cQueue

1.4

Generated by Doxygen 1.8.13

# **Contents**

| 1   | Dep  | recated  | List                           | 1  |
|-----|------|----------|--------------------------------|----|
| 2   | Data | Struct   | ure Index                      | 2  |
|     | 2.1  | Data S   | Structures                     | 2  |
| 3   | File | Index    |                                | 2  |
|     | 3.1  | File Lis | st                             | 2  |
| 4   | Data | a Struct | ure Documentation              | 2  |
|     | 4.1  | Queue    | e_t Struct Reference           | 2  |
|     |      | 4.1.1    | Detailed Description           | 3  |
|     |      | 4.1.2    | Field Documentation            | 3  |
| 5   | File | Docum    | entation                       | 5  |
|     | 5.1  | src/cQ   | ueue.c File Reference          | 5  |
|     |      | 5.1.1    | Detailed Description           | 6  |
|     |      | 5.1.2    | Macro Definition Documentation | 6  |
|     |      | 5.1.3    | Function Documentation         | 7  |
|     | 5.2  | src/cQ   | ueue.h File Reference          | 12 |
|     |      | 5.2.1    | Detailed Description           | 14 |
|     |      | 5.2.2    | Macro Definition Documentation | 15 |
|     |      | 5.2.3    | Typedef Documentation          | 15 |
|     |      | 5.2.4    | Enumeration Type Documentation | 16 |
|     |      | 5.2.5    | Function Documentation         | 16 |
| Inc | lex  |          |                                | 27 |

# 1 Deprecated List

# Global q\_clean

q\_clean was already used in cQueue lib, alias is made to keep compatibility with earlier versions

# Global q\_nbRecs

q\_nbRecs was already used in cQueue lib, alias is made to keep compatibility with earlier versions

# Global q\_pull

q\_pull was already used in cQueue lib, alias is made to keep compatibility with earlier versions

## 2 Data Structure Index

#### 2.1 Data Structures

Here are the data structures with brief descriptions:

#### Queue t

Queue type structure holding all variables to handle the queue

2

# 3 File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

#### src/cQueue.c

Queue handling library (designed in c on STM32)

5

### src/cQueue.h

Queue handling library (designed in c on STM32)

12

# 4 Data Structure Documentation

# 4.1 Queue\_t Struct Reference

Queue type structure holding all variables to handle the queue.

```
#include <src/cQueue.h>
```

## Data Fields

· QueueType impl

Queue implementation: FIFO LIFO.

bool ovw

Overwrite previous records when queue is full allowed.

• uint16 t rec nb

number of records in the queue

• uint16\_t rec\_sz

Size of a record.

• uint32\_t queue\_sz

Size of the full queue.

• uint8\_t \* queue

Queue start pointer (when allocated)

• uint16\_t in

number of records pushed into the queue

• uint16\_t out

number of records pulled from the queue (only for FIFO)

• uint16\_t cnt

number of records not retrieved from the queue

• uint16 t init

set to QUEUE\_INITIALIZED after successful init of the queue and reset when killing queue

### 4.1.1 Detailed Description

Queue type structure holding all variables to handle the queue.

### 4.1.2 Field Documentation

### 4.1.2.1 cnt

```
uint16_t Queue_t::cnt
```

number of records not retrieved from the queue

# 4.1.2.2 impl

```
QueueType Queue_t::impl
```

Queue implementation: FIFO LIFO.

### 4.1.2.3 in

```
uint16_t Queue_t::in
```

number of records pushed into the queue

### 4.1.2.4 init

```
uint16_t Queue_t::init
```

set to QUEUE\_INITIALIZED after successful init of the queue and reset when killing queue

# 4.1.2.5 out

```
uint16_t Queue_t::out
```

number of records pulled from the queue (only for FIFO)

```
4.1.2.6 ovw
bool Queue_t::ovw
Overwrite previous records when queue is full allowed.
4.1.2.7 queue
uint8_t* Queue_t::queue
Queue start pointer (when allocated)
4.1.2.8 queue_sz
uint32_t Queue_t::queue_sz
Size of the full queue.
4.1.2.9 rec_nb
uint16_t Queue_t::rec_nb
number of records in the queue
4.1.2.10 rec_sz
uint16_t Queue_t::rec_sz
Size of a record.
The documentation for this struct was generated from the following file:
```

src/cQueue.h

5 File Documentation 5

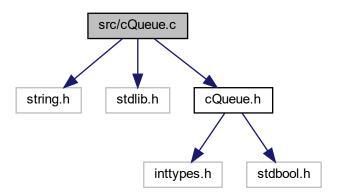
## 5 File Documentation

### 5.1 src/cQueue.c File Reference

Queue handling library (designed in c on STM32)

```
#include <string.h>
#include <stdlib.h>
#include "cQueue.h"
```

Include dependency graph for cQueue.c:



### Macros

• #define INC\_IDX(ctr, end, start)

Increments buffer index ctr rolling back to start when limit end is reached.

