



# **DW APB SPI Master Controller User Manual**

---

Version 6283-001 March 2013

## Copyright Notice and Proprietary Information Notice

Copyright © 2013 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

Synopsys, Inc.  
700 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)

# Contents

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| Customer Support.....                                                                  | 4  |
| Accessing SolvNet.....                                                                 | 4  |
| Contacting the Synopsys Technical Support Center.....                                  | 4  |
| 1 Introduction .....                                                                   | 5  |
| Definitions .....                                                                      | 5  |
| 2 DW APB SPI/SSI Driver .....                                                          | 6  |
| About DW APB SPI Driver .....                                                          | 6  |
| Software Architecture.....                                                             | 7  |
| Error Codes.....                                                                       | 8  |
| Functions .....                                                                        | 8  |
| _dw_apb_spi_poll_install().....                                                        | 8  |
| _dw_apb_spi_poll_deinit() .....                                                        | 8  |
| _dw_apb_spi_poll_init() .....                                                          | 9  |
| _dw_apb_spi_poll_read().....                                                           | 9  |
| _dw_apb_spi_poll_write() .....                                                         | 9  |
| _dw_apb_spi_ioctl() .....                                                              | 10 |
| _dw_apb_spi_int_install() .....                                                        | 10 |
| _dw_apb_spi_int_deinit() .....                                                         | 11 |
| _dw_apb_spi_int_init() .....                                                           | 11 |
| _dw_apb_spi_int_read() .....                                                           | 11 |
| _dw_apb_spi_int_write() .....                                                          | 12 |
| _dw_apb_spi_isr() .....                                                                | 12 |
| Data Types.....                                                                        | 12 |
| _bsp_dw_apb_spi_init .....                                                             | 12 |
| I/O Controllers and I/O Structures.....                                                | 13 |
| IO_IOCTL_SPI_GET_BAUD/IO_IOCTL_SPI_SET_BAUD.....                                       | 14 |
| IO_IOCTL_SPI_DEVICE_DISABLE/IO_IOCTL_SPI_DEVICE_ENABLE .....                           | 14 |
| IO_IOCTL_SPI_SET_TRANSFER_MODE/IO_IOCTL_SPI_GET_TRANSFER_MODE .....                    | 14 |
| IO_IOCTL_SPI_GET_TX_STATS/IO_IOCTL_SPI_GET_RX_STATS<br>/IO_IOCTL_SPI_CLEAR_STATS ..... | 15 |
| IO_IOCTL_SPI_FLUSH_OUTPUT .....                                                        | 15 |

IO\_IOCTL\_SPI\_GET\_FRAMESIZE/IO\_IOCTL\_SPI\_SET\_FRAMESIZE..... 15

IO\_IOCTL\_SPI\_CS..... 16

IO\_IOCTL\_SPI\_READ\_WRITE ..... 16

Default Configuration ..... 17

3 Test Application..... 18

*Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.*

---

## Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at <http://solvnet.synopsys.com/>.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click SolvNet Help in the Support Resources section.

---

## Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to <http://solvnet.synopsys.com/> (Synopsys user name and password required), then clicking “Enter a Call to the Support Center.”
- Send an e-mail message to your local support center.
  - E-mail [support\\_center@synopsys.com](mailto:support_center@synopsys.com) from within North America.
  - Find other local support center e-mail addresses at <http://www.synopsys.com/Support/GlobalSupportCenters>.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at <http://www.synopsys.com/Support/GlobalSupportCenters>.

*This document describes the usage of SPI Master Driver for writing applications. It aims to make the development of embedded application easier for developers. This document contains the description of all data types and API functions.*

## Definitions

The following table explains terms associated with this product.

*Table 1 Definitions*

| Term       | Definition                                                                                                                                                                                                                         |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ARC 700    | Family of 32-bit ARC processor cores with MMU support                                                                                                                                                                              |
| AOM        | Automated Overlay Manager                                                                                                                                                                                                          |
| HLA        | High-Level Architecture                                                                                                                                                                                                            |
| MAS        | Micro-Architecture Specification                                                                                                                                                                                                   |
| ARChitect™ | System Configuration tool developed by ARC. ARChitect can be run from the command line, or as a GUI to provide point-and-click definition of a processor architecture, and generates Verilog source code of the configured design. |
| HDL        | Hardware Description Language — a programming language used to describe the hardware implementation of a system. HDL is usually synthesized to produce a chip (ASIC). Two common HDLs are Verilog and VHDL.                        |
| MetaWare   | C/C++ development toolkit for ARC processors                                                                                                                                                                                       |
| ML509      | Xilinx FPGA development platform used as reference platform by ARC.                                                                                                                                                                |
| MQX        | RTOS supplied by ARC                                                                                                                                                                                                               |

