uart6551_wb32

Overview

A UART component (Universal Asynchronous Transmitter / Receiver) is used for the asynchronous transmission and reception of data. Asynchronous referring to the lack of a clock signal during transmission or reception.

uart6551 is a WDC6551 register compatible UART. The UART is a 32-bit peripheral device. It may be used as an eight-bit peripheral by connecting the high order 24-bit data input lines to ground, and grounding select lines one to three.

Baud rate is controlled by clock divider which assumes a 200MHz baud reference clock input. If a different clock frequency is used, then the divider table will need to be updated. The baud rate may also be controlled via a clock divider register. This register is 24 bits so gives a minimum frequency of 11.92 Hz assuming a 200MHz clock. (200MHz / 2^24).

Special Features

- WDC6551 register compatibility
- Support for message signaled interrupts (QMSI)

Register Description

There are only six registers in the design. The function of the low order eight bits of the registers matches the 6551 function. The controller honors byte lane selects so only the portion of the register selected is written.

Reg	Moniker	Description
0	UART_TRB	Transmit and receive buffer. Data written is transmitted, on a read data available is read. Also reads / writes the clock multiplier if access to clock multiplier is enabled.
1	UART_STAT	Status Register. Returns status bits on a read, a write of any value will cause a reset of some of the command register bits
2	UART_CMD	Command register
3	UART_CTRL	Control register
4	UART_MSI	Message signaled interrupt control reg, vector and device index
5	UART_MSI2	Message signaled interrupt control reg, interrupt controller number, interrupt level number and bus priority

UART_TRB

This register is 32-bits wide of which only the lower eight bits are used to transmit or receive data by the uart. Data written to the register is transmitted. A register read returns data received by the uart. When the fifo's are enabled writing to this register writes to the transmit fifo. Reading this register reads the receive fifo. If clock divider access is enabled (via control register bit 31) then this register allows modifying or reading the clock divider value. Writing a clock divider value to this register automatically switches the function back to transmit / receive.

UART_STAT

Uart status register. Writing any value to the status register resets some of the uart's command bits.

Bit	Status	

0	Parity Error	1 = parity error occurred, 0 = no error
1	Framing Error	1 = framing error
2	Overrun	1 = overrun
3	Rx Full	1 = receiver data available
4	Tx Empty	1 = open slot in transmit fifo
5	DCD	0 = data carrier present
6	DSR	0 = data set ready
7	IRQ	1 = irq occurred
	Additional Line Status Byte	
8	reserved	
9	reserved	
10	reserved	
11	reserved	
12	Break received	1 if a break signal is received
13	Tx Full	1 = transmit fifo full
14	reserved	
15	G Rcv Err	1 = global receiver error (set if any error status is set)
	Additional Modem Status Byte	
16	CTS	1 = CTS line changed state
17	DSR	1 = DSR line changed state
18	RI	1 = RI line changed state
19	DCD	1 = DCD line changed state
20	CTS	CTS state
21	reserved	
22	RI	RI state
23	reserved	
	IRQ Status	
24,25	zero	these two bits are zero
26 to 28	IRQENC	encoded irq value (0 to 7)
29 to 30	reserved	
31	irq	IRQ is set

UART_CMD

Bit		
0	DTR	output 1 = low, 0 = high
1	RxIe	receiver interrupt enable 0 = enabled, 1 = disabled
2,3	RTS Control	
	00	output RTS high
	01	output RTS low, enable transmit interrupt
	10	output RTS low,
	11	output RTS low, send a break signal
4	LLB	1 = local loopback (receiver echo)
5 to 7	Parity Control	
	000	no parity
	001	odd parity
	011	even parity
	101	transmit mark parity (parity error disabled)
	111	transmit space parity (parity error disabled)

8	LSIe	line status change interrupt enable 1 = enabled
9	MSIe	modem status change interrupt enable 1 = enabled
10	RxToIe	receiver timeout interrupt enable 1 = enabled
11 to 31	reserved	

UART_CTRL

<table>
<tr><td>Bit</td><td></td><td></td></tr>
<tr><td>0 to 3</td><td>Baud Rate</td><td></td></tr>
<tr><td></td><td><table>
<tr><td>0000</td><td>Use 16x external clock</td></tr>
<tr><td>0001</td><td>50</td></tr>
<tr><td>0010</td><td>75</td></tr>
<tr><td>0011</td><td>109.92</td></tr>
<tr><td>0100</td><td>134.58</td></tr>
<tr><td>0101</td><td>150</td></tr>
<tr><td>0110</td><td>300</td></tr>
<tr><td>0111</td><td>600</td></tr>
<tr><td>1000</td><td>1200</td></tr>
<tr><td>1001</td><td>1800</td></tr>
<tr><td>1010</td><td>2400</td></tr>
<tr><td>1011</td><td>3600</td></tr>
<tr><td>1100</td><td>4800</td></tr>
<tr><td>1101</td><td>7200</td></tr>
<tr><td>1110</td><td>9600</td></tr>
<tr><td>1111</td><td>19200</td></tr>
</table></td><td>This table is expanded using an extra control bit #27.</td></tr>
<tr><td>4</td><td>Rx clock source</td><td>0 = external, 1= baud rate generator</td></tr>
<tr><td>5,6</td><td>Word length<table>
<tr><td>00</td><td>8</td></tr>
<tr><td>01</td><td>7</td></tr>
<tr><td>10</td><td>6</td></tr>
<tr><td>11</td><td>5</td></tr>
</table></td><td>code for word length in bits</td></tr>
<tr><td>7</td><td>Stop Bit<table>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1 if 8 bits and parity</td></tr>
<tr><td>1</td><td>1.5 if 5 bits and no parity</td></tr>
<tr><td>1</td><td>2 otherwise</td></tr>
</table></td><td></td></tr>
<tr><td>8 to 15</td><td>reserved</td><td>do not use</td></tr>
<tr><td>16</td><td>Fifo enable</td><td>1 = fifo’s enabled</td></tr>
<tr><td>17</td><td>Rx Fifo Clear</td><td>1 = clear receiver fifo</td></tr>
<tr><td>18</td><td>Tx Fifo Clear</td><td>1 = clear transmit fifo</td></tr>
<tr><td>19</td><td>reserved</td><td></td></tr>
<tr><td>20,21</td><td>Transmit Threshold<table>
<tr><td>0</td><td>1 byte</td></tr>
<tr><td>1</td><td>¼ full</td></tr>
<tr><td>2</td><td>½ full</td></tr>
<tr><td>3</td><td>¾ full</td></tr>
</table></td><td>Threshold for DMA signal activation
If the transit fifo count is less than the threshold then a DMA transfer is triggered.</td></tr>
</table>

