fault_handler

Generated by Doxygen 1.8.6

Thu Apr 3 2014 10:21:20

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 SCB_Type Struct Reference

Data Fields

- __I uint32_t CPUID
- __IO uint32_t ICSR
- __IO uint32_t VTOR
- __IO uint32_t AIRCR
- __IO uint32_t SCR
- __IO uint32_t CCR
- __IO uint8_t SHP [12]
- __IO uint32_t SHCSR
- __IO uint32_t CFSR
- __IO uint32_t HFSR
- __IO uint32_t DFSR
- __IO uint32_t MMFAR
- __IO uint32_t BFAR
- __IO uint32_t AFSR
- __I uint32_t PFR [2]
- __I uint32_t DFR
- __I uint32_t ADR
- __I uint32_t MMFR [4]
- __I uint32_t ISAR [5]

3.1.1 Detailed Description

memory mapped structure for System Control Block (SCB)

Definition at line 22 of file fault_handler.c.

3.1.2 Field Documentation

3.1.2.1 __I uint32_t ADR

Offset: 0x4C Auxiliary Feature Register

Definition at line 39 of file fault_handler.c.

3.1.2.2 __IO uint32_t AFSR

Offset: 0x3C Auxiliary Fault Status Register

Definition at line 36 of file fault_handler.c.

3.1.2.3 __IO uint32_t AIRCR

Offset: 0x0C Application Interrupt / Reset Control Register

Definition at line 26 of file fault_handler.c.

3.1.2.4 __IO uint32_t BFAR

Offset: 0x38 Bus Fault Address Register

Definition at line 35 of file fault_handler.c.

3.1.2.5 __IO uint32_t CCR

Offset: 0x14 Configuration Control Register

Definition at line 28 of file fault_handler.c.

3.1.2.6 __IO uint32_t CFSR

Offset: 0x28 Configurable Fault Status Register

Definition at line 31 of file fault_handler.c.

3.1.2.7 __I uint32_t CPUID

Offset: 0x00 CPU ID Base Register

Definition at line 23 of file fault_handler.c.

3.1.2.8 __I uint32_t DFR

Offset: 0x48 Debug Feature Register

Definition at line 38 of file fault_handler.c.

3.1.2.9 __IO uint32_t DFSR

Offset: 0x30 Debug Fault Status Register

Definition at line 33 of file fault_handler.c.

3.1.2.10 __IO uint32_t HFSR

Offset: 0x2C Hard Fault Status Register

Definition at line 32 of file fault_handler.c.

3.1.2.11 __IO uint32_t ICSR

Offset: 0x04 Interrupt Control State Register

Definition at line 24 of file fault_handler.c.

3.1.2.12 __I uint32_t ISAR[5]

Offset: 0x60 ISA Feature Register

Definition at line 41 of file fault_handler.c.

3.1.2.13 __IO uint32_t MMFAR

Offset: 0x34 Mem Manage Address Register

Definition at line 34 of file fault_handler.c.

3.1.2.14 __I uint32_t MMFR[4]

Offset: 0x50 Memory Model Feature Register

Definition at line 40 of file fault_handler.c.

3.1.2.15 __I uint32_t PFR[2]

Offset: 0x40 Processor Feature Register

Definition at line 37 of file fault_handler.c.

3.1.2.16 __IO uint32_t SCR

Offset: 0x10 System Control Register

Definition at line 27 of file fault_handler.c.

3.1.2.17 __IO uint32_t SHCSR

Offset: 0x24 System Handler Control and State Register

Definition at line 30 of file fault_handler.c.

3.1.2.18 __IO uint8_t SHP[12]

Offset: 0x18 System Handlers Priority Registers (4-7, 8-11, 12-15)

Definition at line 29 of file fault_handler.c.

3.1.2.19 __IO uint32_t VTOR

Offset: 0x08 Vector Table Offset Register

Definition at line 25 of file fault_handler.c.

The documentation for this struct was generated from the following file:

- C:/Stefano/GitHub/MyGitHubRepositories/CM3_Fault_Handler/src/fault_handler.c

Chapter 4

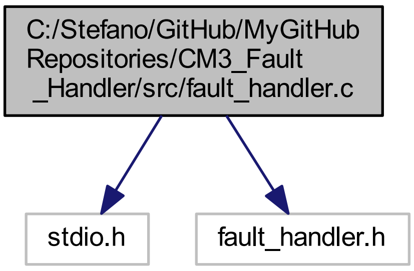
File Documentation

4.1 C:/Stefano/GitHub/MyGitHubRepositories/CM3_Fault_Handler/src/fault_handler.c File Reference

Hard Fault Management.

```
#include <stdio.h>
#include "fault_handler.h"
```

Include dependency graph for fault_handler.c:


C:/Stefano/GitHub/MyGitHub Repositories/CM3_Fault _Handler/src/fault_handler.c
stdio.h
fault_handler.h

Data Structures

- struct SCB_Type

Macros

- #define __IO volatile
- #define __I volatile

- #define SCS_BASE (0xE000E000)
- #define SCB_BASE (SCS_BASE + 0x0D00)
- #define SCB ((SCB_Type ∗) SCB_BASE)
- #define SCB_CFSR_IACCVIOL ((uint32_t)0x00000001)

