edit these constants in [buffered_serial.h](#)

```
#define BUFFERED_SERIAL_SERIALS_QUANTITY 1
#define BUFFERED_SERIAL_BUFFERS_SIZE 500
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Serial buffers size and quantity	9
Error handling	10
Serial buffers	11

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

buffered_serial_serial_descriptor	
Meta data of a buffered serial	13

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

buffered_serial.c	Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings	15
buffered_serial.h	Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings	20

Chapter 5

Module Documentation

5.1 Serial buffers size and quantity

Constants to configure the quantity of serials and the size of their buffers.

Macros

- `#define BUFFERED_SERIAL_SERIALS_QUANTITY 1`
- `#define BUFFERED_SERIAL_BUFFERS_SIZE 500`

5.1.1 Detailed Description

Constants to configure the quantity of serials and the size of their buffers.

5.2 Error handling

Error codes.

Macros

- `#define BUFFERED_SERIAL_ERROR_CODE_STATIC_STRINGS_ERROR 0`
- `#define BUFFERED_SERIAL_ERROR_CODE_NO_LINE_ENDING_DETECTED 1`

Variables

- `uint8_t buffered_serial_error_code`
Global variable to store error code.

5.2.1 Detailed Description

Error codes.

5.2.2 Variable Documentation

5.2.2.1 buffered_serial_error_code

```
uint8_t buffered_serial_error_code
```

Global variable to store error code.

```
static_strings_error_code
```

5.3 Serial buffers

rx and tx buffers to receive and transmit data.

Variables

- `uint8_t buffered_serial_rx_buffers` [BUFFERED_SERIAL_SERIALIZES_QUANTITY][BUFFERED_SERIAL_BUFFERS_SIZE]
- `uint8_t buffered_serial_tx_buffers` [BUFFERED_SERIAL_SERIALIZES_QUANTITY][BUFFERED_SERIAL_BUFFERS_SIZE]

5.3.1 Detailed Description

rx and tx buffers to receive and transmit data.

Chapter 6

Data Structure Documentation

6.1 buffered_serial_serial_descriptor Struct Reference

Meta data of a buffered serial.

```
#include <buffered_serial.h>
```

Data Fields

- UART_HandleTypeDef * **huart**
- uint8_t * **rx_buffer**
- uint8_t * [rx_buffer_data_start](#)
- uint8_t * [rx_buffer_data_finish](#)
- uint8_t * **tx_buffer**

6.1.1 Detailed Description

Meta data of a buffered serial.

6.1.2 Field Documentation

6.1.2.1 rx_buffer_data_finish

```
uint8_t* rx_buffer_data_finish
```

Pointer to the position ahead the last readable character on buffer.

6.1.2.2 rx_buffer_data_start

```
uint8_t* rx_buffer_data_start
```

Pointer to the first readable character on the buffer.

The documentation for this struct was generated from the following file:

- [buffered_serial.h](#)

Chapter 7

File Documentation

7.1 buffered_serial.c File Reference

Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings.

```
#include "buffered_serial.h"
```

Functions

- void [buffered_serial_init](#) (UART_HandleTypeDef **huarts)
Link huarts and buffers with serial descriptors and init rx data receiving and idle interrupt. Also init the Static Strings library.
- [buffered_serial_serial_descriptor](#) * [buffered_serial_get_huart_serial_descriptor](#) (UART_HandleTypeDef *huart)
Returns the serial_descriptor of the provided huart.
- uint16_t [buffered_serial_available](#) ([buffered_serial_serial_descriptor](#) *serial)
Calculates and returns the number of characters that can be read from the rx buffer.
- void [buffered_serial_print_character](#) ([buffered_serial_serial_descriptor](#) *serial, uint8_t character)
Transmit a character with the specific huart in the serial descriptor.
- void [buffered_serial_print_string](#) ([buffered_serial_serial_descriptor](#) *serial, static_strings_string_descriptor *string_descriptor)
Transmit a string with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFER↔S_SIZE will be transmitted in blocks of that size.
- int [buffered_serial_print_line](#) ([buffered_serial_serial_descriptor](#) *serial, static_strings_string_descriptor *string_descriptor)
Add new line (\r\n) to the string and then transmit it with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFERS_SIZE will be transmitted in blocks of that size.
- static_strings_string_descriptor * [buffered_serial_read_line](#) ([buffered_serial_serial_descriptor](#) *serial)
Read a string in the specific huart buffer in the serial descriptor. String must have \r\n line ending.
- void [buffered_serial_update_rx_buffer_data](#) (UART_HandleTypeDef *huart)
When IDLE line interruption is fired this function updates the rx buffer meta data.
- void [HAL_UART_ErrorCallback](#) (UART_HandleTypeDef *huart)

7.1.1 Detailed Description

Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings.

7.1.2 Function Documentation

7.1.2.1 buffered_serial_available()

```
uint16_t buffered_serial_available (
    buffered_serial_serial_descriptor * serial )
```

Calculates and returns the number of characters that can be read from the rx buffer.

