

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

eeprom.c File Reference

Author:

MCD Application Team

[More...](#)

```
#include "eeprom.h"
```

[Go to the source code of this file.](#)

Functions

static uint16_t	EE FindValidPage (uint8_t Operation) <i>Find valid Page for write or read operation.</i>
static FLASH_Status	EE Format (void) <i>Erases PAGE0 and PAGE1 and writes VALID_PAGE header to PAGE0.</i>
uint16_t	EE Init (void) <i>Restore the pages to a known good state in case of page's status corruption after a power loss.</i>
static uint16_t	EE PageTransfer (uint16_t VirtAddress, uint16_t Data) <i>Transfers last updated variables data from the full Page to an empty one.</i>
uint16_t	EE ReadVariable (uint16_t VirtAddress, uint16_t *Data) <i>Returns the last stored variable data, if found, which correspond to the passed virtual address.</i>
static uint16_t	EE VerifyPageFullWriteVariable (uint16_t VirtAddress, uint16_t Data) <i>Verify if active page is full and Writes variable in EEPROM.</i>
uint16_t	EE WriteVariable (uint16_t VirtAddress, uint16_t Data) <i>Writes/upadtes variable data in EEPROM.</i>

Variables

uint16_t	DataVar = 0
----------	-----------------------------

uint16_t	VirtAddVarTab [NumbOfVar]
----------	---

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 This file provides all the EEPROM emulation firmware functions.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [eeprom.c](#).

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

eeprom.h File Reference

Author:

MCD Application Team

[More...](#)

```
#include "stm32f10x.h"
```

[Go to the source code of this file.](#)

Defines

#define	EEPROM_START_ADDRESS	((uint32_t)0x08010000)
#define	ERASED	((uint16_t)0xFFFF)
#define	NO_VALID_PAGE	((uint16_t)0x00AB)
#define	NumbofVar	((uint8_t)0x03)
#define	PAGE0	((uint16_t)0x0000)
#define	PAGE0_BASE_ADDRESS	((uint32_t)(EEPROM_START_ADDRESS + 0x000))
#define	PAGE0_END_ADDRESS	((uint32_t)(EEPROM_START_ADDRESS + (PAGE_SIZE - 1)))
#define	PAGE1	((uint16_t)0x0001)
#define	PAGE1_BASE_ADDRESS	((uint32_t)(EEPROM_START_ADDRESS + PAGE_SIZE))
#define	PAGE1_END_ADDRESS	((uint32_t)(EEPROM_START_ADDRESS + (2 * PAGE_SIZE - 1)))
#define	PAGE_FULL	((uint8_t)0x80)
#define	READ_FROM_VALID_PAGE	((uint8_t)0x00)
#define	RECEIVE_DATA	((uint16_t)0xEEEE)
#define	VALID_PAGE	((uint16_t)0x0000)
#define	WRITE_IN_VALID_PAGE	((uint8_t)0x01)

Functions

uint16_t	EE_Init (void) <i>Restore the pages to a known good state in case of page's status corruption after a power loss.</i>
uint16_t	EE_ReadVariable (uint16_t VirtAddress, uint16_t *Data) <i>Returns the last stored variable data, if found, which correspond to the passed virtual address.</i>
uint16_t	EE_WriteVariable (uint16_t VirtAddress, uint16_t Data) <i>Writes/upadtes variable data in EEPROM.</i>

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 This file contains all the functions prototypes for the EEPROM emulation firmware library.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [eeprom.h](#).

Define Documentation

```
#define EEPROM_START_ADDRESS ((uint32_t)0x08010000)
```

Definition at line [38](#) of file [eeprom.h](#).

Referenced by [EE_ReadVariable\(\)](#), and [EE_VerifyPageFullWriteVariable\(\)](#).

```
#define ERASED ((uint16_t)0xFFFF)
```

Definition at line [56](#) of file [eeprom.h](#).

Referenced by [EE_Init\(\)](#).

```
#define NO_VALID_PAGE ((uint16_t)0x00AB)
```

Definition at line [53](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_PageTransfer\(\)](#), [EE_ReadVariable\(\)](#), and [EE_VerifyPageFullWriteVariable\(\)](#).

```
#define NumOfVar ((uint8_t)0x03)
```

Definition at line [68](#) of file [eeprom.h](#).

Referenced by [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define PAGE0 ((uint16_t)0x0000)
```

Definition at line [49](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_PageTransfer\(\)](#), [EE_ReadVariable\(\)](#), and [EE_VerifyPageFullWriteVariable\(\)](#).

```
#define PAGE0_BASE_ADDRESS ((uint32_t)(EEPROM_START_ADDRESS + 0x000))
```

Definition at line [42](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_Format\(\)](#), [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define PAGE0_END_ADDRESS ((uint32_t)(EEPROM_START_ADDRESS + (PAGE_SIZE -
```

Definition at line [43](#) of file [eeprom.h](#).

```
#define PAGE1 ((uint16_t)0x0001)
```

Definition at line [50](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define PAGE1_BASE_ADDRESS ((uint32_t)(EEPROM_START_ADDRESS + PAGE_SIZE))
```

Definition at line [45](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_Format\(\)](#), [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define PAGE1_END_ADDRESS ((uint32_t)(EEPROM_START_ADDRESS + (2 * PAGE_SIZE)))
```

Definition at line [46](#) of file [eeprom.h](#).

```
#define PAGE_FULL ((uint8_t)0x80)
```

Definition at line [65](#) of file [eeprom.h](#).

Referenced by [EE_VerifyPageFullWriteVariable\(\)](#), and [EE_WriteVariable\(\)](#).

```
#define READ_FROM_VALID_PAGE ((uint8_t)0x00)
```

Definition at line [61](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_PageTransfer\(\)](#), and [EE_ReadVariable\(\)](#).

```
#define RECEIVE_DATA ((uint16_t)0xEEEE)
```

Definition at line [57](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define VALID_PAGE ((uint16_t)0x0000)
```

Definition at line [58](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), [EE_Format\(\)](#), [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
#define WRITE_IN_VALID_PAGE ((uint8_t)0x01)
```

Definition at line [62](#) of file [eeprom.h](#).

Referenced by [EE_FindValidPage\(\)](#), and
[EE_VerifyPageFullWriteVariable\(\)](#).

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

main.c File Reference

Author:

MCD Application Team

[More...](#)

```
#include "eeprom.h"
```

[Go to the source code of this file.](#)

Functions

int	main (void)
	<i>Main program.</i>

Variables

FLASH_Status	FlashStatus
ErrorStatus	HSEStartUpStatus
uint16_t	VarValue = 0
uint16_t	VirtAddVarTab [NumOfVar] = {0x5555, 0x6666, 0x7777}

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 Main program body

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [main.c](#).

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

stm32f10x_conf.h File Reference

Author:

MCD Application Team

[More...](#)

```
#include "stm32f10x_flash.h"
```

[Go to the source code of this file.](#)

Defines

```
#define assert\_param(expr) ((void)0)
```

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 Library configuration file.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [stm32f10x_conf.h](#).

Define Documentation

```
#define assert_param ( expr ) ((void)0)
```

Definition at line [69](#) of file [stm32f10x_conf.h](#).

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

stm32f10x_it.c File Reference

Author:

MCD Application Team

[More...](#)

```
#include "stm32f10x\_it.h"
```

[Go to the source code of this file.](#)

Functions

void	BusFault_Handler (void) <i>This function handles Bus Fault exception.</i>
void	DebugMon_Handler (void) <i>This function handles Debug Monitor exception.</i>
void	HardFault_Handler (void) <i>This function handles Hard Fault exception.</i>
void	MemManage_Handler (void) <i>This function handles Memory Manage exception.</i>
void	NMI_Handler (void) <i>This function handles NMI exception.</i>
void	PendSV_Handler (void) <i>This function handles PendSVC exception.</i>
void	SVC_Handler (void) <i>This function handles SVCcall exception.</i>
void	SysTick_Handler (void) <i>This function handles SysTick Handler.</i>
void	UsageFault_Handler (void) <i>This function handles Usage Fault exception.</i>

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 Main Interrupt Service Routines. This file provides template for all exceptions handler and peripherals interrupt service routine.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [stm32f10x_it.c](#).

