AES132 Library

2.1.0

Generated by Doxygen 1.8.3.1

Fri Sep 27 2013 09:40:19

Contents

1 AES132 Library Source Code **1**

1.1 Introduction . . . 1

1.2 Getting Started . . . 1

1.3 AES132 Communication Interfaces . . . 1

2 Building The Projects **3**

2.1 Work Space and Project Structure . . . 3

2.1.1 Interface Independent Modules . . . 3

2.1.2 Interface Dependent Modules . . . 3

2.1.3 Example Project . . . 3

2.2 Doxygen Generated Documentation . . . 4

3 File Index **5**

3.1 File List . . . 5

4 File Documentation **7**

4.1 aes132.c File Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Enumeration Type Documentation . . . 8

4.1.2.1 aes132_read_write_flag . . . 8

4.1.2.2 aes132_bit_set_flag . . . 9

4.1.3 Function Documentation . . . 9

4.1.3.1 aes132m_execute . . . 9

4.1.3.2 aes132c_calculate_crc . . . 9

4.1.3.3 aes132c_reset_io_address . . . 10

4.1.3.4 aes132c_resync . . . 10

4.1.3.5 aes132c_read_device_status_register . . . 10

4.1.3.6 aes132c_wait_for_status_register_bit . . . 10

4.1.3.7 aes132c_wait_for_device_ready . . . 11

4.1.3.8 aes132c_send_sleep_command . . . 11
4.1.3.9 aes132c_wakeup . . . 11
4.1.3.10 aes132c_sleep . . . 11
4.1.3.11 aes132c_standby . . . 12
4.1.3.12 aes132c_access_memory . . . 12
4.1.3.13 aes132c_write_memory . . . 12
4.1.3.14 aes132c_read_memory . . . 13
4.1.3.15 aes132c_send_command . . . 13
4.1.3.16 aes132c_receive_response . . . 13
4.1.3.17 aes132c_send_and_receive . . . 14
4.2 aes132.h File Reference . . . 14
4.2.1 Detailed Description . . . 19
4.2.2 Macro Definition Documentation . . . 19
4.2.2.1 AES132_RESPONSE_READY_TIMEOUT . . . 19
4.2.3 Function Documentation . . . 19
4.2.3.1 aes132m_execute . . . 19
4.2.3.2 aes132c_read_memory . . . 20
4.2.3.3 aes132c_write_memory . . . 20
4.2.3.4 aes132c_access_memory . . . 20
4.2.3.5 aes132c_read_device_status_register . . . 21
4.2.3.6 aes132c_send_command . . . 21
4.2.3.7 aes132c_receive_response . . . 22
4.2.3.8 aes132c_send_and_receive . . . 22
4.2.3.9 aes132c_wakeup . . . 22
4.2.3.10 aes132c_sleep . . . 23
4.2.3.11 aes132c_standby . . . 23
4.2.3.12 aes132c_resync . . . 23
4.2.3.13 aes132c_wait_for_status_register_bit . . . 23
4.2.3.14 aes132c_wait_for_device_ready . . . 24
4.2.3.15 aes132c_send_sleep_command . . . 24
4.2.3.16 aes132c_reset_io_address . . . 24
4.2.3.17 aes132c_calculate_crc . . . 24
4.3 aes132_example_main.c File Reference . . . 25
4.3.1 Detailed Description . . . 25
4.3.2 Function Documentation . . . 25
4.3.2.1 main . . . 25
4.4 aes132_i2c.c File Reference . . . 25

4.4.1 Detailed Description . . . 27
4.4.2 Macro Definition Documentation . . . 27
4.4.2.1 AES132_I2C_DEFAULT_ADDRESS . . . 27
4.4.3 Enumeration Type Documentation . . . 27
4.4.3.1 aes132_i2c_read_write_flag . . . 27
4.4.4 Function Documentation . . . 27
4.4.4.1 i2c_send_start . . . 27
4.4.4.2 i2c_send_stop . . . 27
4.4.4.3 i2c_send_bytes . . . 28
4.4.4.4 i2c_receive_byte . . . 28
4.4.4.5 i2c_receive_bytes . . . 28
4.4.4.6 aes132p_send_slave_address . . . 29
4.4.4.7 aes132p_select_device . . . 29
4.4.4.8 aes132p_write_memory_physical . . . 29
4.4.4.9 aes132p_read_memory_physical . . . 30
4.4.4.10 aes132p_resync_physical . . . 30
4.5 aes132_i2c.h File Reference . . . 30
4.5.1 Detailed Description . . . 32
4.5.2 Macro Definition Documentation . . . 32
4.5.2.1 POLL_TIME_MEASURED . . . 32
4.5.2.2 AES132_STATUS_REG_POLL_TIME_ACK . . . 32
4.5.2.3 AES132_RETRY_COUNT_RESPONSE_READY . . . 33
4.5.2.4 I2C_START_TIMEOUT . . . 33
4.5.2.5 I2C_BYTE_TIMEOUT . . . 33
4.5.2.6 I2C_STOP_TIMEOUT . . . 33
4.5.3 Function Documentation . . . 33
4.5.3.1 i2c_send_start . . . 33
4.5.3.2 i2c_send_stop . . . 34
4.5.3.3 i2c_send_bytes . . . 34
4.5.3.4 i2c_receive_byte . . . 34
4.5.3.5 i2c_receive_bytes . . . 34
4.5.3.6 aes132p_select_device . . . 35
4.5.3.7 aes132p_read_memory_physical . . . 35
4.5.3.8 aes132p_write_memory_physical . . . 36
4.5.3.9 aes132p_resync_physical . . . 36
4.6 aes132_spi.c File Reference . . . 36
4.6.1 Detailed Description . . . 38

4.6.2 Function Documentation . . . 38
4.6.2.1 spi_select_device . . . 38
4.6.2.2 spi_select_slave . . . 38
4.6.2.3 spi_deselect_slave . . . 39
4.6.2.4 spi_send_bytes . . . 39
4.6.2.5 spi_receive_bytes . . . 39
4.6.2.6 aes132p_enable_interface . . . 39
4.6.2.7 aes132p_disable_interface . . . 40
4.6.2.8 aes132p_select_device . . . 40
4.6.2.9 aes132p_write_memory_physical . . . 40
4.6.2.10 aes132p_read_memory_physical . . . 40
4.6.2.11 aes132p_resync_physical . . . 41
4.7 aes132_spi.h File Reference . . . 41
4.7.1 Detailed Description . . . 43
4.7.2 Macro Definition Documentation . . . 43
4.7.2.1 AES132_STATUS_REG_POLL_TIME . . . 43
4.7.3 Function Documentation . . . 43
4.7.3.1 spi_select_slave . . . 43
4.7.3.2 spi_deselect_slave . . . 43
4.7.3.3 spi_select_device . . . 43
4.7.3.4 spi_send_bytes . . . 44
4.7.3.5 spi_receive_bytes . . . 44
4.7.3.6 aes132p_select_device . . . 44
4.7.3.7 aes132p_read_memory_physical . . . 45
4.7.3.8 aes132p_write_memory_physical . . . 45
4.7.3.9 aes132p_resync_physical . . . 46

Index **46**

Chapter 1

AES132 Library Source Code

1.1 Introduction

This library enables a user to more quickly implement an application that uses an Atmel CryptoMemory ATAES132 device.

The library is distributed as source code. It is licensed according to terms of the included license agreement, which the user agreed to during the installation process.

The code comes in two layers, Communication and Interface layer.

The Communication layer handles communication sequences (send command, receive response, sleep, etc.). It retries such sequences in case of certain communication failures.

The Interface layer puts the command packets on the chosen interface bus, and reads responses from it.

1.2 Getting Started

The user should be able to use most of the library as is, simply adding the library modules into her C project. Functions in the Interface layer will have to be modified or re-written if the processor is not an eight-bit AVR. Starting values for timeout loop counters have to be adjusted.

To start development, add the library files to your project, modify / implement the functions in the Interface layer module, and supply values for the timeout loop counters that match the execution time of your CPU (and the interface clock, I^2C or SPI).

1.3 AES132 Communication Interfaces

The device comes in two types of communication interfaces, SPI or I^2C. With the distribution of this library, example configurations are provided for both interfaces.

Chapter 2

Building The Projects

2.1 Work Space and Project Structure

The source files for the AES132 library are contained in a single folder "aes132_library".

2.1.1 Interface Independent Modules

aes132_example_main.c

aes132.h

aes132.c

Function names in aes132.c start with aes132m_ or aes132c_.

2.1.2 Interface Dependent Modules

Hardware dependent modules are provided that support 8-bit AVR micro-controllers. If you are not using an AVR CPU, either implement the functions listed in aes132_i2c.h or aes132_spi.h, or choose the appropriate module for the communication with the device from one of the communication related modules:

- aes132_spi.c: implementation as Serial Peripheral Interface (SPI)

  Function names in this module start with aes132p_ or spi_.

- aes132_i2c.c: implementation as Inter-Integrated Circuit (I^2C)

  Function names in this module start with aes132p_ or i2c_.

2.1.3 Example Project

A solution file is supplied for the AVR Studio 5 IDE. (Solution files have an "sln" extension.) AVR Studio 5 solution files are located in the root folder. Choose the project that fits the communication interface you like to use.

You can easily create a project under the IDE you are using by following the few steps listed below.

- Supply communication interface independent modules by adding aes132_example_main.c, aes132.c, and aes132.h to the project.

- Supply communication interface modules.
  - SPI:
    - Add aes132_spi.c and aes132_spi.h.
    - Define AES132_SPI.
  - I^2C:
    - Add aes132_i2c.c and aes132_i2c.h.
    - Define AES132_I2C.
    - Define F_CPU=16000000.

  You might have to modify the modules, especially for 32-bit CPUs, since their peripherals often implement sophisticated interface functionality in hardware. For instance, they might not support the generation of individual I^2C Start and Stop conditions.

```
<b>Tools</b>\n
AVR Studio 5 Beta 2
(http://www.atmel.com/dyn/products/tools_card.asp?tool_id=17212)
```

2.2 Doxygen Generated Documentation

Important comments (functions, type and macro definitions, etc.) follow a syntax that the Doxygen document generator for source code can parse.

