

General Description

The StdUart is a Universal Asynchronous Receiver/Transmitter (UART) IP core based on the industry standard 16550 UART. The StdUart performs a serial-to-parallel conversion on the receive data stream, checking the start, stop and parity bits as required. In the transmit direction the StdUart performs a parallel-to-serial conversion on data received from the CPU, generating the required start, stop and parity bits to form the transmit data stream.

Build-Time Options

- Transmit FIFO depth: 1 to 256 (16 by default).
- Receive FIFO depth: 1 to 256 (16 by default).
- Receive FIFO interrupt trigger levels.
- 1 cycle or 2 cycle read accesses.

Features

- Compatible with all existing 16550 UART driver software.
- 16 byte (by default) transmit FIFO.
- 16 byte (by default) receive FIFO with error flags.
- Programmable baud rate generator.
- 5, 6, 7, or 8 bit character length.
- Even, odd, forced-1, forced-0, or no parity bit generation and checking.
- 1, 1.5, or 2 stop bit generation.
- Line break generation and detection.
- Polled and interrupt operation supported.
- Modem control signals nCTS, nRTS, nDSR, nDTR, nDCD, and nRI supported.
- Internal loopback mode included for self-test.
- Core uses a single clock domain.
- Industry standard PPCI compliant processor interface

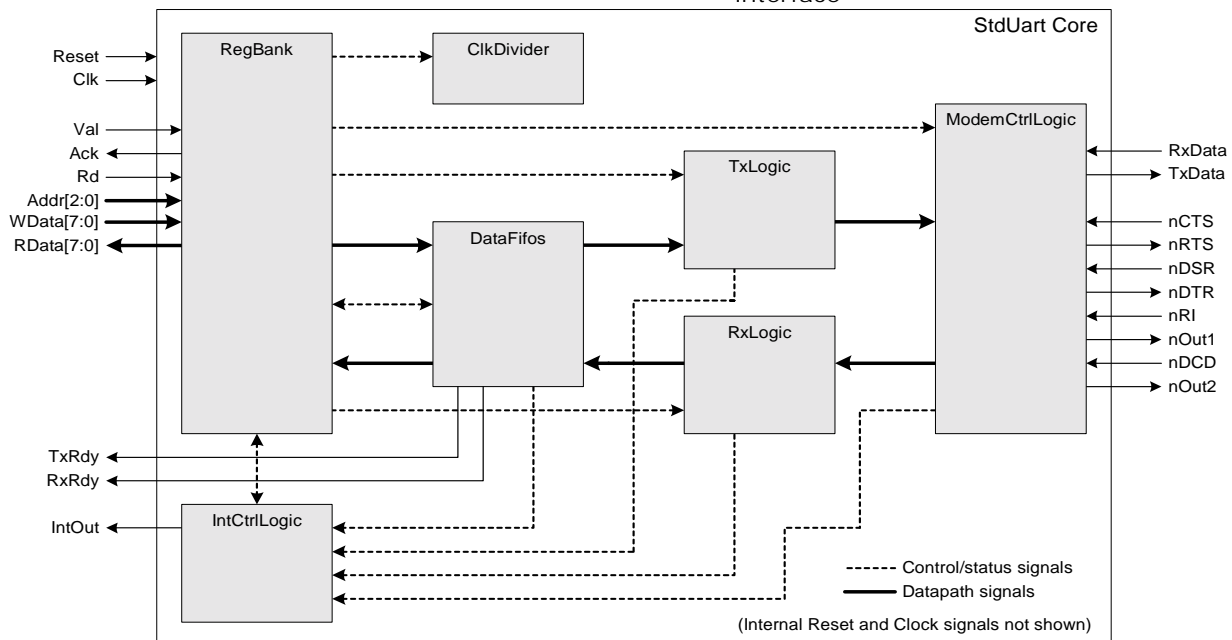


Figure 1: StdUart Core Block Diagram

Functional Description

The StdUart core block diagram shows the seven blocks that form the StdUart. The function of each of these blocks is described in turn next.