• #define DEC\_IDX(ctr, end, start)

Decrements buffer index ctr rolling back to end when limit start is reached.

### **Functions**

 void \* q\_init (Queue\_t \*q, const uint16\_t size\_rec, const uint16\_t nb\_recs, const QueueType type, const bool overwrite)

Queue initialization.

void q\_kill (Queue\_t \*q)

Queue destructor: release dynamically allocated queue.

void q\_flush (Queue\_t \*q)

Flush queue, restarting from empty queue.

bool q\_push (Queue\_t \*q, const void \*record)

Push record to queue.

bool q\_pop (Queue\_t \*q, void \*record)

Pop record from queue.

• bool q\_peek (Queue\_t \*q, void \*record)

Peek record from queue.

bool q\_drop (Queue\_t \*q)

Drop current record from queue.

### 5.1.1 Detailed Description

Queue handling library (designed in c on STM32)

Author

**SMFSW** 

Date

2018/05/26

Copyright

```
BSD 3-Clause License (c) 2017-2018, SMFSW
```

Queue handling library (designed in c on STM32)

## 5.1.2 Macro Definition Documentation

### 5.1.2.1 DEC\_IDX

### Value:

Decrements buffer index ctr rolling back to end when limit start is reached.

# 5.1.2.2 INC\_IDX

### Value:

Increments buffer index ctr rolling back to start when limit end is reached.

### 5.1.3 Function Documentation

## 5.1.3.1 q\_drop()

Drop current record from queue.

## Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

### **Parameters**

| in,out | q | - pointer of queue to handle |
|--------|---|------------------------------|
|--------|---|------------------------------|

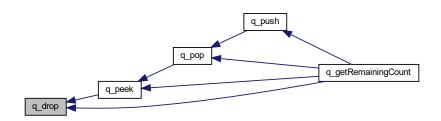
## Returns

drop status

## Return values

| true  | if successfully dropped from queue |
|-------|------------------------------------|
| false | if queue is empty                  |

Here is the caller graph for this function:



# 5.1.3.2 q\_flush()

```
void q_flush ( \label{eq:queue_t * q } \mbox{Queue\_t * q } \mbox{)}
```

Flush queue, restarting from empty queue.

## **Parameters**

| in, out $q$ | - pointer of queue to handle |
|-------------|------------------------------|
|-------------|------------------------------|

Here is the caller graph for this function:



# 5.1.3.3 q\_init()

Queue initialization.

## **Parameters**

| in,out | q         | - pointer of queue to handle                    |
|--------|-----------|---|
| in     | size_rec  | - size of a record in the queue                 |
| in     | nb_recs   | - number of records in the queue                |
| in     | type      | - Queue implementation type: FIFO, LIFO         |
| in     | overwrite | - Overwrite previous records when queue is full |

## Returns

NULL when allocation not possible, Queue tab address when successful

## 5.1.3.4 q\_kill()

```
void q_kill (
          Queue_t * q )
```

Queue destructor: release dynamically allocated queue.

## **Parameters**

| in,out $ q $ | <ul> <li>pointer of queue to handle</li> </ul> |
|--------------|--|
|--------------|--|

Here is the call graph for this function:



# 5.1.3.5 q\_peek()

Peek record from queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

# **Parameters**

| in     | q      | - pointer of queue to handle                |
|--------|--------|---|
| in,out | record | - pointer to record to be peeked from queue |

## Returns

Peek status

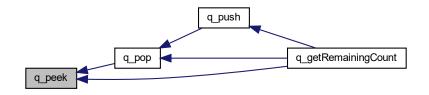
## Return values

| true  | if successfully pulled from queue |
|-------|-----------------------------------|
| false | if queue is empty                 |

Here is the call graph for this function:



Here is the caller graph for this function:



# 5.1.3.6 q\_pop()

Pop record from queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

## **Parameters**

| in     | q      | - pointer of queue to handle                |
|--------|--------|---|
| in,out | record | - pointer to record to be popped from queue |

# Returns

Pop status

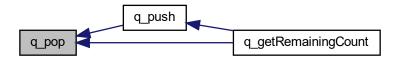
## Return values

| true  | if successfully pulled from queue |  |
|-------|-----------------------------------|--|
| false | if queue is empty                 |  |