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

aes132.c
This file contains implementations of interface independent functions for the AES132 device 7
aes132.h
Header file for aes132.c 14
aes132_example_main.c
This file contains the main function for AES132 library example 25
aes132_i2c.c
This file contains implementations of I2C functions 25
aes132_i2c.h
This file contains definitions of the I2C layer of the AES132 library 30
aes132_spi.c
This file contains implementations of SPI functions of the AES132 library 36
aes132_spi.h
This file contains definitions of the SPI layer of the AES132 library 41

Chapter 4

File Documentation

4.1 aes132.c File Reference

This file contains implementations of interface independent functions for the AES132 device.

```
#include <stdint.h>
#include <string.h>
#include "aes132.h"
```

Enumerations

- enum aes132_read_write_flag { AES132_WRITE = (uint8_t) 0, AES132_READ = (uint8_t) 1 }

  These enumerations are used as arguments when calling aes132c_access_memory().
- enum aes132_bit_set_flag { AES132_BIT_CLEARED = (uint8_t) 0, AES132_BIT_SET = (uint8_t) 1 }

  These enumerations are used as arguments when calling aes132c_wait_for_status_register_bit().

Functions

- uint8_t aes132m_execute (uint8_t op_code, uint8_t mode, uint16_t param1, uint16_t param2, uint8_t datalen1, uint8_t *data1, uint8_t datalen2, uint8_t *data2, uint8_t datalen3, uint8_t *data3, uint8_t datalen4, uint8_t *data4, uint8_t *tx_buffer, uint8_t *rx_buffer)

  This function creates a command packet, sends it, and receives its response. The caller has to allocate enough space for txBuffer and rxBuffer so that the generated command and the expected response respectively do not overflow these buffers.
- void aes132c_calculate_crc (uint8_t length, uint8_t *data, uint8_t *crc)

  This function calculates a 16-bit CRC.
- uint8_t aes132c_reset_io_address (void)

  This function resets the command and response buffer address.
- uint8_t aes132c_resync ()

  This function resynchronizes communication with the device.
- uint8_t aes132c_read_device_status_register (uint8_t *device_status_register)

  This function reads the device status register.
- uint8_t aes132c_wait_for_status_register_bit (uint8_t mask, uint8_t is_set, uint16_t n_retries)

  This function waits until a bit in the device status register is set or reset. Reading this register will wake up the device.

- uint8_t aes132c_wait_for_device_ready (void)

  This function waits for the Write-In-Progress (WIP) bit in the device status register to be cleared.
- uint8_t aes132c_wait_for_response_ready (void)

  This function waits for the Response-Ready (RRDY) bit in the device status register to be set. \ return status of the operation.
- uint8_t aes132c_send_sleep_command (uint8_t standby)

  This function sends a Sleep command to the device.
- uint8_t aes132c_wakeup (void)

  This function wakes up a device.
- uint8_t aes132c_sleep (void)

  This function puts a device into Sleep mode.
- uint8_t aes132c_standby (void)

  This function puts a device into Standby mode.
- uint8_t aes132c_access_memory (uint8_t count, uint16_t word_address, uint8_t *data, uint8_t read)

  This function writes to or reads from memory with retries.
- uint8_t aes132c_write_memory (uint8_t count, uint16_t word_address, uint8_t *data)

  This function sends data to the device.
- uint8_t aes132c_read_memory (uint8_t size, uint16_t word_address, uint8_t *data)

  This function reads data from the device.
- uint8_t aes132c_send_command (uint8_t *command, uint8_t options)

  This function writes a command into the I/O buffer of the device.
- uint8_t aes132c_receive_response (uint8_t size, uint8_t *response)

  This function reads a response from the I/O buffer of the device.
- uint8_t aes132c_send_and_receive (uint8_t *command, uint8_t size, uint8_t *response, uint8_t options)

  This function sends a command and reads its response.

4.1.1 Detailed Description

This file contains implementations of interface independent functions for the AES132 device.

Author

Atmel Crypto Products

Date

June 16, 2011

4.1.2 Enumeration Type Documentation

4.1.2.1 enum aes132_read_write_flag

These enumerations are used as arguments when calling aes132c_access_memory().

Enumerator

AES132_WRITE write flag

AES132_READ read flag

4.1.2.2 enum aes132_bit_set_flag

These enumerations are used as arguments when calling aes132c_wait_for_status_register_bit().

Enumerator

AES132_BIT_CLEARED bit cleared flag

AES132_BIT_SET bit set flag

4.1.3 Function Documentation

4.1.3.1 uint8_t aes132m_execute (uint8_t *op_code*, uint8_t *mode*, uint16_t *param1*, uint16_t *param2*, uint8_t *datalen1*, uint8_t * *data1*, uint8_t *datalen2*, uint8_t * *data2*, uint8_t *datalen3*, uint8_t * *data3*, uint8_t *datalen4*, uint8_t * *data4*, uint8_t * *tx_buffer*, uint8_t * *rx_buffer*)

This function creates a command packet, sends it, and receives its response. The caller has to allocate enough space for txBuffer and rxBuffer so that the generated command and the expected response respectively do not overflow these buffers.

Parameters

in	*op_code*	command op-code
in	*mode*	command mode
in	*param1*	first parameter
in	*param2*	second parameter
in	*datalen1*	number of bytes in first data block
in	*data1*	pointer to first data block
in	*datalen2*	number of bytes in second data block
in	*data2*	pointer to second data block
in	*datalen3*	number of bytes in third data block
in	*data3*	pointer to third data block
in	*datalen4*	number of bytes in fourth data block
in	*data4*	pointer to fourth data block
in	*tx_buffer*	pointer to command buffer
out	*rx_buffer*	pointer to response buffer

Returns

status of the operation

References AES132_COMMAND_SIZE_MIN, AES132_OPTION_DEFAULT, AES132_RESPONSE_SIZE_MAX, and aes132c_send_and_receive().

Referenced by main().

4.1.3.2 void aes132c_calculate_crc (uint8_t *length*, uint8_t * *data*, uint8_t * *crc*)

This function calculates a 16-bit CRC.

Parameters

in	*length*	number of bytes in data buffer
in	*data*	pointer to data
out	*crc*	pointer to calculated CRC (high byte at crc[0])

Referenced by aes132c_receive_response(), and aes132c_send_command().

4.1.3.3 uint8_t aes132c_reset_io_address (void)

This function resets the command and response buffer address.

Returns

status of the operation

References AES132_RESET_ADDR, and aes132p_write_memory_physical().

Referenced by aes132c_resync(), and aes132c_send_sleep_command().

4.1.3.4 uint8_t aes132c_resync (void)

This function resynchronizes communication with the device.

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, aes132c_reset_io_address(), and aes132p_resync_physical().

Referenced by aes132c_access_memory(), aes132c_receive_response(), and aes132c_send_command().

4.1.3.5 uint8_t aes132c_read_device_status_register (uint8_t * *device_status_register*)

This function reads the device status register.

Parameters

out	*device_status_register*	pointer to byte where the register value is stored

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_RETRY_COUNT_ERROR, AES132_STATUS_ADDR, and aes132p_read_memory_physical().

4.1.3.6 uint8_t aes132c_wait_for_status_register_bit (uint8_t *mask*, uint8_t *is_set*, uint16_t *n_retries*)

This function waits until a bit in the device status register is set or reset. Reading this register will wake up the device.

Parameters

in	*mask*	contains bit pattern to wait for
in	*is_set*	specifies whether to wait until bit is set (AES132_BIT_SET) or reset (AES132_BIT_SET)
in	*n_retries*	16-bit number that indicates the number of retries before stopping to poll.

Returns

status of the operation

References AES132_BIT_SET, AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK, AES132_FUNCTION_RETCODE_SUCCESS, AES132_FUNCTION_RETCODE_TIMEOUT, AES132_STATUS_ADDR, and aes132p_read_memory_physical().

Referenced by aes132c_wait_for_device_ready(), and aes132c_wait_for_response_ready().

4.1.3.7 uint8_t aes132c_wait_for_device_ready (void)

This function waits for the Write-In-Progress (WIP) bit in the device status register to be cleared.

Returns

status of the operation

References AES132_BIT_CLEARED, AES132_RETRY_COUNT_DEVICE_READY, AES132_WIP_BIT, and aes132c_wait_for_status_register_bit().

Referenced by aes132c_access_memory(), and aes132c_wakeup().

4.1.3.8 uint8_t aes132c_send_sleep_command (uint8_t *standby*)

This function sends a Sleep command to the device.

Parameters

in	*standby*	mode (0: sleep, non-zero: standby)

Returns

status of the operation

References AES132_COMMAND_MODE_SLEEP, AES132_COMMAND_MODE_STANDBY, AES132_COMMAND_SIZE_MIN, AES132_FUNCTION_RETCODE_SUCCESS, AES132_OPTION_NO_APPEND_CRC, AES132_OPTION_NO_STATUS_READ, aes132c_reset_io_address(), and aes132c_send_command().

Referenced by aes132c_sleep(), and aes132c_standby().

4.1.3.9 uint8_t aes132c_wakeup (void)

This function wakes up a device.

It takes about 1.5 ms for the device to wake up when in Sleep mode, and about 0.3 ms when in Standby mode.

Returns

status of the operation

References aes132c_wait_for_device_ready().

4.1.3.10 uint8_t aes132c_sleep (void)

This function puts a device into Sleep mode.

Returns

status of the operation

References AES132_COMMAND_MODE_SLEEP, and aes132c_send_sleep_command().

4.1.3.11 uint8_t aes132c_standby (void)

This function puts a device into Standby mode.

Returns

status of the operation

References AES132_COMMAND_MODE_STANDBY, and aes132c_send_sleep_command().

