

Universal Asynchronous Receiver/Transmitter (UART) Data Sheet



••••• Proven System Block (PSB) for QuickLogic Customer Specific Standard Products (CSSPs)

Device Highlights

- Transmitter and receiver with independent 256-byte FIFOs to reduce the number of interrupts to the processor.
- Accommodates 5, 6, 7, or 8 bits per character
- Accommodates 1, 1.5, or 2 stop bits
- Accommodates even, odd, or stick parity with enable/disable
- Prioritized and independently controlled interrupts
- Maximum baud rate up to 3 Mbits/sec.
- Programmable baud generator
- Modem control interface
- Complete status reporting capabilities
- Line break and false start bit detection
- DMA signaling
- Software compatible with 16550 UARTs

Overview

UART is a popular method of serial asynchronous communication. Typically, the UART is connected between a processor and a peripheral. To the processor, the UART appears as an 8-bit read-write parallel port that performs serial-to-parallel conversions for the processor, and vice versa for the peripheral.

This 16550 UART core emulates the functionality of National Semiconductor's PC16550D UART with FIFOs. It is capable of communicating with any other 16550-compliant UART.

UART Architecture

The UART core consists of a transmitter, a receiver, a modem interface, a baud generator, an interrupt controller, and various control and status registers.

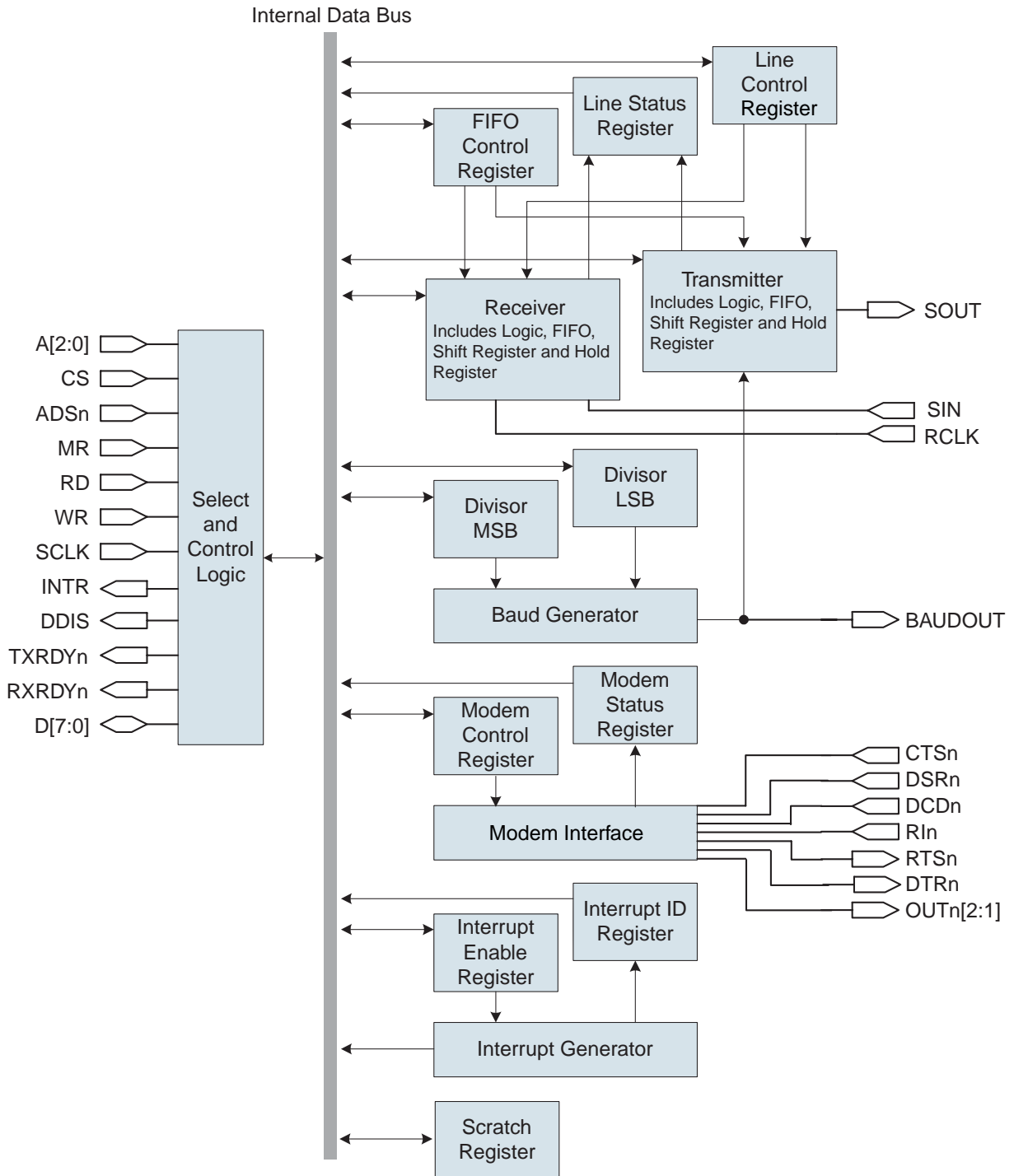
The transmitter and receiver perform parallel-to-serial and serial-to-parallel conversions, respectively. Each has a 256-byte FIFO to buffer data flow. The use of FIFO buffers increases the overall transmission rate by allowing slower processors to respond, and reducing the amount of time wasted on context switching. Besides data transfer, they also facilitate start/stop framing bits, check various parity options, as well as detect transmission errors.

