

# axis\_1553.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/06/24

---

## INFORMATION

---

### Brief

---

AXIS 1553 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.

## axis\_1553

---

```
module axis_1553 #(
    parameter
    CLOCK_SPEED
    =
    20000000,
    parameter
    RX_BAUD_DELAY
    =
    0,
    parameter
    TX_BAUD_DELAY
    =
    0
) ( input wire aclk, input wire arstn, output wire parity_err, output wire t
```

---

AXIS 1553, simple core for encoding and decoding 1553 bus messages.

## Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz
<b>RX_BAUD_DELAY</b> parameter	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).
<b>TX_BAUD_DELAY</b> parameter	Delay in tx baud enable. This will delay the time the bit output starts.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>parity_err</b>	Indicates error with parity check for receive (active high)
<b>frame_err</b>	Indicates the diff line went to no diff before data capture finished.
<b>s_axis_tdata</b>	Input data for UART TX.
<b>s_axis_tuser</b>	Information about the AXIS data {S,D,TYY} (4:0) Bits explained below:

```
- S = SYNC ONLY (4)
  - 1 = Send only a sync pulse specified by TYY
  - 0 = Send normal sync + data.
- D = DELAY ENABLED (3)
  - 1 = Make sure there is a delay of 4us
  - 0 = Send out immediately
- TYY = TYPE OF DATA (2:0)
  - 000 = NA
  - 001 = REG (NOT IMPLIMENTED)
  - 010 = DATA
  - 100 = CMD/STATUS
```

s\_axis\_tvalid - When set active high the input data is valid s\_axis\_tready - When active high the device is ready for input data. m\_axis\_tdata - Output data from UART RX m\_axis\_tuser - Information about the AXIS data {S,D,TYY} (4:0)

Bits explained below:

```
- S = SYNC ONLY (4)
  - 1 = Only received a sync pulse specified by TYY
  - 0 = Normal sync + data received.
- D = DELAY BEFORE DATA (3)
  - 1 = Delay of 4us or more before data
  - 0 = No delay between data
- TYY = TYPE OF DATA (2:0)
  - 000 NA
  - 001 REG (NOT IMPLIMENTED)
  - 010 DATA
  - 100 CMD/STATUS
```

m\_axis\_tvalid - When active high the output data is valid m\_axis\_tready - When set active high the output device is ready for data. tx\_active - Active high indicates transmit is in progress. tx\_diff - transmit for 1553 (output to RX) rx\_diff - receive for 1553 (input from TX)

## BASE\_1553\_CLOCK\_RATE

---

```
localparam integer BASE_1553_CLOCK_RATE = 1000000
```

1553 base clock rate

## BASE\_1553\_SAMPLE\_RATE

---

```
localparam integer BASE_1553_SAMPLE_RATE = 2000000
```

Sample rate to use for the 1553 bus, set to 2 MHz

## SAMPLES\_PER\_MHZ

---

```
localparam integer SAMPLES_PER_MHZ = BASE_1553_SAMPLE_RATE /  
    BASE_1553_CLOCK_RATE
```

sample rate of 2 MHz to capture transmission bits at

## cycles\_per\_mhz

---

```
localparam integer CYCLES_PER_MHZ = CLOCK_SPEED / BASE_1553_CLOCK_RATE
```

calculate the number of cycles the clock changes per period

## BIT\_RATE\_PER\_MHZ

---

```
localparam integer BIT_RATE_PER_MHZ = SAMPLES_PER_MHZ
```

bit rate per mhz

## DELAY\_TIME

---

```
localparam integer DELAY_TIME = CYCLES_PER_MHZ * 4
```

delay time, 4 is for 4 us (min 1553 time)

## SYNC\_BITS\_PER\_TRANS

---

```
localparam integer SYNC_BITS_PER_TRANS = 3
```

sync bits per transmission

## SYNTH\_SYNC\_BITS\_PER\_TRANS

---

```
localparam integer SYNTH_SYNC_BITS_PER_TRANS = SYNC_BITS_PER_TRANS *  
    BIT_RATE_PER_MHZ
```

sync pulse length

## PARITY\_BITS\_PER\_TRANS

---

```
localparam integer PARITY_BITS_PER_TRANS = 1
```

parity bits per transmission

## SYNTH\_PARITY\_BITS\_PER\_TRANS

---

```
localparam integer SYNTH_PARITY_BITS_PER_TRANS = PARITY_BITS_PER_TRANS *  
    BIT_RATE_PER_MHZ
```