4.1.3.12 uint8_t aes132c_access_memory (uint8_t *count*, uint16_t *word_address*, uint8_t * *data*, uint8_t *read*)

This function writes to or reads from memory with retries.

Parameters

in	*count*	number of bytes to send
in	*word_address*	word address
in,out	*data*	pointer to tx or rx data
in	*read*	flag indicating whether to read (AES132_READ) or write (AES132_WRITE)

Returns

status of the operation or response return code

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_RESPONSE_INDEX_RETURN_CODE, AES132_RESPONSE_SIZE_MIN, AES132_RETRY_COUNT_ERROR, AES132_RETRY_COUNT_RESYNC, aes132c_receive_response(), aes132c_resync(), aes132c_wait_for_device_ready(), aes132c_wait_for_response_ready(), aes132p_read_memory_physical(), and aes132p_write_memory_physical().

Referenced by aes132c_read_memory(), and aes132c_write_memory().

4.1.3.13 uint8_t aes132c_write_memory (uint8_t *count*, uint16_t *word_address*, uint8_t * *data*)

This function sends data to the device.

Parameters

in	*count*	number of bytes to send
in	*word_address*	word address
in	*data*	pointer to tx data

Returns

status of the operation

References AES132_WRITE, and aes132c_access_memory().

Referenced by aes132c_send_command(), and main().

4.1.3.14 uint8_t aes132c_read_memory (uint8_t *size*, uint16_t *word_address*, uint8_t * *data*)

This function reads data from the device.

Parameters

in	*size*	number of bytes to read
in	*word_address*	pointer to word address
out	*data*	pointer to rx data

Returns

status of the operation

References AES132_READ, and aes132c_access_memory().

Referenced by main().

4.1.3.15 uint8_t aes132c_send_command (uint8_t * *command*, uint8_t *options*)

This function writes a command into the I/O buffer of the device.

The fields of the command buffer are described below:

<count, 1 byte> <op-code, 1 byte> <mode, 1 byte> <param1, 2 bytes> <param2, 2 bytes> <data, n bytes (optional)> <crc, 2 bytes)>

The "count" field and size of the command buffer has to include the two CRC bytes independent from the AES132_OPTION_NO_APPEND_CRC flag in the options parameter.

The function retries sending the command if the device indicates a CRC error.

Parameters

in	*command*	pointer to command buffer
in	*options*	flags for communication behavior

Returns

status of the operation

References AES132_COMMAND_INDEX_COUNT, AES132_CRC_ERROR_BIT, AES132_CRC_SIZE, AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK, AES132_FUNCTION_RETCODE_BAD_CRC_TX, AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_OPTION_NO_APPEND_CRC, AES132_OPTION_NO_STATUS_READ, AES132_RETRY_COUNT_ERROR, AES132_STATUS_ADDR, aes132c_calculate_crc(), aes132c_resync(), aes132c_write_memory(), and aes132p_read_memory_physical().

Referenced by aes132c_send_and_receive(), and aes132c_send_sleep_command().

4.1.3.16 uint8_t aes132c_receive_response (uint8_t *size*, uint8_t * *response*)

This function reads a response from the I/O buffer of the device.

Parameters

in	*size*	number of bytes to retrieve (<= response buffer size allocated by caller)
out	*response*	pointer to retrieved response

Returns

status of the operation

References AES132_COMMAND_INDEX_COUNT, AES132_CRC_SIZE, AES132_FUNCTION_RETCODE_BAD_CRC_RX, AES132_FUNCTION_RETCODE_COUNT_INVALID, AES132_FUNCTION_RETCODE_SIZE_TOO_SMALL, AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_RESPONSE_INDEX_COUNT, AES132_RESPONSE_INDEX_RETURN_CODE, AES132_RESPONSE_SIZE_MAX, AES132_RESPONSE_SIZE_MIN, AES132_RETRY_COUNT_ERROR, aes132c_calculate_crc(), aes132c_resync(), aes132c_wait_for_response_ready(), and aes132p_read_memory_physical().

Referenced by aes132c_access_memory(), and aes132c_send_and_receive().

4.1.3.17 uint8_t aes132c_send_and_receive (uint8_t * *command*, uint8_t *size*, uint8_t * *response*, uint8_t *options*)

This function sends a command and reads its response.

Parameters

in	*command*	pointer to command buffer
in	*size*	size of response buffer
out	*response*	pointer to response buffer
in	*options*	flags for communication behavior

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, aes132c_receive_response(), and aes132c_send_command().

Referenced by aes132m_execute().

4.2 aes132.h File Reference

Header file for aes132.c.

```
#include <stdint.h>
```

Macros

- #define AES132_DEVICE_READY_TIMEOUT (100)

  Poll this many ms for the device being ready for access.
- #define AES132_RESPONSE_READY_TIMEOUT (145)

  Poll this many ms for the response buffer being ready for reading.
- #define AES132_RETRY_COUNT_ERROR ((uint8_t) 2)

number of retries for sending a command, receiving a response, and accessing memory

- #define AES132_RETRY_COUNT_RESYNC ((uint8_t) 2)

  number of re-synchronization retries
- #define AES132_CRC_SIZE ((uint8_t) 2)

  size of CRC
- #define AES132_COMMAND_SIZE_MIN ((uint8_t) 9)

  minimum command size
- #define AES132_COMMAND_SIZE_MAX ((uint8_t) 63)

  maximum command size (KeyImport command)
- #define AES132_RESPONSE_SIZE_MIN ((uint8_t) 4)

  minimum response size (EncRead and Encrypt command)
- #define AES132_RESPONSE_SIZE_MAX ((uint8_t) 52)

  maximum response size
- #define AES132_MEM_ACCESS_MAX ((uint8_t) 32)

  maximum number of bytes to write to or read from memory
- #define AES132_IO_ADDR ((uint16_t) 0xFE00)

  word address of command / response buffer
- #define AES132_RESET_ADDR ((uint16_t) 0xFFE0)

  Write to this word address to reset the index of the command / response buffer.
- #define AES132_STATUS_ADDR ((uint16_t) 0xFFF0)

  word address of device status register
- #define AES132_WIP_BIT ((uint8_t) 0x01)

  bit position of the Write-In-Progress bit (WIP) in the device status register
- #define AES132_WEN_BIT ((uint8_t) 0x02)

  bit position of the Write-Enabled bit in the device status register (SPI only)
- #define AES132_WAKE_BIT ((uint8_t) 0x04)

  bit position of the power state bit in the device status register
- #define AES132_RESERVED3_BIT ((uint8_t) 0x08)

  bit position of reserved bit 3 in the device status register
- #define AES132_CRC_ERROR_BIT ((uint8_t) 0x10)

  bit position of the CRC error bit in the device status register
- #define AES132_RESERVED5_BIT ((uint8_t) 0x20)

  bit position of reserved bit 5 in the device status register
- #define AES132_RESPONSE_READY_BIT ((uint8_t) 0x40)

  bit position of the CRC error bit in the device status register
- #define AES132_DEVICE_ERROR_BIT ((uint8_t) 0x80)

  bit position of bit in the device status register that indicates error
- #define AES132_DEVICE_RETCODE_SUCCESS ((uint8_t) 0x00)

  no error in executing a command and receiving a response, or writing data to memory
- #define AES132_DEVICE_RETCODE_BOUNDARY_ERROR ((uint8_t) 0x02)

  error when crossing a page or key boundary for a Write, BlockRead or EncRead
- #define AES132_DEVICE_RETCODE_RW_CONFIG ((uint8_t) 0x04)

  Access to the specified User Zone is not permitted due to the current configuration or internal state.
- #define AES132_DEVICE_RETCODE_BAD_ADDR ((uint8_t) 0x08)

  Address is not implemented, or address is illegal for this command, or attempted to write locked memory.
- #define AES132_DEVICE_RETCODE_COUNT_ERROR ((uint8_t) 0x10)

  Counter limit reached, or count usage error, or restricted key error.

- #define AES132_DEVICE_RETCODE_NONCE_ERROR ((uint8_t) 0x20)

  no nonce available, or nonce invalid, or nonce does not include a random source, or MacCount limit has been reached
- #define AES132_DEVICE_RETCODE_MAC_ERROR ((uint8_t) 0x40)

  Authorization MAC input is missing, or MAC compare failed.
- #define AES132_DEVICE_RETCODE_PARSE_ERROR ((uint8_t) 0x50)

  bad opcode, bad mode, bad parameter, invalid length, or other encoding failure
- #define AES132_DEVICE_RETCODE_DATA_MISMATCH ((uint8_t) 0x60)

  EEPROM post-write automatic data verification failed due to data mismatch.
- #define AES132_DEVICE_RETCODE_LOCK_ERROR ((uint8_t) 0x70)

  Lock command contained bad checksum or bad MAC.
- #define AES132_DEVICE_RETCODE_KEY_ERROR ((uint8_t) 0x80)

  Key is not permitted to be used for this operation, or wrong key was used for operation, or prior authentication has not been performed, or other authentication error, or other key error has occurred.
- #define AES132_DEVICE_RETCODE_TEMP_SENSE_ERROR ((uint8_t) 0x90)

  temperature sensor timeout error
- #define AES132_OPTION_DEFAULT ((uint8_t) 0x00)

  default flags for option parameter
- #define AES132_OPTION_NO_APPEND_CRC ((uint8_t) 0x01)

  flag for option parameter that indicates whether or not to calculate and append a CRC.
- #define AES132_OPTION_NO_STATUS_READ ((uint8_t) 0x02)

  flag for option parameter that indicates whether or not to read the device status register after sending a command.
- #define AES132_COMMAND_INDEX_COUNT (0)

  count at index 0 (1 byte)
- #define AES132_COMMAND_INDEX_OPCODE (1)

  op-code at index 1 (1 byte)
- #define AES132_COMMAND_INDEX_MODE (2)

  mode at index 2 (1 byte)
- #define AES132_COMMAND_INDEX_PARAM1_MSB (3)

  msb of param1 (2 bytes) at index 3
- #define AES132_COMMAND_INDEX_PARAM1_LSB (4)

  lsb of param1 (2 bytes) at index 4
- #define AES132_COMMAND_INDEX_PARAM2_MSB (5)

  msb of param2 (2 bytes) at index 5
- #define AES132_COMMAND_INDEX_PARAM2_LSB (6)

  msb of param2 (2 bytes) at index 5
- #define AES132_COMMAND_MODE_SLEEP ((uint8_t) 0x00)

  value of mode byte for the Sleep command to put device into Sleep mode
- #define AES132_COMMAND_MODE_STANDBY ((uint8_t) 0x40)

  value of mode byte for the Sleep command to put device into Standby mode
- #define AES132_RESPONSE_INDEX_COUNT ((uint8_t) 0)

  count at index 0 (1 byte)
- #define AES132_RESPONSE_INDEX_RETURN_CODE ((uint8_t) 1)

  response return code at index 1 (1 byte)
- #define AES132_RESPONSE_INDEX_DATA ((uint8_t) 2)