```
uint16_t buffered_serial_available(buffered_serial_serial_descriptor *serial)
```

Parameters

<i>serial</i>	Pointer to the serial descriptor of the target huart.
---------------	---

Returns

Number of characters that can be read from the rx buffer.

7.1.2.2 buffered_serial_get_huart_serial_descriptor()

```
buffered_serial_serial_descriptor* buffered_serial_get_huart_serial_descriptor (
    UART_HandleTypeDef * huart )
```

Returns the serial_descriptor of the provided huart.

```
buffered_serial_serial_descriptor buffered_serial_get_huart_serial_descriptor(UART_HandleTypeDef *huart)
```

Parameters

<i>huart</i>	Pointer to a UART_HandleTypeDef.
--------------	----------------------------------

Returns

A pointer to the serial descriptor of the provided huart. Return NULL if there is no serial descriptor attached to the huart provided.

7.1.2.3 buffered_serial_init()

```
void buffered_serial_init (
    UART_HandleTypeDef ** huarts )
```

Link huarts and buffers with serial descriptors and init rx data receiving and idle interrupt. Also init the Static Strings library.

```
void buffered_serial_init(UART_HandleTypeDef **huarts)
```

Parameters

<i>huarts</i>	Array of pointers to huart pointer.
---------------	-------------------------------------

7.1.2.4 buffered_serial_print_character()

```
void buffered_serial_print_character (
    buffered_serial_serial_descriptor * serial,
    uint8_t character )
```

Transmit a character with the specific huart in the serial descriptor.

```
void buffered_serial_print_character(buffered_serial_serial_descriptor *serial,uint8_t character)
```

Parameters

<i>character</i>	character to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

7.1.2.5 buffered_serial_print_line()

```
int buffered_serial_print_line (
    buffered_serial_serial_descriptor * serial,
    static_strings_string_descriptor * string_descriptor )
```

Add new line (\r\n) to the string and then transmit it with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFERS_SIZE will be transmitted in blocks of that size.

```
int buffered_serial_print_line(static_strings_string_descriptor *string,buffered_serial_serial_descriptor *serial)
```

Parameters

<i>string_descriptor</i>	Pointer to the descriptor of the string to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

Returns

Return 0 if success and 1 if error.

7.1.2.6 buffered_serial_print_string()

```
void buffered_serial_print_string (
    buffered_serial_serial_descriptor * serial,
    static_strings_string_descriptor * string_descriptor )
```

Transmit a string with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFER_SIZE will be transmitted in blocks of that size.

```
void buffered_serial_print_string(static_strings_string_descriptor *string, buffered_serial_serial_descriptor *serial)
```

Parameters

<i>string_descriptor</i>	Pointer to the descriptor of the string to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

7.1.2.7 buffered_serial_read_line()

```
static_strings_string_descriptor* buffered_serial_read_line (
    buffered_serial_serial_descriptor * serial )
```

Read a string in the specific huart buffer in the serial descriptor. String must have \r\n line ending.

```
static_strings_string_descriptor *buffered_serial_read_line(buffered_serial_serial_descriptor *serial)
```

Parameters

<i>serial</i>	Pointer to the serial descriptor of the target huart.
---------------	---

Returns

Pointer to the string descriptor of the line read (See library Static Strings), if NULL check buffered_serial_error_code.

7.1.2.8 buffered_serial_update_rx_buffer_data()

```
void buffered_serial_update_rx_buffer_data (
    UART_HandleTypeDef * huart )
```

When IDLE line interruption is fired this function updates the rx buffer meta data.

```
void buffered_serial_update_rx_buffer_data(UART_HandleTypeDef *huart)
```

Parameters

<i>Pointer</i>	to the huart IDLE line interruption source.
----------------	---

7.2 buffered_serial.h File Reference

Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings.

