kBuffer

1.0

Generated by Doxygen 1.8.9.1

Sat May 14 2016 21:58:07

Contents

Chapter 1

Main Page

1.1 Introduction

kBuffer is a universal library for a ring- / circular buffer.

1.2 Functions and Datatypes

buffer_t
bufferStatus_t

bufferInit()

bufferWrite();
bufferRead();

bufferFill();
bufferIsFull()
bufferIsEmpty()

bufferWriteToIndex()
bufferReadFromIndex()

bufferMean()
bufferMeanRMS()

1.3 Usage

Have a look Fundamental Usage for an explenation of the main kBuffer functions (with some examples)
If you want to take the mean of your buffer, have a look at Mean of buffer

1.4 Example code

An example code project is available under ../../test/x86. It isn't well documented, but you can compile it for your system.

Chapter 2

Fundamental Usage

2.1 Buffer datatype definition

A ringbuffer consists of variables, which can be accessed in a continuous way.
You have to define, which datatype you want to have the elements.
By default, the elements are unsigned 16bit integers (uint16_t).
The datatype is defined in kBuffer.h :

```
#define bufferDatatype uint16_t
```

Instead of uint16_t, you can insert (almost) any datatype you want.

2.2 Initializing a ringbuffer

At first, you have to include the kBuffer library into your project. This can be done by copying the files from src/kBuffer to your project's directory. You can include the header as usual:

```
#include "kBuffer.h"
```

In your code, you have to define an instance of buffer_t. You have to init this instance with the function bufferInit().
If you want to have a ringbuffer with 8 elements:

```
buffer_t ringbuffer;
bufferInit(&ringbuffer, 8);
```

To check, if the initialization was successfull, you need to parse the return value of bufferInit():

```
buffer_t ringbuffer;
if(bufferInit(&ringbuffer, 8) == bufferOK){
 do_something_it_worked_ok();
}else{
 do_something_there_was_an_error();
}
```

2.3 Writing data to the buffer

To write data to the buffer, you can use the bufferWrite() function:

```
#include "kBuffer.h"

int main(void){
 buffer_t ringbuffer;            // Declare an buffer instance
 bufferInit(&ringbuffer, 8);     // Init the buffer with 8 elements
 //Notice, that no errorhandling has been done. We just expect a success

 bufferWrite(&ringbuffer, 42);   // Write the integer "42" to the buffer.

 return 0;
}
```

2.4 Reading data from the buffer

To read data from the buffer, you can use the bufferRead() function:

```
#include "kBuffer.h"

int main(void){

 buffer_t ringbuffer;                // Declare an buffer instance
 bufferInit(&ringbuffer, 8);         // Init the buffer with 8 elements
 //Notice, that no errorhandling has been done. We just expect a success

 bufferWrite(&ringbuffer, 42);       // Write the integer "42" to the buffer.

 uint16_t dataRead;                  // Declare an integer, where the read data should be stored
 bufferRead(&ringbuffer, &dataRead); // We expect, that dataRead is now 42 (because we have
       written 42 to the buffer before)

 return 0;
}
```

Chapter 3

Mean of buffer

3.1 Enabling of mean functions

Notice: Only enable the mean functions, if the element datatype (i.e. buffer datatype) is some sort of numeric type (i.e. integer, float, ...) To enable the buffer mean functions, you have to uncommented the following define in kBuffer.h:

```
#define BUFFER_ENABLE_MEAN
```

3.2 Caution!

There might be problems with this functions. The sum of the values (or the squared values) must be stored in a variable.
This variable is currently a long, but under certain conditions it might overflow.
You could replace it with an "unsigned long long" (or something smaller)

3.3 Mean of the buffer

You can take the mean of the buffer with the function bufferMean():

```
uint16_t mean;

bufferMean(&buffer, &mean);
```

You can also get the RMS (Root Mean Square), by calling the function bufferMeanRMS() (Parameters are the same)

Chapter 4

Bug List

globalScope$>$ Global bufferMean (buffer_t $*$buffer, bufferDatatype $*$meanOut)

The sum of the buffer is taken. Take precautions, that this variable won't overflow

globalScope$>$ Global bufferMeanRMS (buffer_t $*$buffer, bufferDatatype $*$meanOut)

The sum of squared buffer elements is taken. Take precautions, that this variable won't overflow

