

Circular Buffer Library

Generated by Doxygen 1.8.17

1 Circular Buffer Library	1
1.1 Getting started	1
1.2 Running the tests	1
1.3 Acknowledgments	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 circular_buf_t Struct Reference	7
4.1.1 Detailed Description	7
4.2 circular_buffer< T > Class Template Reference	7
4.2.1 Detailed Description	8
4.2.2 Constructor & Destructor Documentation	8
4.2.2.1 circular_buffer()	8
4.2.3 Member Function Documentation	8
4.2.3.1 capacity()	8
4.2.3.2 empty()	9
4.2.3.3 full()	9
4.2.3.4 get()	9
4.2.3.5 put()	10
4.2.3.6 size()	10
5 File Documentation	11
5.1 C_implementation/include/CircularBuffer.h File Reference	11
5.1.1 Detailed Description	12
5.1.2 Function Documentation	12
5.1.2.1 circular_buf_capacity()	12
5.1.2.2 circular_buf_empty()	12
5.1.2.3 circular_buf_free()	13
5.1.2.4 circular_buf_full()	13
5.1.2.5 circular_buf_get()	14
5.1.2.6 circular_buf_init()	14
5.1.2.7 circular_buf_put()	15
5.1.2.8 circular_buf_reset()	15
5.1.2.9 circular_buf_size()	15
5.2 Cpp_implementation/include/CircularBuffer.h File Reference	16
5.2.1 Detailed Description	16
5.3 C_implementation/src/CircularBuffer.c File Reference	16
5.3.1 Detailed Description	17
5.3.2 Function Documentation	17

5.3.2.1 circular_buf_capacity()	17
5.3.2.2 circular_buf_empty()	18
5.3.2.3 circular_buf_free()	18
5.3.2.4 circular_buf_full()	19
5.3.2.5 circular_buf_get()	19
5.3.2.6 circular_buf_init()	20
5.3.2.7 circular_buf_put()	20
5.3.2.8 circular_buf_reset()	21
5.3.2.9 circular_buf_size()	21
5.4 Cpp_implementation/src/CircularBuffer.ipp File Reference	22
5.4.1 Detailed Description	22

Index	23
--------------	-----------

Chapter 1

Circular Buffer Library

Circular buffer library written in C and C++.

1.1 Getting started

Clone repository with:

```
git clone https://github.com/jpare006/Circular-Buffer-Library.git
```

Since submodules are used, run the following command to initialize them:

```
git submodule update --init --recursive
```

1.2 Running the tests

Specific information on running the tests can be found in the respective directory.

1.3 Acknowledgments

- [Unity](#) test framework. GtiHub [repo](#)
- [CppUTest](#) test framework. Github [repo](#)
- EmbeddedArtistry for their [tutorial](#) on creating a circular buffer.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

circular_buf_t	7
circular_buffer< T >	7

Chapter 3

File Index

3.1 File List

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

C_implementation/include/ CircularBuffer.h	
Circular buffer library for embedded systems using uint8_t	11
C_implementation/src/ CircularBuffer.c	
Circular buffer library for embedded systems. Supports uint8_t elements. Functions can be easily modified to support the required data type	16
Cpp_implementation/include/ CircularBuffer.h	
Circular buffer library for embedded systems. Uses class template to support various data types	16
Cpp_implementation/src/ CircularBuffer.hpp	
Circular buffer library for embedded systems. Uses class template to support various data types	22

Chapter 4

Class Documentation

4.1 circular_buf_t Struct Reference

Public Attributes

- uint8_t * **p_buffer**
- size_t **head**
- size_t **tail**
- size_t **max**
- BOOL **b_is_buffer_full**

4.1.1 Detailed Description

Definition at line 11 of file CircularBuffer.c.

