WB_UART8 Serial Communications Port

Summary

This document provides detailed reference information with respect to the UART peripheral device.

Core Reference
CR0157 (v3.1) August 01, 2008

Serial ports on embedded systems often provide a 2-wire communication channel only. The WB_UART8 is a Wishbone-compliant, serial communications port, providing serial communication with hardware handshake and FIFO buffers.

The WB_UART8 can be used with any of the Wishbone-compliant processors available in Altium Designer.

Features

- 8-bit UART (fixed to no parity, 8 data bits and 1 stop bit)
- Full Duplex
- 16-byte FIFO input buffer
- 8-byte FIFO output buffer
- Automatic RTS/CTS hardware-controlled handshake, with user-definable watermark levels
- Dedicated, high precision internal Baud rate generator
- Wishbone-compliant.

Available Devices

The WB_UART8 device can be found in the FPGA Peripherals integrated library (`FPGA Peripherals.IntLib`), located in the (`\Library\Fpga` folder of the installation.

Functional Description

Symbol

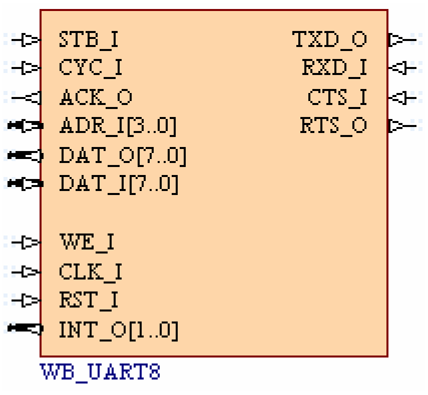

Figure 1. WB_UART8 Symbol

Pin Description

Table 1. WB_UART8 pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK_I	I	Rise	External (system) clock
RST_I	I	High	External (system) reset
Microcontroller Interface Signals			
STB_I	I	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
CYC_I	I	High	Cycle signal. When asserted, indicates the start of a valid Wishbone cycle
ACK_O	O	High	Standard Wishbone device acknowledgement signal. When this signal goes high, the WB_UART8 (Wishbone Slave) has finished execution of the requested action and the current bus cycle is terminated
ADR_I	I	4	Standard Wishbone address bus, used to select an internal register of the Wishbone slave device for writing to/reading from
DAT_O	O	8	Data to be sent to an external Wishbone master device (e.g. host processor)
DAT_I	I	8	Data received from an external Wishbone master device (e.g. host processor)
WE_I	I	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle: 0 = Read 1 = Write
INT_O	O	2/High	Interrupt output lines. Two interrupts are sent to the connected host processor on this 2-bit bus. bit 0 = set High if any of the lower 4 bits of the Interrupt Status register (INTSTAT.3..INTSTAT.0) are High. These bits are all related to the transmitter. bit 1 = set High if any of the upper 4 bits of the Interrupt Status register (INTSTAT.7..INTSTAT.4) are High. These bits are all related to the receiver.

Name	Type	Polarity/ Bus size	Description
Serial Interface Signals			
TXD_O	O	-	Serial data transmit
RXD_I	I	-	Serial data receive
CTS_I	I	Level	Clear To Send. Part of the automatic hardware-controlled handshake. 0 = device connected to the WB_UART8 on the serial side is not ready to receive data 1 = device connected to the WB_UART8 on the serial side is ready to receive data. **Note**: If the `ctsen` bit in the Handshake register (HSK.0) is cleared, transmission will not depend on the readiness of the connected (remote) serial device.
RTS_O	O	Level	Request To Send. Part of the automatic hardware-controlled handshake. 0 = WB_UART8 is ready to receive data from the connected serial device. RTS_O will go Low when: • there are less bytes in the Receive Buffer than the specified low watermark in the RXLOWMARK register, or • the `forcerts` bit in the Handshake register (HSK.1) is High and the `rtsval` bit in the same register (HSK.2) is Low. 1 = WB_UART8 is not ready to receive data from the connected serial device. RTS_O will go High when: • there are more bytes in the Receive Buffer than the specified high watermark in the RXHIGHMARK register, or • the `forcerts` bit in the Handshake register (HSK.1) is High and the `rtsval` bit in the same register (HSK.2) is High.