The modem interface allows the processor to transmit data to another terminal or handshake to the modem input signals. It also makes the modem's status available to the processor. The baud generator creates a baud rate clock and a receiver reference clock.

The interrupt controller works with the transmitter, receiver, and modem interface to notify the processor of events that require intervention. Some examples of these events are transmission errors and modem status changes.

The processor has access to the complete status of the UART at any time. Information reported includes the line condition, modem status, as well as any errors such as parity, overrun, framing, or line break. The processor can also configure the UART's operation via various control registers. **Figure 1** is an architectural view of the UART core.

Figure 1: UART Core Architecture



Interface List and Description

Table 1 provides details on the core's input and output signals.

Table 1: UART Pin Description

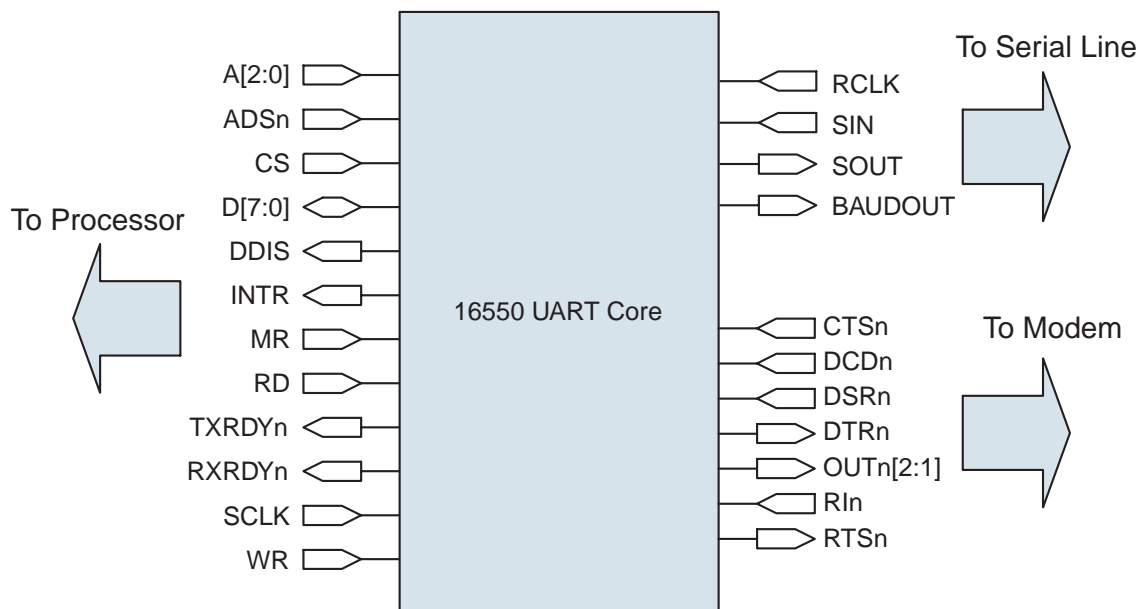
Pin	Type	Description
System Interface		
A[2:0]	I	Address lines. Selects an internal UART register during data transfer to and from the processor.
ADSn	I	Address strobe. Active low. The UART recognizes A[2:0] and CS when ADSn = 0. This signal is used when A[2:0] and CS are unstable for the duration of an access; else, tie ADSn to low.
CS	I	Chip select. The UART module is selected if CS = 1.
D[7:0]	I/O	Data bus. A bi-directional bus between the UART and the processor.
DDIS	O	Driver disable. Active low. This signal goes low when the UART is driving the data bus; it can be used to control a transceiver between the processor and the UART.
INTR	O	Interrupt. This signal goes high whenever an interrupt situation occurs.
MR	I	Master reset. When this signal is high, it initializes all registers and control logic.
RD	I	Read. When RD = 1, the processor can read from the UART. This signal is used in conjunction with A[2:0], CS, and ADSn.
TXRDYn	O	Transmitter ready. Active low. This is used for DMA signaling, and has two modes of operation: mode 0 supports single transfer DMA, while mode 1 supports multiple transfer DMA.
RXRDYn	O	Receiver ready. Active low. This is used for DMA signaling, and has two modes of operation: mode 0 supports single transfer DMA, while mode 1 supports multiple transfer DMA.
SCLK	I	System clock. Internal state machines and the baud rate generator all operate at this frequency.
WR	I	Write. When WR is high and chip is selected, the processor can write to the UART. This signal is used in conjunction with A[2:0], CS, and ADSn.
Serial Interface		
BAUDOUT	O	Baud rate X 16 clock output. This clock is obtained by dividing the SCLK input by the Baud Generator Divisor Registers.
RCLK	I	Receiver reference clock. This is the baud rate X 16 clock for the receiver module.
SIN	I	Serial input. Serial data input from the transmission line.
SOUT	O	Serial output. Serial data output to the transmission line.
Modem Interface		
CTSn	I	Clear to send. Active low. When low, this indicates that the modem is ready to send data to the UART. CTS is a modem status signal. Its condition can be checked by reading bit 4 in the Modem Status Register. Bit 0 for the Modem Status Register indicates that CTS has changed states since the last read from the Modem Status Register.
DCDn	I	Data carrier detect. Active low. When low, this indicates that the data carrier has been detected by the modem.