```
#include "stm32f1xx_hal.h"
#include "stm32f1xx_hal_uart.h"
#include "static_strings.h"
```

Data Structures

- struct [buffered_serial_serial_descriptor](#)
Meta data of a buffered serial.

Macros

- #define **BUFFERED_SERIAL_SERIALS_QUANTITY** 1
- #define **BUFFERED_SERIAL_BUFFERS_SIZE** 500
- #define **BUFFERED_SERIAL_ERROR_CODE_STATIC_STRINGS_ERROR** 0
- #define **BUFFERED_SERIAL_ERROR_CODE_NO_LINE_ENDING_DETECTED** 1

Typedefs

- typedef struct [buffered_serial_serial_descriptor](#) **buffered_serial_serial_descriptor**

Functions

- void [buffered_serial_init](#) (UART_HandleTypeDef **huarts)
Link huarts and buffers with serial descriptors and init rx data receiving and idle interrupt. Also init the Static Strings library.
- [buffered_serial_serial_descriptor](#) * [buffered_serial_get_huart_serial_descriptor](#) (UART_HandleTypeDef *huart)
Returns the serial_descriptor of the provided huart.
- uint16_t [buffered_serial_available](#) ([buffered_serial_serial_descriptor](#) *serial)
Calculates and returns the number of characters that can be read from the rx buffer.
- void [buffered_serial_print_character](#) ([buffered_serial_serial_descriptor](#) *serial, uint8_t character)
Transmit a character with the specific huart in the serial descriptor.
- void [buffered_serial_print_string](#) ([buffered_serial_serial_descriptor](#) *serial, static_strings_string_descriptor *string_descriptor)
Transmit a string with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFER↵S_SIZE will be transmitted in blocks of that size.
- int [buffered_serial_print_line](#) ([buffered_serial_serial_descriptor](#) *serial, static_strings_string_descriptor *string_descriptor)
Add new line (\r\n) to the string and then transmit it with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFERS_SIZE will be transmitted in blocks of that size.
- static_strings_string_descriptor * [buffered_serial_read_line](#) ([buffered_serial_serial_descriptor](#) *serial)
Read a string in the specific huart buffer in the serial descriptor. String must have \r\n line ending.
- void [buffered_serial_update_rx_buffer_data](#) (UART_HandleTypeDef *huart)
When IDLE line interruption is fired this function updates the rx buffer meta data.

Variables

- `uint8_t buffered_serial_error_code`
Global variable to store error code.
- `uint8_t buffered_serial_rx_buffers` [BUFFERED_SERIAL_SERIALS_QUANTITY][BUFFERED_SERIAL_BUFFERS_SIZE]
- `uint8_t buffered_serial_tx_buffers` [BUFFERED_SERIAL_SERIALS_QUANTITY][BUFFERED_SERIAL_BUFFERS_SIZE]
- `buffered_serial_serial_descriptor buffered_serial_serial_descriptors` [BUFFERED_SERIAL_SERIALS_QUANTITY]

7.2.1 Detailed Description

Serial communication based on a circular buffer, dma and huart with hal controls and Static Strings.

7.2.2 Function Documentation

7.2.2.1 buffered_serial_available()

```
uint16_t buffered_serial_available (
    buffered_serial_serial_descriptor * serial )
```

Calculates and returns the number of characters that can be read from the rx buffer.

```
uint16_t buffered_serial_available(buffered_serial_serial_descriptor *serial)
```

Parameters

<i>serial</i>	Pointer to the serial descriptor of the target huart.
---------------	---

Returns

Number of characters that can be read from the rx buffer.

7.2.2.2 buffered_serial_get_huart_serial_descriptor()

```
buffered_serial_serial_descriptor* buffered_serial_get_huart_serial_descriptor (
    UART_HandleTypeDef * huart )
```

Returns the serial_descriptor of the provided huart.

```
buffered_serial_serial_descriptor buffered_serial_get_huart_serial_descriptor(UART_HandleTypeDef *huart)
```

Parameters

<i>huart</i>	Pointer to a UART_HandleTypeDef.
--------------	----------------------------------

Returns

A pointer to the serial descriptor of the provided huart. Return NULL if there is no serial descriptor attached to the huart provided.

7.2.2.3 buffered_serial_init()

```
void buffered_serial_init (
    UART_HandleTypeDef ** huarts )
```

Link huarts and buffers with serial descriptors and init rx data receiving and idle interrupt. Also init the Static Strings library.

```
void buffered_serial_init(UART_HandleTypeDef **huarts)
```

Parameters

<i>huarts</i>	Array of pointers to huart pointer.
---------------	-------------------------------------

7.2.2.4 buffered_serial_print_character()

```
void buffered_serial_print_character (
    buffered_serial_serial_descriptor * serial,
    uint8_t character )
```

Transmit a character with the specific huart in the serial descriptor.