Hardware Description

Block Diagram

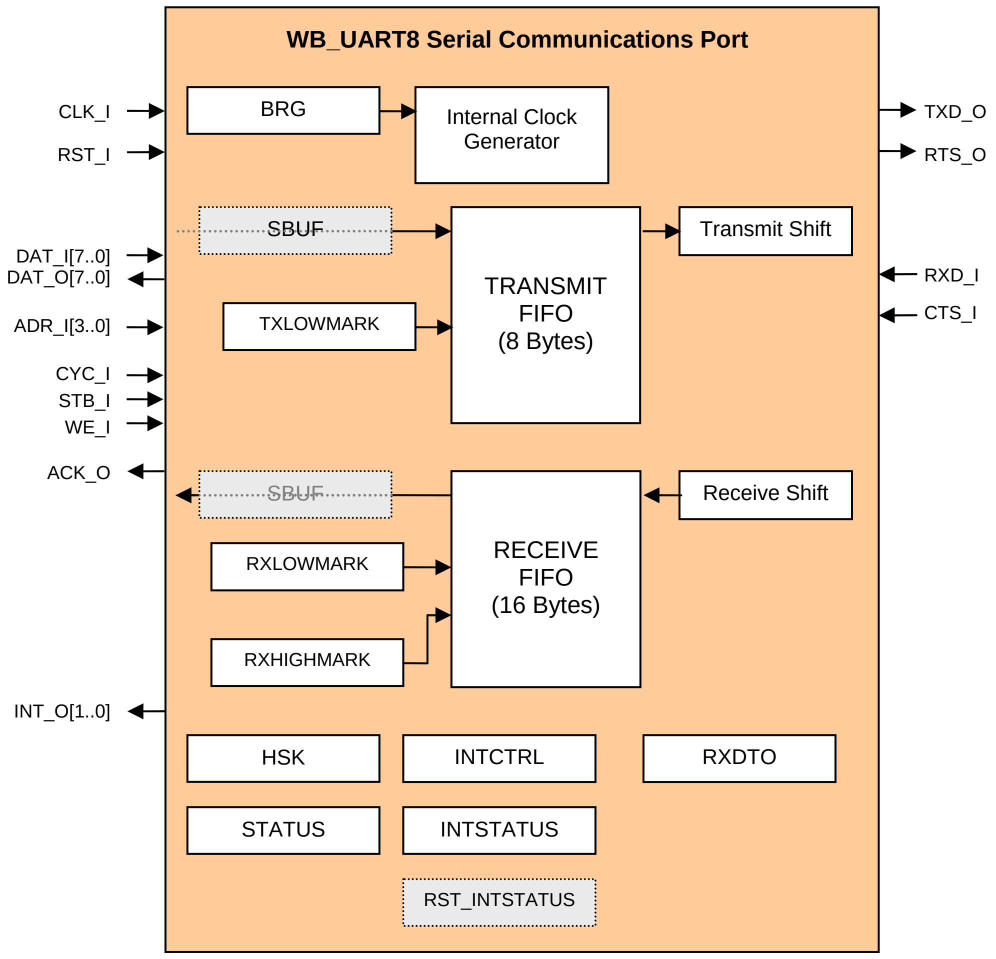

Figure 2. WB_UART8 block diagram

Internal Registers

The following sections detail the internal registers for the WB_UART8.

Baud Rate Generator Register (BRG)

Access: Read/Write

The upper (BRG[23..16]), middle (BRG[15..8]) and lower (BRG[7..0]) bytes of this register are accessed separately. The register's 24-bit value is used in the generation of the serial transmit and receive clocks.

The baud rate is generated based on the overflow rate of an internal 24-bit baud rate adder. This adder increments every timer tick with the 24-bit value set in the BRG register. When the most significant bit in the adder changes state from '0' to '1', the baud rate counter fires. A serial bit takes 8 clock cycles from this counter. The value to be loaded into the BRG register can be calculated using the following expression:

$$\text{BRG} = \frac{\text{Baud Rate} * \text{8000000h}}{F_{\text{CLK\_I}}}$$

Note: The internal adder is not reset when the baud rate counter changes.