Table 1: UART Pin Description (Continued)

Pin	Type	Description
DSRn	I	Data set ready. Active low. When low, this signal indicates that the modem is ready to establish the communications link with the UART.
DTRn	O	Data terminal ready. Active low. When low, this informs the modem that the UART is ready to establish a communications link.
OUTn[2:1]	O	User designated output. Active low. These can be controlled via the Modem Control Register.
RIn	I	Ring indicator. Active low. When low, this signal indicates that a telephone ringing signal has been detected by the modem.
RTSn	O	Request to send. Active low. When low, this informs the modem that the UART is ready to receive data. RTS is set to the active level when the RTS Modem Control Register bit is set and the Rx FIFO contains a number of data symbols less than 3/4 of its size (less than 192 bytes). RTS is set to the inactive (HIGH) level either as a result of clearing bit 1 (RTS) of the Modem Control Register, or when Rx FIFO contains a number of data symbols more than 3/4 of its size (192 bytes or more).

Figure 2 shows the UART core I/O pins.

Figure 2: UART Ports



Register Sets

Memory Map

Divisor Register Access Bit (DRAB) is the most significant bit of the Line Control Register. It is used in conjunction with the address lines to select the UART register. **Table 2** shows the register addresses.

Table 2: Register Addresses

DRAB	A[2:0]	Selected Register	Acronym
0	000	Receive FIFO (Read) Transmit FIFO (Write)	DATA
0	001	Interrupt Enable	IER
x	010	Interrupt Identification (Read) FIFO Control (Write)	IIR FCR
x	011	Line Control	LCR
x	100	Modem Control	MCR
x	101	Line Status	LSR
x	110	Modem Status	MSR
x	111	Scratch	SR
1	000	Divisor, least significant byte	DRLSB
1	001	Divisor, most significant type	DRMSB

Register Description

This section describes the core registers' functions and usage.

Line Control Register

This Read-Write register specifies the details of serial communication, such as data format, parity, and break control. **Table 3** shows the Line Control Register.

Table 3: Line Control Register

Bit Number	Function
7	Divisor registers access bit 0 = Read/Write cycles access the receive/transmit FIFOs, respectively. 1 = Read/Write cycles access the Divisor Registers. See Table 2 on page 5 for more details.
6	Break condition 0 = transmitter has control of SOUT. 1 = force SOUT to 0 and cause a break condition. Under a break condition, the transmitter finishes shifting out the bits in the shift register, and remains idle until the break condition is cleared.
5	Stick parity 0 = disable stick parity. 1 = enable stick parity. Used in conjunction with LCR[4:3]. When enabled with even parity, the parity bit is sent as 0. When enabled with odd parity, the parity bit is sent as 1.
4	Even parity 0 = enable odd parity. Used in conjunction with LCR[5] and LCR[3]. 1 = enable even parity. Used in conjunction with LCR[5] and LCR[3]. Even parity means the total number of '1's transmitted (data bits and parity bit) is even.
3	Parity enable 0 = send no parity bit after data bits. 1 = generate or check a parity bit after data bits. This has precedence over LCR[5:4].
2	Stop bits 0 = transmit 1 stop bit after data bits are sent. 1 = transmit 1.5 stop bits for a 5-bit character, or 2 stop bits for a 6-, 7-, or 8-bit character. The receiver checks the first stop bit only regardless of the number of stop bits selected.
[1:0]	Character length 00 = transmit 5 bits of data per frame. 01 = transmit 6 bit of data per frame. 10 = transmit 7 bit of data per frame. 11 = transmit 8 bit of data per frame.

Line Status Register

This Read-only register provides serial communication status to the processor, such as errors and flags.

Table 4 shows the Line Status Register.