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

stm32f10x_it.h File Reference

Author:

MCD Application Team

[More...](#)

```
#include "stm32f10x.h"
```

[Go to the source code of this file.](#)

Functions

void	BusFault_Handler (void) <i>This function handles Bus Fault exception.</i>
void	DebugMon_Handler (void) <i>This function handles Debug Monitor exception.</i>
void	HardFault_Handler (void) <i>This function handles Hard Fault exception.</i>
void	MemManage_Handler (void) <i>This function handles Memory Manage exception.</i>
void	NMI_Handler (void) <i>This function handles NMI exception.</i>
void	PendSV_Handler (void) <i>This function handles PendSVC exception.</i>
void	SVC_Handler (void) <i>This function handles SVCcall exception.</i>
void	SysTick_Handler (void) <i>This function handles SysTick Handler.</i>
void	UsageFault_Handler (void) <i>This function handles Usage Fault exception.</i>

Detailed Description

Author:

MCD Application Team

Version:

V3.1.0

Date:

07/27/2009 This file contains the headers of the interrupt handlers.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

© **COPYRIGHT 2009 STMicroelectronics**

Definition in file [stm32f10x_it.h](#).

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- a -

- assert_param : [stm32f10x_conf.h](#)

- b -

- BusFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)

- d -

- DataVar : [eeprom.c](#)
- DebugMon_Handler() : [stm32f10x_it.c](#) , [stm32f10x_it.h](#)

- e -

- EE_FindValidPage() : [eeprom.c](#)
- EE_Format() : [eeprom.c](#)
- EE_Init() : [eeprom.c](#) , [eeprom.h](#)
- EE_PageTransfer() : [eeprom.c](#)
- EE_ReadVariable() : [eeprom.h](#) , [eeprom.c](#)
- EE_VerifyPageFullWriteVariable() : [eeprom.c](#)
- EE_WriteVariable() : [eeprom.c](#) , [eeprom.h](#)
- EEPROM_START_ADDRESS : [eeprom.h](#)
- ERASED : [eeprom.h](#)

- f -

- FlashStatus : [main.c](#)

- h -

- HardFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- HSEStartUpStatus : [main.c](#)

- m -

- main() : [main.c](#)
- MemManage_Handler() : [stm32f10x_it.c](#) , [stm32f10x_it.h](#)

- n -

- NMI_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- NO_VALID_PAGE : [eeprom.h](#)
- NumbOfVar : [eeprom.h](#)

- p -

- PAGE0 : [eeprom.h](#)
- PAGE0_BASE_ADDRESS : [eeprom.h](#)
- PAGE0_END_ADDRESS : [eeprom.h](#)
- PAGE1 : [eeprom.h](#)
- PAGE1_BASE_ADDRESS : [eeprom.h](#)
- PAGE1_END_ADDRESS : [eeprom.h](#)
- PAGE_FULL : [eeprom.h](#)
- PendSV_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)

- r -

- READ_FROM_VALID_PAGE : [eeprom.h](#)
- RECEIVE_DATA : [eeprom.h](#)

- s -

- SVC_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- SysTick_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)

- u -

- UsageFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)

- v -

- VALID_PAGE : [eeprom.h](#)
- VarValue : [main.c](#)
- VirtAddVarTab : [eeprom.c](#) , [main.c](#)

- w -

- WRITE_IN_VALID_PAGE : [eeprom.h](#)

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

AN2594 EEPROM_Emulation

Directory Content

src

- [EEPROM_Emulation/src/main.c](#)
- [EEPROM_Emulation/src/stm32f10x_it.c](#)
- [EEPROM_Emulation/src/eeprom.c](#)

inc

- [EEPROM_Emulation/inc/stm32f10x_conf.h](#)
- [EEPROM_Emulation/inc/stm32f10x_it.h](#)
- [EEPROM_Emulation/inc/eeprom.h](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

AN2594 EEPROM_Emulation

Readme file

```
***** (C) COPYRIGHT 2009 STMicroelectronics *****
* @file      EEPROM_Emulation/readme.txt
* @author    MCD Application Team
* @version   V3.1.0
* @date      07/27/2009
* @brief     Description of the AN2594 "EEPROM emulation in STM32F10x
*****
* THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING C
* WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM
* TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR
* DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIM
* FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
* CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUC
*****
```

Example Description

This AN describes a software method for emulating EEPROM using the onchip Flash memory of the STM32F10xx devices.

Directory contents

+ inc

- [EEPROM_Emulation/inc/stm32f10x_conf.h](#) Library Configuration file
- [EEPROM_Emulation/inc/stm32f10x_it.h](#) Interrupt handlers header file
- [EEPROM_Emulation/inc/eeprom.h](#) EEPROM emulation Header file

+ src

- [EEPROM_Emulation/src/stm32f10x_it.c](#) Interrupt handlers

- [EEPROM Emulation/src/eeeprom.c](#) EEPROM emulation firmware functions
- [EEPROM Emulation/src/main.c](#) Main program

Hardware and Software environment

- This example runs on STM32F10x Connectivity line, High-Density, Medium-Density and Low-Density Devices.
- This example has been tested with STMicroelectronics STM3210C-EVAL (STM32F10x Connectivity line), STM3210E-EVAL (STM32F10x High-Density) and STM3210B-EVAL (STM32F10x Medium-Density) evaluation boards and can be easily tailored to any other supported device and development board.

How to use it ?

- RVMDK
 - Open the EEPROM_Emulation.Uv2 project
 - In the build toolbar select the project config:
 - STM3210C-EVAL: to configure the project for STM32 Connectivity line devices
 - STM3210B-EVAL: to configure the project for STM32 Medium-density devices
 - STM3210E-EVAL: to configure the project for STM32 High-density devices
 - Rebuild all files: Project->Rebuild all target files
 - Load project image: Debug->Start/Stop Debug Session
 - Run program: Debug->Run (F5)
- EWARMv5
 - Open the EEPROM_Emulation.eww workspace.
 - In the workspace toolbar select the project config:
 - STM3210C-EVAL: to configure the project for STM32 Connectivity line devices
 - STM3210B-EVAL: to configure the project for STM32 Medium-density devices
 - STM3210E-EVAL: to configure the project for STM32 High-

density devices

- Rebuild all files: Project->Rebuild all
- Load project image: Project->Debug
- Run program: Debug->Go(F5)

- RIDE

- Open the EEPROM_Emulation.rprj project.
- In the configuration toolbar(Project->properties) select the project config:
 - STM3210C-EVAL: to configure the project for STM32 Connectivity line devices

STM3210B-EVAL: to configure the project for STM32 Medium-density devices

- STM3210E-EVAL: to configure the project for STM32 High-density devices

Rebuild all files: Project->build project

- Load project image: Debug->start(ctrl+D)
- Run program: Debug->Run(ctrl+F9)

Note:

- Low-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.
- Medium-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 32 and 128 Kbytes.
- High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.
- Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

© ***COPYRIGHT 2009 STMicroelectronics***

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

AN2594: EEPROM emulation in STM32F10x microcontrollers Modules

Here is a list of all modules:

- [EEPROM Emulation](#)

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[File List](#)

[Globals](#)

AN2594: EEPROM emulation in STM32F10x microcontrollers File List

Here is a list of all files with brief descriptions:

eeprom.c [code]	Author: <i>MCD Application Team</i>
eeprom.h [code]	Author: <i>MCD Application Team</i>
main.c [code]	Author: <i>MCD Application Team</i>
stm32f10x_conf.h [code]	Author: <i>MCD Application Team</i>
stm32f10x_it.c [code]	Author: <i>MCD Application Team</i>
stm32f10x_it.h [code]	Author: <i>MCD Application Team</i>

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

AN2594: EEPROM emulation in STM32F10x microcontrollers Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

- [STM32F10x AN2594 FW V3.1.0](#)
 - [Project](#)
 - [EEPROM Emulation](#)
 - [inc](#)
 - [src](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

AN2594: EEPROM emulation in STM32F10x microcontrollers Related Pages

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x AN2594 FW V3.1.0](#)

STM32F10x_AN2594_FW_V3.1.0

Directory Reference

Directories

directory [Project](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x AN2594 FW V3.1.0](#) » [Project](#)

Project Directory Reference

Directories

directory [EEPROM Emulation](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x AN2594 FW V3.1.0](#) » [Project](#) » [EEPROM Emulation](#)

EEPROM_Emulation Directory Reference

Directories

directory	inc
directory	src

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

src Directory Reference

Files

file	eeprom.c [code]
	Author: <i>MCD Application Team</i>
file	main.c [code]
	Author: <i>MCD Application Team</i>
file	stm32f10x_it.c [code]
	Author: <i>MCD Application Team</i>

