

Demo

V1.0

Generated by Doxygen 1.9.6

1 Module Index	1
1.1 Modules	1
2 File Index	3
2.1 File List	3
3 Module Documentation	5
3.1 W25Qxx_Exported_Functions	5
3.1.1 Detailed Description	5
3.1.2 Function Documentation	5
3.1.2.1 W25Qx_Coerce_Reset()	6
3.1.2.2 W25Qx_Erase_Sector()	6
3.1.2.3 W25Qx_EraseALL()	6
3.1.2.4 W25Qx_Read_Data()	6
3.1.2.5 W25Qx_Read_ID_16()	7
3.1.2.6 W25Qx_Read_ID_8()	7
3.1.2.7 W25Qx_Reset()	8
3.1.2.8 W25Qx_Write_Data()	8
3.1.2.9 W25Qx_WriteEnable()	9
4 File Documentation	11
4.1 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25Qxx.c File Reference	11
4.1.1 Detailed Description	12
4.1.2 Function Documentation	12
4.1.2.1 Flash_Read_Byte()	12
4.1.2.2 Flash_Send_Byte()	13
4.2 W25Qxx.c	13
4.3 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25Qxx.h File Reference	16
4.3.1 Detailed Description	18
4.3.2 Macro Definition Documentation	18
4.3.2.1 FLASH_All_Erase	18
4.3.2.2 Flash_CS_Discard	18
4.3.2.3 Flash_CS_Select	18
4.3.2.4 FLASH_Empty	19
4.3.2.5 FLASH_Enable_Reset	19
4.3.2.6 FLASH_ENABLE_Write	19
4.3.2.7 FLASH_Erase	19
4.3.2.8 FLASH_ID_16Byte	19
4.3.2.9 FLASH_ID_8Byte	19
4.3.2.10 FLASH_Read	20
4.3.2.11 FLASH_Reset_Device	20
4.3.2.12 FLASH_Write	20
4.3.2.13 Read_Status_Reg_1	20

4.3.2.14 Read_Status_Reg_2	20
4.3.2.15 Read_Status_Reg_3	20
4.3.2.16 W25Qx_Erase_TIMEOUT_VALUE	21
4.3.2.17 W25QX_PAGE_SIZE	21
4.3.2.18 W25Qx_TIMEOUT_VALUE	21
4.3.3 Enumeration Type Documentation	21
4.3.3.1 W25Qx_StatusTypeDef	21
4.4 W25Qxx.h	21
Index	23

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

W25Qxx_Exported_Functions	5
-------------------------------------	-------------------

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/ W25Qxx.c	
This file provides information about the W25Qxx firmware functions	11
F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/ W25Qxx.h	
Header file of W25Qxx module	16

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

<i>EraseAddr</i>	: start addr of erase
------------------	-----------------------

Return values

<i>W25Qxx</i>	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

<i>W25Qxx</i>	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

<i>Data</i>	: pointer to data buffer
<i>Addr</i>	: start addr of read
<i>ReadSize</i>	: amount of data to be read

Return values

<i>W25Qxx</i>	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

<i>ID:Data</i>	to be receive
----------------	---------------

Return values

<i>None</i>	
-------------	--

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

<i>ID:Data</i>	to be receive
----------------	---------------

Return values

<i>None</i>	
-------------	--

Definition at line 27 of file [W25Qxx.c](#).

3.1.2.7 W25Qx_Reset()

```
uint8_t W25Qx_Reset (
    void )
```

Reset W25Qxx.

Return values

<i>W25Qxx</i>	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

<i>Data</i>	: pointer to data buffer
<i>WriteAddr</i>	: start addr of Write
<i>Size</i>	: amount of data to be write

Note

Size No more than 256byte.

Return values

<i>W25Qxx</i>	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

<i>W25Qxx</i>	status
---------------	--------

Definition at line 96 of file [W25Qxx.c](#).

