Circular Buffer Library

Generated by Doxygen 1.8.17

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:

Chapter 3

File Index

3.1 File List

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

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.
- $\sim$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.

```
{
    max_size = s;
    p_buffer = new T[s];
    b_is_buffer_full = false;
    head = 0;
    tail = 0;
}
```

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.

```
{
    return max_size;
}
```

4.2.3.2 empty()

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

@breif 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.

```
{
    if ((head == tail) && !b_is_buffer_full)
    {
        return true;
    }

    return false;
}
```

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.

```
{
    return b_is_buffer_full;
}
```

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.

```
{
    if (empty())
    {
        return T();
    }

    T value = p_buffer[head];
    advance_head();

    return value;
}
```

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	*data*	Data element to be stored in circular buffer.

Definition at line 72 of file CircularBuffer.ipp.

```
{
    p_buffer[tail] = data;

    advance_tail();
}
```

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.

```
{
    size_t value = -1;

    if (b_is_buffer_full)
    {
        value = max_size;
    }
    else if (tail > head)
    {
        value = tail - head;
    }
    else if (tail < head)
    {
        //Use buffer max value to calculate amount of elements
        value = (max_size - head) + tail;
    }
    else
    {
        //At this point, the only option left is: (tail == head  == 0) && !full
        value = 0;
    }

    return value;
}
```

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	*cbuf*	Handle for ciruclar buffer.

Returns

Maximum size of the circular buffer.

Definition at line 122 of file CircularBuffer.c.

```
{
    return cbuf->max;
}
```

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	*cbuf*	Handle for circular buffer.

Returns

Boolean value answering if circular buffer is empty or not.

Definition at line 83 of file CircularBuffer.c.

```
{
    if((cbuf->head == cbuf->tail) && !cbuf->b_is_buffer_full)
    {
        return TRUE;
    }

    return FALSE;
}
```

5.1.2.3 circular_buf_free()

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

Free memory that was allocated during initialization.

Parameters

in	*cbuf*	Hanlde 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.

```
{
    free(cbuf);

    return 0;
}
```

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	*cbuf*	Handle for circular buffer.

Returns

Boolean value answering if circular buffer is full or not.

Definition at line 98 of file CircularBuffer.c.

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

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	*cbuf*	Handle for circular buffer.
in	*p_value*	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.

```
{
    int status = -1;

    if(!circular_buf_empty(cbuf))
    {
        status = 0;

        *p_value = cbuf->p_buffer[cbuf->head];
        advance_head(cbuf);
    }

    return status;
}
```

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	*p_buffer*	A pointer to the user declared buffer.
in	*size*	The size of the buffer.

Returns

The handle type used to access the circular buffer internals.

Definition at line 68 of file CircularBuffer.c.

```
{
    cbuf_handle_t cbuf = malloc(sizeof(circular_buf_t));
    cbuf->p_buffer = p_buffer;
    cbuf->max = size;
    circular_buf_reset(cbuf);

    return cbuf;
}
```

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	*cbuf*	Handle for circular buffer.
in	*data*	Data element to be stored in circular buffer.

Definition at line 110 of file CircularBuffer.c.

```
{
    cbuf->p_buffer[cbuf->tail] = data;

    advance_tail(cbuf);
}
```

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	*cbuf*	Handle for circular buffer.

Definition at line 131 of file CircularBuffer.c.

```
{
    cbuf->head = 0;
    cbuf->tail = 0;
    cbuf->b_is_buffer_full = FALSE;
}
```

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

in	*cbuf*	Handle for circular buffer.

Returns

Number of elements currently in circular buffer.

Definition at line 164 of file CircularBuffer.c.

```
{
    size_t value = 0;

    if(circular_buf_full(cbuf))
    {
        value = cbuf->max;
    }
    else if(cbuf->tail > cbuf->head)
    {
        value = cbuf->tail - cbuf->head;
    }
    else if (cbuf->tail < cbuf->head)
    {
        //Use buffer max value to calculate amount of elements
        value = (cbuf->max - cbuf->head) + cbuf->tail;
    }
    else
    {
        //At this point, the only option left is: (tail == head  == 0) && !full
        value = 0;
    }

    return value;
}
```

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	*cbuf*	Handle for ciruclar buffer.

Returns

Maximum size of the circular buffer.

Definition at line 122 of file CircularBuffer.c.
```
{
    return cbuf->max;
}
```

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	*cbuf*	Handle for circular buffer.

Returns

Boolean value answering if circular buffer is empty or not.

Definition at line 83 of file CircularBuffer.c.
```
{
    if((cbuf->head == cbuf->tail) && !cbuf->b_is_buffer_full)
    {
        return TRUE;
    }

    return FALSE;
}
```

5.3.2.3 circular_buf_free()

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

Free memory that was allocated during initialization.

Parameters

in	*cbuf*	Hanlde 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.

```
{
    free(cbuf);

    return 0;
}
```

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	*cbuf*	Handle for circular buffer.

Returns

Boolean value answering if circular buffer is full or not.

Definition at line 98 of file CircularBuffer.c.

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

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	*cbuf*	Handle for circular buffer.
in	*p_value*	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.

```
{
    int status = -1;

    if(!circular_buf_empty(cbuf))
    {
        status = 0;

        *p_value = cbuf->p_buffer[cbuf->head];
        advance_head(cbuf);
    }

    return status;
}
```

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	*p_buffer*	A pointer to the user declared buffer.
in	*size*	The size of the buffer.

Returns

The handle type used to access the circular buffer internals.

Definition at line 68 of file CircularBuffer.c.

```
{
    cbuf_handle_t cbuf = malloc(sizeof(circular_buf_t));
    cbuf->p_buffer = p_buffer;
    cbuf->max = size;
    circular_buf_reset(cbuf);

    return cbuf;
}
```

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	*cbuf*	Handle for circular buffer.
in	*data*	Data element to be stored in circular buffer.

Definition at line 110 of file CircularBuffer.c.

```
{
    cbuf->p_buffer[cbuf->tail] = data;

    advance_tail(cbuf);
}
```

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	*cbuf*	Handle for circular buffer.

Definition at line 131 of file CircularBuffer.c.

```
{
    cbuf->head = 0;
    cbuf->tail = 0;
    cbuf->b_is_buffer_full = FALSE;
}
```

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	*cbuf*	Handle for circular buffer.

Returns

Number of elements currently in circular buffer.

Definition at line 164 of file CircularBuffer.c.

```
{
    size_t value = 0;

    if(circular_buf_full(cbuf))
    {
        value = cbuf->max;
    }
    else if(cbuf->tail > cbuf->head)
    {
        value = cbuf->tail - cbuf->head;
    }
    else if (cbuf->tail < cbuf->head)
    {
        //Use buffer max value to calculate amount of elements
        value = (cbuf->max - cbuf->head) + cbuf->tail;
    }
    else
    {
        //At this point, the only option left is: (tail == head  == 0) && !full
        value = 0;
    }

    return value;
}
```

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