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.1.3.7 q\_push()

Push record to queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

### **Parameters**

| in,out | q      | - pointer of queue to handle                |
|--------|--------|---|
| in     | record | - pointer to record to be pushed into queue |

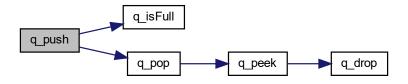
## Returns

Push status

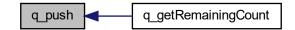
### Return values

| true  | if successfully pushed into queue |
|-------|-----------------------------------|
| false | if queue is full                  |

Here is the call graph for this function:



Here is the caller graph for this function:

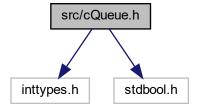


# 5.2 src/cQueue.h File Reference

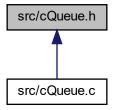
Queue handling library (designed in c on STM32)

#include <inttypes.h>
#include <stdbool.h>

Include dependency graph for cQueue.h:



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



## **Data Structures**

struct Queue\_t

Queue type structure holding all variables to handle the queue.

## Macros

• #define QUEUE\_INITIALIZED 0x5AA5

Queue initialized control value.

• #define q\_init\_def(q, sz) q\_init(q, sz, 20, FIFO, false)

Some kind of average default for queue initialization.

- #define q\_pull q\_pop
- #define q\_nbRecs q\_getCount
- #define q\_clean q\_flush

## Typedefs

- typedef enum enumQueueType QueueType
- typedef struct Queue\_t Queue\_t

## **Enumerations**

• enum enumQueueType { FIFO = 0, LIFO = 1 }

Queue behavior enumeration (FIFO, LIFO)

#### **Functions**

```
• void * q_init (Queue_t *q, const uint16_t size_rec, const uint16_t nb_recs, const QueueType type, const bool
      overwrite)
          Queue initialization.
    void q_kill (Queue_t *q)
          Queue destructor: release dynamically allocated queue.
    void q_flush (Queue_t *q)
          Flush queue, restarting from empty queue.

    bool q_isInitialized (const Queue_t *q)

          get initialization state of the queue

    bool q_isEmpty (const Queue_t *q)

          get emptiness state of the queue

    bool q_isFull (const Queue_t *q)

          get fullness state of the queue

    uint32_t q_sizeof (const Queue_t *q)

          get size of queue

    uint16_t q_getCount (const Queue_t *q)

          get number of records in the queue

    uint16_t q_getRemainingCount (const Queue_t *q)

          get number of records left in the queue

    bool q_push (Queue_t *q, const void *record)

          Push record to queue.

    bool q_pop (Queue_t *q, void *record)

          Pop record from queue.
    • bool q_peek (Queue_t *q, void *record)
          Peek record from queue.
    bool q_drop (Queue_t *q)
          Drop current record from queue.
5.2.1 Detailed Description
Queue handling library (designed in c on STM32)
Author
      SMFSW
Date
      2018/05/26
Copyright
      BSD 3-Clause License (c) 2017-2018, SMFSW
```

Queue handling library (designed in c on STM32)

### 5.2.2 Macro Definition Documentation

```
5.2.2.1 q_clean
```

```
#define q_clean q_flush
```

Deprecated q\_clean was already used in cQueue lib, alias is made to keep compatibility with earlier versions

## 5.2.2.2 q\_init\_def

Some kind of average default for queue initialization.

# 5.2.2.3 q\_nbRecs

```
#define q_nbRecs q_getCount
```

Deprecated q\_nbRecs was already used in cQueue lib, alias is made to keep compatibility with earlier versions

## 5.2.2.4 q\_pull

```
\#define q_pull q_pop
```

Deprecated q\_pull was already used in cQueue lib, alias is made to keep compatibility with earlier versions

# 5.2.2.5 QUEUE\_INITIALIZED

```
#define QUEUE_INITIALIZED 0x5AA5
```

Queue initialized control value.

## 5.2.3 Typedef Documentation

### 5.2.3.1 Queue\_t

```
typedef struct Queue_t Queue_t
```

# 5.2.3.2 QueueType

```
typedef enum enumQueueType QueueType
```

# 5.2.4 Enumeration Type Documentation

# 5.2.4.1 enumQueueType

```
enum enumQueueType
```

Queue behavior enumeration (FIFO, LIFO)

### Enumerator

| FIFO | First In First Out behavior. |
|------|------------------------------|
| LIFO | Last In First Out behavior.  |

### 5.2.5 Function Documentation

# 5.2.5.1 q\_drop()

Drop current record from queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

### **Parameters**

| in,out | q | - pointer of queue to handle |
|--------|---|------------------------------|

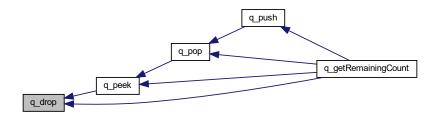
# Returns

drop status

## Return values

| true  | if successfully dropped from queue |
|-------|------------------------------------|
| false | if queue is empty                  |

Here is the caller graph for this function:



# 5.2.5.2 q\_flush()

Flush queue, restarting from empty queue.