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

eeprom.h

[Go to the documentation of this file.](#)

```
00001 /**
00002  ****
00003  * @file    EEPROM_Emulation/inc/eeprom.h
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   This file contains all the functions prototypes for
00008  *          emulation firmware library.
00009  ****
00010  * @copy
00011  *
00012  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00013  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR
00014  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR
00015  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00016  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00017  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR
00018  *
00019  * <h2><center>COPYRIGHT 2009 STMicroelectronics</center>
00020  */
00021
00022 /* Define to prevent recursive inclusion -----
00023 #ifndef __EEPROM_H
00024 #define __EEPROM_H
00025
00026 /* Includes -----
00027 #include "stm32f10x.h"
00028
00029 /* Exported constants -----
00030 /* Define the STM32F10Xxx Flash page size depending on the used S
00031 #if defined (STM32F10X_LD) || defined (STM32F10X_MD)
00032 #define PAGE_SIZE (uint16_t)0x400 /* Page size = 1KByte */
00033 #elif defined (STM32F10X_HD) || defined (STM32F10X_CL)
00034 #define PAGE_SIZE (uint16_t)0x800 /* Page size = 2KByte */
00035 #endif
00036
00037 /* EEPROM start address in Flash */
00038 #define EEPROM_START_ADDRESS ((uint32_t)0x08010000) /* EEPROM start address
00039 after 64KByte of Flash */
00040
00041 /* Pages 0 and 1 base and end addresses */
00042 #define PAGE0_BASE_ADDRESS ((uint32_t)(EEPROM_START_ADDRESS +
```

```

00043 #define PAGE0_END_ADDRESS      ((uint32_t)(EEPROM_START_ADDRESS
00044
00045 #define PAGE1_BASE_ADDRESS      ((uint32_t)(EEPROM_START_ADDRESS
00046 #define PAGE1_END_ADDRESS      ((uint32_t)(EEPROM_START_ADDRESS
00047
00048 /* Used Flash pages for EEPROM emulation */
00049 #define PAGE0                    ((uint16_t)0x0000)
00050 #define PAGE1                    ((uint16_t)0x0001)
00051
00052 /* No valid page define */
00053 #define NO_VALID_PAGE            ((uint16_t)0x00AB)
00054
00055 /* Page status definitions */
00056 #define ERASED                   ((uint16_t)0xFFFF) /* PAGE is
00057 #define RECEIVE_DATA             ((uint16_t)0xEEEE) /* PAGE is
00058 #define VALID_PAGE              ((uint16_t)0x0000) /* PAGE co
00059
00060 /* Valid pages in read and write defines */
00061 #define READ_FROM_VALID_PAGE     ((uint8_t)0x00)
00062 #define WRITE_IN_VALID_PAGE     ((uint8_t)0x01)
00063
00064 /* Page full define */
00065 #define PAGE_FULL                ((uint8_t)0x80)
00066
00067 /* Variables' number */
00068 #define NumOfVar                 ((uint8_t)0x03)
00069
00070 /* Exported types -----
00071 /* Exported macro -----
00072 /* Exported functions -----
00073 uint16_t EE_Init(void);
00074 uint16_t EE_ReadVariable(uint16_t VirtAddress, uint16_t* Data);
00075 uint16_t EE_WriteVariable(uint16_t VirtAddress, uint16_t Data);
00076
00077 #endif /* __EEPROM_H */
00078
00079 /***** (C) COPYRIGHT 2009 STMicroelectronics *****/

```

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

eeeprom.c

[Go to the documentation of this file.](#)

```
00001 /**
00002  *****
00003  * @file    EEPROM_Emulation/src/eeeprom.c
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   This file provides all the EEPROM emulation firmware
00008  *****
00009  * @copy
00010  *
00011  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00012  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR
00013  * TIME. AS A RESULT, STMicroelectronics SHALL NOT BE HELD LIABLE FOR
00014  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00015  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00016  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR
00017  *
00018  * <h2><center>COPYRIGHT 2009 STMicroelectronics</center>
00019  */
00020 /** @addtogroup EEPROM_Emulation
00021  * @{
00022  */
00023
00024 /* Includes -----
00025 #include "eeeprom.h"
00026
00027 /* Private typedef -----
00028 /* Private define -----
00029 /* Private macro -----
00030 /* Private variables -----
00031
00032 /* Global variable used to store variable value in read sequence
00033 uint16_t DataVar = 0;
00034
00035 /* Virtual address defined by the user: 0xFFFF value is prohibited
00036 extern uint16_t VirtAddVarTab[NumbOfVar];
00037
00038 /* Private function prototypes -----
00039 /* Private functions -----
00040 static FLASH_Status EE_Format(void);
00041 static uint16_t EE_FindValidPage(uint8_t Operation);
00042 static uint16_t EE_VerifyPageFullWriteVariable(uint16_t VirtAddress);
```

```

00043 static uint16_t EE_PageTransfer(uint16_t VirtAddress, uint16_t Da
00044
00045 /**
00046  * @brief Restore the pages to a known good state in case of pa
00047  * corruption after a power loss.
00048  * @param None.
00049  * @retval - Flash error code: on write Flash error
00050  *         - FLASH_COMPLETE: on success
00051  */
00052 uint16_t EE_Init(void)
00053 {
00054     uint16_t PageStatus0 = 6, PageStatus1 = 6;
00055     uint16_t VarIdx = 0;
00056     uint16_t EepromStatus = 0, ReadStatus = 0;
00057     int16_t x = -1;
00058     uint16_t FlashStatus;
00059
00060     /* Get Page0 status */
00061     PageStatus0 = (*(__IO uint16_t*)PAGE0_BASE_ADDRESS);
00062     /* Get Page1 status */
00063     PageStatus1 = (*(__IO uint16_t*)PAGE1_BASE_ADDRESS);
00064
00065     /* Check for invalid header states and repair if necessary */
00066     switch (PageStatus0)
00067     {
00068         case ERASED:
00069             if (PageStatus1 == VALID_PAGE) /* Page0 erased, Page1 valid
00070             {
00071                 /* Erase Page0 */
00072                 FlashStatus = FLASH_ErasePage(PAGE0_BASE_ADDRESS);
00073                 /* If erase operation was failed, a Flash error code is r
00074                 if (FlashStatus != FLASH_COMPLETE)
00075                 {
00076                     return FlashStatus;
00077                 }
00078             }
00079             else if (PageStatus1 == RECEIVE_DATA) /* Page0 erased, Page:
00080             {
00081                 /* Erase Page0 */
00082                 FlashStatus = FLASH_ErasePage(PAGE0_BASE_ADDRESS);
00083                 /* If erase operation was failed, a Flash error code is r
00084                 if (FlashStatus != FLASH_COMPLETE)
00085                 {
00086                     return FlashStatus;
00087                 }
00088                 /* Mark Page1 as valid */
00089                 FlashStatus = FLASH_ProgramHalfword(PAGE1_BASE_ADDRESS, V
00090                 /* If program operation was failed, a Flash error code is
00091                 if (FlashStatus != FLASH_COMPLETE)
00092                 {

```

```

00093         return FlashStatus;
00094     }
00095 }
00096 else /* First EEPROM access (Page0&1 are erased) or invalid
00097 {
00098     /* Erase both Page0 and Page1 and set Page0 as valid page
00099     FlashStatus = EE_Format();
00100     /* If erase/program operation was failed, a Flash error code
00101     if (FlashStatus != FLASH_COMPLETE)
00102     {
00103         return FlashStatus;
00104     }
00105 }
00106 break;
00107
00108 case RECEIVE_DATA:
00109     if (PageStatus1 == VALID_PAGE) /* Page0 receive, Page1 valid
00110     {
00111         /* Transfer data from Page1 to Page0 */
00112         for (VarIdx = 0; VarIdx < NumOfVar; VarIdx++)
00113         {
00114             if (( *(__IO uint16_t*)(PAGE0_BASE_ADDRESS + 6)) == VirtAddr)
00115             {
00116                 x = VarIdx;
00117             }
00118             if (VarIdx != x)
00119             {
00120                 /* Read the last variables' updates */
00121                 ReadStatus = EE_ReadVariable(VirtAddrVarTab[VarIdx], &VirtAddr);
00122                 /* In case variable corresponding to the virtual address is not found
00123                 if (ReadStatus != 0x1)
00124                 {
00125                     /* Transfer the variable to the Page0 */
00126                     EepromStatus = EE_VerifyPageFullWriteVariable(VirtAddr, VirtAddr);
00127                     /* If program operation was failed, a Flash error code
00128                     if (EepromStatus != FLASH_COMPLETE)
00129                     {
00130                         return EepromStatus;
00131                     }
00132                 }
00133             }
00134         }
00135         /* Mark Page0 as valid */
00136         FlashStatus = FLASH_ProgramHalfWord(PAGE0_BASE_ADDRESS, VirtAddr);
00137         /* If program operation was failed, a Flash error code is
00138         if (FlashStatus != FLASH_COMPLETE)
00139         {
00140             return FlashStatus;
00141         }

