Demo

V1.0

Generated by Doxygen 1.9.6

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 W25Qxx_Exported_Functions

Functions

- void W25Qx_Read_ID_8 (uint8_t ∗ID)

  Read W25Qxx Device ID(ID7-ID0)
- void W25Qx_Read_ID_16 (uint16_t ∗ID)

  Read W25Qxx Manufacturer ID + Device ID.
- uint8_t W25Qx_WriteEnable (void)

  Enable Write for W25Qxx.
- void W25Qx_Coerce_Reset (void)

  Coerce W25Qxx to Reset.
- uint8_t W25Qx_Reset (void)

  Reset W25Qxx.
- uint8_t W25Qx_Erase_Sector (uint32_t EraseAddr)

  Erase the specified sector of the W25Qxx.
- uint8_t W25Qx_EraseALL (void)

  Erase all block of the W25Qxx.
- uint8_t W25Qx_Read_Data (uint8_t ∗Data, uint32_t Addr, uint32_t ReadSize)

  Read an amount of Data to the W25Qxx.
- uint8_t W25Qx_Write_Data (uint8_t ∗Data, uint32_t WriteAddr, uint32_t Size)

  Write an amount of data to the W25Qxx.

3.1.1 Detailed Description

3.1.2 Function Documentation

3.1.2.1 W25Qx_Coerce_Reset()

```
void W25Qx_Coerce_Reset (
              void  )
```

Coerce W25Qxx to Reset.

Definition at line 124 of file W25Qxx.c.

3.1.2.2 W25Qx_Erase_Sector()

```
uint8_t W25Qx_Erase_Sector (
              uint32_t EraseAddr )
```

Erase the specified sector of the W25Qxx.

Parameters

EraseAddr	: start addr of erase

Return values

W25Qxx	status

Definition at line 263 of file W25Qxx.c.

3.1.2.3 W25Qx_EraseALL()

```
uint8_t W25Qx_EraseALL (
              void  )
```

Erase all block of the W25Qxx.

Return values

W25Qxx	status

Definition at line 294 of file W25Qxx.c.

3.1.2.4 W25Qx_Read_Data()

```
uint8_t W25Qx_Read_Data (
              uint8_t * Data,
              uint32_t Addr,
              uint32_t ReadSize )
```

Read an amount of Data to the W25Qxx.

Parameters

Data	: pointer to data buffer
Addr	: start addr of read
ReadSize	: amount of data to be read

Return values

W25Qxx	status

Definition at line 162 of file W25Qxx.c.

3.1.2.5 W25Qx_Read_ID_16()

```
void W25Qx_Read_ID_16 (
             uint16_t * ID )
```

Read W25Qxx Manufacturer ID + Device ID.

Note

Manufacturer ID EFh Device ID 17h

Parameters

ID:Data	to be receive

Return values

None	

Definition at line 49 of file W25Qxx.c.

3.1.2.6 W25Qx_Read_ID_8()

```
void W25Qx_Read_ID_8 (
             uint8_t * ID )
```

Read W25Qxx Device ID(ID7-ID0)

Note

Device ID(ID7-ID0) VALUE 17h

Parameters

ID:Data	to be receive

Return values

None	

Definition at line 27 of file W25Qxx.c.

3.1.2.7 W25Qx_Reset()

```
uint8_t W25Qx_Reset (
             void  )
```

Reset W25Qxx.

Return values

W25Qxx	status

Definition at line 137 of file W25Qxx.c.

3.1.2.8 W25Qx_Write_Data()

```
uint8_t W25Qx_Write_Data (
             uint8_t * Data,
             uint32_t WriteAddr,
             uint32_t Size )
```

Write an amount of data to the W25Qxx.

Parameters

Data	: pointer to data buffer
WriteAddr	: start addr of Write
Size	: amount of data to be write

Note

Size No more than 256byte.

Return values

W25Qxx	status

Definition at line 189 of file W25Qxx.c.

3.1.2.9 W25Qx_WriteEnable()

```
uint8_t W25Qx_WriteEnable (
            void  )
```

Enable Write for W25Qxx.

Return values

W25Qxx	status

Definition at line 96 of file W25Qxx.c.

