

SPI library for AVR uC

Version : 1.0

Generated by Doxygen 1.7.4

Tue Jul 26 2011 21:20:35

## Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>File Documentation</b>	<b>1</b>
2.1	spi.h File Reference . . . . .	1
2.1.1	Detailed Description . . . . .	2
2.1.2	License . . . . .	2
2.1.3	Specifications . . . . .	3
2.1.4	More informations . . . . .	3
2.1.5	Datasheet . . . . .	3
2.1.6	Function Documentation . . . . .	4

## 1 Todo List

**File `spi.h`** Find a way to use multiple peripherals (currently only support one slave)  
Functions for slave mode

## 2 File Documentation

### 2.1 spi.h File Reference

Public SPI functions for AVR uC.

```
#include <avr/io.h>
#include <stdbool.h>
#include "utils/utils.h"
```

#### Defines

##### Bit order, Mode and clock divider

- `#define spi_MSB 0`  
*Most Significant Bit mode.*
- `#define spi_LSB !spi_MSB`  
*Least Significant Bit mode.*
- `#define spi_MODE0 0`

- Mode 0 CPOL=0, CPHA=0.*
- #define `spi_MODE1` 1
- Mode 1 CPOL=0, CPHA=1.*
- #define `spi_MODE2` 2
- Mode 2 CPOL=1, CPHA=0.*
- #define `spi_MODE3` 3
- Mode 3 CPOL=1, CPHA=1.*
- #define `spi_CLOCK_2` 2
- Clock/2.*
- #define `spi_CLOCK_4` 4
- Clock/4.*
- #define `spi_CLOCK_8` 8
- Clock/8.*
- #define `spi_CLOCK_16` 16
- Clock/8.*
- #define `spi_CLOCK_32` 32
- Clock/32.*
- #define `spi_CLOCK_64` 64
- Clock/64.*
- #define `spi_CLOCK_128` 128
- Clock/128.*

## Functions

- void `spi_MasterInit` (uint8\_t data\_order, uint8\_t mode, uint8\_t clock\_rate, uint8\_t interrupt\_enable)  
*Initialize the SPI registers and the SlaveSelect pin.*
- uint8\_t `spi_MasterTransmit` (uint8\_t data)  
*Send a byte and return the received byte.*
- void `spi_SlaveSelect` (uint8\_t action)  
*Enable or disable the SlaveSelect pin.*

### 2.1.1 Detailed Description

Public SPI functions for AVR uC.

### 2.1.2 License

Copyright (c) 2009, 2010, 2011 Patrice Nadeau All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code

must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Patrice Nadeau nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL Patrice Nadeau BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 2.1.3 Specifications

Low-level functions to interface a peripheral with an AVR via the SPI protocol

#### 2.1.3.1 Language

- C (c99)

#### 2.1.3.2 Target

- ATmega48
- ATmega88
- ATmega168 (Tested)
- ATmega328 (Tested)

### 2.1.4 More informations

#### 2.1.5 Datasheet

-ATmega168, section 19

- [http://www.atmel.com/dyn/resources/prod\\_documents/doc8271.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf)

### Author

Patrice Nadeau

email : [patricen@telwarwick.net](mailto:patricen@telwarwick.net)

www : <http://nadeaup.homeip.net:8080/>

### Note

The following SPI port and pins are defined at compile time depending of the AVR used

- **spi\_PORT**
- **spi\_MOSI**
- **spi\_SCK**
- **spi\_SS**

### Todo

Find a way to use multiple peripherals (currently only support one slave)

Functions for slave mode

#### 2.1.6 Function Documentation

2.1.6.1 `void spi_MasterInit ( uint8_t data_order, uint8_t mode, uint8_t clock_rate, uint8_t interrupt_enable )`

Initialize the SPI registers and the SlaveSelect pin.

#### Parameters

in	<i>data_order</i>	Possible values : <ul style="list-style-type: none"><li>• spi_MSB</li><li>• spi_LSB</li></ul>
in	<i>mode</i>	Possible values : <ul style="list-style-type: none"><li>• spi_MODE0</li><li>• spi_MODE1</li><li>• spi_MODE2</li><li>• spi_MODE3</li></ul>

in	<i>clock_rate</i>	Possible values : <ul style="list-style-type: none"> <li>• spi_CLOCK_2</li> <li>• spi_CLOCK_4</li> <li>• spi_CLOCK_8</li> <li>• spi_CLOCK_16</li> <li>• spi_CLOCK_32</li> <li>• spi_CLOCK_64</li> <li>• spi_CLOCK_128</li> </ul>
in	<i>interrupt_ - enable</i>	Possible values : <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>

**Example**

Activate the SPI interface in MSB mode0 with a clock/16 and interrupt disabled

```
spi_MasterInit(spi_MSB, spi_MODE0, spi_CLOCK_16, false);
```

**2.1.6.2 uint8\_t spi\_MasterTransmit ( uint8\_t data )**

Send a byte and return the received byte.

**Parameters**

in	<i>data</i>	A byte to transmit
----	-------------	--------------------

**Returns**

The value received (if any)

**Example**

Transmit a value and read the result

```
uint8_t value;
spi_SlaveSelect(true);
value = spi_MasterTransmit(23);
spi_SlaveSelect(false);
```

**2.1.6.3 void spi\_SlaveSelect ( uint8\_t action )**

Enable or disable the SlaveSelect pin.

**Parameters**

<code>in</code>	<code>action</code>	Possible values : <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>
-----------------	---------------------	------------------------------------------------------------------------------------------

**Example**

Transmit a value and read the result

```
uint8_t value;  
spi_SlaveSelect(true);  
value = spi_MasterTransmit(23);  
spi_SlaveSelect(false);
```

**Note**

Active low

## Index

spi.h, [1](#)  
    spi\_MasterInit, [4](#)  
    spi\_MasterTransmit, [4](#)  
    spi\_SlaveSelect, [5](#)  
spi\_MasterInit  
    spi.h, [4](#)  
spi\_MasterTransmit  
    spi.h, [4](#)  
spi\_SlaveSelect  
    spi.h, [5](#)