RegBank

The *Register Bank* block implements the processor interface and the register bank. The values written to the control registers in the register bank are used to control all the other blocks in the StdUart.

ClkDivider

The *Clock Divider* block divides the main clock supplied to the StdUart to form the 16x data rate

clock used by both the receive and transmit logic.

DataFifos

The *Data FIFOs* block contains the transmit and receive FIFOs as well as the THR (Transmit Holding Register) and the RHR (Receive Holding Register).

TxLogic

The *Transmit Logic* block takes the data, one character at a time, from either the transmit FIFO or the THR and performs the parallel-to-serial conversion. At this time it also inserts the required start, stop, and parity bits to generate the final transmit datastream.

RxLogic

The *Receive Logic* block takes the received datastream, detects and checks the start, parity, and stop bit. The Receive Logic block then performs the serial-to-parallel conversion on the received data and writes the data to either the RHR or the receive FIFO ready for the processor to read.

ModemCtrlLogic

The *Modem Control Logic* block is responsible for generating the internal loop-back mode.

IntCtrlLogic

The *Interrupt Control Logic* block contains the logic for generating interrupts on the IntOut line. It is also responsible for generating the contents for the ISR (Interrupt Status Register), LSR (Line Status Register), and the MSR (Modem Status Register).

Port & Generic Descriptions

The StdUart core has 37 ports at its top level, which are described in Table 1 below.

| Port | Direction | Description |
|----------------------------------|-----------|--|
| System Ports | | |
| Reset | Input | Asynchronous reset signal |
| Clk | Input | Master clock input |
| Processor Interface Ports | | |
| Val | Input | Valid strobe input. |
| Ack | Output | Acknowledge strobe output. |
| Rd | Input | Read/Not Write strobe input. |
| Addr[2:0] | Input | 3-bit address bus for register read/writes. |
| WData[7:0] | Input | 8-bit Write Data bus for register writes. |
| RData[7:0] | Output | 8-bit Read Data bus for register reads. |
| IntOut | Output | Processor interrupt signal. |
| DMA Interface Ports | | |
| RxRdy | Output | Receiver Ready DMA output. |
| TxRdy | Output | Transmitter Ready DMA output. |
| Modem Interface Ports | | |
| RxData | Input | Receive serial data input. |
| TxData | Output | Transmit serial data output. |
| nCTS | Input | Modem CLEAR TO SEND signal (active low). |
| nRTS | Output | Modem REQUEST TO SEND signal (active low). |
| nDSR | Input | Modem DATA SET READY signal (active low). |
| nDTR | Output | Modem DATA TERMINAL READY signal (active low). |
| nRI | Input | Modem RING INDICATOR signal (active low). |
| nOUT1 | Output | Modem general purpose output 1 (active low). |
| nDCD | Input | Modem DATA CARRIER DETECT (active low). |
| nOUT2 | Output | Modem general purpose output 2 (active low). |

Table 1: StdUart core ports

The StdUart core has 7 generics than can be modified when the component is instantiated to control the build-time options. These are listed in Table 2 below.

| Generic | Range | Description |
|--------------|----------|--|
| gTxFifoDepth | 1 to 256 | Transmit FIFO depth (16 by default). |
| gRxFifoDepth | 1 to 256 | Receive FIFO depth (16 by default). |
| gFifoTrig0 | 1 to 256 | Receive FIFO interrupt trigger level when FCR[7:6] = 00 (1 by default). |
| gFifoTrig1 | 1 to 256 | Receive FIFO interrupt trigger level when FCR[7:6] = 01 (4 by default). |
| gFifoTrig2 | 1 to 256 | Receive FIFO interrupt trigger level when FCR[7:6] = 10 (8 by default). |
| gFifoTrig3 | 1 to 256 | Receive FIFO interrupt trigger level when FCR[7:6] = 11 (14 by default). |
| gExtndRd | 0 to 1 | 0 = Normal read accesses. 1 = Extended read accesses. (0 by default) |

Table 2: StdUart core generics

Processor Interface

The processor interface is a synchronous interface, with all signals being generated on the rising edge of Clk and sampled on the subsequent rising edge of Clk.