```

```

00142     /* Erase Page1 */
00143     FlashStatus = FLASH_ErasePage(PAGE1_BASE_ADDRESS);
00144     /* If erase operation was failed, a Flash error code is re
00145     if (FlashStatus != FLASH_COMPLETE)
00146     {
00147         return FlashStatus;
00148     }
00149 }
00150 else if (PageStatus1 == ERASED) /* Page0 receive, Page1 erase
00151 {
00152     /* Erase Page1 */
00153     FlashStatus = FLASH_ErasePage(PAGE1_BASE_ADDRESS);
00154     /* If erase operation was failed, a Flash error code is re
00155     if (FlashStatus != FLASH_COMPLETE)
00156     {
00157         return FlashStatus;
00158     }
00159     /* Mark Page0 as valid */
00160     FlashStatus = FLASH_ProgramHalfWord(PAGE0_BASE_ADDRESS, V
00161     /* If program operation was failed, a Flash error code is
00162     if (FlashStatus != FLASH_COMPLETE)
00163     {
00164         return FlashStatus;
00165     }
00166 }
00167 else /* Invalid state -> format eeprom */
00168 {
00169     /* Erase both Page0 and Page1 and set Page0 as valid page
00170     FlashStatus = EE_Format();
00171     /* If erase/program operation was failed, a Flash error co
00172     if (FlashStatus != FLASH_COMPLETE)
00173     {
00174         return FlashStatus;
00175     }
00176 }
00177 break;
00178
00179 case VALID_PAGE:
00180     if (PageStatus1 == VALID_PAGE) /* Invalid state -> format e
00181     {
00182         /* Erase both Page0 and Page1 and set Page0 as valid page
00183         FlashStatus = EE_Format();
00184         /* If erase/program operation was failed, a Flash error co
00185         if (FlashStatus != FLASH_COMPLETE)
00186         {
00187             return FlashStatus;
00188         }
00189     }
00190     else if (PageStatus1 == ERASED) /* Page0 valid, Page1 erase
00191     {

```

```

00192     /* Erase Page1 */
00193     FlashStatus = FLASH_ErasePage(PAGE1_BASE_ADDRESS);
00194     /* If erase operation was failed, a Flash error code is r
00195     if (FlashStatus != FLASH_COMPLETE)
00196     {
00197         return FlashStatus;
00198     }
00199 }
00200 else /* Page0 valid, Page1 receive */
00201 {
00202     /* Transfer data from Page0 to Page1 */
00203     for (VarIdx = 0; VarIdx < NumbOfVar; VarIdx++)
00204     {
00205         if ((__IO uint16_t*)(PAGE1_BASE_ADDRESS + 6)) == Virt
00206         {
00207             x = VarIdx;
00208         }
00209         if (VarIdx != x)
00210         {
00211             /* Read the last variables' updates */
00212             ReadStatus = EE_ReadVariable(VirtAddVarTab[VarIdx], &
00213             /* In case variable corresponding to the virtual addr
00214             if (ReadStatus != 0x1)
00215             {
00216                 /* Transfer the variable to the Page1 */
00217                 EepromStatus = EE_VerifyPageFullWriteVariable(VirtA
00218                 /* If program operation was failed, a Flash error c
00219                 if (EepromStatus != FLASH_COMPLETE)
00220                 {
00221                     return EepromStatus;
00222                 }
00223             }
00224         }
00225     }
00226     /* Mark Page1 as valid */
00227     FlashStatus = FLASH_ProgramHalfWord(PAGE1_BASE_ADDRESS, V
00228     /* If program operation was failed, a Flash error code is
00229     if (FlashStatus != FLASH_COMPLETE)
00230     {
00231         return FlashStatus;
00232     }
00233     /* Erase Page0 */
00234     FlashStatus = FLASH_ErasePage(PAGE0_BASE_ADDRESS);
00235     /* If erase operation was failed, a Flash error code is r
00236     if (FlashStatus != FLASH_COMPLETE)
00237     {
00238         return FlashStatus;
00239     }
00240 }

```

```

00241     break;
00242
00243     default: /* Any other state -> format eeprom */
00244         /* Erase both Page0 and Page1 and set Page0 as valid page */
00245         FlashStatus = EE_Format();
00246         /* If erase/program operation was failed, a Flash error code */
00247         if (FlashStatus != FLASH_COMPLETE)
00248             {
00249                 return FlashStatus;
00250             }
00251         break;
00252     }
00253
00254     return FLASH_COMPLETE;
00255 }
00256
00257 /**
00258  * @brief Returns the last stored variable data, if found, which
00259  *        the passed virtual address
00260  * @param VirtAddress: Variable virtual address
00261  * @param Data: Global variable contains the read variable value
00262  * @retval Success or error status:
00263  *         - 0: if variable was found
00264  *         - 1: if the variable was not found
00265  *         - NO_VALID_PAGE: if no valid page was found.
00266  */
00267 uint16_t EE_ReadVariable(uint16_t VirtAddress, uint16_t* Data)
00268 {
00269     uint16_t ValidPage = PAGE0;
00270     uint16_t AddressValue = 0x5555, ReadStatus = 1;
00271     uint32_t Address = 0x08010000, PageStartAddress = 0x08010000;
00272
00273     /* Get active Page for read operation */
00274     ValidPage = EE_FindValidPage(READ_FROM_VALID_PAGE);
00275
00276     /* Check if there is no valid page */
00277     if (ValidPage == NO_VALID_PAGE)
00278     {
00279         return NO_VALID_PAGE;
00280     }
00281
00282     /* Get the valid Page start Address */
00283     PageStartAddress = (uint32_t)(EEPROM_START_ADDRESS + (uint32_t)
00284
00285     /* Get the valid Page end Address */
00286     Address = (uint32_t)((EEPROM_START_ADDRESS - 2) + (uint32_t)((1
00287
00288     /* Check each active page address starting from end */
00289     while (Address > (PageStartAddress + 2))
00290     {

```

```

00291     /* Get the current location content to be compared with virtu
00292     AddressValue = (*(__IO uint16_t*)Address);
00293
00294     /* Compare the read address with the virtual address */
00295     if (AddressValue == VirtAddress)
00296     {
00297         /* Get content of Address-2 which is variable value */
00298         *Data = (*(__IO uint16_t*)(Address - 2));
00299
00300         /* In case variable value is read, reset ReadStatus flag */
00301         ReadStatus = 0;
00302
00303         break;
00304     }
00305     else
00306     {
00307         /* Next address location */
00308         Address = Address - 4;
00309     }
00310 }
00311
00312 /* Return ReadStatus value: (0: variable exist, 1: variable doe
00313 return ReadStatus;
00314 }
00315
00316 /**
00317  * @brief Writes/upadtes variable data in EEPROM.
00318  * @param VirtAddress: Variable virtual address
00319  * @param Data: 16 bit data to be written
00320  * @retval Success or error status:
00321  *         - FLASH_COMPLETE: on success
00322  *         - PAGE_FULL: if valid page is full
00323  *         - NO_VALID_PAGE: if no valid page was found
00324  *         - Flash error code: on write Flash error
00325  */
00326 uint16_t EE_WriteVariable(uint16_t VirtAddress, uint16_t Data)
00327 {
00328     uint16_t Status = 0;
00329
00330     /* Write the variable virtual address and value in the EEPROM */
00331     Status = EE_VerifyPageFullWriteVariable(VirtAddress, Data);
00332
00333     /* In case the EEPROM active page is full */
00334     if (Status == PAGE_FULL)
00335     {
00336         /* Perform Page transfer */
00337         Status = EE_PageTransfer(VirtAddress, Data);
00338     }
00339

```

```

00340  /* Return last operation status */
00341  return Status;
00342 }
00343
00344 /**
00345  * @brief Erases PAGE0 and PAGE1 and writes VALID_PAGE header to
00346  * @param None
00347  * @retval Status of the last operation (Flash write or erase) done
00348  *         EEPROM formatting
00349  */
00350 static FLASH_Status EE_Format(void)
00351 {
00352     FLASH_Status FlashStatus = FLASH_COMPLETE;
00353
00354     /* Erase Page0 */
00355     FlashStatus = FLASH_ErasePage(PAGE0_BASE_ADDRESS);
00356
00357     /* If erase operation was failed, a Flash error code is returned
00358     if (FlashStatus != FLASH_COMPLETE)
00359     {
00360         return FlashStatus;
00361     }
00362
00363     /* Set Page0 as valid page: Write VALID_PAGE at Page0 base address
00364     FlashStatus = FLASH_ProgramHalfWord(PAGE0_BASE_ADDRESS, VALID_PAGE);
00365
00366     /* If program operation was failed, a Flash error code is returned
00367     if (FlashStatus != FLASH_COMPLETE)
00368     {
00369         return FlashStatus;
00370     }
00371
00372     /* Erase Page1 */
00373     FlashStatus = FLASH_ErasePage(PAGE1_BASE_ADDRESS);
00374
00375     /* Return Page1 erase operation status */
00376     return FlashStatus;
00377 }
00378
00379 /**
00380  * @brief Find valid Page for write or read operation
00381  * @param Operation: operation to achieve on the valid page.
00382  *         This parameter can be one of the following values:
00383  *         @arg READ_FROM_VALID_PAGE: read operation from valid page
00384  *         @arg WRITE_IN_VALID_PAGE: write operation from valid page
00385  * @retval Valid page number (PAGE0 or PAGE1) or NO_VALID_PAGE if
00386  *         of no valid page was found
00387  */
00388 static uint16_t EE_FindValidPage(uint8_t Operation)
00389 {