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

- C_implementation/src/[CircularBuffer.c](#)

4.2 circular_buffer< T > Class Template Reference

Public Member Functions

- size_t [capacity](#) ()
Find the maximum number of elements the circular buffer can store.
- [circular_buffer](#) (size_t s)
Circular buffer constructor.
- [~circular_buffer](#) ()
Circular buffer destructor.
- bool [full](#) ()
Check if the circular buffer is full or not.
- bool [empty](#) ()
- void [put](#) (T data)
Place a single data element into the underlying buffer.
- T [get](#) ()
Read the next value in the circular buffer.
- void [reset](#) ()
Reset the circular buffer to its initial state.
- size_t [size](#) ()
Calculate the current number of elements in the circular buffer.

4.2.1 Detailed Description

```
template<class T>
class circular_buffer< T >
```

Definition at line 12 of file CircularBuffer.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 circular_buffer()

```
template<class T >
circular_buffer< T >::circular_buffer (
    size_t s )
```

Circular buffer constructor.

Parameters

in	s	Size with which to initialize buffer
----	---	--------------------------------------

Definition at line 14 of file CircularBuffer.ipp.

```
15 {
16     max_size = s;
17     p_buffer = new T[s];
18     b_is_buffer_full = false;
19     head = 0;
20     tail = 0;
21 }
```

4.2.3 Member Function Documentation

4.2.3.1 capacity()

```
template<class T >
size_t circular_buffer< T >::capacity
```

Find the maximum number of elements the circular buffer can store.

Returns

Maximum size of the circular buffer.

Definition at line 37 of file CircularBuffer.ipp.

```
38 {
39     return max_size;
40 }
```

4.2.3.2 empty()

```
template<class T >
bool circular_buffer< T >::empty
```

@brief Check if circular buffer is empty or not.

Returns

Boolean value answering if circular buffer is empty or not.

Definition at line 57 of file CircularBuffer.ipp.

```
58 {
59     if ((head == tail) && !b_is_buffer_full)
60     {
61         return true;
62     }
63
64     return false;
65 }
```

4.2.3.3 full()

```
template<class T >
bool circular_buffer< T >::full
```

Check if the circular buffer is full or not.

Returns

Boolean value answering if circular buffer is full or not.

Definition at line 47 of file CircularBuffer.ipp.

```
48 {
49     return b_is_buffer_full;
50 }
```

4.2.3.4 get()

```
template<class T >
T circular_buffer< T >::get
```

Read the next value in the circular buffer.

Returns

Value read from buffer.

Definition at line 84 of file CircularBuffer.ipp.

```
85 {
86     if (empty())
87     {
88         return T();
89     }
90
91     T value = p_buffer[head];
92     advance_head();
93
94     return value;
95 }
```

4.2.3.5 put()

```
template<class T >
void circular_buffer< T >::put (
    T data )
```

Place a single data element into the underlying buffer.

Parameters

in	<i>data</i>	Data element to be stored in circular buffer.
----	-------------	---

Definition at line 72 of file CircularBuffer.ipp.

```
73 {
74     p_buffer[tail] = data;
75
76     advance_tail();
77 }
```

4.2.3.6 size()

```
template<class T >
size_t circular_buffer< T >::size
```

Calculate the current number of elements in the circular buffer.

Returns

Number of elements currently in circular buffer.

Definition at line 113 of file CircularBuffer.ipp.

```
114 {
115     size_t value = -1;
116
117     if (b_is_buffer_full)
118     {
119         value = max_size;
120     }
121     else if (tail > head)
122     {
123         value = tail - head;
124     }
125     else if (tail < head)
126     {
127         //Use buffer max value to calculate amount of elements
128         value = (max_size - head) + tail;
129     }
130     else
131     {
132         //At this point, the only option left is: (tail == head == 0) && !full
133         value = 0;
134     }
135
136     return value;
137 }
```

The documentation for this class was generated from the following files:

- Cpp_implementation/include/CircularBuffer.h
- Cpp_implementation/src/CircularBuffer.ipp

Chapter 5

File Documentation

5.1 C_implementation/include/CircularBuffer.h File Reference

Circular buffer library for embedded systems using uint8_t.

```
#include <stdint.h>
#include <stdlib.h>
```

Typedefs

- typedef int **BOOL**
- typedef struct [circular_buf_t](#) **circular_buf_t**
- typedef [circular_buf_t](#) * **cbuf_handle_t**

