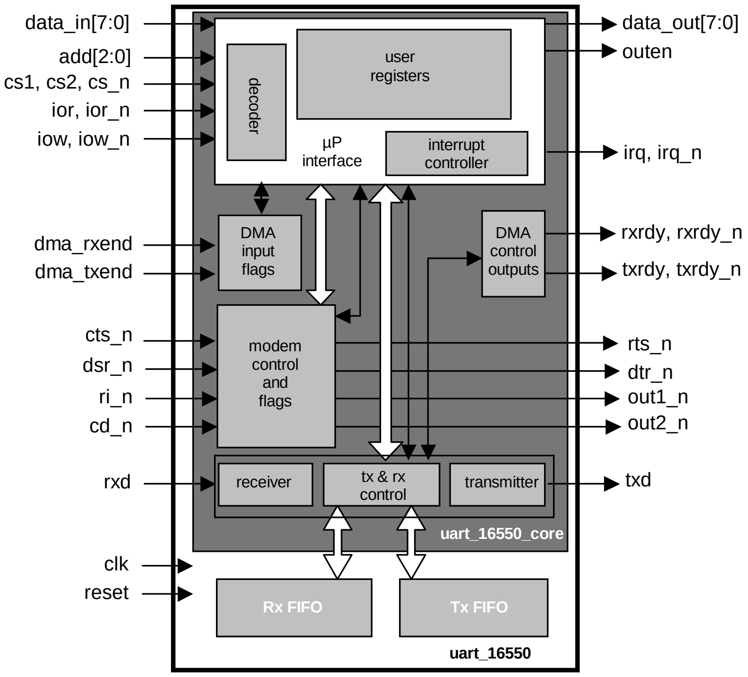
SIDSA

Semiconductor Design Solutions

UART 16550

IP



Overview

The UART_16550 IP is a Universal Asynchronous Receiver Transmitter module fully compatible with the de-facto standard 16550. This is the standard that can be found in most personal computers and for which a lot of software knowledge and programs is available.

The UART provides a full-featured transmitter-receiver pair, configurable by software for different speeds, character widths, parity codification, etc. The receiver provides information status with several error indications.

Both the transmitter and the receiver can be equipped, if selected at synthesis time, with a 16-character First In First Out (FIFO) buffer. If equipped, the software can decide to put the UART in non-FIFO (16450) mode or in FIFO (16550) mode. For designs requiring low area, the module can be implemented with a 1-character buffer instead of the 16-byte FIFO.

DMA operation is allowed with two output signals that inform the DMA controller about when is new received data available and when the UART is able to accept new data for transmission. Moreover, the module provides an extension to the 16550 standard providing two DMA transfer flags and associated interrupts.

Management of modem control outputs and inputs (with their associated interrupt) is included.

This IP is fully synchronous and static, and well suited for implementation in standard-cells circuits, gate-arrays and FPGAs.

Key Features

- Fully compatible with industry-standard 16550.
- Full-featured, software configurable transmission -reception pair.
- FIFO capability (default 16 words for both transmission and reception). The implementation of the FIFO can be omitted for a smaller area.
- DMA capability.
- Modem control signals.
- Fully synchronous and static design. No clock gating. Simple RTL codification to easy the use with any Verilog synthesizer.
- Provided as a communication core plus FIFO modules, to make possible the connection to technology dependent FIFO memories.
- Easy connection to microprocessors. Example connection provided for ARM and AVR.
- Fully tested on Xilinx and Altera FPGAs.
- High fault coverage for silicon implementations.

Deliverables

- RTL Verilog source code & test bench.
- Synthesis script for Synopsys Design Compiler®.
- Reference technology netlist

Pin Description

Table 1 below describes the functionality of each pin.