  Response data start at index 2 (1 or more bytes).
- #define AES132_OPCODE_AUTH ((uint8_t) 0x03)

  op-code for the Auth command

- #define AES132_OPCODE_AUTH_CHECK ((uint8_t) 0x15)

  op-code for the AuthCheck command
- #define AES132_OPCODE_AUTH_COMPUTE ((uint8_t) 0x14)

  op-code for the AuthCompute command
- #define AES132_OPCODE_BLOCK_READ ((uint8_t) 0x10)

  op-code for the BlockRead command
- #define AES132_OPCODE_COUNTER ((uint8_t) 0x0A)

  op-code for the Counter command
- #define AES132_OPCODE_CRUNCH ((uint8_t) 0x0B)

  op-code for the Crunch command
- #define AES132_OPCODE_DECRYPT ((uint8_t) 0x07)

  op-code for the Decrypt command
- #define AES132_OPCODE_ENC_READ ((uint8_t) 0x04)

  op-code for the EncRead command
- #define AES132_OPCODE_ENCRYPT ((uint8_t) 0x06)

  op-code for the Encrypt command
- #define AES132_OPCODE_ENC_WRITE ((uint8_t) 0x05)

  op-code for the EncWrite command
- #define AES132_OPCODE_INFO ((uint8_t) 0x0C)

  op-code for the Info command
- #define AES132_OPCODE_KEY_CREATE ((uint8_t) 0x08)

  op-code for the KeyCreate command
- #define AES132_OPCODE_KEY_IMPORT ((uint8_t) 0x19)

  op-code for the KeyImport command
- #define AES132_OPCODE_KEY_LOAD ((uint8_t) 0x09)

  op-code for the KeyLoad command
- #define AES132_OPCODE_KEY_TRANSFER ((uint8_t) 0x1A)

  op-code for the KeyTransfer command
- #define AES132_OPCODE_LEGACY ((uint8_t) 0x0F)

  op-code for the Legacy command
- #define AES132_OPCODE_LOCK ((uint8_t) 0x0D)

  op-code for the Lock command
- #define AES132_OPCODE_NONCE ((uint8_t) 0x01)

  op-code for the Nonce command
- #define AES132_OPCODE_NONCE_COMPUTE ((uint8_t) 0x13)

  op-code for the NonceCompute command
- #define AES132_OPCODE_RANDOM ((uint8_t) 0x02)

  op-code for the Random command
- #define AES132_OPCODE_RESET ((uint8_t) 0x00)

  op-code for the Reset command
- #define AES132_OPCODE_SLEEP ((uint8_t) 0x11)

  op-code for the Sleep command
- #define AES132_OPCODE_TEMP_SENSE ((uint8_t) 0x0E)

  op-code for the TempSense command

Functions

- uint8_t aes132m_execute (uint8_t op_code, uint8_t mode, uint16_t param1, uint16_t param2, uint8_t datalen1, uint8_t *data1, uint8_t datalen2, uint8_t *data2, uint8_t datalen3, uint8_t *data3, uint8_t datalen4, uint8_t *data4, uint8_t *tx_buffer, uint8_t *rx_buffer)

  This function creates a command packet, sends it, and receives its response. The caller has to allocate enough space for txBuffer and rxBuffer so that the generated command and the expected response respectively do not overflow these buffers.
- uint8_t aes132c_read_memory (uint8_t count, uint16_t word_address, uint8_t *data)

  This function reads data from the device.
- uint8_t aes132c_write_memory (uint8_t count, uint16_t word_address, uint8_t *data)

  This function sends data to the device.
- uint8_t aes132c_access_memory (uint8_t count, uint16_t word_address, uint8_t *data, uint8_t read)

  This function writes to or reads from memory with retries.
- uint8_t aes132c_read_device_status_register (uint8_t *deviceStatus)

  This function reads the device status register.
- uint8_t aes132c_send_command (uint8_t *command, uint8_t options)

  This function writes a command into the I/O buffer of the device.
- uint8_t aes132c_receive_response (uint8_t count, uint8_t *response)

  This function reads a response from the I/O buffer of the device.
- uint8_t aes132c_send_and_receive (uint8_t *command, uint8_t size, uint8_t *response, uint8_t options)

  This function sends a command and reads its response.
- uint8_t aes132c_wakeup (void)

  This function wakes up a device.
- uint8_t aes132c_sleep (void)

  This function puts a device into Sleep mode.
- uint8_t aes132c_standby (void)

  This function puts a device into Standby mode.
- uint8_t aes132c_resync (void)

  This function resynchronizes communication with the device.
- uint8_t aes132c_wait_for_status_register_bit (uint8_t mask, uint8_t is_set, uint16_t n_retries)

  This function waits until a bit in the device status register is set or reset. Reading this register will wake up the device.
- uint8_t aes132c_wait_for_response_ready (void)

  This function waits for the Response-Ready (RRDY) bit in the device status register to be set. \ return status of the operation.
- uint8_t aes132c_wait_for_device_ready (void)

  This function waits for the Write-In-Progress (WIP) bit in the device status register to be cleared.
- uint8_t aes132c_send_sleep_command (uint8_t standby)

  This function sends a Sleep command to the device.
- uint8_t aes132c_reset_io_address (void)

  This function resets the command and response buffer address.
- void aes132c_calculate_crc (uint8_t count, uint8_t *data, uint8_t *crc)

  This function calculates a 16-bit CRC.
- void aes132p_enable_interface (void)

  This function initializes and enables the I2C hardware peripheral.
- void aes132p_disable_interface (void)

  This function disables the I2C hardware peripheral.

4.2.1 Detailed Description

Header file for aes132.c.

Author

Atmel Crypto Products

Date

June 16, 2011

4.2.2 Macro Definition Documentation

4.2.2.1 #define AES132_RESPONSE_READY_TIMEOUT (145)

Poll this many ms for the response buffer being ready for reading.

When adjusting this number, consider the command execution delays used. As these delays lie closer to or farther from the minimum command execution delays, this number has to be made bigger or smaller accordingly. With other words: The earlier we start polling after sending a command, the longer we have to make the wait-for-response-ready time-out.

4.2.3 Function Documentation

4.2.3.1 uint8_t aes132m_execute (uint8_t *op_code*, uint8_t *mode*, uint16_t *param1*, uint16_t *param2*, uint8_t *datalen1*, uint8_t * *data1*, uint8_t *datalen2*, uint8_t * *data2*, uint8_t *datalen3*, uint8_t * *data3*, uint8_t *datalen4*, uint8_t * *data4*, uint8_t * *tx_buffer*, uint8_t * *rx_buffer*)

This function creates a command packet, sends it, and receives its response. The caller has to allocate enough space for txBuffer and rxBuffer so that the generated command and the expected response respectively do not overflow these buffers.

Parameters

in	*op_code*	command op-code
in	*mode*	command mode
in	*param1*	first parameter
in	*param2*	second parameter
in	*datalen1*	number of bytes in first data block
in	*data1*	pointer to first data block
in	*datalen2*	number of bytes in second data block
in	*data2*	pointer to second data block
in	*datalen3*	number of bytes in third data block
in	*data3*	pointer to third data block
in	*datalen4*	number of bytes in fourth data block
in	*data4*	pointer to fourth data block
in	*tx_buffer*	pointer to command buffer
out	*rx_buffer*	pointer to response buffer

Returns

status of the operation

References AES132_COMMAND_SIZE_MIN, AES132_OPTION_DEFAULT, AES132_RESPONSE_SIZE_MAX, and aes132c_send_and_receive().

Referenced by main().

4.2.3.2 uint8_t aes132c_read_memory (uint8_t *size*, uint16_t *word_address*, uint8_t * *data*)

This function reads data from the device.

Parameters

in	*size*	number of bytes to read
in	*word_address*	pointer to word address
out	*data*	pointer to rx data

Returns

status of the operation

References AES132_READ, and aes132c_access_memory().

Referenced by main().

4.2.3.3 uint8_t aes132c_write_memory (uint8_t *count*, uint16_t *word_address*, uint8_t * *data*)

This function sends data to the device.

Parameters

in	*count*	number of bytes to send
in	*word_address*	word address
in	*data*	pointer to tx data

Returns

status of the operation

References AES132_WRITE, and aes132c_access_memory().

Referenced by aes132c_send_command(), and main().

4.2.3.4 uint8_t aes132c_access_memory (uint8_t *count*, uint16_t *word_address*, uint8_t * *data*, uint8_t *read*)

This function writes to or reads from memory with retries.

Parameters

in	*count*	number of bytes to send
in	*word_address*	word address
in,out	*data*	pointer to tx or rx data
in	*read*	flag indicating whether to read (AES132_READ) or write (AES132_WRITE)

Returns

status of the operation or response return code

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_RESPONSE_INDEX_RETURN_CODE, AES132_RESPONSE_SIZE_MIN, AES132_RETRY_COUNT_ERROR, AES132_RETRY_COUNT_RESYNC, aes132c_receive_response(), aes132c_resync(), aes132c_wait_for_device_ready(), aes132c_wait_for_response_ready(), aes132p_read_memory_physical(), and aes132p_write_memory_physical().