```



```

00390 uint16_t PageStatus0 = 6, PageStatus1 = 6;
00391
00392 /* Get Page0 actual status */
00393 PageStatus0 = (*(__IO uint16_t*)PAGE0_BASE_ADDRESS);
00394
00395 /* Get Page1 actual status */
00396 PageStatus1 = (*(__IO uint16_t*)PAGE1_BASE_ADDRESS);
00397
00398 /* Write or read operation */
00399 switch (Operation)
00400 {
00401     case WRITE_IN_VALID_PAGE: /* ---- Write operation ---- */
00402         if (PageStatus1 == VALID_PAGE)
00403         {
00404             /* Page0 receiving data */
00405             if (PageStatus0 == RECEIVE_DATA)
00406             {
00407                 return PAGE0; /* Page0 valid */
00408             }
00409             else
00410             {
00411                 return PAGE1; /* Page1 valid */
00412             }
00413         }
00414         else if (PageStatus0 == VALID_PAGE)
00415         {
00416             /* Page1 receiving data */
00417             if (PageStatus1 == RECEIVE_DATA)
00418             {
00419                 return PAGE1; /* Page1 valid */
00420             }
00421             else
00422             {
00423                 return PAGE0; /* Page0 valid */
00424             }
00425         }
00426         else
00427         {
00428             return NO_VALID_PAGE; /* No valid Page */
00429         }
00430
00431     case READ_FROM_VALID_PAGE: /* ---- Read operation ---- */
00432         if (PageStatus0 == VALID_PAGE)
00433         {
00434             return PAGE0; /* Page0 valid */
00435         }
00436         else if (PageStatus1 == VALID_PAGE)
00437         {
00438             return PAGE1; /* Page1 valid */

```

```

00439     }
00440     else
00441     {
00442         return NO_VALID_PAGE ; /* No valid Page */
00443     }
00444
00445     default:
00446         return PAGE0;          /* Page0 valid */
00447 }
00448 }
00449
00450 /**
00451  * @brief Verify if active page is full and Writes variable in I
00452  * @param VirtAddress: 16 bit virtual address of the variable
00453  * @param Data: 16 bit data to be written as variable value
00454  * @retval Success or error status:
00455  *         - FLASH_COMPLETE: on success
00456  *         - PAGE_FULL: if valid page is full
00457  *         - NO_VALID_PAGE: if no valid page was found
00458  *         - Flash error code: on write Flash error
00459  */
00460 static uint16_t EE_VerifyPageFullWriteVariable(uint16_t VirtAddress,
00461 {
00462     FLASH_Status FlashStatus = FLASH_COMPLETE;
00463     uint16_t ValidPage = PAGE0;
00464     uint32_t Address = 0x08010000, PageEndAddress = 0x080107FF;
00465
00466     /* Get valid Page for write operation */
00467     ValidPage = EE_FindValidPage(WRITE_IN_VALID_PAGE);
00468
00469     /* Check if there is no valid page */
00470     if (ValidPage == NO_VALID_PAGE)
00471     {
00472         return NO_VALID_PAGE;
00473     }
00474
00475     /* Get the valid Page start Address */
00476     Address = (uint32_t)(EEPROM_START_ADDRESS + (uint32_t)(ValidPage *
00477
00478     /* Get the valid Page end Address */
00479     PageEndAddress = (uint32_t)((EEPROM_START_ADDRESS - 2) + (uint32_t)
00480
00481     /* Check each active page address starting from beginning */
00482     while (Address < PageEndAddress)
00483     {
00484         /* Verify if Address and Address+2 contents are 0xFFFFFFFF */
00485         if ((__IO uint32_t*)Address) == 0xFFFFFFFF)
00486         {
00487             /* Set variable data */
00488             FlashStatus = FLASH_ProgramHalfWord(Address, Data);

```

```

00489     /* If program operation was failed, a Flash error code is re
00490     if (FlashStatus != FLASH_COMPLETE)
00491     {
00492         return FlashStatus;
00493     }
00494     /* Set variable virtual address */
00495     FlashStatus = FLASH_ProgramHalfWord(Address + 2, VirtAddress);
00496     /* Return program operation status */
00497     return FlashStatus;
00498 }
00499 else
00500 {
00501     /* Next address location */
00502     Address = Address + 4;
00503 }
00504 }
00505
00506 /* Return PAGE_FULL in case the valid page is full */
00507 return PAGE_FULL;
00508 }
00509
00510 /**
00511  * @brief Transfers last updated variables data from the full Page
00512  *        an empty one.
00513  * @param VirtAddress: 16 bit virtual address of the variable
00514  * @param Data: 16 bit data to be written as variable value
00515  * @retval Success or error status:
00516  *         - FLASH_COMPLETE: on success
00517  *         - PAGE_FULL: if valid page is full
00518  *         - NO_VALID_PAGE: if no valid page was found
00519  *         - Flash error code: on write Flash error
00520  */
00521 static uint16_t EE_PageTransfer(uint16_t VirtAddress, uint16_t Data)
00522 {
00523     FLASH_Status FlashStatus = FLASH_COMPLETE;
00524     uint32_t NewPageAddress = 0x080103FF, OldPageAddress = 0x08010000;
00525     uint16_t ValidPage = PAGE0, VarIdx = 0;
00526     uint16_t EepromStatus = 0, ReadStatus = 0;
00527
00528     /* Get active Page for read operation */
00529     ValidPage = EE_FindValidPage(READ_FROM_VALID_PAGE);
00530
00531     if (ValidPage == PAGE1) /* Page1 valid */
00532     {
00533         /* New page address where variable will be moved to */
00534         NewPageAddress = PAGE0_BASE_ADDRESS;
00535
00536         /* Old page address where variable will be taken from */
00537         OldPageAddress = PAGE1_BASE_ADDRESS;

```

```

00538 }
00539 else if (ValidPage == PAGE0) /* Page0 valid */
00540 {
00541     /* New page address where variable will be moved to */
00542     NewPageAddress = PAGE1_BASE_ADDRESS;
00543
00544     /* Old page address where variable will be taken from */
00545     OldPageAddress = PAGE0_BASE_ADDRESS;
00546 }
00547 else
00548 {
00549     return NO_VALID_PAGE; /* No valid Page */
00550 }
00551
00552 /* Set the new Page status to RECEIVE_DATA status */
00553 FlashStatus = FLASH_ProgramHalfWord(NewPageAddress, RECEIVE_DATA);
00554 /* If program operation was failed, a Flash error code is returned */
00555 if (FlashStatus != FLASH_COMPLETE)
00556 {
00557     return FlashStatus;
00558 }
00559
00560 /* Write the variable passed as parameter in the new active page */
00561 EepromStatus = EE_VerifyPageFullWriteVariable(VirtAddress, Data);
00562 /* If program operation was failed, a Flash error code is returned */
00563 if (EepromStatus != FLASH_COMPLETE)
00564 {
00565     return EepromStatus;
00566 }
00567
00568 /* Transfer process: transfer variables from old to the new active page */
00569 for (VarIdx = 0; VarIdx < NumbOfVar; VarIdx++)
00570 {
00571     if (VirtAddVarTab[VarIdx] != VirtAddress) /* Check each variable */
00572     {
00573         /* Read the other last variable updates */
00574         ReadStatus = EE_ReadVariable(VirtAddVarTab[VarIdx], &DataVariable);
00575         /* In case variable corresponding to the virtual address was not found */
00576         if (ReadStatus != 0x1)
00577         {
00578             /* Transfer the variable to the new active page */
00579             EepromStatus = EE_VerifyPageFullWriteVariable(VirtAddVarTab[VarIdx], DataVariable);
00580             /* If program operation was failed, a Flash error code is returned */
00581             if (EepromStatus != FLASH_COMPLETE)
00582             {
00583                 return EepromStatus;
00584             }
00585         }
00586     }
00587 }