Table 1 – Pin Description

Pin Name	Size	Direction	Active	Description
System signals				
clk	1	I	Rising	System clock. Every register moves on its positive edge.
reset	1	I	High	System reset. Every register resets asynchronously when it is one.
Microprocessor data interface				
add[2:0]	3	I	-	Address to select one of the registers in the UART
cs1	1	I	High	Chip select # 1 (positive). Must be 1 to perform a read or write operation
cs2	1	I	High	Chip select # 2 (positive). Must be 1 to perform a read or write operation
cs_n	1	I	Low	Chip select # 3 (negative). Must be 0 to perform a read or write operation
ior	1	I	High	Read strobe (positive). Must be 1 to execute a read operation
ior_n	1	I	Low	Read strobe (negative). Must be 0 to execute a read operation
iow	1	I	High	Write strobe (positive). Must be 1 to execute a write operation
iow_n	1	I	Low	Write strobe (negative). Must be 0 to execute a write operation
data_in[7:0]	8	I	-	Input data to be written in a register
data_out[7:0]	8	O	-	Output data read from a register
outen	1	O	High	When high, a valid read operation is being executed (data_out valid)
Interrupt signals				
irq	1	O	High	Interrupt signal for the microprocessor. It is kept high until the interrupt condition is removed.
irq_n	1	O	Low	Negative version of irq. It is just its complement.
DMA interface				
rxrdy	1	O	High	The UART has new received data to be transferred to memory.
rxrdy_n	1	O	Low	Negative version of rxrdy. It is just its complement.
txrdy	1	O	High	The UART is ready to receive characters from memory to be sent.
txrdy_n	1	O	Low	Negative version of txrdy. It is just its complement.
dma_rxend	1	I	High	A DMA transfer for received data has finished. Used to provide a flag and an interrupt.
dma_txend	1	I	High	A DMA transfer for data to be transmitted has finished. Used to provide a flag and an interrupt.

Table 1 – Pin Description (continued)

Pin Name	Size	Direction	Active	Description
Modem control signals				
cts_n	1	I	Low	Clear To Send. Used to provide flags and an interrupt.
dsr_n	1	I	Low	Data Set Ready. Used to provide flags and an interrupt.
ri_n	1	I	Low	Ring Indicator. Used to provide flags and an interrupt.
cd_n	1	I	Low	Carrier Detect. Used to provide flags and an interrupt.
rts_n	1	O	Low	Request To Send. Controlled by a register's bit.
dtr_n	1	O	Low	Data Terminal Ready. Controlled by a register's bit.
General purpose outputs				
out1_n	1	O	Low	General-purpose output # 1.
out2_n	1	O	Low	General-purpose output # 2. Depending on a synthesis option may correspond to a global interrupt mask
Serial communication signals				
rxd	1	I	-	Serial input. (When there is no communication, it stays at 1).
txd	1	O	-	Serial output. (When there is no communication, it stays at 1).

Following are some explanations about each of the signal groups shown in the table above.

SYSTEM SIGNALS

The complete UART circuitry is driven by the rising edge of the *clk* clock. This clock is not gated at any time.

Every UART register is reset by the *reset* signal. In order to allow for smaller area in standard cell implementations, asynchronous reset technique has been used.

The above sentences are valid for all the provided modules: the UART core and the register-based implementation of the FIFO buffers.

MICROPROCESSOR DATA INTERFACE

The UART is designed as an 8-bit peripheral and so it provides 8-bit input and output data buses, *data_in* and *data_out* respectively.

A 3-bit address input *add* selects which register is being access by the microprocessor. As it is explained below, there are more than 8 registers and so the selection is also based on the state of a special bit (see Register Description section below).

In order to enable any read or write operation on the registers, a "chip select" condition must be met. This condition is defined by three different signals: *cs1*, *cs2* and *cs_n*. The first two must be set at high level and the third at low level in order for any operation to take place.

A read operation is executed when the chip select condition is met and the signal *ior* is 1 while *ior_n* is 0. Similarly, a write operation occurs when the chip select condition is met and the signal *iow* is 1 while *iow_n* is 0.

Splitting the chip select input and the read and write strobes into several signals provides more versatility to the module while few more gates are added to the circuitry. In this way, the UART can be connected in a straightforward manner to different microprocessor architectures.

Finally, the UART provides an output signal, *outen*, which validates the data output. This signal indicates that a valid read operation is being executed in the current clock cycle. It may be used in an external data multiplexer or just left unconnected.

INTERRUPT SIGNALS

The UART provides an interrupt line *irq* to the microprocessor. This line will rise as soon as an interrupt condition appears for which the interrupt generation has been enabled. It will stay high until the interrupt condition disappears. To reset this condition the microprocessor will have to execute an appropriate read or write operation.

Again to provide more flexibility with low area cost, a complementary signal *irq_n* is also generated.

