

# axis\_1553\_decoder.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2021/05/24

---

## INFORMATION

---

### Brief

---

AXIS MIL-STD-1553 DECODER

### License MIT

---

Copyright 2021 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\_decoder

---

```
module axis_1553_decoder #(
    parameter
    CLOCK_SPEED
    =
    20000000,
    parameter
    SAMPLE_RATE
    =
    2000000,
    parameter
    BIT_SLICE_OFFSET
    =
    0,
    parameter
```

```

INVERT_DATA
=
0,
parameter
SAMPLE_SELECT
=
0
) ( input aclk, input arstn, output [15:0] m_axis_tdata, output m_axis_tvalid

```

This core is a MIL-STD-1553 to AXI streaming decoder. It uses the positive edge of a clock to sample data. This restricts the core to 2 Mhz and above for a sample clock.

## Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz, must be 2 MHz or above.
<b>SAMPLE_RATE</b> parameter	2 MHz or above rate that is an even divisor of CLOCK_SPEED
<b>BIT_SLICE_OFFSET</b> parameter	Changes the bit that is selected for data reduction.
<b>INVERT_DATA</b> parameter	Will invert all decoded data.
<b>SAMPLE_SELECT</b> parameter	Changes the bit that is sampled for data capture.

## Ports

<b>aclk</b>	Clock for all logic
<b>arstn</b>	Negative reset
<b>m_axis_tdata</b>	Output data for 1553 decoder.
<b>m_axis_tvalid</b>	When active high the output data is valid.
<b>m_axis_tuser</b>	Information about the AXIS data {TYY,NA,D,I,P} Bits explained below:

- TYY = TYPE OF DATA
  - 000 NA
  - 001 REG (NOT IMPLIMENTED)
  - 010 DATA
  - 100 CMD/STATUS
- NA = RESERVED FOR FUTURE USE.
- D = DELAY BEFORE DATA
  - 1 = Delay of 4us or more before data
  - 0 = No delay between data
- I = INVERT DATA
  - 1 = INVERT
  - 0 = NORMAL
- P = PARITY
  - 1 = GOOD
  - 0 = BAD

<b>m_axis_tready</b>	When active high the destination device is ready for data.
<b>diff</b>	Output data in TTL differential format.

## base\_1553\_clock\_rate

```
localparam integer base_1553_clock_rate = 1000000
```

1553 base clock rate

## **samples\_per\_mhz**

---

```
localparam integer samples_per_mhz = SAMPLE_RATE / base_1553_clock_rate
```

sample rate 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

## **delay\_time**

---

```
localparam integer delay_time = cycles_per_mhz * 4
```

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

## **samples\_to\_skip**

---

```
localparam integer samples_to_skip = (
    (cycles_per_mhz > samples_per_mhz) ? cycles_per_mhz /
    samples_per_mhz
    :
    0
)
```

calculate the number of samples to skip

## **round\_SAMPLE\_SELECT**

---

```
localparam integer round_SAMPLE_SELECT = (
    (SAMPLE_SELECT == 0) ? samples_to_skip /
    2
    :
    SAMPLE_SELECT % samples_to_skip
)
```

SAMPLE\_SELECT rounded

## **bit\_rate\_per\_mhz**

---

```
localparam integer bit_rate_per_mhz = samples_per_mhz
```

bit rate per mhz

## round\_BIT\_SLICE\_OFFSET

---

```
localparam integer round_BIT_SLICE_OFFSET = (
    (cycles_per_mhz > samples_per_mhz) ? ((
        BIT_SLICE_OFFSET == 0) ? bit_rate_per_mhz/4 : BIT_SLICE_OFFSET % bit_rate_per_mhz) :
    0
)
```

pick the middle of the samples generated by default

## sync\_pulse\_len

---

```
localparam integer sync_pulse_len = bit_rate_per_mhz * 3
```

sync pulse length

## bits\_per\_trans

---

```
localparam integer bits_per_trans = 20
```

bits per transmission

## synth\_bits\_per\_trans

---

```
localparam integer synth_bits_per_trans = (
    bits_per_trans*bit_rate_per_mhz
)
```

sync bits per trans

## sync\_cmd\_stat

---

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

Command sync pulse

## sync\_data

---

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

Data sync pulse

## cmd\_data

---

```
localparam cmd_data = 3'b010
```

data tuser encode

## cmd\_data

---

command tuser encode

## 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_bits_per_trans-1:0]synth_clk = {  
    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.

## STATE MACHINE

---

Constants that makeup the decoder state machine.

## diff\_wait

---

```
localparam diff_wait = 5'h01
```

wait for diff

## data\_cap

---

```
localparam data_cap = 5'h03
```

data capture

## data\_reduce

---

```
localparam data_reduce = 5'h07
```

reduce data

## parity\_gen

---

```
localparam parity_gen = 5'h0F
```

parity generator

## trans

---

```
localparam trans = 5'h1F
```

transmit data

## error

---

```
localparam error = 5'h00
```

someone made a whoops