

# 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\_classic\_1553

---

```
module wishbone_classic_1553 #(
    parameter
    ADDRESS_WIDTH
    =
    32,
    parameter
    BUS_WIDTH
    =
    4,
    parameter
    CLOCK_SPEED
```

```

=
100000000,
parameter
SAMPLE_RATE
=
2000000,
parameter
BIT_SLICE_OFFSET
=
0,
parameter
INVERT_DATA
=
0,
parameter
SAMPLE_SELECT
=
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_sel, input s_wb_bte, input s_wb_cti, input s_wb_ack, input s_wb_data_o, input s_wb_err )

```

Wishbone Calssic based 1553 communications device.

## Parameters

|                                   |  |
|-----------------------------------|--|
| <b>ADDRESS_WIDTH</b><br>parameter | Width of the address bus in bits   |
| <b>BUS_WIDTH</b><br>parameter     | Width of the data bus in bytes.  |
| <b>CLOCK_SPEED</b><br>parameter   | This is the aclk frequency in Hz   |
| <b>SAMPLE_RATE</b><br>parameter   | Rate of in which to sample the 1553 bus. Must be 2 MHz or more and less than aclk. This is in Hz. BIT_SLICE_OFFSET- Adjust where the sample is taken from the input. |
| <b>INVERT_DATA</b><br>parameter   | Invert all 1553 bits coming in and out.  |
| <b>SAMPLE_SELECT</b><br>parameter | Adjust where in the array of samples to select a bit.  |

## 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_bte</b>    | Burst Type Extension                    |
| <b>s_wb_cti</b>    | Cycle Type                              |
| <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>i_diff</b>    | Input differential signal for 1553 bus                                     |
| <b>o_diff</b>    | Output differential signal for 1553 bus                                    |
| <b>en_o_diff</b> | Enable output of differential signal (for signal switching on 1553 module) |
| <b>irq</b>       | 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-3:0] 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-3:0] up_waddr
```

uP write bus address

## up\_wdata

---

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

uP write bus data

## INSTANTIATED MODULES

---

### inst\_up\_wishbone\_classic

---

```
up_wishbone_classic #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH)
) inst_up_wishbone_classic ( .clk(clk), .rst(rst), .s_wb_cyc(s_wb_cyc), .s_v
```

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

### inst\_up\_1553

---

```
up_1553 #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH),
    CLOCK_SPEED(CLOCK_SPEED),
    SAMPLE_RATE(SAMPLE_RATE),
    BIT_SLICE_OFFSET(BIT_SLICE_OFFSET),
    INVERT_DATA(INVERT_DATA),
    SAMPLE_SELECT(SAMPLE_SELECT)
) inst_up_1553 ( .clk(aclk), .rstn(arstn), .up_rreq(up_rreq), .up_rack(up_rack)
```

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