Chapter 4

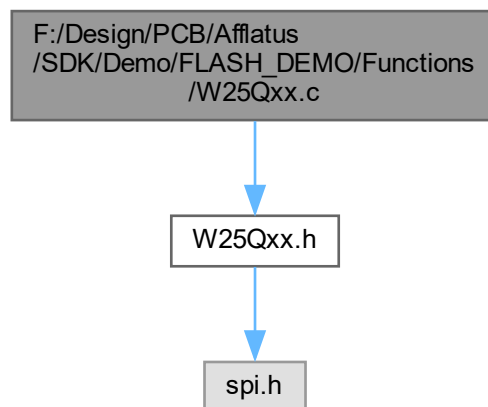
File Documentation

4.1 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25Qxx.c File Reference

This file provides information about the W25Qxx firmware functions.

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

Include dependency graph for W25Qxx.c:



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)
```

```

00097 {
00098     uint8_t cmd[] = {FLASH_ENABLE_Write};
00099     uint32_t StartTime = HAL_GetTick();
00100
00101     /*Select the FLASH: Chip Select low */
00102     Flash_CS_Select();
00103     /* Send the read ID command */
00104     HAL_SPI_Transmit(&hspi2, cmd, 1, W25Qx_TIMEOUT_VALUE);
00105     /*Deselect the FLASH: Chip Select high */
00106     Flash_CS_Discard();
00107
00108     /* Wait the end of Flash writing */
00109     while (W25Qx_Read_Busy() == W25Qx_BUSY)
00110     {
00111         /* Check for the Timeout */
00112         if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
00113         {
00114             return W25Qx_TimeOut;
00115         }
00116         HAL_Delay(1);
00117     }
00118     return W25Qx_OK;
00119 }
00120
00124 void W25Qx_Coerce_Reset(void)
00125 {
00126     uint8_t cmd[] = {FLASH_Enable_Reset, FLASH_Reset_Device};
00127     Flash_CS_Select();
00128     /* Send the read ID command */
00129     HAL_SPI_Transmit(&hspi2, cmd, 2, W25Qx_TIMEOUT_VALUE);
00130     Flash_CS_Discard();
00131     HAL_Delay(30);
00132 }
00137 uint8_t W25Qx_Reset(void)
00138 {
00139     uint8_t cmd[] = {FLASH_Enable_Reset, FLASH_Reset_Device};
00140     uint32_t StartTime = HAL_GetTick();
00141     while (W25Qx_Read_Busy() == W25Qx_BUSY)
00142     {
00143         if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
00144         {
00145             return W25Qx_TimeOut;
00146         }
00147     }
00148     Flash_CS_Select();
00149     /* Send the read ID command */
00150     HAL_SPI_Transmit(&hspi2, cmd, 2, W25Qx_TIMEOUT_VALUE);
00151     Flash_CS_Discard();
00152     HAL_Delay(30);
00153     return W25Qx_OK;
00154 }
00162 uint8_t W25Qx_Read_Data(uint8_t *Data, uint32_t Addr, uint32_t ReadSize)
00163 {
00164     uint8_t cmd[4];
00165     /* Configure the command */
00166     cmd[0] = FLASH_Read;
00167     cmd[1] = (uint8_t)(Addr >> 16);
00168     cmd[2] = (uint8_t)(Addr >> 8);
00169     cmd[3] = (uint8_t)(Addr);
00170     Flash_CS_Select();
00171     /* Send the read ID command */
00172     HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
00173     /* Reception of the data */
00174     if (HAL_SPI_Receive(&hspi2, Data, ReadSize, W25Qx_TIMEOUT_VALUE) != HAL_OK)
00175     {
00176         return W25Qx_ERROR;
00177     }
00178     Flash_CS_Discard();
00179     return W25Qx_OK;
00180 }
00189 uint8_t W25Qx_Write_Data(uint8_t *Data, uint32_t WriteAddr, uint32_t Size)
00190 {
00191     uint8_t cmd[4];
00192     uint32_t end_addr, current_size, current_addr;
00193     uint32_t StartTime = HAL_GetTick();
00194
00195     /* Calculation of the size between the write address and the end of the page */
00196     current_addr = 0;
00197
00198     while (current_addr <= WriteAddr)
00199     {
00200         current_addr += W25Qx_PAGE_SIZE;
00201     }
00202     current_size = current_addr - WriteAddr;
00203
00204     /* Check if the size of the data is less than the remaining place in the page */
00205     if (current_size > Size)

```