Functions

- [cbuf_handle_t](#) [circular_buf_init](#) (uint8_t *buffer, size_t size)
Initialize the circular buffer structure with user declared buffer and size.
- BOOL [circular_buf_empty](#) ([cbuf_handle_t](#) cbuf)
Check if circular buffer is empty or not.
- BOOL [circular_buf_full](#) ([cbuf_handle_t](#) cbuf)
Check if circular buffer is full or not.
- void [circular_buf_put](#) ([cbuf_handle_t](#) cbuf, uint8_t data)
Place a single data element into the underlying buffer.
- size_t [circular_buf_capacity](#) ([cbuf_handle_t](#) cbuf)
Find the maximum number of elements the circular buffer can store.
- void [circular_buf_reset](#) ([cbuf_handle_t](#) cbuf)
Reset the circular buffer to its initial state.
- int [circular_buf_get](#) ([cbuf_handle_t](#) cbuf, uint8_t *p_value)
Read the next value in the circular buffer.
- size_t [circular_buf_size](#) ([cbuf_handle_t](#) cbuf)
Calculate the current number of elements in the circular buffer.
- int [circular_buf_free](#) ([cbuf_handle_t](#) cbuf)
Free memory that was allocated during initialization.

5.1.1 Detailed Description

Circular buffer library for embedded systems using uint8_t.

5.1.2 Function Documentation

5.1.2.1 circular_buf_capacity()

```
size_t circular_buf_capacity (  
    cbuf_handle_t cbuf )
```

Find the maximum number of elements the circular buffer can store.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Maximum size of the circular buffer.

Definition at line 122 of file CircularBuffer.c.

```
123 {  
124     return cbuf->max;  
125 }
```

5.1.2.2 circular_buf_empty()

```
BOOL circular_buf_empty (  
    cbuf_handle_t cbuf )
```

Check if circular buffer is empty or not.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Boolean value answering if circular buffer is empty or not.

Definition at line 83 of file CircularBuffer.c.

```
84 {  
85     if ((cbuf->head == cbuf->tail) && !cbuf->b_is_buffer_full)  
86     {  
87         return TRUE;  
88     }
```



```

88     }
89
90     return FALSE;
91 }

```

5.1.2.3 circular_buf_free()

```

int circular_buf_free (
    cbuf_handle_t cbuf )

```

Free memory that was allocated during initialization.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

It is the users responsibility to call this function at the end of the program once for every time `circular_buf_init` was called.

Definition at line 198 of file CircularBuffer.c.

```

199 {
200     free(cbuf);
201
202     return 0;
203 }

```

5.1.2.4 circular_buf_full()

```

BOOL circular_buf_full (
    cbuf_handle_t cbuf )

```

Check if circular buffer is full or not.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Boolean value answering if circular buffer is full or not.

Definition at line 98 of file CircularBuffer.c.

```

99 {
100     // b_is_buffer_full is updated in other functions, therefore, this function
101     // simply returns whatever value is stored in the struct.
102     return cbuf->b_is_buffer_full;
103 }

```

5.1.2.5 circular_buf_get()

```
int circular_buf_get (
    cbuf_handle_t cbuf,
    uint8_t * p_value )
```

Read the next value in the circular buffer.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
in	<i>p_value</i>	Pointer to var where value that is read is to be stored.

Returns

Status indicating wether read was succesful (0) or not (-1).

Definition at line 144 of file CircularBuffer.c.

```
145 {
146     int status = -1;
147
148     if(!circular_buf_empty(cbuf))
149     {
150         status = 0;
151
152         *p_value = cbuf->p_buffer[cbuf->head];
153         advance_head(cbuf);
154     }
155
156     return status;
157 }
```

5.1.2.6 circular_buf_init()

```
cbuf_handle_t circular_buf_init (
    uint8_t * p_buffer,
    size_t size )
```

Initialize the circular buffer structure with user declared buffer and size.

Parameters

in	<i>p_buffer</i>	A pointer to the user declared buffer.
in	<i>size</i>	The size of the buffer.