### **Parameters**

| in, out $q$ | - pointer of queue to handle |
|-------------|------------------------------|
|-------------|------------------------------|

Here is the caller graph for this function:



## 5.2.5.3 q\_getCount()

```
uint16_t q_getCount (  {\tt const\ Queue\_t\ *\ } q\ ) \quad [{\tt inline}]
```

get number of records in the queue

## **Parameters**

| in | q | - pointer of queue to handle |
|----|---|------------------------------|
|----|---|------------------------------|

### Returns

Number of records stored in the queue

# 5.2.5.4 q\_getRemainingCount()

```
uint16_t q_getRemainingCount ( {\tt const\ Queue\_t\ *\ }q\ ) \quad [inline]
```

get number of records left in the queue

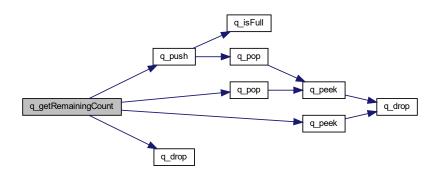
## **Parameters**

|  | in | q | - pointer of queue to handle |
|--|----|---|------------------------------|
|--|----|---|------------------------------|

# Returns

Number of records left in the queue

Here is the call graph for this function:



# 5.2.5.5 q\_init()

## Queue initialization.

### **Parameters**

| in,out | q         | - pointer of queue to handle                    |
|--------|-----------|---|
| in     | size_rec  | - size of a record in the queue                 |
| in     | nb_recs   | - number of records in the queue                |
| in     | type      | - Queue implementation type: FIFO, LIFO         |
| in     | overwrite | - Overwrite previous records when queue is full |

### Returns

NULL when allocation not possible, Queue tab address when successful

# 5.2.5.6 q\_isEmpty()

```
bool q_isEmpty ( {\tt const\ Queue\_t\ *\ q\ )} \quad [{\tt inline}]
```

get emptiness state of the queue

### **Parameters**

| in | q | - pointer of queue to handle |
|----|---|------------------------------|
|----|---|------------------------------|

### Returns

Queue emptiness status

## Return values

| true  | if queue is empty |
|-------|-------------------|
| false | is not empty      |

# 5.2.5.7 q\_isFull()

```
bool q_isFull ( {\tt const\ Queue\_t\ *\ }q\ ) \quad [{\tt inline}]
```

get fullness state of the queue

### **Parameters**

| in | q | - pointer of queue to handle |
|----|---|------------------------------|
|----|---|------------------------------|

## Returns

Queue fullness status

### Return values

| true  | if queue is full |
|-------|------------------|
| false | is not full      |

Here is the caller graph for this function:



# 5.2.5.8 q\_isInitialized()

```
bool q_isInitialized ( {\tt const\ Queue\_t\ *\ }q\ ) \quad [{\tt inline}]
```

get initialization state of the queue

## **Parameters**

| 1 |    |   |  |
|---|----|---|--|
|   | in | q | <ul> <li>pointer of queue to handle</li> </ul> |

# Returns

Queue initialization status

## Return values

| true  | if queue is allocated     |
|-------|---------------------------|
| false | is queue is not allocated |

## 5.2.5.9 q\_kill()

Queue destructor: release dynamically allocated queue.

## **Parameters**

```
in, out | q | - pointer of queue to handle
```

Here is the call graph for this function:



## 5.2.5.10 q\_peek()

Peek record from queue.

## Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

## **Parameters**

| in     | q      | - pointer of queue to handle                |
|--------|--------|---|
| in,out | record | - pointer to record to be peeked from queue |

#### Returns

Peek status

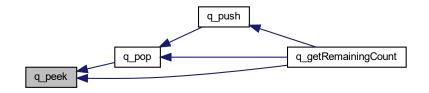
### Return values

| true  | if successfully pulled from queue |
|-------|-----------------------------------|
| false | if queue is empty                 |

Here is the call graph for this function:



Here is the caller graph for this function:



# 5.2.5.11 q\_pop()

Pop record from queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

## **Parameters**

| in     | q      | - pointer of queue to handle                |  |
|--------|--------|---|--|
| in,out | record | - pointer to record to be popped from queue |  |

# Returns

Pop status

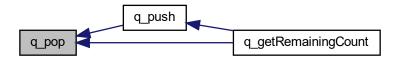
## Return values

| true  | if successfully pulled from queue |
|-------|-----------------------------------|
| false | if queue is empty                 |

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.2.5.12 q\_push()

Push record to queue.