```

00206     {
00207         current_size = Size;
00208     }
00209
00210     /* Initialize the adress variables */
00211     current_addr = WriteAddr;
00212     end_addr = WriteAddr + Size;
00213
00214     /* Perform the write page by page */
00215     do
00216     {
00217         /* Configure the command */
00218         cmd[0] = FLASH_Write;
00219         cmd[1] = (uint8_t)(current_addr >> 16);
00220         cmd[2] = (uint8_t)(current_addr >> 8);
00221         cmd[3] = (uint8_t)(current_addr);
00222
00223         /* Enable write operations */
00224         W25Qx_WriteEnable();
00225
00226         Flash_CS_Select();
00227         /* Send the command */
00228         if (HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE) != HAL_OK)
00229         {
00230             return W25Qx_ERROR;
00231         }
00232
00233         /* Transmission of the data */
00234         if (HAL_SPI_Transmit(&hspi2, Data, current_size, W25Qx_TIMEOUT_VALUE) != HAL_OK)
00235         {
00236             return W25Qx_ERROR;
00237         }
00238         Flash_CS_Discard();
00239         /* Wait the end of Flash writing */
00240         while (W25Qx_Read_Busy() == W25Qx_BUSY)
00241         {
00242             /* Check for the Timeout */
00243             if ((HAL_GetTick() - StartTime) > W25Qx_TIMEOUT_VALUE)
00244             {
00245                 return W25Qx_TimeOut;
00246             }
00247             // delay(1);
00248         }
00249
00250         /* Update the address and size variables for next page programming */
00251         current_addr += current_size;
00252         Data += current_size;
00253         current_size = ((current_addr + W25QX_PAGE_SIZE) > end_addr) ? (end_addr - current_addr) :
W25QX_PAGE_SIZE;
00254     } while (current_addr < end_addr);
00255
00256     return W25Qx_OK;
00257 }
00263 uint8_t W25Qx_Erase_Sector(uint32_t EraseAddr)
00264 {
00265     uint8_t cmd[4];
00266     uint32_t StartTime = HAL_GetTick();
00267     cmd[0] = FLASH_Erase;
00268     cmd[1] = (uint8_t)(EraseAddr >> 16);
00269     cmd[2] = (uint8_t)(EraseAddr >> 8);
00270     cmd[3] = (uint8_t)(EraseAddr);
00271
00272     /* Enable write operations */
00273     W25Qx_WriteEnable();
00274
00275     /*Select the FLASH: Chip Select low */
00276     Flash_CS_Select();
00277     /* Send the read ID command */
00278     HAL_SPI_Transmit(&hspi2, cmd, 4, W25Qx_TIMEOUT_VALUE);
00279     /*Deselect the FLASH: Chip Select high */
00280     Flash_CS_Discard();
00281     while (W25Qx_Read_Busy() == W25Qx_BUSY)
00282     {
00283         if ((HAL_GetTick() - StartTime) > W25Qx_Erase_TIMEOUT_VALUE)
00284         {
00285             return W25Qx_TimeOut;
00286         }
00287     }
00288     return W25Qx_OK;
00289 }
00294 uint8_t W25Qx_EraseALL(void)
00295 {
00296     uint8_t cmd[] = {FLASH_All_Erase};
00297     uint32_t StartTime = HAL_GetTick();
00298     W25Qx_WriteEnable();
00299     Flash_CS_Select();
00300     /* Send the read ID command */

```