- #define SCB_CFSR_DACCVIOL ((uint32_t)0x00000002)
- #define SCB_CFSR_MUNSTKERR ((uint32_t)0x00000008)
- #define SCB_CFSR_MSTKERR ((uint32_t)0x00000010)
- #define SCB_CFSR_MMARVALID ((uint32_t)0x00000080)
- #define SCB_CFSR_IBUSERR ((uint32_t)0x00000100)
- #define SCB_CFSR_PRECISERR ((uint32_t)0x00000200)
- #define SCB_CFSR_IMPRECISERR ((uint32_t)0x00000400)
- #define SCB_CFSR_UNSTKERR ((uint32_t)0x00000800)
- #define SCB_CFSR_STKERR ((uint32_t)0x00001000)
- #define SCB_CFSR_BFARVALID ((uint32_t)0x00008000)
- #define SCB_CFSR_UNDEFINSTR ((uint32_t)0x00010000)
- #define SCB_CFSR_INVSTATE ((uint32_t)0x00020000)
- #define SCB_CFSR_INVPC ((uint32_t)0x00040000)
- #define SCB_CFSR_NOCP ((uint32_t)0x00080000)
- #define SCB_CFSR_UNALIGNED ((uint32_t)0x01000000)
- #define SCB_CFSR_DIVBYZERO ((uint32_t)0x02000000)
- enum {
  r0, r1, r2, r3,
  r12, lr, pc, psr }
- volatile int dontoptimize = 1

  This function tries to divide by zero (and enables div_by_zero trap)
- static void printErrorMsg (const char ∗errMsg)

  Print Messages using semihosting.
- static void printUsageErrorMsg (uint32_t CFSRValue)

  Print Usage Errors.
- static void printBusFaultErrorMsg (uint32_t CFSRValue)

  Print Bus Fault Errors.
- static void printMemoryManagementErrorMsg (uint32_t CFSRValue)

  Print Memory Management Errors.
- static void DumpStack (uint32_t stack[])

  Dump Stack, printing all registers ARM core pushes on stack on hard fault exception.
- static void HardFaultHandlerUser (uint32_t stack[])

  Fill-in this function with your code to handle the exception.
- void Hard_Fault_Handler (uint32_t stack[])

  The Hard Fault Handler.
- uint8_t bus_fault_code (void)

  This function does a buffer overflow.
- uint8_t divide_by_zero (void)
- uint8_t call_to_null_function (void)

  This function creates a null pointer and then calls it.
- uint8_t dangling_pointer (void)

  This function accesses an invalid RAM address.
- uint32_t dangling_pointer2 (void)

  This function accesses an RAM address usually not available.

4.1.1 Detailed Description

Hard Fault Management. This module gives information about a hard fault exception. There are also some functions to generate exceptions, so you can call them and have an idea of what help this module can give you!

Definition in file fault_handler.c.

4.1.2 Macro Definition Documentation

4.1.2.1 #define __I volatile

defines 'read only' permissions

Definition at line 17 of file fault_handler.c.

4.1.2.2 #define __IO volatile

defines 'read / write' permissions

Definition at line 16 of file fault_handler.c.

4.1.2.3 #define SCB ((SCB_Type *) SCB_BASE)

SCB configuration struct

Definition at line 46 of file fault_handler.c.

4.1.2.4 #define SCB_BASE (SCS_BASE + 0x0D00)

System Control Block Base Address

Definition at line 45 of file fault_handler.c.

4.1.2.5 #define SCB_CFSR_BFARVALID ((uint32_t)0x00008000)

Bus Fault Address Register address valid flag UFSR

Definition at line 61 of file fault_handler.c.

4.1.2.6 #define SCB_CFSR_DACCVIOL ((uint32_t)0x00000002)

Data access violation

Definition at line 51 of file fault_handler.c.

4.1.2.7 #define SCB_CFSR_DIVBYZERO ((uint32_t)0x02000000)

Fault occurs when SDIV or DIV instruction is used with a divisor of 0

Definition at line 68 of file fault_handler.c.

4.1.2.8 #define SCB_CFSR_IACCVIOL ((uint32_t)0x00000001)

< MFSR Instruction access violation

Definition at line 50 of file fault_handler.c.

4.1.2.9 #define SCB_CFSR_IBUSERR ((uint32_t)0x00000100)

Instruction bus error flag

Definition at line 56 of file fault_handler.c.

4.1.2.10 #define SCB_CFSR_IMPRECISERR ((uint32_t)0x00000400)

Imprecise data bus error

Definition at line 58 of file fault_handler.c.

4.1.2.11 #define SCB_CFSR_INVPC ((uint32_t)0x00040000)

Attempt to load EXC_RETURN into pc illegally

Definition at line 65 of file fault_handler.c.

4.1.2.12 #define SCB_CFSR_INVSTATE ((uint32_t)0x00020000)

Invalid combination of EPSR and instruction

Definition at line 64 of file fault_handler.c.