synth parity bits per transmission

## DATA\_BITS\_PER\_TRANS

---

```
localparam integer DATA_BITS_PER_TRANS = 16
```

data bits per transmission

## SYNTH\_DATA\_BITS\_PER\_TRANS

---

```
localparam integer SYNTH_DATA_BITS_PER_TRANS = DATA_BITS_PER_TRANS *  
    BIT_RATE_PER_MHZ
```

synth data bits per transmission

## BITS\_PER\_TRANS

---

```
localparam integer BITS_PER_TRANS = DATA_BITS_PER_TRANS +  
    PARITY_BITS_PER_TRANS
```

non sync bits per transmission

## TOTAL\_BITS\_PER\_TRANS

---

```
localparam integer TOTAL_BITS_PER_TRANS = DATA_BITS_PER_TRANS +  
    PARITY_BITS_PER_TRANS + SYNC_BITS_PER_TRANS
```

bits per transmission with sync

## SYNTH\_BITS\_PER\_TRANS

---

```
localparam integer SYNTH_BITS_PER_TRANS = SYNTH_DATA_BITS_PER_TRANS +  
    SYNTH_PARITY_BITS_PER_TRANS
```

synth bits per trans without sync

---

## TOTAL\_SYNTH\_BITS\_PER\_TRANS

```
localparam integer TOTAL_SYNTH_BITS_PER_TRANS =  
    SYNTH_DATA_BITS_PER_TRANS + SYNTH_SYNC_BITS_PER_TRANS +  
    SYNTH_PARITY_BITS_PER_TRANS
```

synth bits per trans with sync

---

## TOTAL\_SYNTH\_BYTES\_PER\_TRANS

```
localparam integer TOTAL_SYNTH_BYTES_PER_TRANS =  
    TOTAL_SYNTH_BITS_PER_TRANS/8
```

sync bits per trans with sync

---

## BIT\_PATTERN

```
localparam [(  
    BIT_RATE_PER_MHZ  
)-1:0]BIT_PATTERN = {{BIT_RATE_PER_MHZ/2{1'b1}}, {BIT_RATE_PER_MHZ/2{1'b0}}}
```

create the bit pattern. This is based on outputting data on the negative and positive. This allows the encoder to run down to 1 mhz.

---

## SYNTH\_CLK

```
localparam [SYNTH_DATA_BITS_PER_TRANS-1:0]SYNTH_CLK = {  
    DATA_BITS_PER_TRANS{BIT_PATTERN}  
}
```

synth clock is the clock constructed by the repeating the bit pattern. this is intended to be a representation of the clock. Captured at a bit\_rate\_per\_mhz of a 1mhz clock.

---

## SYNC\_CMD\_STAT

```
localparam [SYNTH_SYNC_BITS_PER_TRANS-1:0]SYNC_CMD_STAT = {  
    SYNTH_SYNC_BITS_PER_TRANS/2{1'b1}},  
    SYNTH_SYNC_BITS_PER_TRANS/2{1'b0}}}
```

sync pulse command

---

## SYNC\_DATA

```
localparam [SYNTH_SYNC_BITS_PER_TRANS-1:0]SYNC_DATA = {  
    {
```

```

    SYNTH_SYNC_BITS_PER_TRANS/2{1'b0}},
    SYNTH_SYNC_BITS_PER_TRANS/2{1'b1}}
}

```

sync pulse data

## CMD\_DATA

```

localparam CMD_DATA = 3'b010

```

tuser decode for data

## CMD\_DATA

tuser decode for command

## INSTANTIATED MODULES

### uart\_baud\_gen\_tx

Generates TX BAUD rate for UART modules using modulo divide method.

### uart\_baud\_gen\_rx

```

mod_clock_ena_gen #(
    CLOCK_SPEED(CLOCK_SPEED),
    DELAY(RX_BAUD_DELAY)
) uart_baud_gen_rx ( .clk(aclk), .rstn(arstn), .start0(1'b0), .clr(s_clr_clk)

```

Generates RX BAUD rate for UART modules using modulo divide method.

### inst\_sipo

```

sipo #(
    BUS_WIDTH(TOTAL_SYNTH_BYTES_PER_TRANS)
) inst_sipo ( .clk(aclk), .rstn(arstn), .ena(ena_rx), .rev(1'b0), .load(r_load)

```

Captures RX data for 1553 receive

### inst\_piso

```

piso #(
    BUS_WIDTH(TOTAL_SYNTH_BYTES_PER_TRANS),
    DEFAULT_RESET_VAL(0),

```

```
DEFAULT_SHIFT_VAL(0)  
) inst_piso ( .clk(aclk), .rstn(arstn), .ena(ena_tx), .rev(1'b0), .load(s_a)
```

Generates TX data for 1553 transmit