Referenced by aes132c_read_memory(), and aes132c_write_memory().

4.2.3.5 uint8_t aes132c_read_device_status_register (uint8_t * *device_status_register*)

This function reads the device status register.

Parameters

out	*device_status_register*	pointer to byte where the register value is stored

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_RETRY_COUNT_ERROR, AES132_STATUS_ADDR, and aes132p_read_memory_physical().

4.2.3.6 uint8_t aes132c_send_command (uint8_t * *command*, uint8_t *options*)

This function writes a command into the I/O buffer of the device.

The fields of the command buffer are described below:

<count, 1 byte> <op-code, 1 byte> <mode, 1 byte> <param1, 2 bytes> <param2, 2 bytes> <data, n bytes (optional)> <crc, 2 bytes)>

The "count" field and size of the command buffer has to include the two CRC bytes independent from the AES132_OPTION_NO_APPEND_CRC flag in the options parameter.

The function retries sending the command if the device indicates a CRC error.

Parameters

in	*command*	pointer to command buffer
in	*options*	flags for communication behavior

Returns

status of the operation

References AES132_COMMAND_INDEX_COUNT, AES132_CRC_ERROR_BIT, AES132_CRC_SIZE, AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK, AES132_FUNCTION_RETCODE_BAD_CRC_TX, AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_OPTION_NO_APPEND_CRC, AES132_OPTION_NO_STATUS_READ, AES132_RETRY_COUNT_ERROR, AES132_STATUS_ADDR, aes132c_calculate_crc(), aes132c_resync(), aes132c_write_memory(), and aes132p_read_memory_physical().

Referenced by aes132c_send_and_receive(), and aes132c_send_sleep_command().

4.2.3.7 uint8_t aes132c_receive_response (uint8_t *size*, uint8_t * *response*)

This function reads a response from the I/O buffer of the device.

Parameters

in	*size*	number of bytes to retrieve (<= response buffer size allocated by caller)
out	*response*	pointer to retrieved response

Returns

status of the operation

References AES132_COMMAND_INDEX_COUNT, AES132_CRC_SIZE, AES132_FUNCTION_RETCODE_BAD_CRC_RX, AES132_FUNCTION_RETCODE_COUNT_INVALID, AES132_FUNCTION_RETCODE_SIZE_TOO_SMALL, AES132_FUNCTION_RETCODE_SUCCESS, AES132_IO_ADDR, AES132_RESPONSE_INDEX_COUNT, AES132_RESPONSE_INDEX_RETURN_CODE, AES132_RESPONSE_SIZE_MAX, AES132_RESPONSE_SIZE_MIN, AES132_RETRY_COUNT_ERROR, aes132c_calculate_crc(), aes132c_resync(), aes132c_wait_for_response_ready(), and aes132p_read_memory_physical().

Referenced by aes132c_access_memory(), and aes132c_send_and_receive().

4.2.3.8 uint8_t aes132c_send_and_receive (uint8_t * *command*, uint8_t *size*, uint8_t * *response*, uint8_t *options*)

This function sends a command and reads its response.

Parameters

in	*command*	pointer to command buffer
in	*size*	size of response buffer
out	*response*	pointer to response buffer
in	*options*	flags for communication behavior

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, aes132c_receive_response(), and aes132c_send_command().

Referenced by aes132m_execute().

4.2.3.9 uint8_t aes132c_wakeup (void)

This function wakes up a device.

It takes about 1.5 ms for the device to wake up when in Sleep mode, and about 0.3 ms when in Standby mode.

Returns

status of the operation

References aes132c_wait_for_device_ready().

4.2.3.10 uint8_t aes132c_sleep (void)

This function puts a device into Sleep mode.

Returns

status of the operation

References AES132_COMMAND_MODE_SLEEP, and aes132c_send_sleep_command().

4.2.3.11 uint8_t aes132c_standby (void)

This function puts a device into Standby mode.

Returns

status of the operation

References AES132_COMMAND_MODE_STANDBY, and aes132c_send_sleep_command().

4.2.3.12 uint8_t aes132c_resync (void)

This function resynchronizes communication with the device.

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, aes132c_reset_io_address(), and aes132p_resync_physical().

Referenced by aes132c_access_memory(), aes132c_receive_response(), and aes132c_send_command().

4.2.3.13 uint8_t aes132c_wait_for_status_register_bit (uint8_t *mask*, uint8_t *is_set*, uint16_t *n_retries*)

This function waits until a bit in the device status register is set or reset. Reading this register will wake up the device.

Parameters

in	*mask*	contains bit pattern to wait for
in	*is_set*	specifies whether to wait until bit is set (AES132_BIT_SET) or reset (AES132_BIT_SET)
in	*n_retries*	16-bit number that indicates the number of retries before stopping to poll.

Returns

status of the operation

References AES132_BIT_SET, AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK, AES132_FUNCTION_RETCODE_SUCCESS, AES132_FUNCTION_RETCODE_TIMEOUT, AES132_STATUS_ADDR, and aes132p_read_memory_physical().

Referenced by aes132c_wait_for_device_ready(), and aes132c_wait_for_response_ready().

4.2.3.14 uint8_t aes132c_wait_for_device_ready (void)

This function waits for the Write-In-Progress (WIP) bit in the device status register to be cleared.

Returns

status of the operation

References AES132_BIT_CLEARED, AES132_RETRY_COUNT_DEVICE_READY, AES132_WIP_BIT, and aes132c_wait_for_status_register_bit().

Referenced by aes132c_access_memory(), and aes132c_wakeup().

4.2.3.15 uint8_t aes132c_send_sleep_command (uint8_t *standby*)

This function sends a Sleep command to the device.

Parameters

in	*standby*	mode (0: sleep, non-zero: standby)

Returns

status of the operation

References AES132_COMMAND_MODE_SLEEP, AES132_COMMAND_MODE_STANDBY, AES132_COMMAND_SIZE_MIN, AES132_FUNCTION_RETCODE_SUCCESS, AES132_OPTION_NO_APPEND_CRC, AES132_OPTION_NO_STATUS_READ, aes132c_reset_io_address(), and aes132c_send_command().

Referenced by aes132c_sleep(), and aes132c_standby().

4.2.3.16 uint8_t aes132c_reset_io_address (void)

This function resets the command and response buffer address.

Returns

status of the operation

References AES132_RESET_ADDR, and aes132p_write_memory_physical().

Referenced by aes132c_resync(), and aes132c_send_sleep_command().

4.2.3.17 void aes132c_calculate_crc (uint8_t *length*, uint8_t * *data*, uint8_t * *crc*)

This function calculates a 16-bit CRC.

Parameters

in	*length*	number of bytes in data buffer
in	*data*	pointer to data
out	*crc*	pointer to calculated CRC (high byte at crc[0])

Referenced by aes132c_receive_response(), and aes132c_send_command().

4.3 aes132_example_main.c File Reference

This file contains the main function for AES132 library example.

```
#include <stdint.h>
#include <string.h>
#include "aes132.h"
```

Functions

- int main (void)

  This function is the entry function for an example application that uses the AES132 library.

4.3.1 Detailed Description

This file contains the main function for AES132 library example.

Author

Atmel Crypto Products

Date

June 16, 2011

4.3.2 Function Documentation

4.3.2.1 int main (void)

This function is the entry function for an example application that uses the AES132 library.

Returns

result (0: success, otherwise failure)

References AES132_COMMAND_SIZE_MIN, AES132_FUNCTION_RETCODE_SUCCESS, AES132_OPCODE_-BLOCK_READ, AES132_RESPONSE_INDEX_DATA, AES132_RESPONSE_SIZE_MIN, aes132c_read_memory(), aes132c_write_memory(), aes132m_execute(), aes132p_disable_interface(), and aes132p_enable_interface().

4.4 aes132_i2c.c File Reference

This file contains implementations of I2C functions.

```
#include <stdint.h>
#include <avr\io.h>
#include <util\twi.h>
#include "aes132_i2c.h"
```

Macros

- #define AES132_I2C_DEFAULT_ADDRESS ((uint8_t) 0xA0)

  < *I2C AVR definitions*

Enumerations

- enum aes132_i2c_read_write_flag { I2C_WRITE = (uint8_t) 0x00 }

  These enumerations are flags for I2C read or write addressing.

Functions

- void i2c_enable (void)

  This function initializes and enables the I2C peripheral.
- void i2c_disable (void)

  This function disables the I2C peripheral.
- uint8_t i2c_send_start (void)

  This function creates a Start condition (SDA low, then SCL low).
- uint8_t i2c_send_stop (void)

  This function creates a Stop condition (SCL high, then SDA high).
- uint8_t i2c_send_bytes (uint8_t count, uint8_t *data)

  This function sends bytes to an I2C device.
- uint8_t i2c_receive_byte (uint8_t *data)

  This function receives one byte from an I2C device.
- uint8_t i2c_receive_bytes (uint8_t count, uint8_t *data)

  This function receives bytes from an I2C device and sends a Stop.
- uint8_t aes132p_send_slave_address (uint8_t read)

  This function creates a Start condition and sends the I2C address.
- void aes132p_enable_interface (void)

  This function initializes and enables the I2C hardware peripheral.
- void aes132p_disable_interface (void)

  This function disables the I2C hardware peripheral.
- uint8_t aes132p_select_device (uint8_t device_id)

  This function selects a I2C AES132 device.
- uint8_t aes132p_write_memory_physical (uint8_t count, uint16_t word_address, uint8_t *data)

  This function writes bytes to the device.
- uint8_t aes132p_read_memory_physical (uint8_t size, uint16_t word_address, uint8_t *data)

  This function reads bytes from the device.
- uint8_t aes132p_resync_physical (void)

  This function resynchronizes communication.

4.4.1 Detailed Description

This file contains implementations of I2C functions.