Handshake Register (HSK)

Access: Read/Write

This 8-bit register is used to control the hardware handshake between the WB_UART8 and the remote device connected to its serial port.

Table 2. The HSK register

MSB							LSB
cts	-	-	-	-	rtsval	forcerts	ctsen

Table 3. HSK register bit functions

Bit	Symbol	Function
HSK.7	cts	Returns current state of CTS_I pin when read. This bit is ignored when writing to the register.
HSK.6	-	Not used. Returns 0 when read
HSK.5	-	Not used. Returns 0 when read
HSK.4	-	Not used. Returns 0 when read
HSK.3	-	Not used. Returns 0 when read
HSK.2	rtsval	RTS Value bit. Controls the state of the RTS_O pin when the `forcerts` bit of this register (HSK.1) is High. 0 = RTS_O taken Low 1 = RTS_O taken High.
HSK.1	forcerts	Force RTS bit. 0 = State of RTS_O pin is purely controlled based on the number of bytes in the Receive Buffer. If the number exceeds the high watermark defined by the RXHIGHMARK register, RTS_O goes High. RTS_O will automatically go Low when the number of bytes drops below the low watermark defined by the RXLOWMARK register. 1 = State of RTS_O pin is controlled by the value of the `rtsval` bit of this register (HSK.2).
HSK.0	ctsen	Enable CTS bit. 0 = WB_UART8 is free to transmit data without regard for the current state of the CTS_I pin 1 = WB_UART8 must wait for the CTS_I pin to be taken High by the connected serial station before transmission can proceed.

Status Register (STATUS)

Access: Read-only

This 8-bit register is used to determine the current state of the WB_UART8 device.

Table 4. The STATUS register

MSB							LSB
rxfull	rxhigh	rxnempty	rxtimeout	txshempty	txlow	txempty	txfull

Table 5. STATUS register bit functions

Bit	Symbol	Function
STATUS.7	rxfull	Receiver Full flag. Taken High if the Receive Buffer is full.
STATUS.6	rxhigh	Receiver High flag. Taken High if the number of bytes in the Receive Buffer exceeds the high watermark defined by the RXHIGHMARK register.
STATUS.5	rxnempty	Receiver Not Empty flag. Taken High if the Receive Buffer is not empty.
STATUS.4	rxtimeout	Receiver Time-out flag. Taken High if the Receive Buffer is not empty and has not been read within time.
STATUS.3	txshempty	Transmitter Shift Register Empty flag. Taken High if the Transmitter shift register is empty.
STATUS.2	txlow	Transmitter Low flag. Taken High if the number of bytes in the Transmit Buffer is less than the low watermark defined by the TXLOWMARK register.
STATUS.1	txempty	Transmitter Empty flag. Taken High if the Transmit Buffer is empty.
STATUS.0	txfull	Transmitter Full. Taken High if the Transmit Buffer is full.

Interrupt Control Register (INTCTRL)

Access: Read/Write

This 8-bit register is used to enable interrupt generation for each of the corresponding bits in the STATUS register. Provided bit INTCTRL.n is High, an interrupt will be generated when the corresponding bit STATUS.n goes High.

Table 6. The INTCTRL register

MSB							LSB
rxfull	rxhigh	rxnempty	rxtimeout	txshempty	txlow	txempty	txfull

Table 7. INTCTRL register bit functions

Bit	Symbol	Function
INTCTRL.7	rxfull	Enables interrupt generation for Receiver Full flag (STATUS.7).
INTCTRL.6	rxhigh	Enables interrupt generation for Receiver High flag (STATUS.6).
INTCTRL.5	rxnempty	Enables interrupt generation for Receiver Not Empty flag (STATUS.5).
INTCTRL.4	rxtimeout	Enables interrupt generation for Receiver Time-out flag (STATUS.4).
INTCTRL.3	txshempty	Enables interrupt generation for Transmitter Shift Register Empty flag (STATUS.3).
INTCTRL.2	txlow	Enables interrupt generation for Transmitter Low flag (STATUS.2).
INTCTRL.1	txempty	Enables interrupt generation for Transmitter Empty flag (STATUS.1).
INTCTRL.0	txfull	Enables interrupt generation for Transmitter Full (STATUS.0).