Processor Read Cycle

The processor initiates a read cycle by driving the Addr[2:0] lines with the address or the register to be read, the Rd signal high to indicate it is a read access and the Val signal high to indicate the other signals are valid.

The processor then waits in this state until the device drives the RData[7:0] lines with the data and drives the Ack line high. Once the processor has seen the Ack signal high and latched the data on the RData lines, the read cycle is complete.

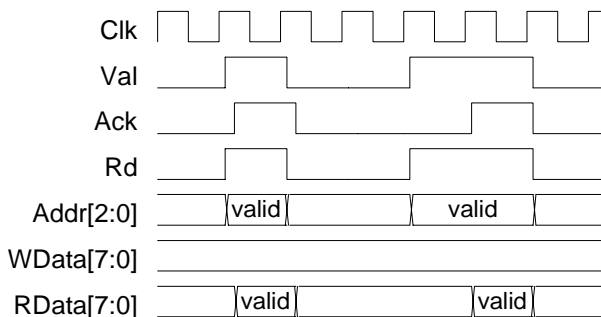


Figure 2: Example processor read cycles

The waveforms in Figure 2 show two separate read accesses. In the first access the peripheral asserts the Ack signal soon after the Val signal has gone high (using combinatorial logic) and the read cycle completes in 1 Clk cycle.

In the second read access, the peripheral does not assert the Ack signal until 1 Clk cycle after the Val signal has been asserted. Because of this the read cycle is extended and takes 2 Clk cycles.

The StdUart core behaves like the first read access in Figure 2 when the generic gExtndRd is set to 0, with the Ack signal effectively being the Val signal fed back to the processor.

When gExtndRd is set to 1 the StdUart core behaves like the second read access in Figure 2, with the Ack signal being asserted 1 Clk cycle after the Val signal is asserted. This option is designed to be used when the frequency of Clk is too high to complete a read access in one Clk cycle.

Processor Write Cycle

The processor initiates a write cycle by driving the required register address onto the Addr[2:0] lines, the data to be written onto the WData lines,

the Rd signal low and the Val signal high to indicate the other signals are valid.

The processor then waits until the peripheral asserts the Ack signal to indicate the write cycle is complete.

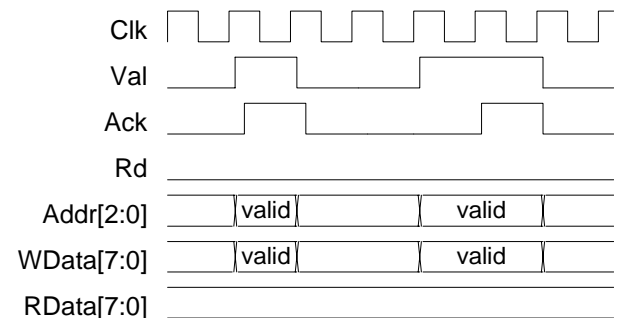


Figure 3: Example processor write cycles

The waveforms in Figure 3 show two separate write accesses. Again, in the first access the peripheral asserts the Ack signal soon after the Val signal has gone high (using combinatorial logic) and the write cycle is complete in 1 Clk cycle.

In the second write access, the peripheral does not assert the Ack signal until 1 cycle after the Val signal has gone high. Again this causes the processor to extend the cycle until the Ack signal is seen, and thus the write cycle takes 2 Clk cycles.

The StdUart core always behaves like the first write access in Figure 3, with the Ack signal just being the Val signal fed back to the processor.