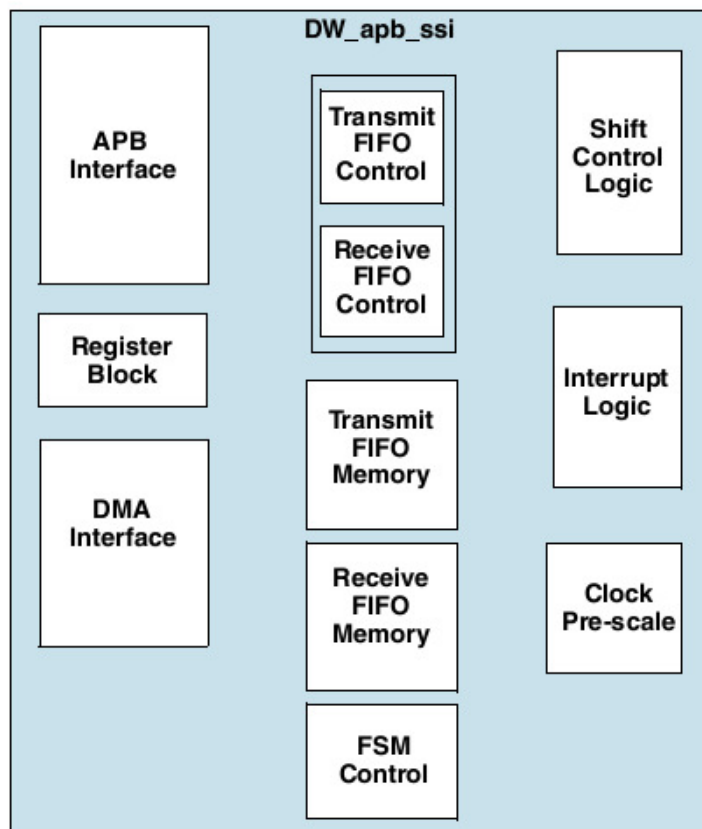
*This chapter describes the functionality and the APIs exposed by the DW APB SPI MQX Master Driver.*

## About the DW APB SPI Driver

The DW APB SPI is a configurable and programmable control bus between integrated circuits in a system. SPI is a serial communication device. It works in a master-slave configuration. The SPI operating frequency is 1MHz – 25MHz. Some examples of slave devices are Flash, ADC, DAC and SD cards.

The following is the block diagram of the DW APB SPI Controller.