Interrupt Status Register (INTSTATUS)

Access: Read/Write

This 8-bit register is used to reflect the state of interrupts for the WB_UART8 device. If an interrupt is generated, the cause of the interrupt can be easily determined by reading this register.

Each bit in this register (INTSTATUS.n) will go HIGH provided:

- the corresponding bit in the Interrupt Control register (INTCTRL.n) is High and
- the corresponding bit in the Status register (STATUS.n) goes High.

A set bit in this register can only be cleared by software, by writing a ‘1’ to the relevant bit of the register (INTSTATUS.n). This may sound strange at first, as the bit is already a ‘1’. When you perform a Wishbone Write to the INTSTATUS register, you are simply using the address of the register only. The byte of data sent from the host processor is actually loaded into an alternate, internal register – RST_INTSTATUS. A '1' in bit RST_INTSTATUS.n is subsequently used to clear the corresponding bit in the Interrupt Status register (INTSTATUS.n).

Note: After writing a '1' to clear bit INTSTATUS.n, the RST_INTSTATUS.n bit will be cleared on the next rising edge of CLK_I provided that the INTSTATUS.n bit is still clear.

Table 8. The INTSTATUS register

MSB							LSB
rxfull	rxhigh	rxnempty	rxtimeout	txshempty	txlow	txempty	txfull

Table 9. INTSTATUS register bit functions

Bit	Symbol	Function
INTSTATUS.7	rxfull	Goes High if INTCTRL.7 is High and STATUS.7 goes High.
INTSTATUS.6	rxhigh	Goes High if INTCTRL.6 is High and STATUS.6 goes High
INTSTATUS.5	rxnempty	Goes High if INTCTRL.5 is High and STATUS.5 goes High
INTSTATUS.4	rxtimeout	Goes High if INTCTRL.4 is High and STATUS.4 goes High
INTSTATUS.3	txshempty	Goes High if INTCTRL.3 is High and STATUS.3 goes High
INTSTATUS.2	txlow	Goes High if INTCTRL.2 is High and STATUS.2 goes High
INTSTATUS.1	txempty	Goes High if INTCTRL.1 is High and STATUS.1 goes High
INTSTATUS.0	txfull	Goes High if INTCTRL.0 is High and STATUS.0 goes High

Serial Data Buffer Register (SBUF)

Access: Read/Write

This is not actually a register in the true sense of the word, but rather is a single address that is used to access the Transmit and Receive Buffers. Performing a Wishbone Write to the SBUF address loads data directly into the Transmit Buffer. If the Buffer is full, transmission may stop and the buffer content is overwritten.

Performing a Wishbone Read from the SBUF address retrieves data directly from the Receive Buffer. If no bytes are available in the Receive Buffer, the returned byte is invalid. Otherwise, the retrieved byte is removed from the buffer, effectively freeing up space.

Transmit Buffer Low Watermark Register (TXLOWMARK)

Access: Read/Write

For optimal performance, the Transmit Buffer should be filled at all times and the transmitter interrupt service should occur as little as possible. Therefore a low watermark can be set. When there are less bytes in the Transmit Buffer than indicated by this register, the corresponding `txlow` status flag (STATUS.2) will be set and an interrupt can be generated to indicate to the processor that it's time to refill the Transmit Buffer.

Note: The interrupt for this condition will only be generated provided that the corresponding interrupt enable bit for this flag is set in the Interrupt Control register (INTCTRL.2).

Receive Buffer Watermark Registers (RXHIGHMARK and RXLOWMARK)

Access: Read/Write

High and low watermarks can be defined for the Receive Buffer, which can then be used to control the state of the RST_O output line sent to the connected serial device from which data is being received. If the number of bytes in the Receive Buffer exceeds the high watermark defined by the RXHIGHMARK register, RTS_O goes High. RTS_O will automatically go Low again when the number of bytes drops below the low watermark defined by the RXLOWMARK register

Upon exceeding the high watermark, the `rxhigh` status flag will be set (STATUS.6). An interrupt can be generated provided that the corresponding interrupt enable bit in the Interrupt Control register is set (INTCTRL.6).