Table 4: Line Status Register

Bit Number	Function
7	Error flag 0 = the receive FIFO has no errors. 1 = the receive FIFO has parity, framing, or break condition errors.
6	Transmitter empty flag 0 = transmitter is shifting data out to the serial line. 1 = the transmit FIFO and transmit shift register are both empty.
5	Transmit FIFO empty flag 0 = transmit FIFO has at least 1 piece of data remaining. 1 = the transmit FIFO is empty.
4	Break indication This is set to 1 if the receiver detects a string of 0 for longer than a full word transmission time. Under this condition, a 0x00 character is loaded into the receive FIFO, and the receiver remains idle until it detects a valid start bit. This is associated with a particular character in the FIFO.
3	Framing error This is set to 1 when the received character has an invalid stop bit. Under this condition, the UART will attempt to resynchronize by assuming that the framing error was due to the next start bit, so it samples this start bit twice and then takes in the following data. This is associated with a particular character in the FIFO.
2	Parity error This is set to 1 when the received character has an incorrect parity bit. This is associated with a particular character in the FIFO.
1	Overrun error This is set to 1 when the receiver FIFO is full and a completed character in the receive shift register is destroyed.
0	Receiver data ready 0 = the receive FIFO is empty. 1 = data is ready to be read from the receive FIFO.

FIFO Control Register

This Write-only register controls the operation of the FIFO buffers. **Table 5** shows the FIFO Control Register.

Table 5: FIFO Control Register

Bit Number	Function
[7:6]	Receiver FIFO trigger level 00 = trigger at 8 byte 01 = trigger at 8 bytes 10 = trigger at 64 bytes 11 = trigger at 128 bytes
[5:4]	Reserved These bits are always 0.
3	DMA signaling mode 0 = use DMA mode 0 (single transfer mode). 1 = use DMA mode 1 (multiple transfer mode). See Table 6 for details.
2	Clear transmit FIFO Set this to 1 to reset the transmit FIFO. This bit is self-clearing, and does not clear the transmit shift register.
1	Clear receive FIFO Set this to 1 to reset the receive FIFO. This bit is self-clearing, and does not clear the receive shift register.
0	Reserved This bit is always 0.

DMA Signaling Modes

The 16550 UART core has two operational modes, and the RXRDY_n and TXRDY_n signals behave differently depending on the mode. See **Table 6** for details.

Table 6: DMA Signaling Modes

Mode	RXRDY _n	TXRDY _n
0: Single Transfer	When there is at least 1 character in the receive FIFO, this signal will be low. This signal will go high again when there are no more characters in the FIFO.	When there are no characters in the transmit FIFO, this signal will be low. This signal will go high again after the first character is loaded into the FIFO.
1: Multiple Transfer	When the trigger level or the timeout has been reached, this signal will go low. This signal will go high again when there are no more characters in the FIFO.	When there are no characters in the transmitter FIFO, this signal will go low. This signal will go high again when the FIFO is completely full.

Interrupt Identification Register

To reduce software overhead, the UART prioritizes interrupts into four levels and records these in this Read-only register. **Table 7** shows the Interrupt Identification Register.

Table 7: Interrupt Identification Register

Bit Number	Function
[7:4]	Reserved These bits are always 4'b1100.
[3:1]	Interrupt ID These bits are used to identify the highest priority interrupt pending. See Table 8 for more details.
0	Interrupt pending, active low 0 = an interrupt is pending 1 = there are no interrupts

Table 8: Interrupt Table

IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset Condition
0110	Highest	Receiver status	1. Overrun error 2. Parity error 3. Framing error 4. Line break	Read the LSR
0100	Second	Received data	Receive FIFO trigger level reached	Receive FIFO drops below the trigger level
1100	Second	Character timeout	No character have been removed from, or input to, the receive FIFO during the last four character times, and there is at least one character in it during this time	Read the receiver
0010	Third	Transmitter empty	Transmit FIFO empty	1. Read the IIR 2. Write to transmitter
0000	Fourth	Modem status	1. Clear to send 2. Data set ready 3. Ring indicator 4. Data carrier detect	Read the MSR
0001	N/A	N/A	No interrupts pending	N/A

Interrupt Enable Register

This Read-Write register allows each type of interrupt to be independently enabled or disabled. **Table 9** shows the Interrupt Enable Register.

Table 9: Interrupt Enable Register

Bit Number	Function
[7:4]	Reserved These bits are always 0.
3	Enable Modem Status Interrupt Set this bit to 1 to enable this interrupt.
2	Enable Receiver Line Status Interrupt Set this bit to 1 to enable this interrupt.
1	Enable Transmit FIFO Empty Interrupt Set this bit to 1 to enable this interrupt.
0	Enable Receive Data Available Interrupt and Timeout Interrupt Set this bit to 1 to enable these interrupts.

Modem Control Register

This Read-Write register controls the modem interface. **Table 10** shows the Modem Control Register.

Table 10: Modem Control Register

Bit Number	Function
[7:4]	Reserved These bits are always 0.
3	User designated output 1 = set OUTn[2] to 0 0 = set OUTn[2] to 1
2	User designated output 1 = set OUTn[1] to 0 0 = set OUTn[1] to 1
1	Request to send 1 = set RTSn to 0 0 = set RTSn to 1
0	Data terminal ready 1 = set DTRn to 0 0 = set DTRn to 1

Modem Status Register

This Read-only register provides the current state of the signals from the modem. It also provides change information, where bits are set to 1 whenever a modem signal toggles, and set to 0 once the processor reads this register. **Table 11** shows the Modem Status Register.