*Figure 1 SPI IO Controller Block Diagram*



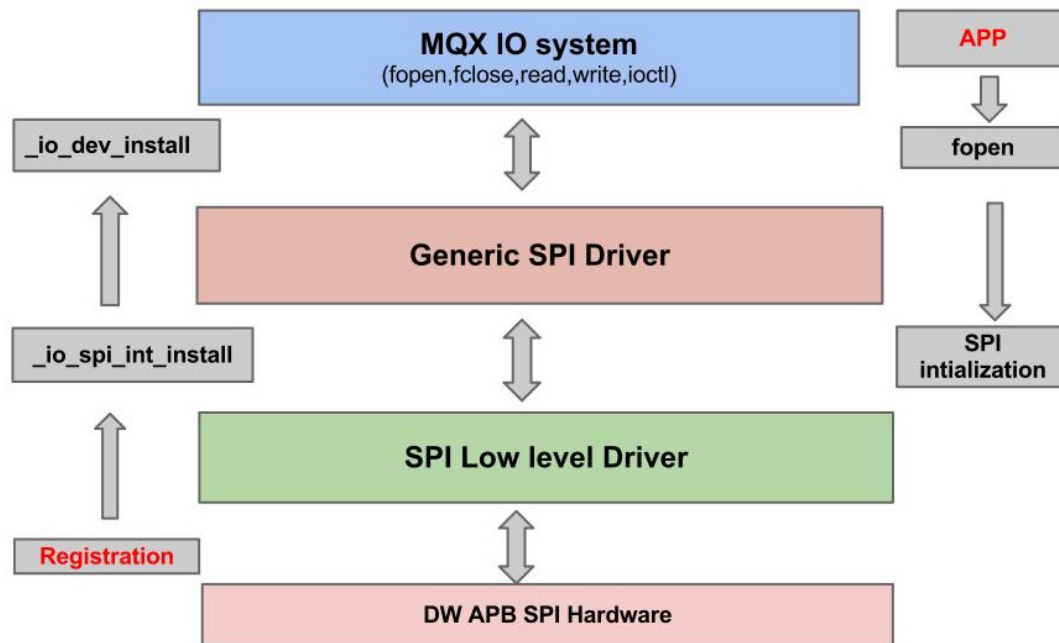
## Software Architecture

The DW APB driver is written using MQX I/O subsystem-defined APIs. The low-level SPI driver registers with the MQX I/O subsystem by providing some callback functions defined by the MQX I/O subsystem. Driver registration is done by calling `_io_dev_install(device name, ..., )`. This call creates an entry for the SPI device instance in the MQX I/O subsystem.

To get the handle of the SPI device, the application should use the same device name used by the driver to install the SPI driver .

Following is the block diagram of the MQX software architecture.

Figure 2 Software Architecture



## Error Codes

The DW APB SPI driver returns standard MQX-defined error codes for each error condition handled in the driver.

The following are the standard MQX error codes used in the driver.

Table 2 Error Codes

| MQX Error Code             | Description              |
|----------------------------|--------------------------|
| MQX_OUT_OF_MEMORY          | Memory allocation failed |
| IO_ERROR_INVALID_IOCTL_CMD | Invalid IOCTL            |

## Functions

### \_dw\_apb\_spi\_poll\_install()

**Description** This function installs the driver's polling mode. It can be called either from the BSP or from the application.

**Syntax**

```
_dw_apb_spi_poll_install(BSP_DW_APB_SPI_POL_NAME,
(pointer) (&(_bsp_dw_apb_spi_init)));
```

**Parameters**

BSP\_DW\_APB\_SPI\_POL\_NAME

This variable is a `char_ptr`, input parameter and holds a string that identifies the device during **fopen()**.

`_bsp_dw_apb_spi_init`

This variable is a pointer input parameter that provides the default configuration of the device.

### \_dw\_apb\_spi\_poll\_deinit()

**Description** This function de-initializes the SPI Controller and frees its memory.

**Syntax**

```
_dw_apb_spi_poll_deinit(DW_APB_SPI_INFO_STRUCT_PTR pdata);
```

**Parameters** `DW_APB_SPI_INFO_STRUCT_PTR`  
 This variable points to the low-level driver structure.

---

### **`_dw_apb_spi_poll_init()`**

**Description** This function initializes the SPI controller and allocates memory to the low-level data structure.

**Syntax** `_dw_apb_spi_poll_init(IO_SPI_POLL_DEVICE_STRUCT_PTR host);`

**Parameters** `IO_SPI_POLL_DEVICE_STRUCT_PTR`  
 This variable points to the `IO_SPI` driver structure.

---

### **`_dw_apb_spi_poll_read()`**

**Description** This function reads data from the FIFO. Use this function when the SPI controller is configured in receive-only mode. You can configure the transfer mode using the `IO_IOCTL_SPI_SET_TRANSFER_MODE` `IOCTL`.

**Syntax** `_dw_apb_spi_poll_read(DW_APB_SPI_INFO_STRUCT_PTR host Char_ptr buf, uint_32 len);`

**Parameters** `DW_APB_SPI_INFO_STRUCT_PTR`  
 This variable points to the low-level driver structure.

`Buf`  
 This variable points to the memory location of the data to read.

`Len`  
 This variable points to the number of bytes to be received.

---

### **`_dw_apb_spi_poll_write()`**

**Description** This function writes data from the FIFO. Use this function when the SPI Controller is configured in transmit-only mode. You can configure the transfer mode using the `IO_IOCTL_SPI_SET_TRANSFER_MODE` `IOCTL`.

**Syntax** `_dw_apb_spi_poll_write(DW_APB_SPI_INFO_STRUCT_PTR host char_ptr buf, uint_32 len);`

**Parameters** `DW_APB_SPI_INFO_STRUCT_PTR`  
 This variable points to the low-level driver structure.

`Buf`  
 This variable points to the memory location of the data to be transmitted.

`Len`  
 This variable points to the number of bytes to be transmitted.