Note: Asserting the RTS_O signal (i.e. RTS_O = '1') should stop the connected serial device from sending more information. However, in many implementations, the remote device will empty its Transmit Buffer before stopping. If a currently transmitting WB_UART8 receives a Low on its CTS_I input, it will finish the current byte in its Transmit shift register and then stop.

Receive Delay Timeout Register (RXDTO)

Access: Read/Write

This 8-bit register provides an 8-bit delay value used in the generation of the rxtimeout interrupt (if enabled). You are essentially specifying the number of bit cycles that can elapse since the last data received by the WB_UART8. The delay time can be in the valid range 0 – 255.

After reception of the last data word (i.e. stop bit received) a delay counter is started. If another data word is received – heralded by the detection of another start bit – the delay counter is stopped. If, however, no further data is received, the delay counter will continue to increment on each change of the Baud rate generator. If the Receive Buffer is not empty and the delay counter reaches the delay time specified by the Receive Delay Timeout register, then the rxtimeout interrupt will be generated provided:

- the `rxtimeout` bit in the Interrupt Control register (INTCTRL.4) is High and
- the `rxtimeout` bit in the Status register (STATUS.4) is High.

Register Reset Values

Table 10 shows the values contained in each of the WB_UART8's internal registers after an external system reset has been received on the RST_I input.

Table 10. Register reset values

Register	Value after reset
BRG[23..16]	00000000
BRG[15..8]	00000000
BRG[7..0]	00000000
HSK	10000000
STATUS	00001010
INTCTRL	00000000
INTSTATUS	00000000
TXLOWMARK	00000000
RXHIGHMARK	11111111
RXLOWMARK	00000000
RXDTO	00000000

Note: After a reset, bit 3 of the Status register (`txshempty`) is initially cleared to '0', then set to '1' at the first cycle of CLK_I. Bit 7 of the Handshake register (`cts`) depends on the state of the CTS_I pin, which in turn is controlled by the remote serial device. Typically this is a '1' as any data sent to the remote device before the reset will have easily been read by that device.

Host to Controller Communications

Communications between the host processor and the WB_UART8 is carried out over the standard Wishbone bus.

The host processor can access any of the WB_UART8's software accessible internal registers. Selection of a particular register is achieved by supplying the 4-bit binary ID address code of the register. This code is sent to the WB_UART8 and appears at its ADR_I input. Table 11 shows the address IDs associated with each of the addressable registers.

Table 11. Internal register address IDs

Register	Register Address ID
BRG[23..16]	0000
BRG[15..8]	0001
BRG[7..0]	0010
HSK	0011
STATUS	0100
INTCTRL	0101
INTSTATUS	0110
SBUF	0111
TXLOWMARK	1000
RXHIGHMARK	1001
RXLOWMARK	1010
RXDTO	1011

Notes

- The STATUS register is read-only.
- The SBUF address is used to access the Transmit and Receive Buffers directly, and not a dedicated SBUF register. Provided you are performing a Write (WE_I input High), you will access the Transmit Buffer. When performing a Read (WE_I input Low) you will access the Receive Buffer.
- When performing a read using address '0110', you will be reading data from the INTSTATUS register. When performing a write using this same address, you will be loading data into the internal RST_INTSTATUS register.

Writing to an Internal Register

Data is written from the host processor to an internal register in the WB_UART8, in accordance with the standard Wishbone data transfer handshaking protocol. The write operation occurs on the rising edge of the CLK_I input and can be summarized as follows:

- The host presents the 4-bit address ID for the register to be written on its ADR_O output and a valid byte of data on its DAT_O output. It then asserts its WE_O signal, to specify a Write cycle.
- The WB_UART8 receives the address ID on its ADR_I input and prepares to receive data into the selected register.
- The host asserts its STB_O and CYC_O outputs, indicating that the transfer is to begin. The WB_UART8, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by latching the byte of data appearing at its DAT_I input, into the target register, and asserting its ACK_O signal – to indicate to the host that the data has been received.
- The host, which monitors its ACK_I input on each rising edge of the CLK_I signal, responds by negating the STB_O and CYC_O signals. At the same time, the WB_UART8 negates the ACK_O signal and the data transfer cycle is naturally terminated.