```

```

00588
00589 /* Erase the old Page: Set old Page status to ERASED status */
00590 FlashStatus = FLASH_ErasePage(OldPageAddress);
00591 /* If erase operation was failed, a Flash error code is returned
00592 if (FlashStatus != FLASH_COMPLETE)
00593 {
00594     return FlashStatus;
00595 }
00596
00597 /* Set new Page status to VALID_PAGE status */
00598 FlashStatus = FLASH_ProgramHalfWord(NewPageAddress, VALID_PAGE);
00599 /* If program operation was failed, a Flash error code is returned
00600 if (FlashStatus != FLASH_COMPLETE)
00601 {
00602     return FlashStatus;
00603 }
00604
00605 /* Return last operation flash status */
00606 return FlashStatus;
00607 }
00608
00609 /**
00610  * @}
00611  */
00612
00613 /***** (C) COPYRIGHT 2009 STMicroelectronics *****/

```

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

EEPROM_Emulation

Functions

void	BusFault_Handler (void) <i>This function handles Bus Fault exception.</i>
void	DebugMon_Handler (void) <i>This function handles Debug Monitor exception.</i>
static uint16_t	EE_FindValidPage (uint8_t Operation) <i>Find valid Page for write or read operation.</i>
static FLASH_Status	EE_Format (void) <i>Erases PAGE0 and PAGE1 and writes VALID_PAGE header to PAGE0.</i>
uint16_t	EE_Init (void) <i>Restore the pages to a known good state in case of page's status corruption after a power loss.</i>
static uint16_t	EE_PageTransfer (uint16_t VirtAddress, uint16_t Data) <i>Transfers last updated variables data from the full Page to an empty one.</i>
uint16_t	EE_ReadVariable (uint16_t VirtAddress, uint16_t *Data) <i>Returns the last stored variable data, if found, which correspond to the passed virtual address.</i>
static uint16_t	EE_VerifyPageFullWriteVariable (uint16_t VirtAddress, uint16_t Data) <i>Verify if active page is full and Writes variable in EEPROM.</i>
uint16_t	EE_WriteVariable (uint16_t VirtAddress, uint16_t Data) <i>Writes/upadtes variable data in EEPROM.</i>
void	HardFault_Handler (void) <i>This function handles Hard Fault exception.</i>
int	main (void) <i>Main program.</i>
void	MemManage_Handler (void) <i>This function handles Memory Manage exception.</i>
void	NMI_Handler (void) <i>This function handles NMI exception.</i>
void	PendSV_Handler (void) <i>This function handles PendSVC exception.</i>
void	SVC_Handler (void) <i>This function handles SVCcall exception.</i>
void	SysTick_Handler (void) <i>This function handles SysTick Handler.</i>
void	UsageFault_Handler (void)

This function handles Usage Fault exception.

Variables

uint16_t	DataVar = 0
FLASH_Status	FlashStatus
ErrorStatus	HSEStartUpStatus
uint16_t	VarValue = 0
uint16_t	VirtAddVarTab [NumbOfVar] = {0x5555, 0x6666, 0x7777}
uint16_t	VirtAddVarTab [NumbOfVar]

Function Documentation

```
void BusFault_Handler ( void )
```

This function handles Bus Fault exception.

Parameters:

None

Return values:

None

Definition at line [81](#) of file [stm32f10x_it.c](#).

```
void DebugMon_Handler ( void )
```

This function handles Debug Monitor exception.

Parameters:

None

Return values:

None

Definition at line [116](#) of file [stm32f10x_it.c](#).

```
static uint16_t EE_FindValidPage ( uint8_t Operation ) [static]
```

Find valid Page for write or read operation.

Parameters:

Operation,: operation to achieve on the valid page. This parameter can be one of the following values:

- READ_FROM_VALID_PAGE: read operation from valid page
- WRITE_IN_VALID_PAGE: write operation from valid page

Return values:

Valid page number (PAGE0 or PAGE1) or NO_VALID_PAGE in case of no valid page was found

Definition at line [388](#) of file [eeprom.c](#).

References [NO_VALID_PAGE](#), [PAGE0](#), [PAGE0_BASE_ADDRESS](#), [PAGE1](#), [PAGE1_BASE_ADDRESS](#), [READ_FROM_VALID_PAGE](#), [RECEIVE_DATA](#), [VALID_PAGE](#), and [WRITE_IN_VALID_PAGE](#).

Referenced by [EE_PageTransfer\(\)](#), [EE_ReadVariable\(\)](#), and [EE_VerifyPageFullWriteVariable\(\)](#).

```
static FLASH_Status EE_Format ( void ) [static]
```

Erases PAGE0 and PAGE1 and writes VALID_PAGE header to PAGE0.

Parameters:

None

Return values:

Status of the last operation (Flash write or erase) done during EEPROM formatting

Definition at line [350](#) of file [eeprom.c](#).

References [FlashStatus](#), [PAGE0_BASE_ADDRESS](#), [PAGE1_BASE_ADDRESS](#), and [VALID_PAGE](#).

Referenced by [EE_Init\(\)](#).

```
uint16_t EE_Init ( void )
```

Restore the pages to a known good state in case of page's status corruption after a power loss.

Parameters:

None.

Return values:

- Flash error code: on write Flash error
 - FLASH_COMPLETE: on success

Definition at line [52](#) of file [eeprom.c](#).

References [DataVar](#), [EE_Format\(\)](#), [EE_ReadVariable\(\)](#), [EE_VerifyPageFullWriteVariable\(\)](#), [ERASED](#), [FlashStatus](#), [NumbOfVar](#), [PAGE0_BASE_ADDRESS](#), [PAGE1_BASE_ADDRESS](#), [RECEIVE_DATA](#), [VALID_PAGE](#), and [VirtAddVarTab](#).

Referenced by [main\(\)](#).

```
static uint16_t EE_PageTransfer ( uint16_t VirtAddress,  
                                uint16_t Data  
                                ) [static]
```

Transfers last updated variables data from the full Page to an empty one.

Parameters:

- VirtAddress*,: 16 bit virtual address of the variable
- Data*,: 16 bit data to be written as variable value

Return values:

- Success* or error status:
- FLASH_COMPLETE: on success
 - PAGE_FULL: if valid page is full

- NO_VALID_PAGE: if no valid page was found
- Flash error code: on write Flash error

Definition at line [521](#) of file [eeprom.c](#).

References [DataVar](#), [EE_FindValidPage\(\)](#), [EE_ReadVariable\(\)](#), [EE_VerifyPageFullWriteVariable\(\)](#), [FlashStatus](#), [NO_VALID_PAGE](#), [NumOfVar](#), [PAGE0](#), [PAGE0_BASE_ADDRESS](#), [PAGE1](#), [PAGE1_BASE_ADDRESS](#), [READ_FROM_VALID_PAGE](#), [RECEIVE_DATA](#), [VALID_PAGE](#), and [VirtAddVarTab](#).

Referenced by [EE_WriteVariable\(\)](#).

```
uint16_t EE_ReadVariable ( uint16_t  VirtAddress,  
                          uint16_t * Data  
                          )
```

Returns the last stored variable data, if found, which correspond to the passed virtual address.

Parameters:

VirtAddress,: Variable virtual address

Data,: Global variable contains the read variable value

Return values:

Success or error status:

- 0: if variable was found
- 1: if the variable was not found
- NO_VALID_PAGE: if no valid page was found.

Definition at line [267](#) of file [eeprom.c](#).

References [EE_FindValidPage\(\)](#), [EEPROM_START_ADDRESS](#), [NO_VALID_PAGE](#), [PAGE0](#), and [READ_FROM_VALID_PAGE](#).

Referenced by [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

```
static uint16_t EE_VerifyPageFullWriteVariable ( uint16_t VirtAddress,  
                                                uint16_t Data  
                                                ) [static]
```

Verify if active page is full and Writes variable in EEPROM.

Parameters:

VirtAddress,: 16 bit virtual address of the variable

Data,: 16 bit data to be written as variable value

Return values:

Success or error status:

- FLASH_COMPLETE: on success
- PAGE_FULL: if valid page is full
- NO_VALID_PAGE: if no valid page was found
- Flash error code: on write Flash error

Definition at line [460](#) of file [eeprom.c](#).

References [EE_FindValidPage\(\)](#),
[EEPROM_START_ADDRESS](#), [FlashStatus](#),
[NO_VALID_PAGE](#), [PAGE0](#), [PAGE_FULL](#), and
[WRITE_IN_VALID_PAGE](#).

Referenced by [EE_Init\(\)](#), [EE_PageTransfer\(\)](#), and
[EE_WriteVariable\(\)](#).

```
uint16_t EE_WriteVariable ( uint16_t VirtAddress,  
                           uint16_t Data  
                           )
```

Writes/upadtes variable data in EEPROM.

Parameters:

VirtAddress,: Variable virtual address

Data,: 16 bit data to be written

Return values:

Success or error status:

- FLASH_COMPLETE: on success
- PAGE_FULL: if valid page is full
- NO_VALID_PAGE: if no valid page was found
- Flash error code: on write Flash error

Definition at line [326](#) of file [eeprom.c](#).