<table>
<tr><td>22, 23</td><td>Receive Threshold<table><tr><td>0</td><td>1 byte</td></tr><tr><td>1</td><td>¼ full</td></tr><tr><td>2</td><td>½ full</td></tr><tr><td>3</td><td>¾ full</td></tr></table></td><td>Threshold for DMA signal activation. If the receive fifo count is greater than the threshold then a DMA transfer is triggered.</td></tr>
<tr><td>24</td><td>hwfc</td><td>1 = automatic hardware flow control</td></tr>
<tr><td>25</td><td>reserved</td><td></td></tr>
<tr><td>26</td><td>dmaEnable</td><td>1 = dma enabled</td></tr>
<tr><td>27</td><td>Baud Rate bit 4<table><tr><td>10000</td><td>38400</td></tr><tr><td>10001</td><td>57600</td></tr><tr><td>10010</td><td>115200</td></tr><tr><td>10011</td><td>230600</td></tr><tr><td>10100</td><td>460800</td></tr><tr><td>10101</td><td>921600</td></tr><tr><td>10110</td><td>reserved</td></tr><tr><td>10111</td><td>reserved</td></tr><tr><td>11xxx</td><td>reserved</td></tr></table></td><td>Extended baud rate selection bit, used in combination with bits 0 to 3.</td></tr>
<tr><td>28,29</td><td>reserved</td><td></td></tr>
<tr><td>30</td><td>selDV</td><td>1 = use clock divider register, 0 = use baud table</td></tr>
<tr><td>31</td><td>accessDV</td><td>1 = access clock divider via TRB register, 0 = normal TRB operation</td></tr>
</table>

Selecting the clock divider register as the baud source allows any programmable baud rate.

UART_MSI

Bits		
0 to 11	Interrupt vector number	The vector number to use which the interrupt controller uses to identify the interrupt subroutine, operating mode, software stack
12 to 21	Device index	An index into a 1024 entry table containing the device addresss: segment, bus, device, function. The entry in the table should reflect the address of the 6551 UART. It is how software knows which device to service.
22 to 31	Data	A raw data field, supplied to the interrupt contoller

UART_MSI2

Bits		
0 to 5	Interrupt priority level	This is the level of the interrupt priority seen by the CPU once the message is received
6	~	Reserved
7 to 12	Interrupt controller number	This is the number of the interrupt controller that should process the message
13 to 16	Message bus priority	This is the priority used to route the interrupt message on the system bus. It may be used by bus bridges.

Ports

Signal	I/O	Wid	Purpose
rst_i	I	1	reset
clk_i	I	1	bus clock input
cs_i	I	1	circuit/core select
irq_o	O	1	interrupt request (not used if using message signaled interrupts)
	WISHBONE SIGNALS		
req	I		request bus input signals from a bus master (contains WISHBONE signals)
resp	O		Response bus output back to the bus master (contains WISHBONE signals)
	Modem Controls		
cts_ni	I	1	clear to send input active low.
rts_no	O	1	request to send output active low
dsr_ni	I	1	data set ready active low
dcd_ni	I	1	data carrier detect active low
dtr_no	O	1	data terminal ready active low
ri_ni	I	1	ring indicator active low
rxd_i	I	1	serial data input (receive)
txd_o	O	1	serial data output (transmit)
data_present	O	1	data is present in the receiver
rxDRQ_o	O	1	receiver DMA request
txDRQ_o	O	1	transmitter DMA request
xclk_i	I	1	external baud rate clock
RxC_i	I	1	external receiver clock

WISHBONE Compatibility Datasheet

The uart6551_wb32 core may be directly interfaced to a WISHBONE compatible bus.

WISHBONE Datasheet WISHBONE SoC Architecture Specification, Revision B.3		
Description:	Specifications:	
General Description:	Uart6551_wb32 – UART	
Supported Cycles:	SLAVE, READ / WRITE SLAVE, BLOCK READ / WRITE SLAVE, RMW	
Data port, size: Data port, granularity: Data port, maximum operand size: Data transfer ordering: Data transfer sequencing	32 bit 8 bit byte lane selects 32 bit Little Endian any (undefined)	
Clock frequency constraints:	Baud rate lookup table depends on clock frequency	
Supported signal list and cross reference to equivalent WISHBONE signals	Signal Name: resp.ack req.sel(3:0) req.adr(31:0) req.clk req.dat(31:0) resp.dat(31:0) req.cyc req.stb req.we req.rst	WISHBONE Equiv. ACK_O SEL_I() ADR_I() CLK_I DAT_I() DAT_O() CYC_I STB_I WE_I RST_I
Special Requirements:	none	