Table 11: Modem Status Register

Bit Number	Function
7	Data Carrier Detect This is the inverted version of DCDn.
6	Ring Indicator This is the inverted version of RIn.
5	Data Send Request This is the inverted version of DSRn.
4	Clear to Send This is the inverted version of CTSn.
3	DCDn changed 1 = DCDn has changed since the last time it was read by the processor. 0 = processor has the most recent value of this signal.
2	RIn rising edge 1 = RIn has transitioned from 0 to 1. 0 = processor has the most recent value of this signal.
1	DSRn changed 1 = DSRn has changed since the last time it was read by the processor. 0 = processor has the most recent value of this signal.
0	CTSn changed 1 = CTSn has changed since the last time it was read by the processor. 0 = processor has the most recent value of this signal.

Scratch Register

This Read-Write scratchpad register does not control the UART in any way. The designer can use this to store any temporary data. **Table 12** shows the Scratch Register.

Table 12: Scratch Register

Bit Number	Function
[7:0]	User temporary data

Divisor Registers

The baud generator divides the system clock by the user-defined value in these read-write registers, and generates a new clock at the frequency of baud rate X 16. The receiving UART on the other side of the serial line can use this clock as its reference clock. The most significant byte and least significant byte of the Divisor Registers are individually accessible.

Table 1: Divisor Register

Bit Number	Function
[15:0]	Divisor value = (system clock frequency) / (baud rate X 16)

UART Functional and Module Description

UART Standard Serial Data Format

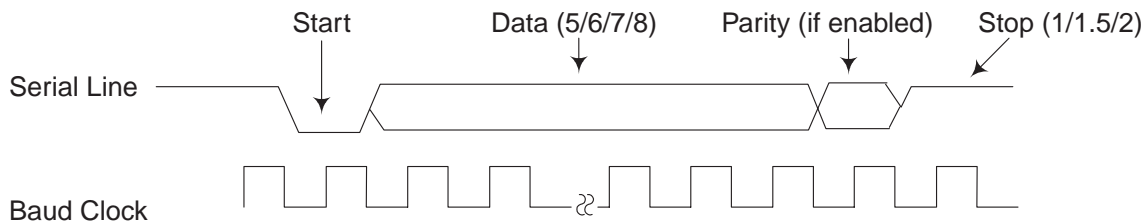
Standard UART characters are contained within frames consisting of start/stop indicators, data, and parity information. The serial bitstream uses the following sequence:

- Start bit
- Data bits
- Parity bit (if enabled)
- Stop bit(s)

When there is nothing to be sent, the serial line is high. The first low bit is the start bit, which indicates the beginning of a new frame. The next five to eight bits are data bits, which convey the actual information to be sent, least significant bit first. The number of data bits per frame is configurable.

If enabled, a parity bit will be sent on the serial line after the data bits. Finally, the serial line is held high again for at least one bit to indicate the end of a frame. This is called the stop bit, which also returns the serial line to the idle state. The type of parity and length of stop bits are also configurable. **Figure 3** shows the waveform of a frame.

Figure 3: Standard Serial Data Format



UART Module Descriptions

The UART core consists of five modules:

- Transmitter
- Receiver
- Modem interface
- Baud generator
- Interrupt controller

UART Transmitter

To send data with the UART, the processor writes data to the transmitter address as if it was a memory space. The transmitter controls the entire transmission process.

The transmitter performs parallel-to-serial conversions, and sends data on the serial line. The processor pushes data (5 to 8 bits wide) into the transmit FIFO, which is 256 bytes deep. The transmitter pops the data off the FIFO, and shifts it out at the baud rate. The Divisor Registers determine the baud rate.

Using the Line Control Register, the user can configure the number of data bits (5, 6, 7, or 8) per frame, as well as the number of stop bits (1, 1.5, or 2) to be sent at the end of a frame. The transmitter also has a parity generation circuit that is capable of creating even, odd, stick even, or stick odd parity. When parity generation is disabled, no parity will be sent after the data bits.

There is one interrupt associated with the transmitter—it is a third-level priority interrupt that occurs when the transmit FIFO is empty.

Logically, the transmitter performs the following steps:

1. If there is data available in the FIFO, loads a byte into the shift register.
2. Sends a start bit on the serial line, indicating the beginning of a frame.
3. Shifts out data bits from the shift register to the serial line.
4. If parity is enabled, sends a Parity Bit after all data bits are sent.
5. Sends stop bit(s) on the serial line, indicating the end of a frame.

UART Receiver

To receive data with the UART, the processor reads data from the receiver address as if it was a memory space. The receiver is responsible for capturing data from the serial line and validating data integrity.

The receiver performs serial-to-parallel conversions, and pushes data into the receive FIFO at the baud rate. The processor can pop data from the 256-byte deep receive FIFO at the system clock frequency. The supplied RCLK must be 16 times faster than the baud rate.

The receiver monitors the serial line. When a valid start bit is detected, the receiver begins to shift data bits from the serial line, and saves the received data in the receive FIFO.