References [EE_PageTransfer\(\)](#),
[EE_VerifyPageFullWriteVariable\(\)](#), and [PAGE_FULL](#).

Referenced by [main\(\)](#).

```
void HardFault_Handler ( void )
```

This function handles Hard Fault exception.

Parameters:

None

Return values:

None

Definition at line [55](#) of file [stm32f10x_it.c](#).

```
int main ( void )
```

Main program.

Parameters:

None

Return values:

None

Definition at line [47](#) of file [main.c](#).

References [EE_Init\(\)](#), [EE_WriteVariable\(\)](#), [VarValue](#), and [VirtAddVarTab](#).

```
void MemManage_Handler ( void )
```

This function handles Memory Manage exception.

Parameters:

None

Return values:

None

Definition at line [68](#) of file [stm32f10x_it.c](#).

```
void NMI_Handler ( void )
```

This function handles NMI exception.

Parameters:

None

Return values:

None

Definition at line [46](#) of file [stm32f10x_it.c](#).

void PendSV_Handler (void)

This function handles PendSVC exception.

Parameters:

None

Return values:

None

Definition at line [125](#) of file [stm32f10x_it.c](#).

void SVC_Handler (void)

This function handles SVCcall exception.

Parameters:

None

Return values:

None

Definition at line [107](#) of file [stm32f10x_it.c](#).

void SysTick_Handler (void)

This function handles SysTick Handler.

Parameters:

None

Return values:

None

Definition at line [134](#) of file [stm32f10x_it.c](#).

```
void UsageFault_Handler ( void )
```

This function handles Usage Fault exception.

Parameters:

None

Return values:

None

Definition at line [94](#) of file [stm32f10x_it.c](#).

Variable Documentation

uint16_t [DataVar](#) = 0

Definition at line [33](#) of file [eeprom.c](#).

Referenced by [EE_Init\(\)](#), and [EE_PageTransfer\(\)](#).

FLASH_Status [FlashStatus](#)

Definition at line [33](#) of file [main.c](#).

Referenced by [EE_Format\(\)](#), [EE_Init\(\)](#), [EE_PageTransfer\(\)](#), and [EE_VerifyPageFullWriteVariable\(\)](#).

ErrorStatus [HSEStartUpStatus](#)

Definition at line [32](#) of file [main.c](#).

uint16_t [VarValue](#) = 0

Definition at line [34](#) of file [main.c](#).

Referenced by [main\(\)](#).

uint16_t [VirtAddVarTab](#)[NumbOfVar] = {0x5555, 0x6666, 0x7777}

Definition at line [37](#) of file [main.c](#).

Referenced by [EE_Init\(\)](#), [EE_PageTransfer\(\)](#), and [main\(\)](#).

`uint16_t VirtAddVarTab[NumbOfVar]`

Definition at line [37](#) of file [main.c](#).

Referenced by [EE_Init\(\)](#), [EE_PageTransfer\(\)](#), and [main\(\)](#).

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

inc Directory Reference

Files

file	eeprom.h [code]
	Author: <i>MCD Application Team</i>
file	stm32f10x_conf.h [code]
	Author: <i>MCD Application Team</i>
file	stm32f10x_it.h [code]
	Author: <i>MCD Application Team</i>

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

main.c

[Go to the documentation of this file.](#)

```
00001 /**
00002  ****
00003  * @file    EEPROM_Emulation/src/main.c
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   Main program body
00008  ****
00009  * @copy
00010  *
00011  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00012  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
00013  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
00014  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00015  * CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00016  * OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
00017  *
00018  * <h2><center>COPYRIGHT 2009 STMicroelectronics</center></h2>
00019  */
00020
00021 /** @addtogroup EEPROM_Emulation
00022  * @{
00023  */
00024
00025 /* Includes -----
00026 #include "eeprom.h"
00027
00028 /* Private typedef -----
00029 /* Private define -----
00030 /* Private macro -----
00031 /* Private variables -----
00032 ErrorStatus HSEStartUpStatus;
00033 FLASH_Status FlashStatus;
00034 uint16_t VarValue = 0;
00035
00036 /* Virtual address defined by the user: 0xFFFF value is prohibited
00037 uint16_t VirtAddVarTab[NumbOfVar] = {0x5555, 0x6666, 0x7777};
00038
00039 /* Private function prototypes -----
00040 /* Private functions -----
00041
00042 /**
```

```

00043  * @brief Main program.
00044  * @param None
00045  * @retval None
00046  */
00047 int main(void)
00048 {
00049     /* Setup STM32 system (clock, PLL and Flash configuration) */
00050     SystemInit();
00051
00052     /* Unlock the Flash Program Erase controller */
00053     FLASH_Unlock();
00054
00055     /* EEPROM Init */
00056     EE_Init();
00057
00058     /* --- Store successively many values of the three variables in t
00059     /* Store 1000 values of Variable1 in EEPROM */
00060     for (VarValue = 0; VarValue < 1000; VarValue++)
00061     {
00062         EE_WriteVariable(VirtAddVarTab[0], VarValue);
00063     }
00064
00065     /* Store 500 values of Variable2 in EEPROM */
00066     for (VarValue = 0; VarValue < 500; VarValue++)
00067     {
00068         EE_WriteVariable(VirtAddVarTab[1], VarValue);
00069     }
00070
00071     /* Store 800 values of Variable3 in EEPROM */
00072     for (VarValue = 0; VarValue < 800; VarValue++)
00073     {
00074         EE_WriteVariable(VirtAddVarTab[2], VarValue);
00075     }
00076
00077     while (1);
00078 }
00079
00080 #ifdef USE_FULL_ASSERT
00081 /**
00082  * @brief Reports the name of the source file and the source li
00083  *       where the assert_param error has occurred.
00084  * @param file: pointer to the source file name
00085  * @param line: assert_param error line source number
00086  * @retval None
00087  */
00088 void assert_failed(uint8_t* file, uint32_t line)
00089 {
00090     /* User can add his own implementation to report the file name ;
00091     ex: printf("Wrong parameters value: file %s on line %d\r\n",
00092

```

```
00093  /* Infinite loop */
00094  while (1)
00095  {}
00096  }
00097  #endif
00098
00099  /**
00100   * @}
00101   */
00102
00103  /** ***** (C) COPYRIGHT 2009 STMicroelectronics *****EI
```

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)[Modules](#)[Files](#)[Directories](#)[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

stm32f10x_conf.h

[Go to the documentation of this file.](#)

```
00001 /**
00002  *****
00003  * @file    EEPROM_Emulation/inc/stm32f10x_conf.h
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   Library configuration file.
00008  *****
00009  * @copy
00010  *
00011  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00012  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR
00013  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR
00014  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00015  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00016  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR
00017  *
00018  * <h2><center>&copy; COPYRIGHT 2009 STMicroelectronics</center>
00019  */
00020
00021 /* Define to prevent recursive inclusion -----
00022 #ifndef __STM32F10x_CONF_H
00023 #define __STM32F10x_CONF_H
00024
00025 /* Includes -----
00026 /* Uncomment the line below to enable peripheral header file inclusion
00027 /* #include "stm32f10x_adc.h" */
00028 /* #include "stm32f10x_bkp.h" */
00029 /* #include "stm32f10x_can.h" */
00030 /* #include "stm32f10x_crc.h" */
00031 /* #include "stm32f10x_dac.h" */
00032 /* #include "stm32f10x_dbgmcu.h" */
00033 /* #include "stm32f10x_dma.h" */
00034 /* #include "stm32f10x_exti.h" */
00035 #include "stm32f10x_flash.h"
00036 /* #include "stm32f10x_fsmc.h" */
00037 /* #include "stm32f10x_gpio.h" */
00038 /* #include "stm32f10x_i2c.h" */
00039 /* #include "stm32f10x_iwdg.h" */
00040 /* #include "stm32f10x_pwr.h" */
00041 /* #include "stm32f10x_rcc.h" */
00042 /* #include "stm32f10x_rtc.h" */
```

```

00043 /* #include "stm32f10x_sdio.h" */
00044 /* #include "stm32f10x_spi.h" */
00045 /* #include "stm32f10x_tim.h" */
00046 /* #include "stm32f10x_usart.h" */
00047 /* #include "stm32f10x_wwdg.h" */
00048 /* #include "misc.h" */ /* High level functions for NVIC and Sys
00049
00050 /* Exported types -----
00051 /* Exported constants -----
00052 /* Uncomment the line below to expanse the "assert_param" macro in
00053 Standard Peripheral Library drivers code */
00054 /* #define USE_FULL_ASSERT 1 */
00055
00056 /* Exported macro -----
00057 #ifdef USE_FULL_ASSERT
00058
00059 /**
00060 * @brief The assert_param macro is used for function's parameter
00061 * @param expr: If expr is false, it calls assert_failed function
00062 * which reports the name of the source file and the source
00063 * line number of the call that failed.
00064 * If expr is true, it returns no value.
00065 * @retval : None
00066 */
00067 #define assert_param(expr) ((expr) ? (void)0 : assert_failed((u
00068 /* Exported functions -----
00069 void assert_failed(uint8_t* file, uint32_t line);
00070 #else
00071 #define assert_param(expr) ((void)0)
00072 #endif /* USE_FULL_ASSERT */
00073
00074 #endif /* __STM32F10x_CONF_H */
00075
00076 /***** (C) COPYRIGHT 2009 STMicroelectronics *****/