Reading from an Internal Register

Data is read from one of the WB_UART8's internal registers, in accordance with the standard Wishbone data transfer handshaking protocol. This data transfer cycle can be summarized as follows:

- The host presents the 4-bit address ID for the register to be read on its ADR_O output. It then negates its WE_O signal, to specify a Read cycle.
- The WB_UART8 receives the address ID on its ADR_I input and prepares to transmit data from the selected register.
- The host asserts its STB_O and CYC_O outputs, indicating that the transfer is to begin. The WB_UART8, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by presenting the valid byte of data at its DAT_O output and asserting its ACK_O signal – to indicate to the host that valid data is present.
- The host, which monitors its ACK_I input on each rising edge of the CLK_I signal, responds by latching the byte of data appearing at its DAT_I input and negating the STB_O and CYC_O signals. At the same time, the WB_UART8 negates the ACK_O signal and the data transfer cycle is naturally terminated.

Interfacing to a 32-Bit Processor

Figure 3 shows an example of how a WB_UART8 device can be wired into a design that uses a 32-bit processor – in this case a TSK3000A. A configurable Wishbone Interconnect device (WB_INTERCON) is used to simplify connection and also handle the addressing – taking the 24-bit address line from the processor and mapping it to the 4-bit address line used to drive the peripheral.

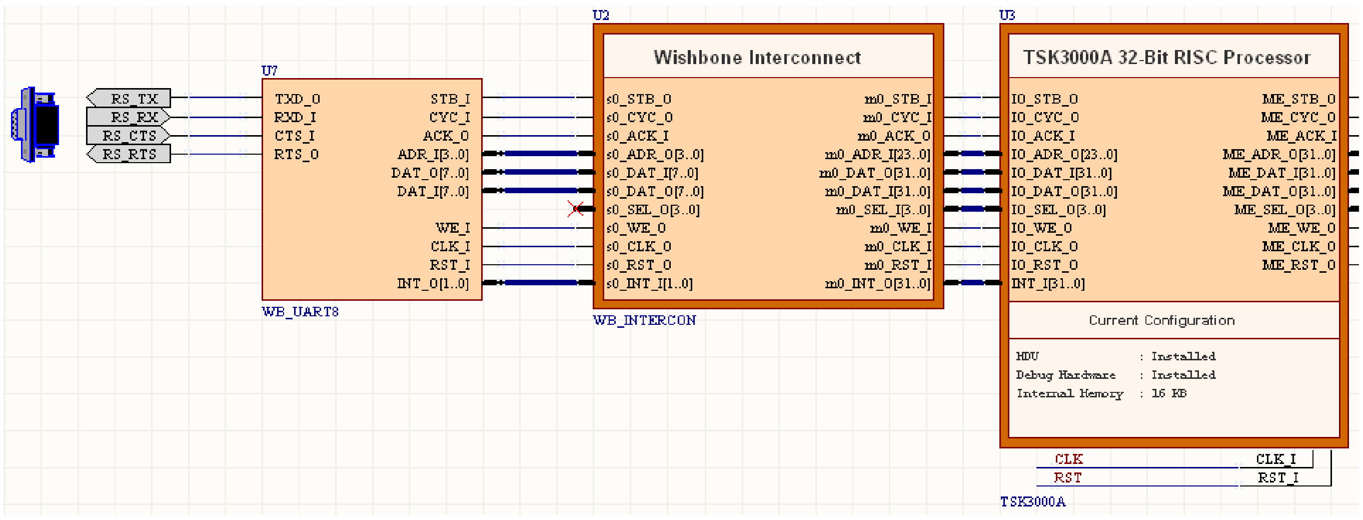

Figure 3. Example interfacing between a 32-bit processor (TSK3000A) and a WB_UART8 Controller

Internal WB_UART8 registers are accessed directly by adding the 4-bit address for the required register to the 24-bit base address of the WB_UART8 device. This base address is specified as part of the peripheral's definition when adding it as a slave to the Wishbone Interconnect. For example, if the base address entered for the device is 100000h (mapping it to address FF10_0000h in the TSK3000A's address space), and you want to write to the Handshake register (HSK) with binary address 0011 (or 3h), the value entered on the processor's 24-bit IO_ADR_O line would be:

100000h + 3h = 100003h

For further information on the Wishbone Interconnect, refer to the *WB_INTERCON Configurable Wishbone Interconnect* core reference.