Notes:

- Read/write accesses can occur back-to-back.
- Although the StdUart core's write access always behaves like the first access in Figure 3, it is advisable that any bridge logic is able to handle the extended version of the accesses also, as they may be used in future versions of the core.

Register Descriptions

The StdUart has 12 internal 8-bit registers. Which register is accessed depends on Addr[2:0], whether the access is a read or a write access and the value of bit 7 of the LCR. Table 3 and Table 4 below summarise these registers.

| Addr[2:0] | R/W | Register |
|-----------|-----|------------------------------|
| 000 | R | RHR: Receiver Holding Reg |
| 000 | W | THR: Transmitter Holding Reg |
| 001 | R/W | IER: Interrupt Enable Reg |
| 010 | R | ISR: Interrupt Status Reg |
| 010 | W | FCR: FIFO Control Reg |
| 011 | R/W | LCR: Line Control Reg |
| 100 | R/W | MCR: Modem Control Reg |
| 101 | R | LSR: Line Status Reg |
| 110 | R | MSR: Modem Status Reg |
| 111 | R/W | Scratchpad Reg |

Table 3: StdUart registers when LCD[7] = 0

| Addr[2:0] | R/W | Register |
|-----------|-----|--------------------------|
| 000 | R/W | DivLo: Clock Divisor LSB |
| 001 | R/W | DivHi: Clock Divisor MSB |

Table 4: StdUart registers when LCD[7] = 1

RHR: Receiver Holding Register (0x0)

This read-only register is where received characters are read from the StdUart by the processor. When the StdUart is not in FIFO mode (FCR[0]=0) this register is the 1 byte Receiver Holding Register, when in FIFO mode it is the Receive FIFO.

THR: Transmitter Holding Register (0x0)

This write-only register is where the processor writes characters to for the StdUart to transmit. When the StdUart is not in FIFO mode (FCR[0]=0) this register is the 1 byte Transmitter Holding Register, when in FIFO mode it is the Transmit FIFO.

IER: Interrupt Enable Register (0x1)

This read/write register controls interrupt masking, setting a bit to 1 enables the associated interrupt.

| Bit | Default | Description |
|-----|---------|-------------------------------------|
| 0 | 0 | Enable Receiver Data Available Int. |
| 1 | 0 | Enable Transmitter Empty Int. |
| 2 | 0 | Enable LSR Interrupt |
| 3 | 0 | Enable MSR Interrupt |
| 7:4 | 0000 | Reserved |

Table 5: IER bits

ISR: Interrupt Status Register (0x2)

This read-only register specifies the source of the highest priority interrupt pending. It also indicates whether or not the FIFOs are enabled.

| Bit | Default | Description |
|-----|---------|-----------------------|
| 0 | 0 | No Interrupt Pending |
| 1 | 0 | Int Source ID – bit 0 |
| 2 | 0 | Int Source ID – bit 1 |
| 3 | 0 | Int Source ID – bit 2 |
| 5:4 | 00 | Reserved |
| 7:6 | 00 | FIFO Mode Enabled |

Table 6: ISR bits

Bits 6 and 7 are both set to 1 when the FIFO mode is enabled.

Bits 0 to 3 indicate whether an interrupt is pending and the highest priority interrupt pending source if one or more interrupts are pending. Table 7 lists the meanings of the Interrupt Source ID bits.

| Priority | ISR[3:0] | Interrupt Source |
|----------|----------|------------------------|
| - | 0001 | No Interrupt Pending |
| 1 | 0110 | LSR |
| 2 | 0100 | Received Data Ready |
| 2 | 1100 | Received Data Time-Out |
| 3 | 0010 | THR or Tx FIFO Empty |
| 4 | 0000 | MSR |

Table 7: ISR Source ID Table

FCR: FIFO Control Register (0x2)

This write-only register is used enable and control the FIFOs, including selecting DMA mode.