Chapter 4

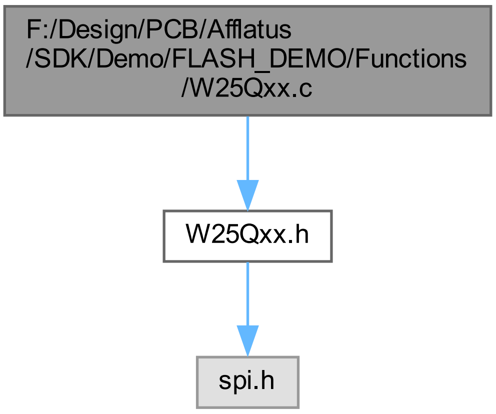
File Documentation

4.1 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25↩ Qxx.c File Reference

This file provides information about the W25Qxx firmware functions.

```
#include "W25Qxx.h"
```

Include dependency graph for W25Qxx.c:


F:/Design/PCB/Afflatus /SDK/Demo/FLASH_DEMO/Functions /W25Qxx.c
W25Qxx.h
spi.h

Functions

- void Flash_Send_Byte (uint8_t Data)
- uint8_t Flash_Read_Byte (uint8_t TxData)
- void W25Qx_Read_ID_8 (uint8_t ∗ID)

    Read W25Qxx Device ID(ID7-ID0)
- void W25Qx_Read_ID_16 (uint16_t ∗ID)

Read W25Qxx Manufacturer ID + Device ID.
- uint8_t W25Qx_WriteEnable (void)

    Enable Write for W25Qxx.
- void W25Qx_Coerce_Reset (void)

    Coerce W25Qxx to Reset.
- uint8_t W25Qx_Reset (void)

    Reset W25Qxx.
- uint8_t W25Qx_Read_Data (uint8_t ∗Data, uint32_t Addr, uint32_t ReadSize)

    Read an amount of Data to the W25Qxx.
- uint8_t W25Qx_Write_Data (uint8_t ∗Data, uint32_t WriteAddr, uint32_t Size)

    Write an amount of data to the W25Qxx.
- uint8_t W25Qx_Erase_Sector (uint32_t EraseAddr)

    Erase the specified sector of the W25Qxx.
- uint8_t W25Qx_EraseALL (void)

    Erase all block of the W25Qxx.

4.1.1 Detailed Description

This file provides information about the W25Qxx firmware functions.

Author

Emotion_Thorn

Version

V1.0

Date

2023-05-05

Definition in file W25Qxx.c.

4.1.2 Function Documentation

4.1.2.1 Flash_Read_Byte()

```
uint8_t Flash_Read_Byte (
            uint8_t TxData )
```

Definition at line 15 of file W25Qxx.c.

4.1.2.2 Flash_Send_Byte()

```
void Flash_Send_Byte (
              uint8_t Data )
```

Definition at line 11 of file W25Qxx.c.