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

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

Chapter 7

Data Structure Documentation

7.1 buffer_t Struct Reference

Struct for buffer handling. If you need a ringbuffer in your software, you should instantiate a buffer_t, and run the neccessary functions with a pointer to your instance.

```
#include <kBuffer.h>
```

Data Fields

- uint8_t isInitialized

  is 0 if the buffer is not initialized
- uint16_t writePointer

  The write pointer of the buffer. At a write procedure, data gets written and the pointer is incremented.
- uint16_t readPointer

  The read pointer of the buffer. At a read procedure, data gets read and the pointer is incremented.
- uint16_t length

  The number of elements in the buffer.
- uint8_t elementLength

  The number of bytes of one buffer element. The total memory consumption in Bytes is equal to length ∗ element↩ Length.
- uint16_t datacount

  A variable which is increased by one when new data gets written and decremented by one when data is read.
- bufferDatatype ∗ data

  A pointer to the first element of the buffer. Length ∗ elementLength bytes of memory are allocated after this pointer.

7.1.1 Detailed Description

Struct for buffer handling. If you need a ringbuffer in your software, you should instantiate a buffer_t, and run the neccessary functions with a pointer to your instance.

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

- kBuffer/kBuffer.h

Chapter 8

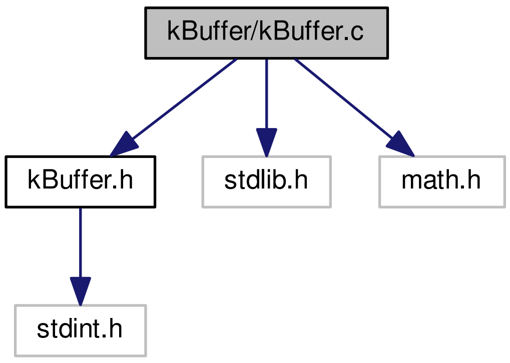
File Documentation

8.1 kBuffer/kBuffer.c File Reference

A universal ringbuffer library.

```
#include "kBuffer.h"
#include <stdlib.h>
#include <math.h>
```

Include dependency graph for kBuffer.c:



Functions

- bufferStatus_t bufferInit (buffer_t ∗buffer, uint16_t bufferSize)

  init a new buffer This function inits a new buffer_t.

- bufferStatus_t bufferWriteToIndex (buffer_t ∗buffer, uint16_t index, bufferDatatype data)

  write data to a specific index of the buffer. WARNING: Take care when using this function, it is against the main concept of a ringbuffer
- bufferStatus_t bufferReadFromIndex (buffer_t ∗buffer, uint16_t index, bufferDatatype ∗data)

  read data from a specifig index of the buffer WARNING: Take care when using this function, it is against the main concept of a ringbuffer
- uint8_t bufferIsEmpty (buffer_t ∗buffer)

  Checks, wheter the buffer is empty.
- uint8_t bufferIsFull (buffer_t ∗buffer)

  Checks, wheter the buffer is full.
- bufferStatus_t bufferWrite (buffer_t ∗buffer, bufferDatatype data)

  add data to the end of the ringbuffer
- bufferStatus_t bufferRead (buffer_t ∗buffer, bufferDatatype ∗data)

  read data from the beginning of the buffer
- bufferStatus_t bufferFill (buffer_t ∗buffer, bufferDatatype data, uint8_t silent)

  fill the whole buffer with given dummy data.
- bufferStatus_t bufferAvailable (buffer_t ∗buffer, uint16_t ∗available)

  return, how many elements are stored and available in the buffer
- bufferStatus_t bufferMean (buffer_t ∗buffer, bufferDatatype ∗meanOut)

  take the average of the whole buffer
- bufferStatus_t bufferMeanRMS (buffer_t ∗buffer, bufferDatatype ∗meanOut)

  take the root mean square of the whole buffer

8.1.1 Detailed Description

A universal ringbuffer library.

Author

Peter Kappelt

See also

`https://github.com/peterkappelt/kBuffer`

Copyright

Peter Kappelt 2016; MIT License (see LICENSE.txt in the root of this repository)

8.1.2 Function Documentation

8.1.2.1 bufferStatus_t bufferAvailable (buffer_t ∗ *buffer,* uint16_t ∗ *available*)

return, how many elements are stored and available in the buffer

Parameters

buffer	pointer to a buffer_t instance
available	pointer to a variable where the number of available elements should be stored

Returns

an element of bufferStatus_t

Return values

bufferOK	it worked as expected

bufferNotInitialized	the buffer wasn't initialized

8.1.2.2 bufferStatus_t bufferFill (buffer_t ∗ *buffer,* bufferDatatype *data,* uint8_t *silent*)

fill the whole buffer with given dummy data.