```

00301     HAL_SPI_Transmit(&hspi2, cmd, 1, W25Qx_TIMEOUT_VALUE);
00302     Flash_CS_Discard();
00303     while (W25Qx_Read_Busy() == W25Qx_BUSY)
00304     {
00305         if ((HAL_GetTick() - StartTime) > W25Qx_Erase_TIMEOUT_VALUE)
00306         {
00307             return W25Qx_TimeOut;
00308         }
00309     }
00310     return W25Qx_OK;
00311 }

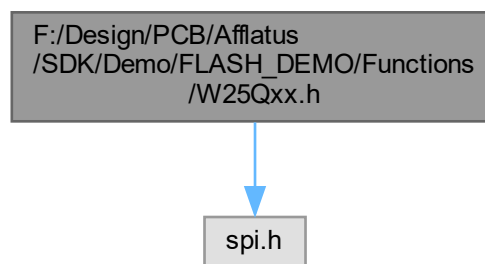
```

4.3 F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Functions/W25Qxx.h File Reference

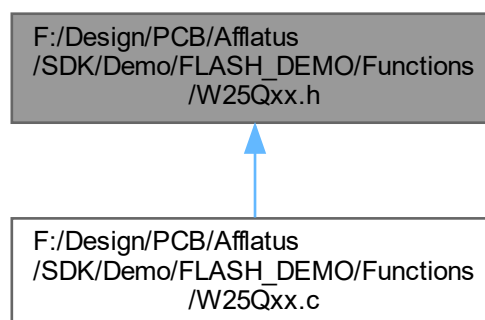
Header file of W25Qxx module.

```
#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.](#)

```

00001
00010 /* Define to prevent recursive inclusion -----*/
00011 #ifndef W25QXX_H_
00012 #define W25QXX_H_
00013 /* Includes -----*/
00014 #include "spi.h"
00015
00016 #define Flash_CS_Select() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_RESET)
00017 #define Flash_CS_Discard() HAL_GPIO_WritePin(Flash_CS_GPIO_Port, Flash_CS_Pin, GPIO_PIN_SET)
00021 typedef enum
00022 {
00023     W25Qx_OK = 0x00U,
00024     W25Qx_ERROR = 0x01U,
00025     W25Qx_BUSY = 0x02U,
00026     W25Qx_TimeOut = 0x03U
00027 } W25Qx_StatusTypeDef;
00031 #define W25Qx_Erase_TIMEOUT_VALUE 5000
00032 #define W25Qx_TIMEOUT_VALUE 1000
00033 #define W25QX_PAGE_SIZE 0x100
00037 #define FLASH_Enable_Reset 0x66
00038 #define FLASH_Reset_Device 0x99
00039 #define FLASH_Empty 0x00
00040 #define FLASH_ID_16Byte 0x90
00041 #define FLASH_ID_8Byte 0xAB
00042 #define FLASH_ENABLE_Write 0x06
00043 #define FLASH_Erase 0x20
00044 #define FLASH_All_Erase 0xC7
00045 #define FLASH_Read 0x03
00046 #define FLASH_Write 0x02
00047 /* Register Operations */
00048 #define Read_Status_Reg_1 0x05
00049 #define Read_Status_Reg_2 0x35
00050 #define Read_Status_Reg_3 015
00051 /* Exported functions -----*/
00057 /* Read W25Qxx ID functions *****/
00058 void W25Qx_Read_ID_8(uint8_t *ID);
00059 void W25Qx_Read_ID_16(uint16_t *ID);
00060
00061 /* Operation for W25Qxx functions *****/
00062 uint8_t W25Qx_WriteEnable(void);
00063 void W25Qx_Coerce_Reset(void);
00064 uint8_t W25Qx_Reset(void);
00065 uint8_t W25Qx_Erase_Sector(uint32_t EraseAddr);
00066 uint8_t W25Qx_EraseALL(void);
00067 uint8_t W25Qx_Read_Data(uint8_t *Data, uint32_t Addr, uint32_t ReadSize);
00068 uint8_t W25Qx_Write_Data(uint8_t *Data, uint32_t WriteAddr, uint32_t Size);
00069
00073 #endif /* W25QXX_H_ */