Returns

The handle type used to access the circular buffer internals.

Definition at line 68 of file CircularBuffer.c.

```
69 {
70     cbuf_handle_t cbuf = malloc(sizeof(circular_buf_t));
71     cbuf->p_buffer = p_buffer;
72     cbuf->max = size;
73     circular_buf_reset(cbuf);
74 }
```

```
75     return cbuf;
76 }
```

5.1.2.7 circular_buf_put()

```
void circular_buf_put (
    cbuf_handle_t cbuf,
    uint8_t data )
```

Place a single data element into the underlying buffer.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
in	<i>data</i>	Data element to be stored in circular buffer.

Definition at line 110 of file CircularBuffer.c.

```
111 {
112     cbuf->p_buffer[cbuf->tail] = data;
113
114     advance_tail(cbuf);
115 }
```

5.1.2.8 circular_buf_reset()

```
void circular_buf_reset (
    cbuf_handle_t cbuf )
```

Reset the circular buffer to its initial state.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Definition at line 131 of file CircularBuffer.c.

```
132 {
133     cbuf->head = 0;
134     cbuf->tail = 0;
135     cbuf->b_is_buffer_full = FALSE;
136 }
```

5.1.2.9 circular_buf_size()

```
size_t circular_buf_size (
    cbuf_handle_t cbuf )
```

Calculate the current number of elements in the circular buffer.

Parameters

<code>in</code>	<code>cbuf</code>	Handle for circular buffer.
-----------------	-------------------	-----------------------------

Returns

Number of elements currently in circular buffer.

Definition at line 164 of file CircularBuffer.c.

```

165 {
166     size_t value = 0;
167
168     if(circular_buf_full(cbuf))
169     {
170         value = cbuf->max;
171     }
172     else if(cbuf->tail > cbuf->head)
173     {
174         value = cbuf->tail - cbuf->head;
175     }
176     else if (cbuf->tail < cbuf->head)
177     {
178         //Use buffer max value to calculate amount of elements
179         value = (cbuf->max - cbuf->head) + cbuf->tail;
180     }
181     else
182     {
183         //At this point, the only option left is: (tail == head == 0) && !full
184         value = 0;
185     }
186
187     return value;
188 }
```

5.2 Cpp_implementation/include/CircularBuffer.h File Reference

Circular buffer library for embedded systems. Uses class template to support various data types.

```
#include "CircularBuffer.ipp"
```

Classes

- class `circular_buffer< T >`

5.2.1 Detailed Description

Circular buffer library for embedded systems. Uses class template to support various data types.

5.3 C_implementation/src/CircularBuffer.c File Reference

Circular buffer library for embedded systems. Supports uint8_t elements. Functions can be easily modified to support the required data type.

```
#include "CircularBuffer.h"
```

Classes

- struct [circular_buf_t](#)

Enumerations

- enum { **TRUE** = 1, **FALSE** = 0 }

Functions

- [cbuf_handle_t circular_buf_init](#) (uint8_t *p_buffer, size_t size)
Initialize the circular buffer structure with user declared buffer and size.
- [BOOL circular_buf_empty](#) ([cbuf_handle_t](#) cbuf)
Check if circular buffer is empty or not.
- [BOOL circular_buf_full](#) ([cbuf_handle_t](#) cbuf)
Check if circular buffer is full or not.
- [void circular_buf_put](#) ([cbuf_handle_t](#) cbuf, uint8_t data)
Place a single data element into the underlying buffer.
- [size_t circular_buf_capacity](#) ([cbuf_handle_t](#) cbuf)
Find the maximum number of elements the circular buffer can store.
- [void circular_buf_reset](#) ([cbuf_handle_t](#) cbuf)
Reset the circular buffer to its initial state.
- [int circular_buf_get](#) ([cbuf_handle_t](#) cbuf, uint8_t *p_value)
Read the next value in the circular buffer.
- [size_t circular_buf_size](#) ([cbuf_handle_t](#) cbuf)
Calculate the current number of elements in the circular buffer.
- [int circular_buf_free](#) ([cbuf_handle_t](#) cbuf)
Free memory that was allocated during initialization.