---

## **`_dw_apb_spi_ioctl()`**

**Description** This function performs operations specific to the controller. Such as getting and setting baud, transfer types, data-frame size, and so on.

**Syntax**

```
_dw_apb_spi_ioctl(DW_APB_SPI_INFO_STRUCT_PTR host _mqx_uint  
cmd, pointer arg);
```

**Parameters** `DW_APB_SPI_INFO_STRUCT_PTR`  
 This variable points to the low-level driver structure.

`cmd`  
 Any define that is described in the I/O controls and structures can be used. The function behavior and type of `param_ptr` is determined according to this parameter.

`param_ptr`  
 This parameter is taken as a `void` pointer and type casted to one of the control data structures depending upon the command type. The type of `param_ptr` is determined according to the command value.

---

## **`_dw_apb_spi_int_install()`**

**Description** This function installs interrupt mode of driver, it can be called either from BSP or from an application.

**Syntax**

```
_dw_apb_spi_int_install(BSP_DW_APB_SPI_INT_NAME,  
(pointer) (&(_bsp_dw_apb_spi_init)));
```

**Parameters** `BSP_DW_APB_SPI_INT_NAME`  
 This variable is `char_ptr`, it is an input parameter to `_dw_apb_spi_int_install` function and it holds string that identifies the device during open.

`_bsp_dw_apb_spi_init`  
 This variable is a pointer, it is an input parameter to `_dw_apb_spi_int_install`

function. Default configuration of the device is provided through this parameter.

---

### **`_dw_apb_spi_int_deinit()`**

**Description** This function de-initializes the SPI controller, frees its memory, and de-registers the ISR handler.

**Syntax** `_dw_apb_spi_int_deinit (DW_APB_SPI_INFO_STRUCT_PTR pdata);`

**Parameters** DW\_APB\_SPI\_INFO\_STRUCT\_PTR  
This variable points to the low-level driver structure.

---

### **`_dw_apb_spi_int_init()`**

**Description** This function Initializes the SPI controller and allocates memory to the low-level data structure. Also it installs the ISR handler.

**Syntax** `_dw_apb_spi_int_init (IO_SPI_INT_DEVICE_STRUCT_PTR host);`

**Parameters** IO\_SPI\_INT\_DEVICE\_STRUCT\_PTR  
This variable points to the IO\_SPI driver structure.

---

### **`_dw_apb_spi_int_read()`**

**Description** This function reads data from the device's FIFO. Use this function when the SPI controller is configured in receive-only mode. You can configure the transfer mode using the `IO_IOCTL_SPI_SET_TRANSFER_MODE` IOCTL.

**Syntax** `_dw_apb_spi_int_read (DW_APB_SPI_INFO_STRUCT_PTR host char_ptr buf, uint_32 len);`

**Parameters** DW\_APB\_SPI\_INFO\_STRUCT\_PTR  
This variable points to the low-level driver structure.

Buf  
This variable points to the memory location where the read data is to be put.

Len  
This variable points to the number of bytes to be received.

---

## **`_dw_apb_spi_int_write()`**

**Description** This function writes data from the device's FIFO. Use this function when the SPI controller is configured in transmit-only mode. You can configure the transfer mode using the `IO_IOCTL_SPI_SET_TRANSFER_MODE` IOCTL.

**Syntax** `_dw_apb_spi_int_write(DW_APB_SPI_INFO_STRUCT_PTR host char_ptr buf, uint_32 len);`

**Parameters** `DW_APB_SPI_INFO_STRUCT_PTR`

This variable points to the low-level driver structure.

`Buf`

This variable points to the memory location where the data to be transmitted is present .

`Len`

This variable points to the number of bytes to be transmitted.

---

## **`_dw_apb_spi_isr()`**

**Description** Interrupt handler for serial I/O interrupts.

**Syntax** `void _dw_apb_spi_isr(pointer parameter);`

**Parameters** `parameter`

This argument is a `void` pointer input argument to the interrupt handler. It holds the address of interrupt I/O initialization information (`IO_SPI_INT_DEVICE_STRUCT`) .