```

Index

F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Function/W25Qxx.c, 11, 13
F:/Design/PCB/Afflatus/SDK/Demo/FLASH_DEMO/Function/W25Qxx.h, 16, 21
FLASH_All_Erase
 W25Qxx.h, 18
Flash_CS_Discard
 W25Qxx.h, 18
Flash_CS_Select
 W25Qxx.h, 18
FLASH_Empty
 W25Qxx.h, 18
FLASH_Enable_Reset
 W25Qxx.h, 19
FLASH_ENABLE_Write
 W25Qxx.h, 19
FLASH_Erase
 W25Qxx.h, 19
FLASH_ID_16Byte
 W25Qxx.h, 19
FLASH_ID_8Byte
 W25Qxx.h, 19
FLASH_Read
 W25Qxx.h, 19
Flash_Read_Byte
 W25Qxx.c, 12
FLASH_Reset_Device
 W25Qxx.h, 20
Flash_Send_Byte
 W25Qxx.c, 12
FLASH_Write
 W25Qxx.h, 20

Read_Status_Reg_1
 W25Qxx.h, 20
Read_Status_Reg_2
 W25Qxx.h, 20
Read_Status_Reg_3
 W25Qxx.h, 20

W25Qx_BUSY
 W25Qxx.h, 21
W25Qx_Coerce_Reset
 W25Qxx_Exported_Functions, 5
W25Qx_Erase_Sector
 W25Qxx_Exported_Functions, 6
W25Qx_Erase_TIMEOUT_VALUE
 W25Qxx.h, 20
W25Qx_EraseALL
 W25Qxx_Exported_Functions, 6

W25Qx_ERROR
 W25Qxx.h, 21
W25Qx_OK
 W25Qxx.h, 21
W25QX_PAGE_SIZE
 W25Qxx.h, 21
W25Qx_Read_Data
 W25Qxx_Exported_Functions, 6
W25Qx_Read_ID_16
 W25Qxx_Exported_Functions, 7
W25Qx_Read_ID_8
 W25Qxx_Exported_Functions, 7
W25Qx_Reset
 W25Qxx_Exported_Functions, 8
W25Qx_StatusTypeDef
 W25Qxx.h, 21
W25Qx_TimeOut
 W25Qxx.h, 21
W25Qx_TIMEOUT_VALUE
 W25Qxx.h, 21
W25Qx_Write_Data
 W25Qxx_Exported_Functions, 8
W25Qx_WriteEnable
 W25Qxx_Exported_Functions, 9
W25Qxx.c
 Flash_Read_Byte, 12
 Flash_Send_Byte, 12
W25Qxx.h
 FLASH_All_Erase, 18
 Flash_CS_Discard, 18
 Flash_CS_Select, 18
 FLASH_Empty, 18
 FLASH_Enable_Reset, 19
 FLASH_ENABLE_Write, 19
 FLASH_Erase, 19
 FLASH_ID_16Byte, 19
 FLASH_ID_8Byte, 19
 FLASH_Read, 19
 FLASH_Reset_Device, 20
 FLASH_Write, 20
 Read_Status_Reg_1, 20
 Read_Status_Reg_2, 20
 Read_Status_Reg_3, 20
 W25Qx_BUSY, 21
 W25Qx_Erase_TIMEOUT_VALUE, 20
 W25Qx_ERROR, 21
 W25Qx_OK, 21
 W25QX_PAGE_SIZE, 21
 W25Qx_StatusTypeDef, 21

- W25Qx_TimeOut, [21](#)
- W25Qx_TIMEOUT_VALUE, [21](#)
- W25Qxx_Exported_Functions, [5](#)
- W25Qx_Coerce_Reset, [5](#)
- W25Qx_Erase_Sector, [6](#)
- W25Qx_EraseALL, [6](#)
- W25Qx_Read_Data, [6](#)
- W25Qx_Read_ID_16, [7](#)
- W25Qx_Read_ID_8, [7](#)
- W25Qx_Reset, [8](#)
- W25Qx_Write_Data, [8](#)
- W25Qx_WriteEnable, [9](#)