4.2 W25Qxx.c

Go to the documentation of this file.

```
00001
00010 #include "W25Qxx.h"
00011 void Flash_Send_Byte(uint8_t Data)
00012 {
00013     HAL_SPI_Transmit(&hspi2, &Data, 1, W25Qx_TIMEOUT_VALUE);
00014 }
00015 uint8_t Flash_Read_Byte(uint8_t TxData)
00016 {
00017     uint8_t RX;
00018     HAL_SPI_TransmitReceive(&hspi2, &TxData, &RX, 1, W25Qx_TIMEOUT_VALUE);
00019     return RX;
00020 }
00027 void W25Qx_Read_ID_8(uint8_t *ID)
00028 {
00029     uint8_t idt;
00030
00031     uint8_t cmd[4] = {FLASH_ID_8Byte, 0x00, 0x00, 0x00};
00032
00033     Flash_CS_Select();
00034     /* Send the read ID command */
00035     HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
00036     /* Reception of the data */
00037     HAL_SPI_Receive(&hspi2, &idt, 1, W25Qx_TIMEOUT_VALUE);
00038
00039     *ID = idt;
00040
00041     Flash_CS_Discard();
00042 }
00049 void W25Qx_Read_ID_16(uint16_t *ID)
00050 {
00051     uint8_t idt[2];
00052
00053     uint8_t cmd[4] = {FLASH_ID_16Byte, 0x00, 0x00, 0x00};
00054
00055     Flash_CS_Select();
00056     /* Send the read ID command */
00057     HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
00058     /* Reception of the data */
00059     HAL_SPI_Receive(&hspi2, idt, 2, W25Qx_TIMEOUT_VALUE);
00060
00061     *ID = (idt[0] « 8) + idt[1];
00062
00063     Flash_CS_Discard();
00064 }
00069 static uint8_t W25Qx_Read_Busy(void)
00070 {
00071     uint8_t cmd[] = {Read_Status_Reg_1};
00072     uint8_t state;
00073     Flash_CS_Select();
00074     /* Send the read ID command */
00075     HAL_SPI_Transmit(&hspi2, cmd, 1, W25Qx_TIMEOUT_VALUE);
00076     /* Reception of the data */
00077     HAL_SPI_Receive(&hspi2, &state, 1, W25Qx_TIMEOUT_VALUE);
00078     Flash_CS_Discard();
00079     if (state == HAL_OK)
00080     {
00081         return W25Qx_OK;
00082     }
00083     else if (state == W25Qx_BUSY)
00084     {
00085         return W25Qx_BUSY;
00086     }
00087     else
00088     {
00089         return W25Qx_ERROR;
00090     }
00091 }
00096 uint8_t W25Qx_WriteEnable(void)
```

```
{
    uint8_t cmd[] = {FLASH_ENABLE_Write};
    uint32_t StartTime = HAL_GetTick();

    /*Select the FLASH: Chip Select low */
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 1, W25Qx_TIMEOUT_VALUE);
    /*Deselect the FLASH: Chip Select high */
    Flash_CS_Discard();

    /* Wait the end of Flash writing */
    while (W25Qx_Read_Busy() == W25Qx_BUSY)
    {
        /* Check for the Timeout */
        if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
        {
            return W25Qx_TimeOut;
        }
        HAL_Delay(1);
    }

    return W25Qx_OK;
}
void W25Qx_Coerce_Reset(void)
{
    uint8_t cmd[] = {FLASH_Enable_Reset, FLASH_Reset_Device};
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 2, W25Qx_TIMEOUT_VALUE);
    Flash_CS_Discard();
    HAL_Delay(30);
}
uint8_t W25Qx_Reset(void)
{
    uint8_t cmd[] = {FLASH_Enable_Reset, FLASH_Reset_Device};
    uint32_t StartTime = HAL_GetTick();
    while (W25Qx_Read_Busy() == W25Qx_BUSY)
    {
        if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
        {
            return W25Qx_TimeOut;
        }
    }
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 2, W25Qx_TIMEOUT_VALUE);
    Flash_CS_Discard();
    HAL_Delay(30);
    return W25Qx_OK;
}
uint8_t W25Qx_Read_Data(uint8_t *Data, uint32_t Addr, uint32_t ReadSize)
{
    uint8_t cmd[4];
    /* Configure the command */
    cmd[0] = FLASH_Read;
    cmd[1] = (uint8_t)(Addr » 16);
    cmd[2] = (uint8_t)(Addr » 8);
    cmd[3] = (uint8_t)(Addr);
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
    /* Reception of the data */
    if (HAL_SPI_Receive(&hspi2, Data, ReadSize, W25Qx_TIMEOUT_VALUE) != HAL_OK)
    {
        return W25Qx_ERROR;
    }
    Flash_CS_Discard();
    return W25Qx_OK;
}
uint8_t W25Qx_Write_Data(uint8_t *Data, uint32_t WriteAddr, uint32_t Size)
{
    uint8_t cmd[4];
    uint32_t end_addr, current_size, current_addr;
    uint32_t StartTime = HAL_GetTick();

    /* Calculation of the size between the write address and the end of the page */
    current_addr = 0;

    while (current_addr <= WriteAddr)
    {
        current_addr += W25QX_PAGE_SIZE;
    }
    current_size = current_addr - WriteAddr;

    /* Check if the size of the data is less than the remaining place in the page */
    if (current_size > Size)
```