For further information on the TSK3000A processor, refer to the *TSK3000A 32-bit RISC Processor* core reference. Similar references can be found for other 32-bit processors supported by Altium Designer, by using the lower section of the **Knowledge Center** panel and navigating to the *Documentation Library » Embedded Processors and Software Development » FPGA Based and Discrete Processors* section.

The following example project includes a WB_UART8 device: `\Examples\NB1 Examples\Processor Examples\Tsk3000 uart\Fpga_tsk3000_uart.PrjFpg`

Operational Overview

After an external reset, the WB_UART8 is effectively ready for use straight away. The hardware controlled handshaking is disabled after a reset – meaning that if data is available in the Transmit Buffer, the device will start sending it, regardless of the state of the CTS_I pin. The Receiver will start receiving data as soon as the connected remote station sends it.

On the transmitter side, the WB_UART8 sends a logical '0' on its TXD_O line to notify the remote station's receiver that a new data word is being sent. The next byte of data in the Transmit Buffer is loaded into the Transmit Shift register. The 8 bits of data are then shifted out onto the TXD_O line on each rising edge of the Transmitter's clock, with the LSB sent first. A stop bit is then sent to notify the remote station's receiver that transmission of this particular data word is finished.

On the receiver side, the WB_UART8 looks for a logical '0' on its RXD_I line, to signify the start of data transmission from the remote station's transmitter. The 8 bits of data are then read into the Receive Shift register on each rising edge of the Receiver's clock. Once the Stop bit has been detected, the byte of data in the shift register is loaded into the Receive Buffer.

The WB_UART8 and the connected remote station should ideally be operating at the same Baud rate. The receiver in either station uses the start bit of each transmission to synchronize its clock to that of the transmitter.

If another byte of data is ready for transmission, the start bit of the next word is transmitted as soon as the stop bit of the previous word has been sent.

When there is nothing to transmit, the TXD_O line remains at logical '1'.

Hardware Handshaking

It is quite possible that the WB_UART8 sends more data than the remote station's receiver can cope with and vice-versa. In this case, some sort of control is required to halt the transmission of data in either direction. The WB_UART8 provides automatic, hardware controlled handshaking – commonly referred to as RTS/CTS handshaking.

Hardware handshaking is enabled by setting the `ctsen` bit of the Handshake register (HSK.0).

When the remote station is ready to receive data, the WB_UART8's CTS_I input goes High. The WB_UART8 is free to transmit provided this input remains High.

When the WB_UART8 is ready to receive, it sets its RTS line Low. The remote station will continue to transmit data as long as this output is Low. The WB_UART8 can control this output automatically, based on the defined watermark levels for the Receive Buffer, or manually using the `forcerts` and `rtsval` bits in the Handshake register (HSK.1 and HSK.2 respectively).

Initialization

After a reset of the WB_UART8, you may want to initialize the device and set it up ready in accordance with design requirements. Initialization could include:

- Enabling hardware handshake by setting the `ctsen` bit in the Handshake register
- Writing the required values to the watermark registers (TXLOWMARK, RXHIGHMARK, RXLOWMARK)
- Writing the required clock divisor value to the Baud Rate Generator register
- Enabling the required interrupt bits in the INTCTRL register
- Loading the required receive delay time value into the RXDTO register.

Revision History

Date	Version No.	Revision
29-Jul-2005	1.0	Initial Release
12-Dec-2005	1.1	Path references updated for Altium Designer 6
03-Oct-2006	1.2	Updated for Altium Designer 6.6.
07-Nov-2006	2.0	Updated for Altium Designer 6.7.
11-Mar-2008	3.0	Updated for Altium Designer Summer 08
01-Aug-2008	3.1	Fixed description for INT_O signal in Table 1.