| Bit | Default | Description |
|-----|---------|-------------------------------|
| 0 | 0 | Enable FIFO Mode |
| 1 | 0 | Reset Rx FIFO (Self Clearing) |
| 2 | 0 | Reset Tx FIFO (Self Clearing) |
| 3 | 0 | DMA Mode 0 or 1 Select |
| 5:4 | 00 | Reserved |
| 6 | 0 | Rx FIFO Trigger Level – bit 0 |
| 7 | 0 | Rx FIFO Trigger Level – bit 1 |

Table 8: FCR bits

The FIFO trigger level bits control the number of characters in the receive FIFO before a Received Data Ready interrupt occurs.

| FCR[7:6] | Rx FIFO Trigger Level |
|----------|----------------------------|
| 00 | gFifoTrig0 (1 by default) |
| 01 | gFifoTrig1 (4 by default) |
| 10 | gFifoTrig2 (8 by default) |
| 11 | gFifoTrig3 (14 by default) |

Table 9: Receive FIFO trigger levels

LCR: Line Control Register (0x3)

This read/write register is used to control the data communication format used for both transmit and receive.

| Bit | Default | Description |
|-----|---------|-------------------------------------|
| 0 | 0 | Character Length – bit 0 |
| 1 | 0 | Character Length – bit 1 |
| 2 | 0 | Number Of Stop bits |
| 3 | 0 | Enable Parity Generation |
| 4 | 0 | Parity Select – bit 0 |
| 5 | 0 | Parity Select – bit 1 |
| 6 | 0 | Set Tx Break Condition |
| 7 | 0 | Enable Clk Divisor Registers Access |

Table 10: LCR bits

Bit 7 of the LCR enables processor access to the clock divisor registers, DivLo and DivHi. When this bit is set to 1, the processor can access the clock divisor registers at addresses 0x0 and 0x01. When set to 0, the processor accesses the RHR, THR and IER at these addresses.

Setting Bit 6 of the LCR to 1 forces a break condition on the transmit line (TxData is forced to 0). Setting this bit back to 0 clears the break condition and restores normal operation.

The remaining LCR bits control the character length, number of stop bits and parity type used for both transmit and receive. Their use are described in Table 11, Table 12 and Table 13 below.

| LCR[1:0] | Character Length |
|----------|------------------|
| 00 | 5 |
| 01 | 6 |
| 10 | 7 |
| 11 | 8 |

Table 11: Character Length Values

| LCR[2] | Character Lengths | Number of Stop Bits |
|--------|-------------------|---------------------|
| 0 | 5, 6, 7, 8 | 1 |
| 1 | 5 | 1.5 |
| 1 | 6, 7, 8 | 2 |

Table 12: Number Of Stop Bits Values

| LCR[5:3] | Parity Type |
|----------|-----------------|
| XX0 | No Parity |
| 001 | Odd Parity |
| 011 | Even Parity |
| 101 | Forced 1 Parity |
| 111 | Forced 0 Parity |

Table 13: Parity Type Values

MCR: Modem Control Register (0x4)

This read/write register controls the modem control output signals and enables loop-back mode.

| Bit | Default | Description |
|-----|---------|------------------------------------|
| 0 | 0 | Assert nDTR (active low) when set |
| 1 | 0 | Assert nRTS (active low) when set |
| 2 | 0 | Assert nOut1 (active low) when set |
| 3 | 0 | Assert nOut2 (active low) when set |
| 4 | 0 | Enable Loop-back mode |
| 7:5 | 000 | Reserved |

Table 14: MCR bits

LSR: Line Status Register (0x5)

This read-only register provides information about the status of receive and transmit data transfers.

| Bit | Default | Description |
|-----|---------|------------------------------------|
| 0 | 0 | RHR or receive FIFO contains data |
| 1 | 0 | Receive data overrun error |
| 2 | 0 | Parity error in received character |
| 3 | 0 | Framing error in rx character |
| 4 | 0 | Break detected by receiver |
| 5 | 0 | THR or transmit FIFO is empty |
| 6 | 0 | Transmitter pipeline is empty |
| 7 | 0 | Error in receive FIFO data |

Table 15: LSR bits

Bit 0 is set when a complete character is received and written to the RHR or the receive FIFO. It is cleared when the processor reads all the data from the RHR (1 character only) or the receive FIFO.

