

# wishbone\_standard\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## INFORMATION

---

### Brief

---

Wishbone Standard SPI Master core.

### 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.

## wishbone\_standard\_spi\_master

---

```
module wishbone_standard_spi_master #(
    parameter
    ADDRESS_WIDTH
    =
    32,
    parameter
    BUS_WIDTH
    =
    4,
    parameter
    CLOCK_SPEED
    =
    100000000,
    parameter
```

```

SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input clk, input rst, input s_wb_cyc, input s_wb_stb, input s_wb_we, input s_wb_addr, input s_wb_data_i, input s_wb_data_o, input s_wb_err, input irq, input sclk, input mosi, input miso, input ss_n

```

Wishbone Standard based SPI Master device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port(can not be less than 2 bytes, max tested is 4).
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 ( $2^{(X+1)}$ X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rst</b>	Positive reset
<b>s_wb_cyc</b>	Bus Cycle in process
<b>s_wb_stb</b>	Valid data transfer cycle
<b>s_wb_we</b>	Active High write, low read
<b>s_wb_addr</b>	Bus address
<b>s_wb_data_i</b>	Input data
<b>s_wb_sel</b>	Device Select
<b>s_wb_ack</b>	Bus transaction terminated
<b>s_wb_data_o</b>	Output data
<b>s_wb_err</b>	Active high when a bus error is present
<b>irq</b>	Interrupt when data is received
<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>ss_n</b>	slave select output

## up\_rreq

---

```
wire up_rreq
```

uP read bus request

## up\_rack

---

```
wire up_rack
```

uP read bus acknowledge

## up\_raddr

---

```
wire [ADDRESS_WIDTH-(  
BUS_WIDTH  
2  
)-1:0] up_raddr /
```

uP read bus address

## up\_rdata

---

```
wire [31:0] up_rdata
```

uP read bus request

## up\_wreq

---

```
wire up_wreq
```

uP write bus request

## up\_wack

---

```
wire up_wack
```

uP write bus acknowledge

## up\_waddr

---

```
wire [ADDRESS_WIDTH-(  
BUS_WIDTH  
2  
)-1:0] up_waddr /
```

uP write bus address

## up\_wdata

---

```
wire [31:0] up_wdata
```

uP write bus data

## INSTANTIATED MODULES

---

### inst\_up\_wishbone\_standard

---

```
up_wishbone_standard #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH)
) inst_up_wishbone_standard ( .clk(clk), .rst(rst), .s_wb_cyc(s_wb_cyc), .s_
```

Module instance of up\_wishbone\_standard for the Wishbone Classic Standard bus to the uP bus.

### inst\_up\_spi\_master

---

```
up_spi_master #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH),
    CLOCK_SPEED(CLOCK_SPEED),
    SELECT_WIDTH(SELECT_WIDTH),
    DEFAULT_RATE_DIV(DEFAULT_RATE_DIV),
    DEFAULT_CPOL(DEFAULT_CPOL),
    DEFAULT_CPHA(DEFAULT_CPHA)
) inst_up_spi_master ( .clk(clk), .rstn(~rst), .up_rreq(up_rreq), .up_rack(u
```

Module instance of up\_spi\_master creating a Logic wrapper for spi master axis bus cores to interface with uP bus.