Author

Atmel Crypto Products

Date

June 16, 2011

4.4.2 Macro Definition Documentation

4.4.2.1 #define AES132_I2C_DEFAULT_ADDRESS ((uint8_t) 0xA0)

< I2C AVR definitions

< C type definitions GPIO definitionsI2C address used at AES132 library startup.

4.4.3 Enumeration Type Documentation

4.4.3.1 enum aes132_i2c_read_write_flag

These enumerations are flags for I2C read or write addressing.

Enumerator

I2C_WRITE write command id

4.4.4 Function Documentation

4.4.4.1 uint8_t i2c_send_start (void)

This function creates a Start condition (SDA low, then SCL low).

Returns

status of the operation

References I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and I2C_START_TIMEOUT.

Referenced by aes132p_resync_physical(), and aes132p_send_slave_address().

4.4.4.2 uint8_t i2c_send_stop (void)

This function creates a Stop condition (SCL high, then SDA high).

Returns

status of the operation

References I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and I2C_STOP_TIMEOUT.

Referenced by aes132p_read_memory_physical(), aes132p_resync_physical(), aes132p_send_slave_address(), aes132p_write_memory_physical(), i2c_receive_byte(), and i2c_receive_bytes().

4.4.4.3 uint8_t i2c_send_bytes (uint8_t *count,* uint8_t * *data*)

This function sends bytes to an I2C device.

Parameters

in	*count*	number of bytes to send
in	*data*	pointer to tx buffer

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_NACK, I2C_FUNCTION_RETCODE_SUCCESS, and I2C_FUNCTION_RETCODE_TIMEOUT.

Referenced by aes132p_read_memory_physical(), aes132p_resync_physical(), aes132p_send_slave_address(), and aes132p_write_memory_physical().

4.4.4.4 uint8_t i2c_receive_byte (uint8_t * *data*)

This function receives one byte from an I2C device.

Parameters

out	*data*	pointer to received byte

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and i2c_send_stop().

4.4.4.5 uint8_t i2c_receive_bytes (uint8_t *count,* uint8_t * *data*)

This function receives bytes from an I2C device and sends a Stop.

Parameters

in	*count*	number of bytes to receive
out	*data*	pointer to rx buffer

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_TIMEOUT, and i2c_send_stop().

Referenced by aes132p_read_memory_physical().

4.4.4.6 uint8_t aes132p_send_slave_address (uint8_t *read*)

This function creates a Start condition and sends the I2C address.

Parameters

in	*read*	I2C_READ for reading, I2C_WRITE for writing

Returns

status of the operation

References AES132_FUNCTION_RETCODE_ADDRESS_READ_NACK, AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK, AES132_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_NACK, i2c_send_bytes(), i2c_send_start(), and i2c_send_stop().

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.4.4.7 uint8_t aes132p_select_device (uint8_t *device_id*)

This function selects a I2C AES132 device.

Parameters

in	*device_id*	I2C address

Returns

always success

4.4.4.8 uint8_t aes132p_write_memory_physical (uint8_t *count,* uint16_t *word_address,* uint8_t * *data*)

This function writes bytes to the device.

Parameters

in	*count*	number of bytes to write
in	*word_address*	word address to write to
in	*data*	pointer to tx buffer

Returns

status of the operation

4.4.4.9 uint8_t aes132p_read_memory_physical (uint8_t *size*, uint16_t *word_address*, uint8_t * *data*)

This function reads bytes from the device.

Parameters

in	*size*	number of bytes to write
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

4.4.4.10 uint8_t aes132p_resync_physical (void)

This function resynchronizes communication.

Returns

status of the operation

4.5 aes132_i2c.h File Reference

This file contains definitions of the I2C layer of the AES132 library.

```
#include <stdint.h>
```

Macros

- #define POLL_TIME_MEASURED

  Status byte polling time was measured.
- #define AES132_STATUS_REG_POLL_TIME_ACK (158)

  time for polling a bit in the device status register in us (measured)
- #define AES132_ITERATIONS_PER_MS_ACK (1000 / AES132_STATUS_REG_POLL_TIME_ACK)

  timeout loop iterations per ms derived from the times it takes to communicate and loop when acknowledged
- #define AES132_ITERATIONS_PER_MS_NACK (1000 / AES132_STATUS_REG_POLL_TIME_NACK)

  timeout loop iterations per ms derived from the times it takes to communicate and loop when not acknowledged
- #define AES132_RETRY_COUNT_DEVICE_READY ((uint16_t) (AES132_DEVICE_READY_TIMEOUT * AES132_ITERATIONS_PER_MS_NACK))

  Poll this many times for the device being ready for access.
- #define AES132_RETRY_COUNT_RESPONSE_READY

  Poll this many times for the response buffer being ready for reading.
- #define I2C_CLOCK (400000.0)

  I2C clock.
- #define I2C_PULLUP

  Use pull-up resistors.
- #define I2C_START_TIMEOUT ((uint8_t) 250)

number of polling iterations for TWINT bit in TWSR after creating a Start condition in i2c_send_start()

- #define I2C_BYTE_TIMEOUT ((uint8_t) 100)

  number of polling iterations for TWINT bit in TWSR after sending or receiving a byte.
- #define I2C_STOP_TIMEOUT ((uint8_t) 250)

  number of polling iterations for TWSTO bit in TWSR after creating a Stop condition in i2c_send_stop().
- #define I2C_FUNCTION_RETCODE_SUCCESS ((uint8_t) 0x00)

  Communication with device succeeded.
- #define I2C_FUNCTION_RETCODE_COMM_FAIL ((uint8_t) 0xF0)

  Communication with device failed.
- #define I2C_FUNCTION_RETCODE_TIMEOUT ((uint8_t) 0xF1)

  Communication timed out.
- #define I2C_FUNCTION_RETCODE_NACK ((uint8_t) 0xF8)

  I2C nack.
- #define AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK ((uint8_t) 0xA0)

  I2C nack when sending a I2C address for writing.
- #define AES132_FUNCTION_RETCODE_ADDRESS_READ_NACK ((uint8_t) 0xA1)

  I2C nack when sending a I2C address for reading.
- #define AES132_FUNCTION_RETCODE_SIZE_TOO_SMALL ((uint8_t) 0xA2)

  Count value in response was bigger than buffer.
- #define AES132_FUNCTION_RETCODE_SUCCESS ((uint8_t) 0x00)

  Function succeeded.
- #define AES132_FUNCTION_RETCODE_BAD_CRC_TX ((uint8_t) 0xD4)

  Device status register bit 4 (CRC) is set.
- #define AES132_FUNCTION_RETCODE_NOT_IMPLEMENTED ((uint8_t) 0xE0)

  interface function not implemented
- #define AES132_FUNCTION_RETCODE_DEVICE_SELECT_FAIL ((uint8_t) 0xE3)

  device index out of bounds
- #define AES132_FUNCTION_RETCODE_COUNT_INVALID ((uint8_t) 0xE4)

  count byte in response is out of range
- #define AES132_FUNCTION_RETCODE_BAD_CRC_RX ((uint8_t) 0xE5)

  incorrect CRC received
- #define AES132_FUNCTION_RETCODE_TIMEOUT ((uint8_t) 0xE7)

  Function timed out while waiting for response.
- #define AES132_FUNCTION_RETCODE_COMM_FAIL ((uint8_t) 0xF0)

  Communication with device failed.

Functions

- void i2c_enable (void)

  This function initializes and enables the I2C peripheral.
- void i2c_disable (void)

  This function disables the I2C peripheral.
- uint8_t i2c_send_start (void)

  This function creates a Start condition (SDA low, then SCL low).
- uint8_t i2c_send_stop (void)

  This function creates a Stop condition (SCL high, then SDA high).
- uint8_t i2c_send_bytes (uint8_t count, uint8_t *data)

This function sends bytes to an I2C device.

- uint8_t i2c_receive_byte (uint8_t *data)

  This function receives one byte from an I2C device.

- uint8_t i2c_receive_bytes (uint8_t count, uint8_t *data)

  This function receives bytes from an I2C device and sends a Stop.

- void aes132p_enable_interface (void)

  This function initializes and enables the I2C hardware peripheral.

- void aes132p_disable_interface (void)

  This function disables the I2C hardware peripheral.

- uint8_t aes132p_select_device (uint8_t device_id)

  This function selects a I2C AES132 device.

- uint8_t aes132p_read_memory_physical (uint8_t size, uint16_t word_address, uint8_t *data)

  This function reads bytes from the device.

- uint8_t aes132p_write_memory_physical (uint8_t count, uint16_t word_address, uint8_t *data)

  This function writes bytes to the device.

- uint8_t aes132p_resync_physical (void)

  This function resynchronizes communication.

4.5.1 Detailed Description

This file contains definitions of the I2C layer of the AES132 library.

Author

Atmel Crypto Products

Date

June 16, 2011

4.5.2 Macro Definition Documentation

4.5.2.1 #define POLL_TIME_MEASURED

Status byte polling time was measured.

Undefine this if you want to use estimated values (see below).

4.5.2.2 #define AES132_STATUS_REG_POLL_TIME_ACK (158)

time for polling a bit in the device status register in us (measured)

With an oscilloscope or logic analyzer, measure the time it takes for one loop iteration in aes132c_wait_for_status_bit() when acknowledged and when not acknowledged, and enter it here.

4.5.2.3 #define AES132_RETRY_COUNT_RESPONSE_READY

Value:

```
((uint16_t) (AES132_RESPONSE_READY_TIMEOUT *
      AES132_ITERATIONS_PER_MS_ACK * 2 \
                      + 2 * AES132_ITERATIONS_PER_MS_NACK))
```

Poll this many times for the response buffer being ready for reading.

The device "nacks" the I2C address while parsing the command during the first few (average of two) milliseconds (second term in formula below). Give some slack by doubling the minimum timeout.

Referenced by aes132c_wait_for_response_ready().

4.5.2.4 #define I2C_START_TIMEOUT ((uint8_t) 250)

number of polling iterations for TWINT bit in TWSR after creating a Start condition in i2c_send_start()

Adjust this value considering how long it takes to check a status bit in the I2C status register, decrement the timeout counter, compare its value with 0, and branch.

