

# axis\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/22

---

## INFORMATION

---

### Brief

---

Stream SPI input/output data over AXIS bus.

### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## axis\_spi\_master

---

```
module axis_spi_master #(
    parameter
    CLOCK_SPEED
    =
    20000000,
    parameter
    BUS_WIDTH
    =
    4,
    parameter
    SELECT_WIDTH
    =
    8
) ( input aclk, input arstn, input [BUS_WIDTH*8-1:0] s_axis_tdata, input s_e
```

---

SPI core with axis input/output data. Read/Write is size of BUS\_WIDTH bytes. Write activates core for read.

### Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz, this is the the frequency used for the bus and is divided by the rate.
<b>BUS_WIDTH</b> parameter	AXIS data width in bytes.
<b>SELECT_WIDTH</b> parameter	Bit width of the slave select.

### Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>s_axis_tdata</b>	Input data for SPI MOSI.
<b>s_axis_tvalid</b>	When set active high the input data is valid
<b>s_axis_tready</b>	When active high the device is ready for input data.
<b>m_axis_tdata</b>	Output data from SPI MISO
<b>m_axis_tvalid</b>	When active high the output data is valid
<b>m_axis_tready</b>	When set active high the output device is ready for data.
<b>sclk</b>	spi clock, should only drive output pins to devices.
<b>mosi</b>	transmit for master output
<b>miso</b>	receive for master input
<b>ssn_i</b>	slave select input
<b>ssn_o</b>	slave select output
<b>rate</b>	output rate of spi core.
<b>cpol</b>	clock polarity of sclk
<b>cpha</b>	clock phase of sclk
<b>miso_dcount</b>	Current number of input bits available from parallel register.
<b>mosi_dcount</b>	current number of output bits available to serial shift output.

## STATE MACHINE

---

Constants that makeup the data\_state machine.

### ready

---

```
localparam ready = 3'd1
```

ready and waiting for data

### processing

---

```
localparam processing = 3'd3
```

data is being processed

## error

---

```
localparam error = 3'd0
```

someone made a whoops

## INSTANTIATED MODULES

---

### inst\_spi\_output\_clk

---

```
mod_clock_ena_gen #(
    .CLOCK_SPEED(CLOCK_SPEED)
) inst_spi_output_clk ( .clk(aclk), .rstn(arstn), .start0(1'b0), .clr(spi_en
```

Generates enable at rate for spi output data.

### inst\_spi\_input\_clk

---

```
mod_clock_ena_gen #(
    .CLOCK_SPEED(CLOCK_SPEED)
) inst_spi_input_clk ( .clk(aclk), .rstn(arstn), .start0(1'b1), .clr(spi_en
```

Generates enable at rate for spi input data.

### inst\_piso

---

```
piso #(
    .BUS_WIDTH(BUS_WIDTH)
) inst_piso ( .clk(aclk), .rstn(arstn), .ena(spi_ena_mosi), .load(spi_mosi
```

take axis input parallel data at bus size, and output the word to the spi bus.

### inst\_sipo

---

```
sipo #(
    .BUS_WIDTH(BUS_WIDTH)
) inst_sipo ( .clk(aclk), .rstn(arstn), .ena(spi_ena_miso), .load(spi_miso
```

take serial input data, and output the world to the parallel data bus.