4.1.2.13 #define SCB_CFSR_MMARVALID ((uint32_t)0x00000080)

Memory Manage Address Register address valid flag BFSR

Definition at line 54 of file fault_handler.c.

4.1.2.14 #define SCB_CFSR_MSTKERR ((uint32_t)0x00000010)

Stacking error

Definition at line 53 of file fault_handler.c.

4.1.2.15 #define SCB_CFSR_MUNSTKERR ((uint32_t)0x00000008)

Unstacking error

Definition at line 52 of file fault_handler.c.

4.1.2.16 #define SCB_CFSR_NOCP ((uint32_t)0x00080000)

Attempt to use a coprocessor instruction

Definition at line 66 of file fault_handler.c.

4.1.2.17 #define SCB_CFSR_PRECISERR ((uint32_t)0x00000200)

Precise data bus error

Definition at line 57 of file fault_handler.c.

4.1.2.18 #define SCB_CFSR_STKERR ((uint32_t)0x00001000)

Stacking error

Definition at line 60 of file fault_handler.c.

4.1.2.19 #define SCB_CFSR_UNALIGNED ((uint32_t)0x01000000)

Fault occurs when there is an attempt to make an unaligned memory access

Definition at line 67 of file fault_handler.c.

4.1.2.20 #define SCB_CFSR_UNDEFINSTR ((uint32_t)0x00010000)

The processor attempt to excecute an undefined instruction

Definition at line 63 of file fault_handler.c.

4.1.2.21 #define SCB_CFSR_UNSTKERR ((uint32_t)0x00000800)

Unstacking error

Definition at line 59 of file fault_handler.c.

4.1.2.22 #define SCS_BASE (0xE000E000)

System Control Space Base Address

Definition at line 44 of file fault_handler.c.

4.1.3 Enumeration Type Documentation

4.1.3.1 anonymous enum

Enumerator

- ***r0***
- ***r1***
- ***r2***
- ***r3***
- ***r12***
- ***lr***
- ***pc***
- ***psr***

Definition at line 263 of file fault_handler.c.

4.1.4 Function Documentation

4.1.4.1 uint8_t bus_fault_code (void)

This function does a buffer overflow.

Definition at line 310 of file fault_handler.c.

4.1.4.2 uint8_t call_to_null_function (void)

This function creates a null pointer and then calls it.

Definition at line 345 of file fault_handler.c.

4.1.4.3 uint8_t dangling_pointer (void)

This function accesses an invalid RAM address.

Definition at line 356 of file fault_handler.c.

4.1.4.4 uint32_t dangling_pointer2 (void)

This function accesses an RAM address usually not available.

Definition at line 366 of file fault_handler.c.

4.1.4.5 uint8_t divide_by_zero (void)

Definition at line 328 of file fault_handler.c.

4.1.4.6 static void DumpStack (uint32_t *stack[]*) `[static]`

Dump Stack, printing all registers ARM core pushes on stack on hard fault exception.

Definition at line 268 of file fault_handler.c.

4.1.4.7 void Hard_Fault_Handler (uint32_t *stack[]*)

The Hard Fault Handler.

Definition at line 86 of file fault_handler.c.

4.1.4.8 static void HardFaultHandlerUser (uint32_t *stack[]*) `[static]`

Fill-in this function with your code to handle the exception.

It is called after debug string prints

Definition at line 126 of file fault_handler.c.

4.1.4.9 static void printBusFaultErrorMsg (uint32_t *CFSRValue*) `[static]`

Print Bus Fault Errors.

Definition at line 174 of file fault_handler.c.

4.1.4.10 static void printErrorMsg (const char * *errMsg*) `[static]`

Print Messages using semihosting.

Definition at line 134 of file fault_handler.c.

4.1.4.11 static void printMemoryManagementErrorMsg (uint32_t *CFSRValue*) `[static]`

Print Memory Management Errors.

Definition at line 211 of file fault_handler.c.

4.1.4.12 static void printUsageErrorMsg (uint32_t *CFSRValue*) [static]

Print Usage Errors.

Definition at line 142 of file fault_handler.c.

4.1.5 Variable Documentation

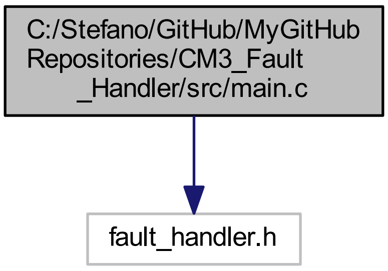
4.1.5.1 volatile int dontoptimize = 1

This function tries to divide by zero (and enables div_by_zero trap)

Definition at line 324 of file fault_handler.c.

4.2 C:/Stefano/GitHub/MyGitHubRepositories/CM3_Fault_Handler/src/main.c File Reference

`#include "fault_handler.h"`
Include dependency graph for main.c:


C:/Stefano/GitHub/MyGitHub Repositories/CM3_Fault _Handler/src/main.c
fault_handler.h

Functions

- int main (void)

4.2.1 Function Documentation

4.2.1.1 int main (void)

Definition at line 3 of file main.c.