Bit 1 is set when a new character is received when the RHR or the receive FIFO is full. The new character is discarded and the data in the RHR or the receive FIFO remains intact. This bit is cleared when the LSR is read.

Bit 2 is set when the character in the RHR or the next character to be read from the receive FIFO contains a parity error. This bit is cleared when the LSR is read.

Bit 3 is set when the character in the RHR or the next character to be read from the receive FIFO has a framing error (no stop bit). This bit is cleared when the LSR is read.

Bit 4 is set when the receiver detects a break condition (RxData is 0 for one character frame time). This bit is cleared when the LSR is read. For each break condition only 1 break character will be loaded into the RHR or the receive FIFO.

Bit 5 is set when the THR or the transmit FIFO is empty. Note that the transmit logic may still

contain data and the transmitter may still be sending data. This bit is cleared when the processor writes any data to the THR or the transmit FIFO.

Bit 6 is set when the THR or the transmit FIFO and the transmit logic is empty. When this bit is set, the transmitter is idle. This bit is cleared when the processor writes any data to the THR or the transmit FIFO.

Bit 7 is always cleared when the FIFOs are not enabled. In FIFO mode this bit is set when there is at least one parity error, framing error or break error in the receive FIFO. This bit is cleared when the LSR is read and no errors remain in the receive FIFO.

MSR: Modem Status Register (0x6)

This read-only register provides information about the state of the modem control inputs.

| Bit | Default | Description |
|-----|---------|---------------------------------|
| 0 | 0 | nCTS has changed since MSR read |
| 1 | 0 | nDTR has changed since MSR read |
| 2 | 0 | nRI has changed since MSR read |
| 3 | 0 | nDCD has changed since MSR read |
| 4 | 0 | Inverted value of nCTS input |
| 5 | 0 | Inverted value of nDTR input |
| 6 | 0 | Inverted value of nRI input |
| 7 | 0 | Inverted value of nDCD input |

Table 16: MSR bits

Bits 3:0 are set when their associated modem control inputs change and are only cleared when the MSR is read, even if their modem control inputs revert to their original values.

Bits 7:4 always reflect the values on their associated modem control inputs.

Scratchpad Register (0x7)

This register has no effect on the operation of the StdUart. It may be used to hold a byte of data. Default value = 0xFF.

DivLo (0x0) & DivHi (0x1): Clock Divisor

These two read/write registers are only accessible when bit 7 of LCR is set to 1. Together these two registers form a 16-bit value ClkDiv, where:

$$\text{ClkDiv} = (\text{DivHi} \times 256) + \text{DivLo}.$$

The Clock Divider block uses the ClkDiv value to produce the 16x data rate clock. The required ClkDiv value should be calculated with:

$$\text{ClkDiv} = (\text{Clk frequency}) / (16 \times \text{serial data rate})$$

Loop-back Mode

The StdUart has a loop-back mode included to allow diagnostic testing. This mode is enabled with bit 4 of MCR.

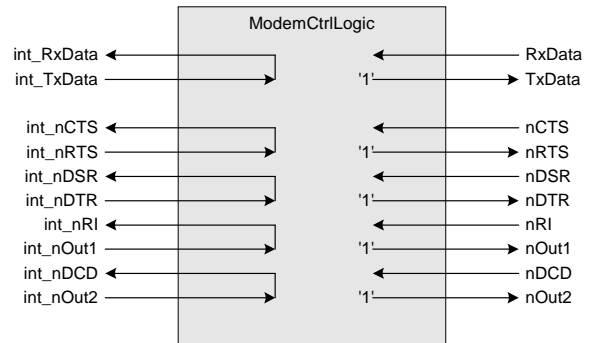


Figure 4: Modem signals in loop-back mode

In loop-back mode, the data signals and modem control signals are looped back in the modem control logic as shown in Figure 4. The signals prefixed with *int_* are signals internal to the StdUart.