```

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [ir](#)

stm32f10x_it.h

[Go to the documentation of this file.](#)

```
00001 /**
00002  ****
00003  * @file    EEPROM_Emulation/inc/stm32f10x_it.h
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   This file contains the headers of the interrupt hand.
00008  ****
00009  * @copy
00010  *
00011  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00012  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
00013  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
00014  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00015  * CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00016  * OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
00017  *
00018  * <h2><center>COPYRIGHT 2009 STMicroelectronics</center></h2>
00019  */
00020
00021 /* Define to prevent recursive inclusion -----
00022 #ifndef __STM32F10x_IT_H
00023 #define __STM32F10x_IT_H
00024
00025 /* Includes -----
00026 #include "stm32f10x.h"
00027
00028 /* Exported types -----
00029 /* Exported constants -----
00030 /* Exported macro -----
00031 /* Exported functions -----
00032
00033 void NMI_Handler(void);
00034 void HardFault_Handler(void);
00035 void MemManage_Handler(void);
00036 void BusFault_Handler(void);
00037 void UsageFault_Handler(void);
00038 void SVC_Handler(void);
00039 void DebugMon_Handler(void);
00040 void PendSV_Handler(void);
00041 void SysTick_Handler(void);
00042
```

```
00043 #endif /* __STM32F10x_IT_H */
00044
00045 /***** (C) COPYRIGHT 2009 STMicroelectronics *****/E
```

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

[Main Page](#)

[Modules](#)

[Files](#)

[Directories](#)

[Related Pages](#)

[STM32F10x_AN2594_FW_V3.1.0](#) » [Project](#) » [EEPROM Emulation](#) » [s](#)

stm32f10x_it.c

[Go to the documentation of this file.](#)

```
00001 /**
00002  *****
00003  * @file    EEPROM_Emulation/src/stm32f10x_it.c
00004  * @author  MCD Application Team
00005  * @version V3.1.0
00006  * @date    07/27/2009
00007  * @brief   Main Interrupt Service Routines.
00008  *          This file provides template for all exceptions handling
00009  *          interrupt service routine.
00010  *****
00011  * @copy
00012  *
00013  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
00014  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR
00015  * TIME. AS A RESULT, STMicroelectronics SHALL NOT BE HELD LIABLE FOR
00016  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY
00017  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS
00018  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR
00019  *
00020  * <h2><center>COPYRIGHT 2009 STMicroelectronics</center>
00021  */
00022
00023 /* Includes -----
00024 #include "stm32f10x_it.h"
00025
00026 /** @addtogroup EEPROM_Emulation
00027  * @{
00028  */
00029
00030 /* Private typedef -----
00031 /* Private define -----
00032 /* Private macro -----
00033 /* Private variables -----
00034 /* Private function prototypes -----
00035 /* Private functions -----
00036
00037 /*****
00038  *          Cortex-M3 Processor Exceptions Handlers
00039 /*****
00040
00041 /**
00042  * @brief This function handles NMI exception.
```

```

00043  * @param None
00044  * @retval None
00045  */
00046 void NMI_Handler(void)
00047 {
00048 }
00049
00050 /**
00051  * @brief This function handles Hard Fault exception.
00052  * @param None
00053  * @retval None
00054  */
00055 void HardFault_Handler(void)
00056 {
00057  /* Go to infinite loop when Hard Fault exception occurs */
00058  while (1)
00059  {
00060  }
00061 }
00062
00063 /**
00064  * @brief This function handles Memory Manage exception.
00065  * @param None
00066  * @retval None
00067  */
00068 void MemManage_Handler(void)
00069 {
00070  /* Go to infinite loop when Memory Manage exception occurs */
00071  while (1)
00072  {
00073  }
00074 }
00075
00076 /**
00077  * @brief This function handles Bus Fault exception.
00078  * @param None
00079  * @retval None
00080  */
00081 void BusFault_Handler(void)
00082 {
00083  /* Go to infinite loop when Bus Fault exception occurs */
00084  while (1)
00085  {
00086  }
00087 }
00088
00089 /**
00090  * @brief This function handles Usage Fault exception.
00091  * @param None
00092  * @retval None

```

```

00093  */
00094 void UsageFault_Handler(void)
00095 {
00096  /* Go to infinite loop when Usage Fault exception occurs */
00097  while (1)
00098  {
00099  }
00100 }
00101
00102 /**
00103  * @brief This function handles SVCcall exception.
00104  * @param None
00105  * @retval None
00106  */
00107 void SVC_Handler(void)
00108 {
00109 }
00110
00111 /**
00112  * @brief This function handles Debug Monitor exception.
00113  * @param None
00114  * @retval None
00115  */
00116 void DebugMon_Handler(void)
00117 {
00118 }
00119
00120 /**
00121  * @brief This function handles PendSVC exception.
00122  * @param None
00123  * @retval None
00124  */
00125 void PendSV_Handler(void)
00126 {
00127 }
00128
00129 /**
00130  * @brief This function handles SysTick Handler.
00131  * @param None
00132  * @retval None
00133  */
00134 void SysTick_Handler(void)
00135 {
00136 }
00137
00138 /*****
00139  /*
00140  /*      STM32F10x Peripherals Interrupt Handlers
00141  /*  Add here the Interrupt Handler for the used peripheral(s) (PPI
00142  /*  available peripheral interrupt handler's name please refer to

```

```
00142 /* file (startup_stm32f10x_xx.s).
00143 /*****
00144
00145 /**
00146  * @brief This function handles PPP interrupt request.
00147  * @param None
00148  * @retval None
00149  */
00150 void PPP_IRQHandler(void)
00151 {
00152 }*/
00153
00154
00155 /**
00156  * @}
00157  */
00158
00159 /***** (C) COPYRIGHT 2009 STMicroelectronics *****/EI
```

For complete documentation on STM32(CORTEX M3)
Microcontrollers platform visit www.st.com/STM32

- BusFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- DebugMon_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- EE_FindValidPage() : [eeprom.c](#)
- EE_Format() : [eeprom.c](#)
- EE_Init() : [eeprom.c](#) , [eeprom.h](#)
- EE_PageTransfer() : [eeprom.c](#)
- EE_ReadVariable() : [eeprom.c](#) , [eeprom.h](#)
- EE_VerifyPageFullWriteVariable() : [eeprom.c](#)
- EE_WriteVariable() : [eeprom.h](#) , [eeprom.c](#)
- HardFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- main() : [main.c](#)
- MemManage_Handler() : [stm32f10x_it.c](#) , [stm32f10x_it.h](#)
- NMI_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- PendSV_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- SVC_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- SysTick_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)
- UsageFault_Handler() : [stm32f10x_it.h](#) , [stm32f10x_it.c](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

[Main Page](#) [Modules](#) [Files](#) [Directories](#) [Related Pages](#)

[File List](#) [Globals](#)

[All](#) [Functions](#) [Variables](#) [Defines](#)

- DataVar : [eeprom.c](#)
- FlashStatus : [main.c](#)
- HSEStartUpStatus : [main.c](#)
- VarValue : [main.c](#)
- VirtAddVarTab : [main.c](#) , [eeprom.c](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32

- assert_param : [stm32f10x_conf.h](#)
- EEPROM_START_ADDRESS : [eeprom.h](#)
- ERASED : [eeprom.h](#)
- NO_VALID_PAGE : [eeprom.h](#)
- NumOfVar : [eeprom.h](#)
- PAGE0 : [eeprom.h](#)
- PAGE0_BASE_ADDRESS : [eeprom.h](#)
- PAGE0_END_ADDRESS : [eeprom.h](#)
- PAGE1 : [eeprom.h](#)
- PAGE1_BASE_ADDRESS : [eeprom.h](#)
- PAGE1_END_ADDRESS : [eeprom.h](#)
- PAGE_FULL : [eeprom.h](#)
- READ_FROM_VALID_PAGE : [eeprom.h](#)
- RECEIVE_DATA : [eeprom.h](#)
- VALID_PAGE : [eeprom.h](#)
- WRITE_IN_VALID_PAGE : [eeprom.h](#)

For complete documentation on STM32(CORTEX M3) Microcontrollers platform visit www.st.com/STM32