DMA INTERFACE

Two standard signals are provided together with their complementary versions in order to manage the operation of an external DMA controller:

The signal *rxrdy* (r*xrdy_n*) is set to its active value when there is new received data that can be read from the UART by the DMA controller to be put in memory. Depending on the DMA mode being used (see DMA operation below), this signal may be set when there is any new data or it may wait until a trigger level has been reached in the receiver's FIFO.

The signal *txrdy* (*txrdy_n*) is set to its active value when the UART is ready to receive new characters taken from memory by the DMA controller, in order to be sent through *txd*. Depending on the DMA mode being used (see DMA operation below), this signal may be set active only when the transmitter FIFO is completely empty or it may be kept active until the transmitter FIFO gets full.

Two additional inputs not found in a 16550 standard UART are provided: *dma_rxend* and *dma_txend*. These ones allow the UART to notify in its registers when a DMA transfer (for reception or transmission respectively) has finished, generating independent interrupt conditions for each signal if desired. This is done in a way so that the UART keeps total compatibility with the standard 16550.

MODEM CONTROL SIGNALS

The standard signals to control a modem equipment are provided: Clear To Send (*cts_n*), Data Set Ready (*dsr_n*), Ring Indicator (*ri_n*) and Carrier Detect (*cd_n*) inputs coming from the modem and Request to Send (*rts_n*) and Data Terminal Ready (*dtr_n*) outputs going to the modem. All of them are active low, which is the standard way to provide them. This just means that the value seen in the signals is the opposite of the one read from or written to the modem status/control registers.

What the UART does with these signals is just putting in the outputs the contents of a register and show in other register the values of the inputs, generating an interrupt if desired when they change. There is no relation between the modem control block and the serial transmission/reception blocks.

The UART includes an anti-metastability filter for the modem control inputs. This inserts a 2-cycle delay in the transfer of the state of these signals to the Modem Status Register (MSR, see below).

GENERAL-PURPOSE OUTPUTS

In the same way that in other standard modules, there are two outputs *out1_n* and *out2_n* with no specific purpose. Their value correspond to the state of a couple of bits in the Modem Control Register. Since one of these two bits can implement a standard global interrupt enable bit (depending on a synthesis option), the *out2_n* may be not so “general-purpose”.

SERIAL COMMUNICATION SIGNALS

The serial input must be connected to the *rxd* pin and the serial output to the *txd* pin.

The UART implements for the *rxd* input an anti-metastability filter followed by a majority filter. This inserts a delay of three clock cycles in the view of the *rxd* that the receiver has respect to a direct sampling of *rxd*.

Operation

During the following paragraphs the various aspects of the UART operation are covered. Some reference will appear to the UART control and status registers, whose detailed description can be found in the next section.

SERIAL DATA REPRESENTATION

Throughout this datasheet the term *character* will appear. This term is commonly used to describe a word of data being transmitted serially. A character can have several possible word lengths, being the possible ones managed by this module 5, 6, 7 or 8 bits.

The serial communication line has an idle state of '1': when no data is being communicated the line is held at high logic level (sometimes called *mark*). The data transmitted through the serial interface (sent through the *txd* pin and received through the *rxd* pin) has the following format:

- A start bit, always '0', is sent first.
- 5 to 8 bits of data follow. The least significant bit is sent first.
- A parity bit may follow the data bits to provide error checking capability.
- Finally 1 or more stop bits separate this character from the next one.

The exact codification used in the line is selected writing to the Line Control Register (LCR).

TRANSMISSION

The transmission of data is done through the *txd* pin. The transmitter has a Transmitter Holding Register (THR) to hold the data to be sent. This buffer is written from by microprocessor. The data in the THR is passed to the Transmitter Shift Register (TSR), which is used to serialize the character data bits. During this serialization process, the start, stop and parity bits are added.

RECEPTION

The serial data comes into the UART through the *rxd* pin. The Receiver Shift Register (RSR) gets this data and converts it to parallel format. When a complete character has been assembled it is passed to the Receiver Holding Register (RHR), where it is read from the microprocessor.

The reception of a character starts when a bit at low level ('0', also called *space*) is detected. At this moment a counter is started. When the counter advances for half a bit time (according to the programmed baud rate), the *rxd* line is checked again. If it is still a zero the start bit is accepted, otherwise it is rejected and the process is repeated the next time a zero is found.

After a start bit has been accepted, the receiver samples the *rxd* line in time points corresponding to the center of each expected bit, passing the sampled value into the RSR.