Referenced by i2c_send_start().

4.5.2.5 #define I2C_BYTE_TIMEOUT ((uint8_t) 100)

number of polling iterations for TWINT bit in TWSR after sending or receiving a byte.

Adjust this value considering how long it takes to check a status bit in the I2C status register, decrement the timeout counter, compare its value with 0, branch, and to send or receive one byte.

Referenced by i2c_receive_byte(), i2c_receive_bytes(), and i2c_send_bytes().

4.5.2.6 #define I2C_STOP_TIMEOUT ((uint8_t) 250)

number of polling iterations for TWSTO bit in TWSR after creating a Stop condition in i2c_send_stop().

Adjust this value considering how long it takes to check a status bit in the I2C control register, decrement the timeout counter, compare its value with 0, and branch.

Referenced by i2c_send_stop().

4.5.3 Function Documentation

4.5.3.1 uint8_t i2c_send_start (void)

This function creates a Start condition (SDA low, then SCL low).

Returns

status of the operation

References I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and I2C_START_TIMEOUT.

Referenced by aes132p_resync_physical(), and aes132p_send_slave_address().

4.5.3.2 uint8_t i2c_send_stop (void)

This function creates a Stop condition (SCL high, then SDA high).

Returns

status of the operation

References I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and I2C_STOP_TIMEOUT.

Referenced by aes132p_read_memory_physical(), aes132p_resync_physical(), aes132p_send_slave_address(), aes132p_write_memory_physical(), i2c_receive_byte(), and i2c_receive_bytes().

4.5.3.3 uint8_t i2c_send_bytes (uint8_t *count*, uint8_t * *data*)

This function sends bytes to an I2C device.

Parameters

in	*count*	number of bytes to send
in	*data*	pointer to tx buffer

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_NACK, I2C_FUNCTION_RETCODE_SUCCESS, and I2C_FUNCTION_RETCODE_TIMEOUT.

Referenced by aes132p_read_memory_physical(), aes132p_resync_physical(), aes132p_send_slave_address(), and aes132p_write_memory_physical().

4.5.3.4 uint8_t i2c_receive_byte (uint8_t * *data*)

This function receives one byte from an I2C device.

Parameters

out	*data*	pointer to received byte

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_SUCCESS, I2C_FUNCTION_RETCODE_TIMEOUT, and i2c_send_stop().

4.5.3.5 uint8_t i2c_receive_bytes (uint8_t *count*, uint8_t * *data*)

This function receives bytes from an I2C device and sends a Stop.

Parameters

in	*count*	number of bytes to receive
out	*data*	pointer to rx buffer

Returns

status of the operation

References I2C_BYTE_TIMEOUT, I2C_FUNCTION_RETCODE_COMM_FAIL, I2C_FUNCTION_RETCODE_TIMEOUT, and i2c_send_stop().

Referenced by aes132p_read_memory_physical().

4.5.3.6 uint8_t aes132p_select_device (uint8_t *device_id*)

This function selects a I2C AES132 device.

Parameters

in	*device_id*	I2C address

Returns

always success

This function selects a I2C AES132 device.

Parameters

in	*device_id*	index into the SPI array for chip select ports and or pins

Returns

success or out-of-bounds error

4.5.3.7 uint8_t aes132p_read_memory_physical (uint8_t *size*, uint16_t *word_address*, uint8_t * *data*)

This function reads bytes from the device.

Parameters

in	*size*	number of bytes to write
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

Parameters

in	*size*	number of bytes to read
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

Referenced by aes132c_access_memory(), aes132c_read_device_status_register(), aes132c_receive_response(), aes132c_send_command(), and aes132c_wait_for_status_register_bit().

4.5.3.8 uint8_t aes132p_write_memory_physical (uint8_t *count,* uint16_t *word_address,* uint8_t * *data*)

This function writes bytes to the device.

Parameters

in	*count*	number of bytes to write
in	*word_address*	word address to write to
in	*data*	pointer to tx buffer

Returns

status of the operation

Referenced by aes132c_access_memory(), and aes132c_reset_io_address().

4.5.3.9 uint8_t aes132p_resync_physical (void)

This function resynchronizes communication.

Returns

status of the operation

This function resynchronizes communication.

Returns

always success

Referenced by aes132c_resync().

4.6 aes132_spi.c File Reference

This file contains implementations of SPI functions of the AES132 library.

```
#include <stdint.h>
#include <avr\io.h>
#include "aes132_spi.h"
```

Macros

- #define SPI_RX_TIMEOUT ((uint8_t) 250)

  Adjust this variable considering how long one iteration of the inner while loop in spi_receive_bytes() takes (receiving one byte and decrementing the timeout counter).
- #define SPI_TX_TIMEOUT ((uint8_t) 50)

  Adjust this variable considering how long one iteration of the inner while loop in spi_send_bytes() takes, (sending one byte and decrementing the timeout counter).
- #define SPI_PORT_OUT PORTB

  port output register
- #define SPI_PORT_DDR DDRB

  port direction register
- #define SPI_CS_0 (1)

  bit value for slave select of first device (pin 1)
- #define SPI_CLOCK (2)

  bit value for clock (pin 2)
- #define SPI_MOSI (4)

  bit value for Master-Out-Slave-In (pin 3)
- #define SPI_MISO (8)

  bit value for Master-In-Slave-Out (pin 4)
- #define SPI_DEVICE_COUNT (1)

  number of AES132 devices
- #define AES132_SPI_WRITE ((uint8_t) 2)

  write command id
- #define AES132_SPI_READ ((uint8_t) 3)

  read command id
- #define AES132_SPI_ENABLE_WRITE ((uint8_t) 6)

  enable-write command id
- #define AES132_SPI_PREFACE_SIZE ((uint8_t) 3)

  three bytes (command id, word address MSB, word address LSB) before sending and receiving actual data

Functions

- uint8_t spi_select_device (uint8_t index)

  This function assigns the chip select pin to be used for communication.
- void spi_select_slave (void)
- void spi_deselect_slave (void)
- void spi_enable (void)

  This function initializes and enables the SPI peripheral.
- void spi_disable (void)

  This function disables the SPI peripheral.
- uint8_t spi_send_bytes (uint8_t count, uint8_t *data)

  This function sends bytes to an SPI device.
- uint8_t spi_receive_bytes (uint8_t count, uint8_t *data)

  This function receives bytes from an SPI device.
- void aes132p_enable_interface (void)

  This function initializes and enables the SPI peripheral.
- void aes132p_disable_interface (void)

This function disables the SPI peripheral.

- uint8_t aes132p_select_device (uint8_t device_id)

  This function selects a device from an array of SPI chip select pins for AES132 devices on the same SPI bus.

- uint8_t aes132p_write_memory_physical (uint8_t count, uint16_t word_address, uint8_t *data)

  This function writes bytes to the device.

- uint8_t aes132p_read_memory_physical (uint8_t size, uint16_t word_address, uint8_t *data)

  This function reads bytes from the device.

- uint8_t aes132p_resync_physical (void)

  This function implements the interface for resynchronization of communication, but cannot be implemented when using SPI.

4.6.1 Detailed Description

This file contains implementations of SPI functions of the AES132 library.

Author

Atmel Crypto Products

Date

June 16, 2011

4.6.2 Function Documentation

4.6.2.1 uint8_t spi_select_device (uint8_t *index*)

This function assigns the chip select pin to be used for communication.

```
The same SPI peripheral is used (clock and data lines), but for the
chip select, a different port and / or pin can be assigned.
```

Parameters

index	index into the port and pin arrays

Returns

error if index argument is out of bounds

References AES132_FUNCTION_RETCODE_DEVICE_SELECT_FAIL, SPI_DEVICE_COUNT, and SPI_FUNCTION-_RETCODE_SUCCESS.

Referenced by aes132p_select_device().

4.6.2.2 void spi_select_slave (void)

This function selects the SPI slave.

References SPI_CLOCK, and SPI_PORT_OUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.6.2.3 void spi_deselect_slave (void)

This function deselects the SPI slave.

References SPI_CLOCK, and SPI_PORT_OUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.6.2.4 uint8_t spi_send_bytes (uint8_t *count*, uint8_t * *data*)

This function sends bytes to an SPI device.

Parameters

in	*count*	number of bytes to send
in	*data*	pointer to tx buffer

Returns

status of the operation

References SPI_FUNCTION_RETCODE_SUCCESS, SPI_FUNCTION_RETCODE_TIMEOUT, and SPI_TX_TIMEOUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.6.2.5 uint8_t spi_receive_bytes (uint8_t *count*, uint8_t * *data*)

This function receives bytes from an SPI device.

Parameters

in	*count*	number of bytes to receive
in	*data*	pointer to rx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_TIMEOUT, SPI_FUNCTION_RETCODE_SUCCESS, and SPI_RX_TIMEOUT.

Referenced by aes132p_read_memory_physical().

4.6.2.6 void aes132p_enable_interface (void)

This function initializes and enables the SPI peripheral.

This function initializes and enables the I2C hardware peripheral.

References i2c_enable(), and spi_enable().

Referenced by main().

4.6.2.7 void aes132p_disable_interface (void)

This function disables the SPI peripheral.

This function disables the I2C hardware peripheral.

References i2c_disable(), and spi_disable().

Referenced by main().

4.6.2.8 uint8_t aes132p_select_device (uint8_t *device_id*)

This function selects a device from an array of SPI chip select pins for AES132 devices on the same SPI bus.

This function selects a I2C AES132 device.

Parameters

in	*device_id*	index into the SPI array for chip select ports and or pins

Returns

success or out-of-bounds error

References AES132_FUNCTION_RETCODE_SUCCESS, and spi_select_device().

4.6.2.9 uint8_t aes132p_write_memory_physical (uint8_t *count,* uint16_t *word_address,* uint8_t * *data*)

This function writes bytes to the device.