Interrupts

The StdUart generates the interrupts listed below when enabled. Interrupts are signalled to the processor by setting the IntOut output signal to a logic 1. The IntOut signal remains in this state until the interrupt (and all other pending interrupts) are cleared.

LSR Interrupt

Enabled by: IER[2]

ISR[3:0]: 0110

Description: This interrupt is generated when any one or more of the LSR bits 1, 2, 3 or 4 are asserted. That is when there is an overrun error, a parity error, a framing error or a break error in either the character in the RHR or next character to be read from the receive FIFO.

Cleared by: Reading the LSR.

Received Data Ready Interrupt

Enabled by: IER[0]

ISR[3:0]: 0100

Description: In FIFO mode this interrupt is generated when the receive FIFO is filled to the trigger level specified by FCR[7:6]. In non-FIFO mode this interrupt is generated when the RHR contains a received character.

Cleared by: Reading enough data from the receive FIFO to empty it below its receive trigger level (or reading the character from the RHR) or resetting the receive FIFO.

Received Data Time-out Interrupt

Enabled by: IER[0]

ISR[3:0]: 1100

Description: This interrupt is generated when the receive FIFO is not empty and at least 4 character frame times has passed since either the processor read a character from the receive FIFO or a new character was received and written to the receive FIFO by the Receive Logic block. This interrupt does not occur in non-FIFO mode.

Cleared by: Reading data from the receive FIFO or resetting the receive FIFO.

THR or Transmit FIFO Empty Interrupt

Enabled by: IER[1]

ISR[3:0]: 0010

Description: This interrupt is generated when the transmit FIFO (in FIFO mode) or the THR is empty.

Cleared by: Reading the ISR or writing data to the THR/transmit FIFO.

MSR Interrupt

Enabled by: IER[3]

ISR[3:0]: 0000

Description: This interrupt is generated when one or more of the MSR bits 0, 1, 2 or 3 are set. That is when one or more of the modem control inputs (nCTS, nDTR, nRI, nDCD) have changed since the MSR was last read.

Cleared by: Reading the MSR.

DMA Mode Operation

The StdUart may be operated using DMA by adding a DMA controller utilising the two output signals RxRdy and TxRdy. The operation of the RxRdy and TxRdy outputs is different in DMA mode 0 (FCR[3]=0) and DMA mode 1 (FCR[3]=1). Table 17 and Table 18 below describe the operation of the RxRdy and TxRdy outputs in each of the DMA modes.

| DMA Mode 0 | DMA Mode 1 |
|---|--|
| RxRdy=0: Rx FIFO is empty. | Falling edge RxRdy: Rx FIFO has emptied. |
| RxRdy=1: Rx FIFO contains at least 1 character. | Rising edge RxRdy: Rx FIFO has reached its trigger level, or a receive data time-out has occurred. |

Table 17: RxRdy Output Operation

| DMA Mode 0 | DMA Mode 1 |
|---|---|
| TxRdy=0: Tx FIFO contains at least 1 character. | TxRdy=0: Tx FIFO is full. |
| TxRdy=1: Tx FIFO is empty. | TxRdy=1: Tx FIFO has space for at least 1 more character. |

Table 18: TxRdy Output Operation

Sweeney Design Ltd

40 Hopkins Close
Cambridge
CB4 1FD
United Kingdom

Tel: +44 (0) 7977 129406
Fax: +44 (0) 7977 045842
E-mail: cores@sweeneydesign.co.uk
URL: www.sweeneydesign.co.uk