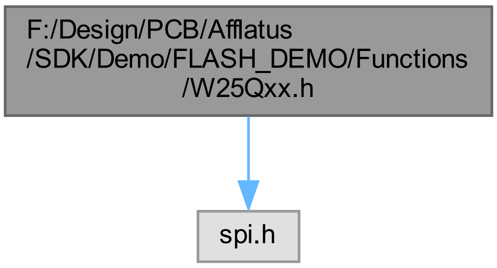
```c
    {
        current_size = Size;
    }

    /* Initialize the adress variables */
    current_addr = WriteAddr;
    end_addr = WriteAddr + Size;

    /* Perform the write page by page */
    do
    {
        /* Configure the command */
        cmd[0] = FLASH_Write;
        cmd[1] = (uint8_t)(current_addr » 16);
        cmd[2] = (uint8_t)(current_addr » 8);
        cmd[3] = (uint8_t)(current_addr);

        /* Enable write operations */
        W25Qx_WriteEnable();

        Flash_CS_Select();
        /* Send the command */
        if (HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE) != HAL_OK)
        {
            return W25Qx_ERROR;
        }

        /* Transmission of the data */
        if (HAL_SPI_Transmit(&hspi2, Data, current_size, W25Qx_TIMEOUT_VALUE) != HAL_OK)
        {
            return W25Qx_ERROR;
        }
        Flash_CS_Discard();
        /* Wait the end of Flash writing */
        while (W25Qx_Read_Busy() == W25Qx_BUSY)
        {
            /* Check for the Timeout */
            if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
            {
                return W25Qx_TimeOut;
            }
            // delay(1);
        }

        /* Update the address and size variables for next page programming */
        current_addr += current_size;
        Data += current_size;
        current_size = ((current_addr + W25QX_PAGE_SIZE) > end_addr) ? (end_addr - current_addr) :
W25QX_PAGE_SIZE;
    } while (current_addr < end_addr);

    return W25Qx_OK;
}
uint8_t W25Qx_Erase_Sector(uint32_t EraseAddr)
{
    uint8_t cmd[4];
    uint32_t StartTime = HAL_GetTick();
    cmd[0] = FLASH_Erase;
    cmd[1] = (uint8_t)(EraseAddr » 16);
    cmd[2] = (uint8_t)(EraseAddr » 8);
    cmd[3] = (uint8_t)(EraseAddr);

    /* Enable write operations */
    W25Qx_WriteEnable();

    /*Select the FLASH: Chip Select low */
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
    /*Deselect the FLASH: Chip Select high */
    Flash_CS_Discard();
    while (W25Qx_Read_Busy() == W25Qx_BUSY)
    {
        if ((HAL_GetTick() - StartTime) > W25Qx_Erase_TIMEOUT_VALUE)
        {
            return W25Qx_TimeOut;
        }
    }
    return W25Qx_OK;
}
uint8_t W25Qx_EraseALL(void)
{
    uint8_t cmd[] = {FLASH_All_Erase};
    uint32_t StartTime = HAL_GetTick();
    W25Qx_WriteEnable();
    Flash_CS_Select();
    /* Send the read ID command */
    HAL_SPI_Transmit(&hspi2, cmd, 1, W25Qx_TIMEOUT_VALUE);
    Flash_CS_Discard();
    while (W25Qx_Read_Busy() == W25Qx_BUSY)
    {
        if ((HAL_GetTick() - StartTime) > W25Qx_Erase_TIMEOUT_VALUE)
        {
            return W25Qx_TimeOut;
        }
    }
    return W25Qx_OK;
}
```

4.3 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25↩ Qxx.h File Reference

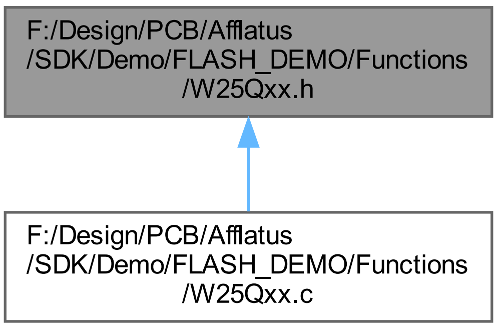
Header file of W25Qxx module.