---

## **Data Types**

---

### **`_bsp_dw_apb_spi_init`**

```
typedef struct dw_apb_spi_init {
    char_ptr  NAME;           /* Name of the SPI
Controller */
    uint_32   BASE_ADDR;     /* Base Address of the SPI
Controller */
    uint_32   VECTOR;        /* Interrupt Vector Number
```

```

        */
        uint_32    OPERATING_FREQ;    /* Operating Frequency
        */
    } DW_APB_SPI_INIT_STRUCT, _PTR_ DW_APB_SPI_INIT_STRUCT_PTR;

```

Table 3 Data Types

| Field Name          | Description                                                                                     |
|---------------------|-------------------------------------------------------------------------------------------------|
| NAME                | Name given to the driver. Two names exist, one for interrupt mode and another for polling mode. |
| BASE_ADDR           | The base address of the SPI controller.                                                         |
| OPERATING_FREQUENCY | The frequency at which the SPI controller should be operated.                                   |
| VECTOR              | Interrupt vector number for this SPI.                                                           |

## I/O Controllers and I/O Structures

IOCTLS are used to interact with the device. They are used to check its status and to change any settings if required.

The following are the IOCTLS that are implemented in the DW SPI Controller Driver.

- IO\_IOCTL\_SPI\_GET\_BAUD
- IO\_IOCTL\_SPI\_SET\_BAUD
- IO\_IOCTL\_SPI\_DEVICE\_DISABLE
- IO\_IOCTL\_SPI\_DEVICE\_ENABLE
- IO\_IOCTL\_SPI\_SET\_TRANSFER\_MODE
- IO\_IOCTL\_SPI\_GET\_TRANSFER\_MODE
- IO\_IOCTL\_SPI\_GET\_TX\_STATS
- IO\_IOCTL\_SPI\_GET\_RX\_STATS
- IO\_IOCTL\_SPI\_CLEAR\_STATS
- IO\_IOCTL\_SPI\_FLUSH\_OUTPUT
- IO\_IOCTL\_SPI\_GET\_FRAME\_SIZE
- IO\_IOCTL\_SPI\_SET\_FRAME\_SIZE
- IO\_IOCTL\_SPI\_CS
- IO\_IOCTL\_SPI\_READ\_WRITE

## IO\_IOCTL\_SPI\_GET\_BAUD/IO\_IOCTL\_SPI\_SET\_BAUD

This IOCTL is used to get and set the current baud rate of the driver.

```
uint_32_ptr params
```

Table 4 IO\_IOCTL\_SPI\_GET\_BAUD/IO\_IOCTL\_SPI\_SET\_BAUD I/O Control Field Descriptions

| Field Name | Description                                                                                        |
|------------|----------------------------------------------------------------------------------------------------|
| params     | This field is used to pass the baud rate as an argument; any valid baud can be passed as argument. |

## IO\_IOCTL\_SPI\_DEVICE\_DISABLE/IO\_IOCTL\_SPI\_DEVICE\_ENABLE

This IOCTL is used to disable and enable the SPI controller

```
uint_32_ptr params
```

Table 5 IO\_IOCTL\_SPI\_DEVICE\_DISABLE/IO\_IOCTL\_SPI\_DEVICE\_ENABLE I/O Control Field Descriptions

| Field Name | Description                                                                                                                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| params     | <p>This field allows the user to enable or disable the controller. The following are the possible values that can be passed arguments:</p> <pre>#define SSI_ENABLE (1)</pre> <pre>#define SSI_DISABLE (0)</pre> |

## IO\_IOCTL\_SPI\_SET\_TRANSFER\_MODE/IO\_IOCTL\_SPI\_GET\_TRANSFER\_MODE

This IOCTL is used to set and get the SPI controller transfer mode.

```
uint_32_ptr params
```

Table 6 IO\_IOCTL\_SPI\_SET\_TRANSFER\_MODE/IO\_IOCTL\_SPI\_GET\_TRANSFER\_MODE I/O Control Field Descriptions

| Field Name | Description                                                                                                                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| params     | <p>This field contains the information about the transfer mode. Possible values that can be passed as argument are</p> <pre>#define SPI_TRANSMIT_RECEIVE_MODE (0x0)</pre> <pre>#define SPI_TRANSMIT_ONLY_MODE (0x1)</pre> <pre>#define SPI_RECEIVE_ONLY_MODE (0x2)</pre> |

```
#define SPI_EEPROM_READ_ONLY_MODE (0x3)
```

## IO\_IOCTL\_SPI\_GET\_TX\_STATS / IO\_IOCTL\_SPI\_GET\_RX\_STATS / IO\_IOCTL\_SPI\_CLEAR\_STATS

This IOCTL is used to get and clear the SPI controller's TX and RX error states.

```
uint_32_ptr params
```