```
void buffered_serial_print_character(buffered_serial_serial_descriptor *serial,uint8_t character)
```

Parameters

<i>character</i>	character to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

7.2.2.5 buffered_serial_print_line()

```
int buffered_serial_print_line (
    buffered_serial_serial_descriptor * serial,
    static_strings_string_descriptor * string_descriptor )
```

Add new line (\r\n) to the string and then transmit it with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFERS_SIZE will be transmitted in blocks of that size.

```
int buffered_serial_print_line(static_strings_string_descriptor *string,buffered_serial_serial_descriptor *serial)
```

Parameters

<i>string_descriptor</i>	Pointer to the descriptor of the string to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

Returns

Return 0 if success and 1 if error.

7.2.2.6 buffered_serial_print_string()

```
void buffered_serial_print_string (
    buffered_serial_serial_descriptor * serial,
    static_strings_string_descriptor * string_descriptor )
```

Transmit a string with the specific huart in the serial descriptor. Strings larger than BUFFERED_SERIAL_BUFFERS_SIZE will be transmitted in blocks of that size.

```
void buffered_serial_print_string(static_strings_string_descriptor *string,buffered_serial_serial_descriptor *serial)
```

Parameters

<i>string_descriptor</i>	Pointer to the descriptor of the string to transmit.
<i>serial</i>	Pointer to the serial descriptor of the target huart.

7.2.2.7 buffered_serial_read_line()

```
static_strings_string_descriptor* buffered_serial_read_line (
    buffered_serial_serial_descriptor * serial )
```

Read a string in the specific huart buffer in the serial descriptor. String must have \r\n line ending.

```
static_strings_string_descriptor *buffered_serial_read_line(buffered_serial_serial_descriptor *serial)
```

Parameters

<i>serial</i>	Pointer to the serial descriptor of the target huart.
---------------	---

Returns

Pointer to the string descriptor of the line read (See library Static Strings), if NULL check buffered_serial_↔ error_code.

7.2.2.8 buffered_serial_update_rx_buffer_data()

```
void buffered_serial_update_rx_buffer_data (
    UART_HandleTypeDef * huart )
```

When IDLE line interruption is fired this function updates the rx buffer meta data.

```
void buffered_serial_update_rx_buffer_data(UART_HandleTypeDef *huart)
```

Parameters

<i>Pointer</i>	to the huart IDLE line interruption source.
----------------	---

Index

- buffered_serial.c, [15](#)
 - buffered_serial_available, [16](#)
 - buffered_serial_get_huart_serial_descriptor, [16](#)
 - buffered_serial_init, [16](#)
 - buffered_serial_print_character, [18](#)
 - buffered_serial_print_line, [18](#)
 - buffered_serial_print_string, [18](#)
 - buffered_serial_read_line, [19](#)
 - buffered_serial_update_rx_buffer_data, [19](#)
- buffered_serial.h, [20](#)
 - buffered_serial_available, [21](#)
 - buffered_serial_get_huart_serial_descriptor, [21](#)
 - buffered_serial_init, [22](#)
 - buffered_serial_print_character, [22](#)
 - buffered_serial_print_line, [22](#)
 - buffered_serial_print_string, [23](#)
 - buffered_serial_read_line, [23](#)
 - buffered_serial_update_rx_buffer_data, [24](#)
- buffered_serial_available
 - buffered_serial.c, [16](#)
 - buffered_serial.h, [21](#)
- buffered_serial_error_code
 - Error handling, [10](#)
- buffered_serial_get_huart_serial_descriptor
 - buffered_serial.c, [16](#)
 - buffered_serial.h, [21](#)
- buffered_serial_init
 - buffered_serial.c, [16](#)
 - buffered_serial.h, [22](#)
- buffered_serial_print_character
 - buffered_serial.c, [18](#)
 - buffered_serial.h, [22](#)
- buffered_serial_print_line
 - buffered_serial.c, [18](#)
 - buffered_serial.h, [22](#)
- buffered_serial_print_string
 - buffered_serial.c, [18](#)
 - buffered_serial.h, [23](#)
- buffered_serial_read_line
 - buffered_serial.c, [19](#)
 - buffered_serial.h, [23](#)
- buffered_serial_serial_descriptor, [13](#)
 - rx_buffer_data_finish, [13](#)
 - rx_buffer_data_start, [13](#)
- buffered_serial_update_rx_buffer_data
 - buffered_serial.c, [19](#)
 - buffered_serial.h, [24](#)
- rx_buffer_data_finish
 - buffered_serial_serial_descriptor, [13](#)
- rx_buffer_data_start
 - buffered_serial_serial_descriptor, [13](#)
- Serial buffers, [11](#)
- Serial buffers size and quantity, [9](#)
- Error handling, [10](#)
 - buffered_serial_error_code, [10](#)