

# wishbone\_classic\_1553.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/10/17

---

## INFORMATION

---

### Brief

---

wishbone classic to uP core for 1553 comms.

### License MIT

---

Copyright 2024 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\_1553

---

```
module wishbone_standard_1553 #(
  parameter
  ADDRESS_WIDTH
  =
  32,
  parameter
  BUS_WIDTH
  =
  4,
  parameter
  CLOCK_SPEED
  =
  100000000
)
```

```

input
clk,
input
rst,
input
s_wb_cyc,
input
s_wb_stb,
input
s_wb_we,
input
[ADDRESS_WIDTH-1:0]
s_wb_addr,
input
[BUS_WIDTH*8-1:0]
s_wb_data_i,
input
[BUS_WIDTH-1:0]
s_wb_sel,
output
s_wb_ack,
output
[BUS_WIDTH*8-1:0]
s_wb_data_o,
output
s_wb_err,
input
[1:0]
rx_diff,
output
[1:0]
tx_diff,
output
tx_active,
output
irq
)

```

Wishbone Stanard based 1553 communications device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the address bus in bits, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the data bus in bytes.
<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz

## Ports

<b>clk</b> input	Clock for all devices in the core
<b>rst</b> input	Positive reset
<b>s_wb_cyc</b> input	Bus Cycle in process
<b>s_wb_stb</b> input	Valid data transfer cycle
<b>s_wb_we</b> input	Active High write, low read

<b>s_wb_addr</b> <code>input [ADDRESS_WIDTH- 1:0]</code>	Bus address
<b>s_wb_data_i</b> <code>input [BUS_WIDTH* 8- 1:0]</code>	Input data
<b>s_wb_sel</b> <code>input [BUS_WIDTH- 1:0]</code>	Device Select
<b>s_wb_ack</b> <code>output [BUS_WIDTH- 1:0]</code>	Bus transaction terminated
<b>s_wb_data_o</b> <code>output [BUS_WIDTH* 8- 1:0]</code>	Output data
<b>s_wb_err</b> <code>output [BUS_WIDTH* 8- 1:0]</code>	Active high when a bus error is present
<b>rx_diff</b> <code>input [1:0]</code>	Input differential signal for 1553 bus
<b>tx_diff</b> <code>output [1:0]</code>	Output differential signal for 1553 bus
<b>tx_active</b> <code>output [1:0]</code>	Enable output of differential signal (for signal switching on 1553 module)
<b>irq</b> <code>output [1:0]</code>	Interrupt when data is received

## 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

---

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

### inst\_up\_1553

---

Module instance of up\_1553 creating a Logic wrapper for 1553 bus cores to interface with uP bus.