FastFalgBlock-Pool

API Reference

Build time : 15 Aug 2018 23:15:35

Version : 1.0

Table of Contents

[Table of Contents 2](#_Toc522142138)

[Chapter 1 Overview 3](#_Toc522142139)

[Chapter 2 API Description 4](#_Toc522142140)

[Chapter 3 Example 6](#_Toc522142141)

1. Overview

* API List

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Return | Parameter | Description |
| reload | FFB\_Pool\_ID | void \*pBuffer | Reload the Pool ID form initialized buffer. |
| init | FFB\_Pool\_ID | void \*pBuffer  uint32\_t bufferSize  uint32\_t blockSize | Initialization the new memory pool. |
| getUseCount | uint16\_t | FFB\_Pool\_ID poolID | Get the memory pool used block count, |
| getFreeCount | uint16\_t | FFB\_Pool\_ID poolID | Get the memory pool free block count, |
| getTotalCount | uint16\_t | FFB\_Pool\_ID poolID | Get the memory pool total block count. |
| alloc | void\* | FFB\_Pool\_ID poolID | Allocate a new block address. |
| free | ffbStatus | FFB\_Pool\_ID poolID  void\* addr | Free this block address. |

* Enum ffbStatus

|  |  |  |
| --- | --- | --- |
| Value | Name | Description |
| 0x0 | ffbOK | Successful. |
| 0x1 | ffbReload |  |
| 0x2 | ffbAddrError |  |
| 0x3 | ffbMutex | Mutex lock. |
| 0xFF | ffbError |  |

1. API Description

* reload

|  |  |
| --- | --- |
| Parameter | |
| void \*pBuffer | A initialized buffer pointer. |
| Return | |
| FFB\_Pool\_ID | FFB\_Pool\_ID=0x0, Reload fail.  FFB\_Pool\_ID!=0x0, Reload finish. |

* init

|  |  |
| --- | --- |
| Parameter | |
| void \*pBuffer | Buffer pointer. |
| uint32\_t bufferSize | Size of buffer. |
| uint32\_t blockSize | Memory pool block size. |
| Return | |
| FFB\_Pool\_ID | FFB\_Pool\_ID=0x0, Reload fail.  FFB\_Pool\_ID!=0x0, Reload finish. |

* getUseCount

|  |  |
| --- | --- |
| Parameter | |
| FFB\_Pool\_ID poolID | FastFlagPool PoolID. |
| Return | |
| uint16\_t | This Pool used block count. |

* getFreeCount

|  |  |
| --- | --- |
| Parameter | |
| FFB\_Pool\_ID poolID | FastFlagPool PoolID. |
| Return | |
| uint16\_t | This Pool free block count. |

* getTotalCount

|  |  |
| --- | --- |
| Parameter | |
| FFB\_Pool\_ID poolID | FastFlagPool PoolID. |
| Return | |
| uint16\_t | This Pool total block count. |

* alloc

|  |  |
| --- | --- |
| Parameter | |
| FFB\_Pool\_ID poolID | FastFlagPool PoolID. |
| Return | |
| Void\* | Memory block point.  return =0x0, Reload fail, memory pool full or mutex lock.  return!=0x0, Reload successful. |
|  |  |

* free

|  |  |
| --- | --- |
| Parameter | |
| FFB\_Pool\_ID poolID | FastFlagPool PoolID. |
| void\* addr | Memory Pool block pointer. |
| Return | |
| ffbStatus | ffbOK: Successful.  ffbMutex: Mutex lock  ffbAddrError: address is not form this pool. |

1. Example

|  |  |
| --- | --- |
| Thread\_A.c | |
|  | #include "ffb\_pool\_API.h" |
|  |  |
|  | uint8\_t poolBuffer[8192]; |
|  |  |
|  | void thread\_A(void){ |
|  | FFB\_Pool\_ID PoolID; |
|  | PoolID = FFB\_POOL\_API.init(&poolBuffer, sizeof(poolBuffer), 64); |
|  |  |
|  | void\* pSpace = FFB\_POOL\_API.alloc(PoolID); |
|  | if(pSpace==0x0) |
|  | while(1); //memory alloc fail, pool full or mutex lock |
|  |  |
|  | memset(pSpace, 0xFF, 64); |
|  |  |
|  | ffbStatus result = FFB\_POOL\_API.free(PoolID, pSpace); |
|  |  |
|  | if(result!=ffbOK) |
|  | while(1);//memory free fail, pool mutex lock |
|  |  |
|  | return; |
|  | } |
|  |  |

|  |  |
| --- | --- |
| Thread\_B.c | |
|  | #include "ffb\_pool\_API.h" |
|  |  |
|  | extern uint8\_t poolBuffer[8192]; |
|  |  |
|  | void thread\_B(void){ |
|  | FFB\_Pool\_ID PoolID; |
|  | PoolID = FFB\_POOL\_API.reload(&poolBuffer); |
|  |  |
|  | if(PoolID==0x0) |
|  | while(1); //memory alloc fail, pool full or mutex lock |
|  |  |
|  | void\* pSpace = FFB\_POOL\_API.alloc(PoolID); |
|  | if(pSpace==0x0) |
|  | while(1); //memory alloc fail, pool full or mutex lock |
|  |  |
|  | memset(pSpace, 0xFF, 64); |
|  |  |
|  | ffbStatus result = FFB\_POOL\_API.free(PoolID, pSpace); |
|  |  |
|  | if(result!=ffbOK) |
|  | while(1);//memory free fail, pool mutex lock |
|  |  |
|  | return; |
|  | } |
|  |  |