Parameters

buffer	pointer buffer_t instance
data	data to fill the buffer with
silent	if this parameter is 1, the buffer will be filled with data, but the write pointer stays at its current position (usefull, if you take the mean but the buffer is not full yet. You can just prefill it, the mean will be taken with the prefilled values)

Returns

an element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the buffer wasn't initialized

8.1.2.3 bufferStatus_t bufferInit (buffer_t ∗ *buffer,* uint16_t *bufferSize*)

init a new buffer This function inits a new buffer_t.

Parameters

buffer	Pointer (&) to a buffer_t object.
bufferSize	desired size of the buffer, the total buffer size (e.g. length-of-datatype ∗ bufferSize) may not exceed 2^16 bytes

Returns

an element of bufferStatus_t

Return values

bufferMemoryAllocation←Failed	The memory allocation with malloc failed. Make sure, you have enough memory available
bufferOK	It seems, like everything went well

8.1.2.4 uint8_t bufferIsEmpty (buffer_t ∗ *buffer*)

Checks, wheter the buffer is empty.

Parameters

buffer	Pointer to a buffer_t instance

Return values

1	buffer is empty
0	buffer is not empty

8.1.2.5 uint8_t bufferIsFull (buffer_t * *buffer*)

Checks, wheter the buffer is full.

Parameters

buffer	Pointer to a buffer_t instance

Return values

1	buffer is full
0	buffer is not full

8.1.2.6 bufferStatus_t bufferMean (buffer_t * *buffer,* bufferDatatype * *meanOut*)

take the average of the whole buffer

Parameters

buffer	pointer to a buffer_t instance
meanOut	pointer to a variable, where the mean will be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected, the mean is stored at the given variable
bufferNotInitialized	the buffer is not initialized

Bug The sum of the buffer is taken. Take precautions, that this variable won't overflow

8.1.2.7 bufferStatus_t bufferMeanRMS (buffer_t * *buffer,* bufferDatatype * *meanOut*)

take the root mean square of the whole buffer

Parameters

buffer	pointer to a buffer_t instance
meanOut	pointer to a variable, where the mean will be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected, the mean is stored at the given variable
bufferNotInitialized	the buffer is not initialized

Bug The sum of squared buffer elements is taken. Take precautions, that this variable won't overflow

8.1.2.8 bufferStatus_t bufferRead (buffer_t * *buffer*, bufferDatatype * *data*)

read data from the beginning of the buffer

Parameters

buffer	pointer to a buffer_t instance
data	pointer to a variable where data should be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the bufferInit() method hasn't been called or failed before
bufferEmpty	the buffer is empty an no more data can be read

8.1.2.9 bufferStatus_t bufferReadFromIndex (buffer_t * *buffer*, uint16_t *index*, bufferDatatype * *data*)

read data from a specifig index of the buffer WARNING: Take care when using this function, it is against the main concept of a ringbuffer

Parameters

buffer	Pointer to a buffer_t instance
index	The index, where data should be written. It can be in range 0 to length - 1
data	Pointer to a variable where the read data should be written to.

Returns

an element of bufferStatus_t

Return values

bufferOK	It went successfull
bufferNotInitialized	The buffer is not initialized. You have to call bufferInit before (or the init failed before)
bufferError	The desired data index is out of range

8.1.2.10 bufferStatus_t bufferWrite (buffer_t * *buffer*, bufferDatatype *data*)

add data to the end of the ringbuffer

Parameters

buffer	pointer to a buffer_t instance
data	data which should be written

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the bufferInit() method hasn't been called or failed before
bufferFull	the buffer is full an no more data can be written

8.1.2.11 bufferStatus_t bufferWriteToIndex (buffer_t ∗ *buffer,* uint16_t *index,* bufferDatatype *data*)

write data to a specific index of the buffer. WARNING: Take care when using this function, it is against the main concept of a ringbuffer

Parameters

buffer	Pointer to a buffer_t instance
index	The index, where data should be written. It can be in range 0 to length - 1
data	The actual data which should be written

Returns

an element of bufferStatus_t

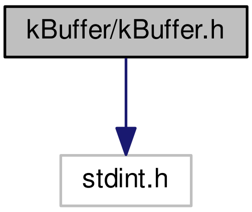
Return values

bufferOK	It went successfull
bufferNotInitialized	The buffer is not initialized. You have to call bufferInit before (or the init failed before)
bufferError	The desired data index is out of range

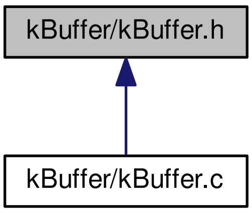
8.2 kBuffer/kBuffer.h File Reference

A universal ringbuffer library.