# Warning

If using q\_push, q\_pop, q\_peek and/or q\_drop in both interrupts and main application, you shall disable interrupts in main application when using these functions

### **Parameters**

| in,out | q      | - pointer of queue to handle                |  |
|--------|--------|---|--|
| in     | record | - pointer to record to be pushed into queue |  |

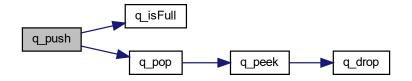
## Returns

Push status

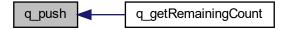
### Return values

| true  | if successfully pushed into queue |
|-------|-----------------------------------|
| false | if queue is full                  |

Here is the call graph for this function:



Here is the caller graph for this function:



# 5.2.5.13 q\_sizeof()

get size of queue

## Remarks

Size in bytes (like sizeof)

## **Parameters**

| in | q | - pointer of queue to handle |
|----|---|------------------------------|

Size of queue in bytes

# Index

| cQueue.c                                | cQueue.h, 15        |
|---|---------------------|
| DEC_IDX, 6                              | q_drop              |
| INC_IDX, 6                              | cQueue.c, 7         |
| q_drop, 7                               | cQueue.h, 16        |
| q_flush, 7                              | q_flush             |
| q_init, 8                               | cQueue.c, 7         |
| q_kill, 8                               | cQueue.h, 17        |
| q_peek, 9                               | q_getCount          |
| q_pop, 10                               |                     |
|   | cQueue.h, 17        |
| q_push, 11                              | q_getRemainingCount |
| cQueue.h                                | cQueue.h, 18        |
| enumQueueType, 16                       | q_init              |
| q_clean, 15                             | cQueue.c, 8         |
| q_drop, 16                              | cQueue.h, 18        |
| q_flush, 17                             | q_init_def          |
| q_getCount, 17                          | cQueue.h, 15        |
| q_getRemainingCount, 18                 | q_isEmpty           |
| q_init, 18                              | cQueue.h, 19        |
| q_init_def, 15                          | q_isFull            |
| q_isEmpty, 19                           | cQueue.h, 19        |
| q_isFull, 19                            | q isInitialized     |
| q_isInitialized, 20                     | <del></del>         |
| q_isinitidalized, 20<br>q_kill, 20      | cQueue.h, 20        |
| <del></del>                             | q_kill              |
| q_nbRecs, 15                            | cQueue.c, 8         |
| q_peek, 21                              | cQueue.h, 20        |
| q_pop, 22                               | q_nbRecs            |
| q_pull, 15                              | cQueue.h, 15        |
| q_push, 23                              | q_peek              |
| q_sizeof, 24                            | cQueue.c, 9         |
| QUEUE_INITIALIZED, 15                   | cQueue.h, 21        |
| Queue_t, 15                             | q_pop               |
| QueueType, 16                           | cQueue.c, 10        |
| cnt                                     | cQueue.h, 22        |
| Queue t, 3                              |                     |
|   | q_pull              |
| DEC IDX                                 | cQueue.h, 15        |
| cQueue.c, 6                             | q_push              |
| 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 | cQueue.c, 11        |
| enumQueueType                           | cQueue.h, 23        |
| cQueue.h, 16                            | q_sizeof            |
| ogudusiii, io                           | cQueue.h, 24        |
| INC IDX                                 | QUEUE_INITIALIZED   |
| cQueue.c, 6                             | cQueue.h, 15        |
| impl                                    | queue               |
| ·                                       | Queue t, 4          |
| Queue_t, 3                              | queue sz            |
| in                                      | Queue t, 4          |
| Queue_t, 3                              | Queue t, 2          |
| init                                    | cQueue.h, 15        |
| Queue_t, 3                              |                     |
|   | cnt, 3              |
| out                                     | impl, 3             |
| Queue_t, 3                              | in, 3               |
| OVW                                     | init, 3             |
| Queue_t, 3                              | out, 3              |
|   | ovw, 3              |
| q_clean                                 | queue, 4            |
|   |                     |

28 INDEX

```
queue_sz, 4
rec_nb, 4
rec_sz, 4
QueueType
cQueue.h, 16

rec_nb
Queue_t, 4
rec_sz
Queue_t, 4

src/cQueue.c, 5
src/cQueue.h, 12
```