Parameters

in	*count*	number of bytes to write
in	*word_address*	word address to write to
in	*data*	pointer to tx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_SPI_ENABLE_WRITE, AES132_SPI_PREFACE_SIZE, AES132_SPI_WRITE, aes132p_send_slave_address(), i2c_send_bytes(), i2c_send_stop(), I2C_WRITE, spi_deselect_slave(), spi_select_slave(), and spi_send_bytes().

Referenced by aes132c_access_memory(), and aes132c_reset_io_address().

4.6.2.10 uint8_t aes132p_read_memory_physical (uint8_t *size,* uint16_t *word_address,* uint8_t * *data*)

This function reads bytes from the device.

Parameters

in	*size*	number of bytes to read
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_SPI_PREFACE_SIZE, AES132_SPI_READ, aes132p_send_slave_address(), i2c_receive_bytes(), i2c_send_bytes(), i2c_send_stop(), I2C_WRITE, spi_deselect_slave(), spi_receive_bytes(), spi_select_slave(), and spi_send_bytes().

Referenced by aes132c_access_memory(), aes132c_read_device_status_register(), aes132c_receive_response(), aes132c_send_command(), and aes132c_wait_for_status_register_bit().

4.6.2.11 uint8_t aes132p_resync_physical (void)

This function implements the interface for resynchronization of communication, but cannot be implemented when using SPI.

This function resynchronizes communication.

Returns

always success

References AES132_FUNCTION_RETCODE_SUCCESS, i2c_disable(), i2c_enable(), i2c_send_bytes(), i2c_send_start(), and i2c_send_stop().

Referenced by aes132c_resync().

4.7 aes132_spi.h File Reference

This file contains definitions of the SPI layer of the AES132 library.

```
#include <stdint.h>
```

Macros

- #define AES132_STATUS_REG_POLL_TIME (23)

  time for polling a bit in the device status register in us (measured)
- #define AES132_ITERATIONS_PER_MS (1000 / AES132_STATUS_REG_POLL_TIME)

  timeout loop iterations per ms derived from the times it takes to communicate and loop when acked
- #define AES132_RETRY_COUNT_DEVICE_READY ((uint16_t) (AES132_DEVICE_READY_TIMEOUT * AES132_ITERATIONS_PER_MS))

  Poll this many times for the device being ready for access.
- #define AES132_RETRY_COUNT_RESPONSE_READY ((uint16_t) (AES132_RESPONSE_READY_TIMEOUT * AES132_ITERATIONS_PER_MS * 2))

  Poll this many times for the response buffer being ready for reading.
- #define SPI_FUNCTION_RETCODE_SUCCESS ((uint8_t) 0x00)

  Communication with device succeeded.
- #define SPI_FUNCTION_RETCODE_COMM_FAIL ((uint8_t) 0xF0)

  Communication with device failed.
- #define SPI_FUNCTION_RETCODE_TIMEOUT ((uint8_t) 0xF1)

  Communication timed out.

- #define AES132_FUNCTION_RETCODE_ADDRESS_WRITE_NACK ((uint8_t) 0xA0)
  I2C nack when sending an I2C address for writing (only returned by I2C)
- #define AES132_FUNCTION_RETCODE_ADDRESS_READ_NACK ((uint8_t) 0xA1)
  I2C nack when sending an I2C address for reading (only returned by I2C)
- #define AES132_FUNCTION_RETCODE_SIZE_TOO_SMALL ((uint8_t) 0xA2)
  Count value in response was bigger than buffer.
- #define AES132_FUNCTION_RETCODE_SUCCESS ((uint8_t) 0x00)
  Function succeeded.
- #define AES132_FUNCTION_RETCODE_BAD_CRC_TX ((uint8_t) 0xD4)
  Device status register bit 4 (CRC) is set.
- #define AES132_FUNCTION_RETCODE_NOT_IMPLEMENTED ((uint8_t) 0xE0)
  interface function not implemented
- #define AES132_FUNCTION_RETCODE_DEVICE_SELECT_FAIL ((uint8_t) 0xE3)
  device index out of bounds
- #define AES132_FUNCTION_RETCODE_COUNT_INVALID ((uint8_t) 0xE4)
  count byte in response is out of range
- #define AES132_FUNCTION_RETCODE_BAD_CRC_RX ((uint8_t) 0xE5)
  incorrect CRC received
- #define AES132_FUNCTION_RETCODE_TIMEOUT ((uint8_t) 0xE7)
  Function timed out while waiting for response.
- #define AES132_FUNCTION_RETCODE_COMM_FAIL ((uint8_t) 0xF0)
  Communication with device failed.

Functions

- void spi_enable (void)
  This function initializes and enables the SPI peripheral.
- void spi_disable (void)
  This function disables the SPI peripheral.
- void spi_select_slave (void)
- void spi_deselect_slave (void)
- uint8_t spi_select_device (uint8_t index)
  This function assigns the chip select pin to be used for communication.
- uint8_t spi_send_bytes (uint8_t count, uint8_t *data)
  This function sends bytes to an SPI device.
- uint8_t spi_receive_bytes (uint8_t count, uint8_t *data)
  This function receives bytes from an SPI device.
- void aes132p_enable_interface (void)
  This function initializes and enables the I2C hardware peripheral.
- void aes132p_disable_interface (void)
  This function disables the I2C hardware peripheral.
- uint8_t aes132p_select_device (uint8_t device_id)
  This function selects a I2C AES132 device.
- uint8_t aes132p_read_memory_physical (uint8_t size, uint16_t word_address, uint8_t *data)
  This function reads bytes from the device.
- uint8_t aes132p_write_memory_physical (uint8_t count, uint16_t word_address, uint8_t *data)
  This function writes bytes to the device.
- uint8_t aes132p_resync_physical (void)
  This function resynchronizes communication.

4.7.1 Detailed Description

This file contains definitions of the SPI layer of the AES132 library.

Author

Atmel Crypto Products

Date

June 16, 2011

4.7.2 Macro Definition Documentation

4.7.2.1 #define AES132_STATUS_REG_POLL_TIME (23)

time for polling a bit in the device status register in us (measured)

With an oscilloscope or logic analyzer, measure the time it takes for one loop iteration in aes132c_wait_for_status_bit() and enter it here. Undefine this if you want to use estimated values (see below).

4.7.3 Function Documentation

4.7.3.1 void spi_select_slave (void)

This function selects the SPI slave.

References SPI_CLOCK, and SPI_PORT_OUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.7.3.2 void spi_deselect_slave (void)

This function deselects the SPI slave.

References SPI_CLOCK, and SPI_PORT_OUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.7.3.3 uint8_t spi_select_device (uint8_t *index*)

This function assigns the chip select pin to be used for communication.

```
The same SPI peripheral is used (clock and data lines), but for the
chip select, a different port and / or pin can be assigned.
```

Parameters

index	index into the port and pin arrays

Returns

error if index argument is out of bounds

References AES132_FUNCTION_RETCODE_DEVICE_SELECT_FAIL, SPI_DEVICE_COUNT, and SPI_FUNCTION_RETCODE_SUCCESS.

Referenced by aes132p_select_device().

4.7.3.4 uint8_t spi_send_bytes (uint8_t *count*, uint8_t * *data*)

This function sends bytes to an SPI device.

Parameters

in	*count*	number of bytes to send
in	*data*	pointer to tx buffer

Returns

status of the operation

References SPI_FUNCTION_RETCODE_SUCCESS, SPI_FUNCTION_RETCODE_TIMEOUT, and SPI_TX_TIMEOUT.

Referenced by aes132p_read_memory_physical(), and aes132p_write_memory_physical().

4.7.3.5 uint8_t spi_receive_bytes (uint8_t *count*, uint8_t * *data*)

This function receives bytes from an SPI device.

Parameters

in	*count*	number of bytes to receive
in	*data*	pointer to rx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_TIMEOUT, SPI_FUNCTION_RETCODE_SUCCESS, and SPI_RX_TIMEOUT.

Referenced by aes132p_read_memory_physical().

4.7.3.6 uint8_t aes132p_select_device (uint8_t *device_id*)

This function selects a I2C AES132 device.

Parameters

in	*device_id*	I2C address

Returns

always success

This function selects a I2C AES132 device.

Parameters

in	*device_id*	index into the SPI array for chip select ports and or pins

Returns

success or out-of-bounds error

References AES132_FUNCTION_RETCODE_SUCCESS, and spi_select_device().

4.7.3.7 uint8_t aes132p_read_memory_physical (uint8_t *size*, uint16_t *word_address*, uint8_t * *data*)

This function reads bytes from the device.

Parameters

in	*size*	number of bytes to write
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

Parameters

in	*size*	number of bytes to read
in	*word_address*	word address to read from
out	*data*	pointer to rx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_SPI_PREFACE_SIZE, AES132_SPI_READ, aes132p_send_slave_address(), i2c_receive_bytes(), i2c_send_bytes(), i2c_send_stop(), I2C_WRITE, spi_deselect_slave(), spi_receive_bytes(), spi_select_slave(), and spi_send_bytes().

4.7.3.8 uint8_t aes132p_write_memory_physical (uint8_t *count*, uint16_t *word_address*, uint8_t * *data*)

This function writes bytes to the device.

Parameters

in	*count*	number of bytes to write
in	*word_address*	word address to write to
in	*data*	pointer to tx buffer

Returns

status of the operation

References AES132_FUNCTION_RETCODE_SUCCESS, AES132_SPI_ENABLE_WRITE, AES132_SPI_PREFACE_SIZE, AES132_SPI_WRITE, aes132p_send_slave_address(), i2c_send_bytes(), i2c_send_stop(), I2C_WRITE, spi_deselect_slave(), spi_select_slave(), and spi_send_bytes().

4.7.3.9 uint8_t aes132p_resync_physical (void)

This function resynchronizes communication.

Returns

status of the operation

This function resynchronizes communication.

Returns

always success

References AES132_FUNCTION_RETCODE_SUCCESS, i2c_disable(), i2c_enable(), i2c_send_bytes(), i2c_send_start(), and i2c_send_stop().