`#include <stdint.h>`
Include dependency graph for kBuffer.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct buffer_t

  Struct for buffer handling. If you need a ringbuffer in your software, you should instantiate a buffer_t, and run the neccessary functions with a pointer to your instance.

Macros

- #define bufferDatatype uint16_t

  The datatype of one buffer element. As default, it is an 16 bit unsigned integer. Feel free to change it to your needs.
- #define BUFFER_ENABLE_MEAN

  Enables mean/ averaging functions. If you uncomment this define, the following functions will be compiled.
  Only enable it, if bufferDatatype is some sort of numeric datatype (integer, float, ...)
  bufferMean();
  bufferMeanRMS();

Enumerations

- enum bufferStatus_t {
  bufferOK = 0, bufferMemoryAllocationFailed, bufferEmpty, bufferFull,

bufferNotInitialized, bufferError }

buffer function return codes

Functions

- bufferStatus_t bufferInit (buffer_t ∗buffer, uint16_t bufferSize)

  init a new buffer This function inits a new buffer_t.

- bufferStatus_t bufferWriteToIndex (buffer_t ∗buffer, uint16_t index, bufferDatatype data)

  write data to a specific index of the buffer. WARNING: Take care when using this function, it is against the main concept of a ringbuffer
- bufferStatus_t bufferReadFromIndex (buffer_t ∗buffer, uint16_t index, bufferDatatype ∗data)

  read data from a specifig index of the buffer WARNING: Take care when using this function, it is against the main concept of a ringbuffer
- uint8_t bufferIsEmpty (buffer_t ∗buffer)

  Checks, wheter the buffer is empty.
- uint8_t bufferIsFull (buffer_t ∗buffer)

  Checks, wheter the buffer is full.
- bufferStatus_t bufferWrite (buffer_t ∗buffer, bufferDatatype data)

  add data to the end of the ringbuffer
- bufferStatus_t bufferRead (buffer_t ∗buffer, bufferDatatype ∗data)

  read data from the beginning of the buffer
- bufferStatus_t bufferFill (buffer_t ∗buffer, bufferDatatype data, uint8_t silent)

  fill the whole buffer with given dummy data.
- bufferStatus_t bufferAvailable (buffer_t ∗buffer, uint16_t ∗available)

  return, how many elements are stored and available in the buffer
- bufferStatus_t bufferMean (buffer_t ∗buffer, bufferDatatype ∗meanOut)

  take the average of the whole buffer
- bufferStatus_t bufferMeanRMS (buffer_t ∗buffer, bufferDatatype ∗meanOut)

  take the root mean square of the whole buffer

8.2.1 Detailed Description

A universal ringbuffer library.

Author

Peter Kappelt

See also

`https://github.com/peterkappelt/kBuffer`

Copyright

Peter Kappelt 2016; MIT License (see LICENSE.txt in the root of this repository)

8.2.2 Enumeration Type Documentation

8.2.2.1 enum bufferStatus_t

buffer function return codes

Enumerator

bufferOK it seems, as everything worked as expected

bufferMemoryAllocationFailed happens while allocating memory,
there is not enough free memory (->malloc failed)

bufferEmpty happens at reading data,
buffer is empty and there is no more data to read

bufferFull happens at writing data,
buffer is full, no more data can be written

bufferNotInitialized The buffer is not initialized

bufferError an error occured, which isn't explained nearer. Have a look at the according function

8.2.3 Function Documentation

8.2.3.1 bufferStatus_t bufferAvailable (buffer_t * *buffer*, uint16_t * *available*)

return, how many elements are stored and available in the buffer

Parameters

buffer	pointer to a buffer_t instance
available	pointer to a variable where the number of available elements should be stored

Returns

an element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the buffer wasn't initialized

8.2.3.2 bufferStatus_t bufferFill (buffer_t * *buffer*, bufferDatatype *data*, uint8_t *silent*)

fill the whole buffer with given dummy data.