5.3.1 Detailed Description

Circular buffer library for embedded systems. Supports uint8_t elements. Functions can be easily modified to support the required data type.

5.3.2 Function Documentation

5.3.2.1 [circular_buf_capacity\(\)](#)

```
size_t circular_buf_capacity (  
    cbuf\_handle\_t cbuf )
```

Find the maximum number of elements the circular buffer can store.

Parameters

in	<i>cbuf</i>	Handle for ciruclar buffer.
----	-------------	-----------------------------

Returns

Maximum size of the circular buffer.

Definition at line 122 of file CircularBuffer.c.

```
123 {  
124     return cbuf->max;  
125 }
```

5.3.2.2 circular_buf_empty()

```
BOOL circular_buf_empty (  
    cbuf_handle_t cbuf )
```

Check if circular buffer is empty or not.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Boolean value answering if circular buffer is empty or not.

Definition at line 83 of file CircularBuffer.c.

```
84 {  
85     if ((cbuf->head == cbuf->tail) && !cbuf->b_is_buffer_full)  
86     {  
87         return TRUE;  
88     }  
89     return FALSE;  
90 }  
91 }
```

5.3.2.3 circular_buf_free()

```
int circular_buf_free (  
    cbuf_handle_t cbuf )
```

Free memory that was allocated during initialization.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

It is the users responsibility to call this function at the end of the program once for every time `circular_buf_init` was called.

Definition at line 198 of file CircularBuffer.c.

```
199 {  
200     free(cbuf);  
201  
202     return 0;  
203 }
```

5.3.2.4 circular_buf_full()

```
BOOL circular_buf_full (  
    cbuf_handle_t cbuf )
```

Check if circular buffer is full or not.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Boolean value answering if circular buffer is full or not.

Definition at line 98 of file CircularBuffer.c.

```
99 {  
100     // b_is_buffer_full is updated in other functions, therefore, this function  
101     // simply returns whatever value is stored in the struct.  
102     return cbuf->b_is_buffer_full;  
103 }
```

5.3.2.5 circular_buf_get()

```
int circular_buf_get (  
    cbuf_handle_t cbuf,  
    uint8_t * p_value )
```

Read the next value in the circular buffer.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
in	<i>p_value</i>	Pointer to var where value that is read is to be stored.

Returns

Status indicating wether read was succesful (0) or not (-1).

Definition at line 144 of file CircularBuffer.c.

```

145 {
146     int status = -1;
147
148     if(!circular_buf_empty(cbuf))
149     {
150         status = 0;
151
152         *p_value = cbuf->p_buffer[cbuf->head];
153         advance_head(cbuf);
154     }
155
156     return status;
157 }
```

5.3.2.6 circular_buf_init()

```

cbuf_handle_t circular_buf_init (
    uint8_t * p_buffer,
    size_t size )
```

Initialize the circular buffer structure with user declared buffer and size.

Parameters

in	<i>p_buffer</i>	A pointer to the user declared buffer.
in	<i>size</i>	The size of the buffer.

Returns

The handle type used to access the circular buffer internals.

Definition at line 68 of file CircularBuffer.c.

```

69 {
70     cbuf_handle_t cbuf = malloc(sizeof(circular_buf_t));
71     cbuf->p_buffer = p_buffer;
72     cbuf->max = size;
73     circular_buf_reset(cbuf);
74
75     return cbuf;
76 }
```

5.3.2.7 circular_buf_put()

```

void circular_buf_put (
    cbuf_handle_t cbuf,
    uint8_t data )
```

Place a single data element into the underlying buffer.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
in	<i>data</i>	Data element to be stored in circular buffer.

Definition at line 110 of file CircularBuffer.c.

```
111 {
112     cbuf->p_buffer[cbuf->tail] = data;
113
114     advance_tail(cbuf);
115 }
```

5.3.2.8 circular_buf_reset()

```
void circular_buf_reset (
    cbuf_handle_t cbuf )
```

Reset the circular buffer to its initial state.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Definition at line 131 of file CircularBuffer.c.

```
132 {
133     cbuf->head = 0;
134     cbuf->tail = 0;
135     cbuf->b_is_buffer_full = FALSE;
136 }
```

5.3.2.9 circular_buf_size()

```
size_t circular_buf_size (
    cbuf_handle_t cbuf )
```

Calculate the current number of elements in the circular buffer.