Using the Line Control Register, the user can configure the number of data bits (5, 6, 7, or 8) per frame, as well as the type of parity to expect on the serial line. The receiver has a parity generation circuit that is capable of creating even, odd, stick even, or stick odd parity as data is shifted in from the serial line. The calculated parity is then checked with the received parity to determine data integrity. When parity generation is disabled, no parity will be expected after the data bits.

Besides parity errors, the receiver is also capable of detecting frame errors, overrun errors, and break errors. Frame errors occur when the receiver is expecting a stop bit but received a '0' on the serial line. Overrun errors occur when the receive FIFO is full and the newly received data is destroyed because it cannot be saved in the FIFO. Break errors occur when the serial line is '0' for more than a full frame. Break errors, frame errors, and parity errors are associated with the particular pieces of data in the FIFO that contains the errors. LSR[7] indicates if there are any errors in the entire receive FIFO.

In addition to error detection capabilities, the receiver also features false start bit detection and self-recovery from frame error. The serial line may be subject to noise, so it is essential that the receiver be able to differentiate noise from a valid start bit. For the receiver to recognize a '0' on the serial line as a valid start bit, this '0' value must remain for at least half a baud cycle from the falling edge of the serial input. If the start bit does not remain stable in that period, it is discarded and the receiver will return to its idle state.

The asynchronous nature of a serial line means it is possible that the receiver is out of synch from the transmitter, resulting in a frame error. When a frame error occurs, the receiver tries to resynchronize itself to the transmitter. To achieve this self-recovery feature, the receiver assumes that the invalid stop bit is actually the start bit of the next frame, and proceeds to shift in data bits from the serial line.

There are three interrupts associated with the receiver. The highest-level priority interrupt occurs when the received data has an error. A second-level priority interrupt occurs when the receive FIFO has reached its trigger level. Another second-level priority interrupt is the timeout interrupt, which occurs when data is sitting idle in the receive FIFO for over a certain period of time.

To perform all these functionalities, the receiver performs the following steps:

1. Hunts for a valid start bit, which indicates the beginning of a frame.
2. Shifts in data bits from the serial line to the shift register.
3. If parity is enabled, compares received parity bit with the expected value.
4. Checks for a valid stop bit on the serial line, which indicates the end of a frame.

Modem Interface

The modem interface allows the processor to transmit data to another terminal, or handshake to the modem input signals. It also makes the modem's status available to the processor.

The Modem Status Register reports status information in two ways. It shows the values of DCDn, Rin, DSRn, and CTSn signals from the modem, and it also indicates whether any of those signals have changed since the last time it was read by the processor. The user can drive the values of OUTn1, OUTn2, RTSn, and DTRn via the Modem Control Register.

Baud Generator

The baud generator is capable of creating a clock by dividing the system clock by any divisor from 2 to $2^{16}-1$. The divisor is the unsigned value stored in the Divisor Register. The output frequency of the baud generator is 16 times the baud rate. This means, the Divisor Register must hold a value equal to the system clock divided by baud rate and further divided by 16. This output clock can be used as the receive reference clock by the receiving UART.

Interrupt Controller

The interrupt controller works with the transmitter, receiver, and modem interface to notify the processor of events that require intervention. The user can configure which interrupts to enable or disable via the Interrupt Enable Register (IER). When the processor is interrupted, it can find out more details regarding the interrupt by reading the Interrupt Identification Register (IIR).

The interrupt controller monitors internal interrupt request lines from other modules. These internal interrupts are masked with the IER. The interrupt controller is then able to determine the interrupt pending with highest priority, and report it in the IIR. After the interrupt has been serviced, the interrupt controller clears the interrupt request and updates the IIR accordingly.

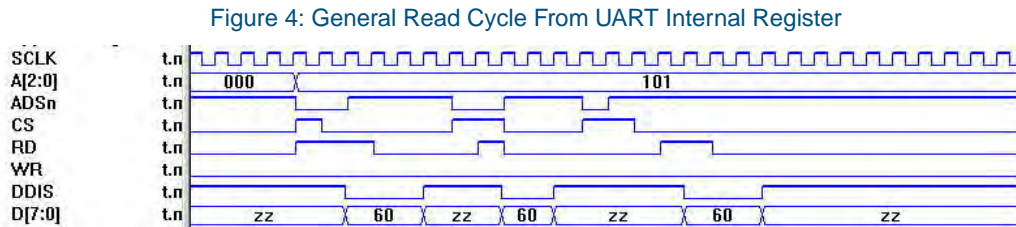
Timing Diagrams

The following diagrams illustrate the timing relationships between various signals during different operations.

General Read Cycle

As shown in **Figure 4**, the general Read cycle from the UART Internal Register is as follows:

1. When $ADSn = 0$ and $CS = 1$, the UART latches in the address.
2. RD can be asserted at any time after $ADSn = 0$ and $CS = 1$, even after $ADSn$ and CS are de-asserted.
3. $DDIS$ is driven low whenever the UART is driving the data bus.
4. After RD is de-asserted, the UART stops driving the data bus.
5. Data output hold time is two clock cycles after RD is de-asserted.