Parameters

buffer	pointer buffer_t instance
data	data to fill the buffer with
silent	if this parameter is 1, the buffer will be filled with data, but the write pointer stays at its current position (usefull, if you take the mean but the buffer is not full yet. You can just prefill it, the mean will be taken with the prefilled values)

Returns

an element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the buffer wasn't initialized

8.2.3.3 bufferStatus_t bufferInit (buffer_t * *buffer*, uint16_t *bufferSize*)

init a new buffer This function inits a new buffer_t.

Parameters

buffer	Pointer (&) to a buffer_t object.
bufferSize	desired size of the buffer, the total buffer size (e.g. length-of-datatype * bufferSize) may not exceed 2^16 bytes

Returns

an element of bufferStatus_t

Return values

bufferMemoryAllocation←↩ Failed	The memory allocation with malloc failed. Make sure, you have enough memory available
bufferOK	It seems, like everything went well

8.2.3.4 uint8_t bufferIsEmpty (buffer_t * *buffer*)

Checks, wheter the buffer is empty.

Parameters

buffer	Pointer to a buffer_t instance

Return values

1	buffer is empty
0	buffer is not empty

8.2.3.5 uint8_t bufferIsFull (buffer_t * *buffer*)

Checks, wheter the buffer is full.

Parameters

buffer	Pointer to a buffer_t instance

Return values

1	buffer is full
0	buffer is not full

8.2.3.6 bufferStatus_t bufferMean (buffer_t * *buffer*, bufferDatatype * *meanOut*)

take the average of the whole buffer

Parameters

buffer	pointer to a buffer_t instance
meanOut	pointer to a variable, where the mean will be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected, the mean is stored at the given variable
bufferNotInitialized	the buffer is not initialized

Bug The sum of the buffer is taken. Take precautions, that this variable won't overflow

8.2.3.7 bufferStatus_t bufferMeanRMS (buffer_t ∗ *buffer,* bufferDatatype ∗ *meanOut*)

take the root mean square of the whole buffer

Parameters

buffer	pointer to a buffer_t instance
meanOut	pointer to a variable, where the mean will be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected, the mean is stored at the given variable
bufferNotInitialized	the buffer is not initialized

Bug The sum of squared buffer elements is taken. Take precautions, that this variable won't overflow

8.2.3.8 bufferStatus_t bufferRead (buffer_t ∗ *buffer,* bufferDatatype ∗ *data*)

read data from the beginning of the buffer

Parameters

buffer	pointer to a buffer_t instance
data	pointer to a variable where data should be stored

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the bufferInit() method hasn't been called or failed before

bufferEmpty	the buffer is empty an no more data can be read

8.2.3.9 bufferStatus_t bufferReadFromIndex (buffer_t ∗ *buffer,* uint16_t *index,* bufferDatatype ∗ *data*)

read data from a specifig index of the buffer WARNING: Take care when using this function, it is against the main concept of a ringbuffer

Parameters

buffer	Pointer to a buffer_t instance
index	The index, where data should be written. It can be in range 0 to length - 1
data	Pointer to a variable where the read data should be written to.

Returns

an element of bufferStatus_t

Return values

bufferOK	It went successfull
bufferNotInitialized	The buffer is not initialized. You have to call bufferInit before (or the init failed before)
bufferError	The desired data index is out of range

8.2.3.10 bufferStatus_t bufferWrite (buffer_t ∗ *buffer,* bufferDatatype *data*)

add data to the end of the ringbuffer

Parameters

buffer	pointer to a buffer_t instance
data	data which should be written

Returns

a element of bufferStatus_t

Return values

bufferOK	it worked as expected
bufferNotInitialized	the bufferInit() method hasn't been called or failed before
bufferFull	the buffer is full an no more data can be written

8.2.3.11 bufferStatus_t bufferWriteToIndex (buffer_t ∗ *buffer,* uint16_t *index,* bufferDatatype *data*)

write data to a specific index of the buffer. WARNING: Take care when using this function, it is against the main concept of a ringbuffer

Parameters

buffer	Pointer to a buffer_t instance

index	The index, where data should be written. It can be in range 0 to length - 1
data	The actual data which should be written

Returns

an element of bufferStatus_t

Return values

bufferOK	It went successfull
bufferNotInitialized	The buffer is not initialized. You have to call bufferInit before (or the init failed before)
bufferError	The desired data index is out of range

Index