`#include "spi.h"`

Include dependency graph for W25Qxx.h:



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



Macros

- #define Flash_CS_Select() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_↩ RESET)
- #define Flash_CS_Discard() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_SET)
- #define W25Qx_Erase_TIMEOUT_VALUE 5000

  W25Qxx Configuration.
- #define W25Qx_TIMEOUT_VALUE 1000
- #define W25QX_PAGE_SIZE 0x100
- #define FLASH_Enable_Reset 0x66

  W25Qxx Commands.
- #define FLASH_Reset_Device 0x99
- #define FLASH_Empty 0x00
- #define FLASH_ID_16Byte 0x90
- #define FLASH_ID_8Byte 0xAB
- #define FLASH_ENABLE_Write 0x06
- #define FLASH_Erase 0x20
- #define FLASH_All_Erase 0xC7
- #define FLASH_Read 0x03
- #define FLASH_Write 0x02
- #define Read_Status_Reg_1 0x05
- #define Read_Status_Reg_2 0x35
- #define Read_Status_Reg_3 015

Enumerations

- enum W25Qx_StatusTypeDef { W25Qx_OK = 0x00U , W25Qx_ERROR = 0x01U , W25Qx_BUSY = 0x02U , W25Qx_TimeOut = 0x03U }

  W25Qxx status Configuration Structure Definition.

Functions

- void W25Qx_Read_ID_8 (uint8_t ∗ID)

  Read W25Qxx Device ID(ID7-ID0)
- void W25Qx_Read_ID_16 (uint16_t ∗ID)

  Read W25Qxx Manufacturer ID + Device ID.
- uint8_t W25Qx_WriteEnable (void)

  Enable Write for W25Qxx.
- void W25Qx_Coerce_Reset (void)

  Coerce W25Qxx to Reset.
- uint8_t W25Qx_Reset (void)

  Reset W25Qxx.
- uint8_t W25Qx_Erase_Sector (uint32_t EraseAddr)

  Erase the specified sector of the W25Qxx.
- uint8_t W25Qx_EraseALL (void)

  Erase all block of the W25Qxx.
- uint8_t W25Qx_Read_Data (uint8_t ∗Data, uint32_t Addr, uint32_t ReadSize)

  Read an amount of Data to the W25Qxx.
- uint8_t W25Qx_Write_Data (uint8_t ∗Data, uint32_t WriteAddr, uint32_t Size)

  Write an amount of data to the W25Qxx.

4.3.1 Detailed Description

Header file of W25Qxx module.

Author

Emotion_Thorn

Version

V1.0

Date

2023-05-05

Definition in file W25Qxx.h.

4.3.2 Macro Definition Documentation

4.3.2.1 FLASH_All_Erase

```
#define FLASH_All_Erase 0xC7
```

Definition at line 44 of file W25Qxx.h.

4.3.2.2 Flash_CS_Discard

```
#define Flash_CS_Discard( ) HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_SET)
```

Definition at line 17 of file W25Qxx.h.

4.3.2.3 Flash_CS_Select

```
#define Flash_CS_Select( ) HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_RESET)
```

Definition at line 16 of file W25Qxx.h.

4.3.2.4 FLASH_Empty

```
#define FLASH_Empty 0x00
```

Definition at line 39 of file W25Qxx.h.

4.3.2.5 FLASH_Enable_Reset

```
#define FLASH_Enable_Reset 0x66
```

W25Qxx Commands.

Definition at line 37 of file W25Qxx.h.

4.3.2.6 FLASH_ENABLE_Write

```
#define FLASH_ENABLE_Write 0x06
```

Definition at line 42 of file W25Qxx.h.

4.3.2.7 FLASH_Erase

```
#define FLASH_Erase 0x20
```

Definition at line 43 of file W25Qxx.h.

4.3.2.8 FLASH_ID_16Byte

```
#define FLASH_ID_16Byte 0x90
```

Definition at line 40 of file W25Qxx.h.

