

# up\_uart.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/02/29

---

## INFORMATION

---

### Brief

---

uP Core for interfacing with axis uart.

### 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.

## up\_1553

---

uP based 1553 communications device.

### Parameters

<b>ADDRESS_WIDTH</b>	Width of the uP address port.
<b>BUS_WIDTH</b>	Width of the uP bus data port.
<b>CLOCK_SPEED</b>	This is the aclk frequency in Hz
<b>BAUD_RATE</b>	Serial Baud, this can be any value including non-standard.
<b>PARITY_ENA</b>	Enable Parity for the data in and out.

<b>PARITY_TYPE</b>	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b>	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b>	Number of data bits, 1 to crazy non-standard amounts.
<b>RX_DELAY</b>	Delay in rx data input.
<b>RX_BAUD_DELAY</b>	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).
<b>TX_DELAY</b>	Delay in tx data output. Delays the time to output of the data.
<b>TX_BAUD_DELAY</b>	Delay in tx baud enable. This will delay the time the bit output starts.

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rstn</b>	Negative reset
<b>up_rreq</b>	uP bus read request
<b>up_rack</b>	uP bus read ack
<b>up_raddr</b>	uP bus read address
<b>up_rdata</b>	uP bus read data
<b>up_wreq</b>	uP bus write request
<b>up_wack</b>	uP bus write ack
<b>up_waddr</b>	uP bus write address
<b>up_wdata</b>	uP bus write data
<b>irq</b>	Interrupt when data is received
<b>tx</b>	transmit for UART (output to RX)
<b>rx</b>	receive for UART (input from TX)
<b>rts</b>	request to send is a loop with CTS
<b>cts</b>	clear to send is a loop with RTS

## FIFO\_DEPTH

```
localparam FIFO_DEPTH = 16
```

Depth of the fifo, matches UART LITE (xilinx), so I kept this just cause

## REGISTER INFORMATION

Core has 4 registers at the offsets that follow.

<b>RX_FIFO_REG</b>	h0
<b>TX_FIFO_REG</b>	h4
<b>STATUS_REG</b>	h8
<b>CONTROL_REG</b>	hC

## RX\_FIFO\_REG

```
localparam RX_FIFO_REG = 4'h0
```

Defines the address offset for RX FIFO

RX FIFO REGISTER	
31:8	7:0
UNUSED	RECEIVED DATA

Valid bits are from DATA\_BITS:0, which are data.

## TX\_FIFO\_REG

```
localparam TX_FIFO_REG = 4'h4
```

Defines the address offset to write the TX FIFO.

TX FIFO REGISTER	
31:8	7:0
UNUSED	TRANSMIT DATA

Valid bits are from DATA\_BITS:0, which are data.

## STATUS\_REG

```
localparam STATUS_REG = 4'h8
```

Defines the address offset to read the status bits.

STATUS REGISTER								
31:8	7	6	5	4	3	2	1	0
UNUSED	PE	FE	OE	irq_en	tx_full	tx_empty	rx_full	rx_valid

## Status Register Bits

<b>PE</b>	7, Parity error, active high on error
<b>FE</b>	6, Frame error, active high on error
<b>OE</b>	5, Overrun error, active high on error
<b>irq_en</b>	4, 1 when the IRQ is enabled by <b>CONTROL_REG</b>
<b>tx_full</b>	3, When 1 the tx fifo is full.
<b>tx_empty</b>	2, When 1 the tx fifo is empty.

**rx\_full**            1, When 1 the rx fifo is full.  
**rx\_valid**           0, When 1 the rx fifo contains valid data.

## CONTROL\_REG

```
localparam CONTROL_REG = 4'hC
```

Defines the address offset to set the control bits.

CONTROL REGISTER				
31:5	4	3:2	1	0
UNUSED	ENA_INTR_BIT	UNUSED	RST_RX_BIT	RST_TX_BIT

See Also: [ENABLE\\_INTR\\_BIT](#), [RESET\\_RX\\_BIT](#), [RESET\\_TX\\_BIT](#)

## Control Register Bits

**ENABLE\_INTR\_BIT**    4, Control Register offset bit for enabling the interrupt.  
**RESET\_RX\_BIT**        1, Control Register offset bit for resetting the RX FIFO.  
**RESET\_TX\_BIT**        0, Control Register offset bit for resetting the TX FIFO.

## INSTANTIATED MODULES

### inst\_axis\_uart

```
axis_uart #(
    BAUD_CLOCK_SPEED(CLOCK_SPEED),
    BAUD_RATE(BAUD_RATE),
    PARITY_ENA(PARITY_ENA),
    PARITY_TYPE(PARITY_TYPE),
    STOP_BITS(STOP_BITS),
    DATA_BITS(DATA_BITS),
    RX_DELAY(RX_DELAY),
    RX_BAUD_DELAY(RX_BAUD_DELAY),
    TX_DELAY(TX_DELAY),
    TX_BAUD_DELAY(TX_BAUD_DELAY)
) inst_axis_uart ( .aclk(clk), .arstn(rstn), .parity_err(s_parity_err), .fr
```

UART instance with AXIS interface for TX/RX

## inst\_rx\_fifo

---

```
fifo #(
    FIFO_DEPTH(FIFO_DEPTH),
    BYTE_WIDTH(BUS_WIDTH),
    COUNT_WIDTH(8),
    FWFT(1),
    RD_SYNC_DEPTH(0),
    WR_SYNC_DEPTH(0),
    DC_SYNC_DEPTH(0),
    COUNT_DELAY(0),
    COUNT_ENA(0),
    DATA_ZERO(0),
    ACK_ENA(0),
    RAM_TYPE("block")
) inst_rx_fifo ( .rd_clk(clk), .rd_rstn(rstn & r_rstn_rx_delay[0]), .rd_en(s
```

Buffer up to 16 items output from the axis\_1553\_encoder.

## inst\_tx\_fifo

---

```
fifo #(
    FIFO_DEPTH(FIFO_DEPTH),
    BYTE_WIDTH(BUS_WIDTH),
    COUNT_WIDTH(8),
    FWFT(1),
    RD_SYNC_DEPTH(0),
    WR_SYNC_DEPTH(0),
    DC_SYNC_DEPTH(0),
    COUNT_DELAY(0),
    COUNT_ENA(0),
    DATA_ZERO(0),
    ACK_ENA(0),
    RAM_TYPE("block")
) inst_tx_fifo ( .rd_clk(clk), .rd_rstn(rstn & r_rstn_tx_delay[0]), .rd_en(s
```

Buffer up to 16 items to input to the axis\_1553\_decoder.