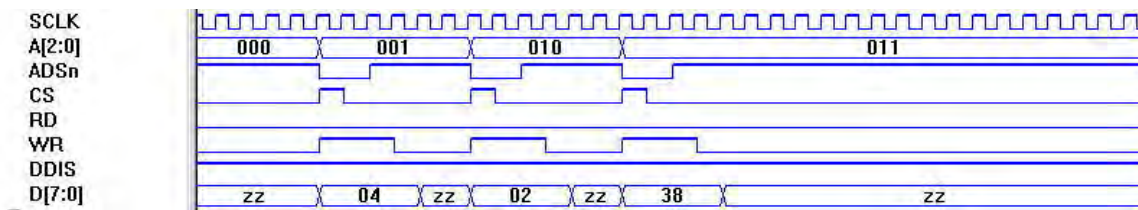


General Write Cycle

As shown in **Figure 5**, the general Write cycle to the UART Internal Register is as follows:

1. When $ADSn = 0$ and $CS = 1$, the UART latches in the address.
2. WR can be asserted at any time after $ADSn = 0$ and $CS = 1$, even after $ADSn$ and CS are de-asserted.
3. $DDIS$ is always high during write cycles.
4. The UART latches the value on the data bus when WR is de-asserted.
5. The UART does not require any data input hold time after WR is de-asserted.

Figure 5: General Write Cycle To UART Internal Register

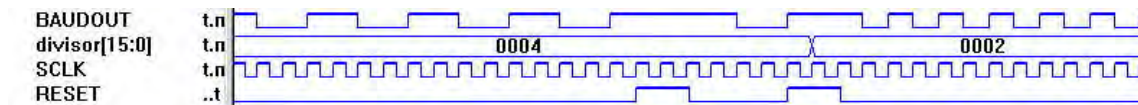


Baud Generator Operation

As shown in **Figure 6**, baud generator operation is as follows:

1. BAUDOUT frequency is the SCLK frequency divided by the value in the Divisor Registers.
2. During a reset, BAUDOUT is held high.
3. Coming out of reset, BAUDOUT starts with value '1'.
4. Whenever the Divisor Registers change value, the baud generator is reset.
5. Duty cycle of BAUDOUT is 50% when the divisor is an even number. When the divisor is an odd number, the duty cycle is less than 50% (by one SCLK cycle).

Figure 6: Baud Generator Operation



Serial Transmission and Reception

As shown in **Figure 7**, sending and receiving serial data is as follows.

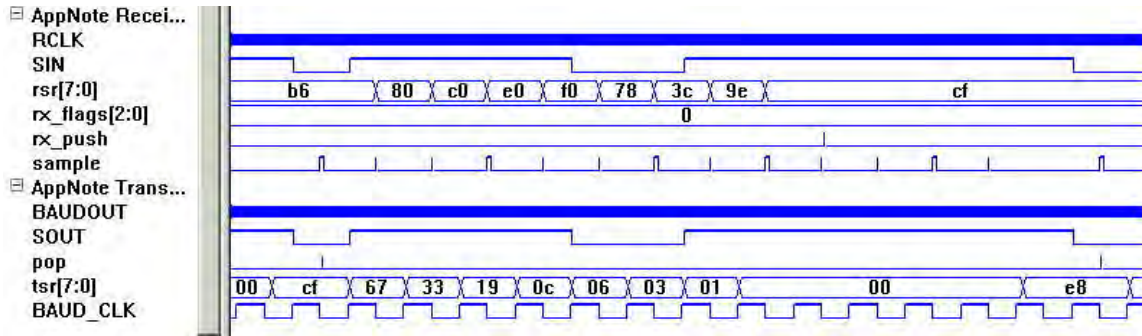
For receiving data:

1. The falling edge of SIN marks the beginning of a start bit.
2. The receiver generates a sample signal, which is the same frequency as the baud rate.
3. Bits are shifted into the receive shift register at every sample. The first sample is the start bit.
4. This example has 8 data bits, and no parity. When all bits are shifted into the receiver and the stop bit is received at the 9th sample, rx_push pushes the data (CFh) into the receive FIFO. rx_flags indicate any break, frame, and parity errors associated with this piece of data.
5. The serial line is sampled at the falling edge of baud clock, where data is stable.

For transmitting data:

1. The transmitter loads CFh from the top of the transmit FIFO into the transmit shift register, and pops the FIFO. At the same time, a start bit is sent by driving SOUT low.
2. At every rising edge of baud clock, a bit is shifted out to SOUT from the transmit shift register, LSB first.
3. After all data bits are shifted out, a stop bit is sent by driving SOUT high.
4. The transmitter remains idle until the next piece of data (E8h) is available.