*Table 7 IO\_IOCTL\_SPI\_GET\_TX\_STATS / IO\_IOCTL\_SPI\_GET\_RX\_STATS / IO\_IOCTL\_SPI\_CLEAR\_STATS Control Field Descriptions*

| Field Name | Description                                                                                                                                                                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Params     | <p>This field contains the information about the required status (TX or RX status). The following are possible values that can be passed as arguments:</p> <pre>#define TX_OVERFLOW_ERROR (0x2)</pre> <pre>#define RX_UNDERFLOW_ERROR (0x4)</pre> <pre>#define RX_OVERFLOW_ERROR (0x8)</pre> |

## IO\_IOCTL\_SPI\_FLUSH\_OUTPUT

This IOCTL is used to clear the SPI Controller TX FIFO.

## IO\_IOCTL\_SPI\_GET\_FRAME\_SIZE / IO\_IOCTL\_SPI\_SET\_FRAME\_SIZE

This IOCTL is used to set and get the SPI controller's data-frame size.

```
uint_32_ptr params
```

*Table 8 IO\_IOCTL\_SPI\_GET\_FRAME\_SIZE / IO\_IOCTL\_SPI\_SET\_FRAME\_SIZE I/O Control Field Descriptions*

| Field Name | Description                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| params     | <p>This field is used to pass the associated value to get or set the frame size. The following are possible values that can be passed as an argument:</p> <pre>#define CTRLR0_DFS_4 (0x3)</pre> <pre>#define CTRLR0_DFS_5 (0x4)</pre> <pre>#define CTRLR0_DFS_6 (0x5)</pre> <pre>#define CTRLR0_DFS_7 (0x6)</pre> <pre>#define CTRLR0_DFS_8 (0x7)</pre> |

|                       |       |
|-----------------------|-------|
| #define CTRLR0_DFS_9  | (0x8) |
| #define CTRLR0_DFS_10 | (0x9) |
| #define CTRLR0_DFS_11 | (0xA) |
| #define CTRLR0_DFS_12 | (0xB) |
| #define CTRLR0_DFS_13 | (0xC) |
| #define CTRLR0_DFS_14 | (0xD) |
| #define CTRLR0_DFS_15 | (0xE) |
| #define CTRLR0_DFS_16 | (0xF) |

## IO\_IOCTL\_SPI\_CS

This IOCTL is used to set and select the SPI slave devices.

```
uint_32_ptr params
```

Table 9 IO\_IOCTL\_SPI\_CS I/O Control Field Descriptions

| Name   | Description                                                                                                                                                                                    |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Params | <p>This field is used to pass information about whether to select or deselect the chip.</p> <p>In order to select it, send param with 1</p> <p>In order to de-select it, send param with 0</p> |

## IO\_IOCTL\_SPI\_READ\_WRITE

```
typedef struct dw_apb_spi_read_write {
    char_ptr TX_BUF;
    char_ptr RX_BUF;
    uint_32  LEN;
    uint_32  TX_LEN;
    uint_32  RX_LEN;
} DW_APB_SPI_READ_WRITE, _PTR_ DW_APB_SPI_READ_WRITE_PTR;
```

*Table 10 IO\_IOCTL\_SPI\_READ\_WRITE I/O Control Field Descriptions*

| Field Name | Description                                                                    |
|------------|--------------------------------------------------------------------------------|
| TX_BUF     | Pointer to Transmit buffer.                                                    |
| RX_BUF     | Pointer to Receive buffer.                                                     |
| LEN        | Total count when the SPI controller is configured in combined write/read mode. |
| TX_LEN     | TX count when the SPI controller is configured in EEPROM mode.                 |
| RX_LEN     | RX count when the SPI controller is configured in EEPROM mode.                 |

---

## Default Configuration

The default value with which the DW APB SPI is configured during the initialization is explained in the table below. These values will be passed to the driver through **dw\_apb\_spi\_init()**.

# 3

## Test Application

---

For a test application, please refer to the SPI application provided with the SPI driver. SPI master controller communicates with SPI SD card slave device and performs the following operations:

- Checks for the supported voltage ranges of the SD card
- Starts and finishes the initialization process
- Determines the capacity of the SD card. (SDSD, SDHC)
- Gets the CID and CSD of the SD card
- Sector write of 512 bytes
- Sector read of 512 bytes