Parameters

in	<i>cbuf</i>	Handle for circular buffer.
----	-------------	-----------------------------

Returns

Number of elements currently in circular buffer.

Definition at line 164 of file CircularBuffer.c.

```
165 {
166     size_t value = 0;
167
168     if(circular_buf_full(cbuf))
169     {
170         value = cbuf->max;
171     }
172     else if(cbuf->tail > cbuf->head)
173     {
174         value = cbuf->tail - cbuf->head;
175     }
176     else if (cbuf->tail < cbuf->head)
177     {
178         //Use buffer max value to calculate amount of elements
```

```
179         value = (cbuf->max - cbuf->head) + cbuf->tail;
180     }
181     else
182     {
183         //At this point, the only option left is: (tail == head == 0) && !full
184         value = 0;
185     }
186
187     return value;
188 }
```

5.4 Cpp_implementation/src/CircularBuffer.ipp File Reference

Circular buffer library for embedded systems. Uses class template to support various data types.

```
#include "CircularBuffer.h"
```

5.4.1 Detailed Description

Circular buffer library for embedded systems. Uses class template to support various data types.

Index

C_implementation/include/CircularBuffer.h, [11](#)

C_implementation/src/CircularBuffer.c, [16](#)

capacity

 circular_buffer< T >, [8](#)

circular_buf_capacity

 CircularBuffer.c, [17](#)

 CircularBuffer.h, [12](#)

circular_buf_empty

 CircularBuffer.c, [18](#)

 CircularBuffer.h, [12](#)

circular_buf_free

 CircularBuffer.c, [18](#)

 CircularBuffer.h, [13](#)

circular_buf_full

 CircularBuffer.c, [19](#)

 CircularBuffer.h, [13](#)

circular_buf_get

 CircularBuffer.c, [19](#)

 CircularBuffer.h, [13](#)

circular_buf_init

 CircularBuffer.c, [20](#)

 CircularBuffer.h, [14](#)

circular_buf_put

 CircularBuffer.c, [20](#)

 CircularBuffer.h, [15](#)

circular_buf_reset

 CircularBuffer.c, [21](#)

 CircularBuffer.h, [15](#)

circular_buf_size

 CircularBuffer.c, [21](#)

 CircularBuffer.h, [15](#)

circular_buf_t, [7](#)

circular_buffer

 circular_buffer< T >, [8](#)

circular_buffer< T >, [7](#)

 capacity, [8](#)

 circular_buffer, [8](#)

 empty, [8](#)

 full, [9](#)

 get, [9](#)

 put, [9](#)

 size, [10](#)

CircularBuffer.c

 circular_buf_capacity, [17](#)

 circular_buf_empty, [18](#)

 circular_buf_free, [18](#)

 circular_buf_full, [19](#)

 circular_buf_get, [19](#)

 circular_buf_init, [20](#)

 circular_buf_put, [20](#)

 circular_buf_reset, [21](#)

 circular_buf_size, [21](#)

CircularBuffer.h

 circular_buf_capacity, [12](#)

 circular_buf_empty, [12](#)

 circular_buf_free, [13](#)

 circular_buf_full, [13](#)

 circular_buf_get, [13](#)

 circular_buf_init, [14](#)

 circular_buf_put, [15](#)

 circular_buf_reset, [15](#)

 circular_buf_size, [15](#)

Cpp_implementation/include/CircularBuffer.h, [16](#)

Cpp_implementation/src/CircularBuffer.ipp, [22](#)

empty

 circular_buffer< T >, [8](#)

full

 circular_buffer< T >, [9](#)

get

 circular_buffer< T >, [9](#)

put

 circular_buffer< T >, [9](#)

size

 circular_buffer< T >, [10](#)