Figure 7: Sending and Receiving Serial Data

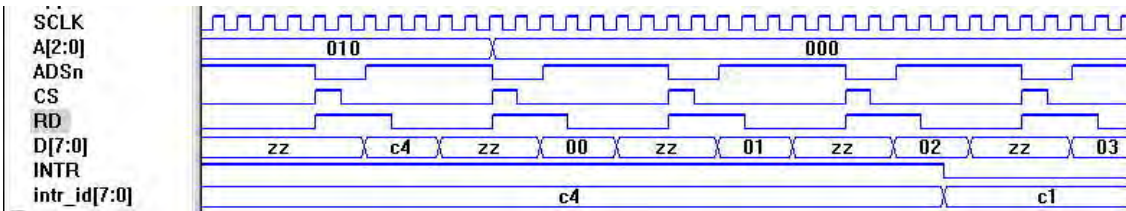


Interrupts

As shown in **Figure 8**, the Interrupt Signal and Interrupt Identification Register are as follows.

1. INTR is high, meaning the UART has at least one interrupt pending.
2. Reading the IIR yields the value of C4h, indicating that the receive FIFO has reached its trigger level.
3. After reading from the receiver a few times, the receive FIFO drops below its trigger level. This is reflected in the IIR, which now indicates no interrupts with a value of C1h.
4. INTR is lowered because IIR is C1h.

Figure 8: Interrupt Signal and Interrupt Identification Register



Resource Utilization

Table 13 shows the resource utilization for the UART IP.

Table 13: UART IP

Device	Utilization	Description
Eclipse II	<ul style="list-style-type: none"> I/Os: 31 I/Os, 2 Clk pads LCs: 300 Flip-flops: 255 RAM: 2 blocks 	Full featured 16550 UART core <ul style="list-style-type: none"> Maximum baud rate up to 3 Mbits/sec. (SCLK = 48 MHz) Programmable baud rate
PolarPro	<ul style="list-style-type: none"> I/Os: 39 I/Os, 2 Clk pads LCs: 275 Flip-flops: 212 RAM: 2 blocks 	Full featured 16550 UART core <ul style="list-style-type: none"> Maximum baud rate up to 3 Mbits/sec. (SCLK = 48 MHz) Programmable baud rate

Table 14 shows the resource utilization for the minimized UART IP.

Table 14: Minimized UART IP

Device	Utilization	Description
Eclipse II	<ul style="list-style-type: none"> • I/Os: 27 I/Os, 2 Clk pads • LCs: 200 • Flip-flops: 186 • RAM: 2 blocks 	Minimized UART core <ul style="list-style-type: none"> • Fixed baud rate • Maximum baud rate up to 3 Mbits/sec. (SCLK = 48 MHz) • 8 bit, no parity, 1 stop bit • Reduced register set
PolarPro	<ul style="list-style-type: none"> • I/Os: 27 I/Os, 2 Clk pads • LCs: 160 • Flip-flops: 140 • RAM: 2 blocks 	Minimized UART core <ul style="list-style-type: none"> • Fixed baud rate • Maximum baud rate up to 3 Mbits/sec. (SCLK = 48 MHz) • 8 bit, no parity, 1 stop bit • Reduced register set

Contact Information

Phone: (408) 990-4000 (US)
 (905) 940-4149 (Canada)
 +(44) 1932-57-9011 (Europe)
 +(852) 2567-5441 (Asia)

E-mail: info@quicklogic.com

Sales: America-sales@quicklogic.com
Europe-sales@quicklogic.com
Asia-sales@quicklogic.com
Japan-sales@quicklogic.com

Support: www.quicklogic.com/support

Internet: www.quicklogic.com

Revision History

Revision	Date	Originator and Comments
A	April 2008	David Shih, Rakesh Moolachari, and Kathleen Murchek
B	June 2008	Kathleen Murchek
C	October 2008	Jason Lew and Kathleen Murchek

Notice of Disclaimer

QuickLogic is providing this design, product or intellectual property "as is." By providing the design, product or intellectual property as one possible implementation of your desired system-level feature, application, or standard, QuickLogic makes no representation that this implementation is free from any claims of infringement and any implied warranties of merchantability or fitness for a particular purpose. You are responsible for obtaining any rights you may require for your system implementation. QuickLogic shall not be liable for any damages arising out of or in connection with the use of the design, product or intellectual property including liability for lost profit, business interruption, or any other damages whatsoever. QuickLogic products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use QuickLogic products in these types of equipment or applications.

QuickLogic does not assume any liability for errors which may appear in this document. However, QuickLogic attempts to notify customers of such errors. QuickLogic retains the right to make changes to either the documentation, specification, or product without notice. Verify with QuickLogic that you have the latest specifications before finalizing a product design.

Copyright and Trademark Information

Copyright © 2008 QuickLogic Corporation. All Rights Reserved.

The information contained in this document and is protected by copyright. All rights are reserved by QuickLogic Corporation. QuickLogic Corporation reserves the right to modify this document without any obligation to notify any person or entity of such revision. Copying, duplicating, selling, or otherwise distributing any part of this product without the prior written consent of an authorized representative of QuickLogic is prohibited.

QuickLogic, PolarPro, and the PolarPro design are registered trademarks; and the QuickLogic logo is a trademark of QuickLogic Corporation. All other trademarks are owned by their respective companies.