4.3.2.9 FLASH_ID_8Byte

```
#define FLASH_ID_8Byte 0xAB
```

Definition at line 41 of file W25Qxx.h.

4.3.2.10 FLASH_Read

```
#define FLASH_Read 0x03
```

Definition at line 45 of file W25Qxx.h.

4.3.2.11 FLASH_Reset_Device

```
#define FLASH_Reset_Device 0x99
```

Definition at line 38 of file W25Qxx.h.

4.3.2.12 FLASH_Write

```
#define FLASH_Write 0x02
```

Definition at line 46 of file W25Qxx.h.

4.3.2.13 Read_Status_Reg_1

```
#define Read_Status_Reg_1 0x05
```

Definition at line 48 of file W25Qxx.h.

4.3.2.14 Read_Status_Reg_2

```
#define Read_Status_Reg_2 0x35
```

Definition at line 49 of file W25Qxx.h.

4.3.2.15 Read_Status_Reg_3

```
#define Read_Status_Reg_3 015
```

Definition at line 50 of file W25Qxx.h.

4.3.2.16 W25Qx_Erase_TIMEOUT_VALUE

```
#define W25Qx_Erase_TIMEOUT_VALUE 5000
```

W25Qxx Configuration.

Definition at line 31 of file W25Qxx.h.

4.3.2.17 W25QX_PAGE_SIZE

```
#define W25QX_PAGE_SIZE 0x100
```

Definition at line 33 of file W25Qxx.h.

4.3.2.18 W25Qx_TIMEOUT_VALUE

```
#define W25Qx_TIMEOUT_VALUE 1000
```

Definition at line 32 of file W25Qxx.h.

4.3.3 Enumeration Type Documentation

4.3.3.1 W25Qx_StatusTypeDef

```
enum W25Qx_StatusTypeDef
```

W25Qxx status Configuration Structure Definition.

Enumerator

W25Qx_OK	
W25Qx_ERROR	
W25Qx_BUSY	
W25Qx_TimeOut	

Definition at line 21 of file W25Qxx.h.

4.4 W25Qxx.h

Go to the documentation of this file.

```

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef W25QXX_H_
#define W25QXX_H_
/* Includes ------------------------------------------------------------------*/
#include "spi.h"

#define Flash_CS_Select() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_RESET)
#define Flash_CS_Discard() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_SET)
typedef enum
{
  W25Qx_OK = 0x00U,
  W25Qx_ERROR = 0x01U,
  W25Qx_BUSY = 0x02U,
  W25Qx_TimeOut = 0x03U
} W25Qx_StatusTypeDef;
#define W25Qx_Erase_TIMEOUT_VALUE 5000
#define W25Qx_TIMEOUT_VALUE 1000
#define W25QX_PAGE_SIZE 0x100
#define FLASH_Enable_Reset 0x66
#define FLASH_Reset_Device 0x99
#define FLASH_Empty 0x00
#define FLASH_ID_16Byte 0x90
#define FLASH_ID_8Byte 0xAB
#define FLASH_ENABLE_Write 0x06
#define FLASH_Erase 0x20
#define FLASH_All_Erase 0xC7
#define FLASH_Read 0x03
#define FLASH_Write 0x02
/* Register Operations */
#define Read_Status_Reg_1 0x05
#define Read_Status_Reg_2 0x35
#define Read_Status_Reg_3 015
/* Exported functions --------------------------------------------------------*/
/* Read W25Qxx ID functions  ********************************/
void W25Qx_Read_ID_8(uint8_t *ID);
void W25Qx_Read_ID_16(uint16_t *ID);

/* Operation for W25Qxx functions  ********************************/
uint8_t W25Qx_WriteEnable(void);
void W25Qx_Coerce_Reset(void);
uint8_t W25Qx_Reset(void);
uint8_t W25Qx_Erase_Sector(uint32_t EraseAddr);
uint8_t W25Qx_EraseALL(void);
uint8_t W25Qx_Read_Data(uint8_t *Data, uint32_t Addr, uint32_t ReadSize);
uint8_t W25Qx_Write_Data(uint8_t *Data, uint32_t WriteAddr, uint32_t Size);

#endif /* W25QXX_H_ */
```

Index