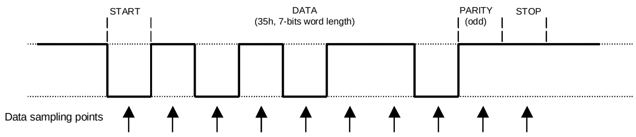


Figure 1 – Data representation and sampling

Before considering any logic value found in the *rxd* line, the receiver applies a filter to reject line noises. Three consecutive samples of the *rxd* line are taken and the logic value to be used is selected by majority.

In order to have some accuracy when choosing the instant to sample the line, each bit time is divided into 16 parts. Due to this the maximum baud rate is the system clock frequency divided by 16.

When a character is received, several error conditions can be detected:

- Overrun Error: It is produced when a character is assembled and there is no space to put it because the microprocessor did not read the RHR fast enough.
- Parity Error: The parity computed on the received data does not agree with that shown in the parity bit.
- Framing Error: The stop bit is not zero. (The receiver only checks the first stop bit, with independence of the number of stop bits programmed).
- Break Interrupt: The *rxd* line has been kept at zero for a complete character time.

Three status flags together with the status of the transmitter are reported in the Line Status Register (LSR).

BAUD RATE

The baud rate is software configurable through two registers:

- The Divisor Latch
- The Pre-Scaler Division

Both registers divide the maximum baud rate (system clock frequency / 16) by a user-selectable factor. The Divisor Latch is a 16550 standard 16-bit register and is accessed through two 8-bit locations: DLL and DLM. Division factors from 1 to 65536 can be programmed. The Pre-Scaler Division register is not standard (although does not restrict module's compatibility at all). It is a 4-bit register that provides an additional division factor (1 to 16) useful when the system clock is a multiple of the most common UART clock frequencies. In this case the same DLL/DLM values used together with those common clocks could be used here just adding a convenient pre-scaling.

Summarizing, the baud rate can be defined as:

$$BR = \frac{f_{clk}}{16 \cdot (PSD + 1) \cdot DL}$$

In order to access these registers, it is necessary to set the DLAB bit in the Line Control Register.

MODEM CONTROL

The UART provides the standard signals for communication with a modem:

- Output signals to the modem: Data Terminal Ready (pin *dtr_n*) and Request To Send (pin *rts_n*).
- Input signals from the modem: Clear To Send (pin *cts_n*), Data Set Ready (pin *dsr_n*), Ring Indication (pin *ri_n*) and Carrier Detect (pin *cd_n*).

All these pins are active low. The output pins are controlled by bits in the Modem Control Register (MCR). The state of the input pins can be read in the Modem Status Register (MSR). Both registers use a positive polarity, so a 1 in the register corresponds to a 0 in the corresponding pin and vice-versa.

The MSR also provides flags to indicate a change in the status of the input pins. These flags can generate an interrupt to the microprocessor if desired.

The behavior of the modem control circuitry is completely independent of the rest of the UART. If modem control is desired, the software has the complete responsibility of managing the output and inputs signals. No automatic flow-control is implemented based on these signals.

THE FIFOS

The holding registers of both the transmitter (THR) and the receiver (RHR) can be formed by a unique 1-word register or by a 16-word First-In-First-Out (FIFO) buffer. At synthesis time the designer can select whether to implement or not the FIFOs (both at the same time, it is not possible to have a FIFO in one direction and a simple register in the opposite direction). If they are implemented, by software they can be enabled or disabled. When the FIFOs are disabled, the behavior is the same as if they had not been implemented. The UART working without FIFOs is what is commonly called the *16450* mode, due to the name of the industry standard UART without FIFOs. The FIFOs are enabled or disabled using the Fifo Control Register (FCR). It is possible to know if an UART has the FIFOs enabled by reading the Interrupt Status Register (ISR).

Note that in both the transmitter and the receiver it is possible to have simultaneously 17 characters. In the transmitter, 1 being sent (TSR) and 16 waiting for transmission (FIFO). In the receiver, 16 ready to be read (FIFO) and 1 being assembled (RSR).

Being 8 bits the maximum data word length, the transmitter FIFO is 8-bits wide. However the receiver FIFO is by default 11-bits wide. This is due to the fact that the receiver does not only put the data in the FIFO, but also the error flags associated to each character. This last size can be reduced to 10-bits wide, without sensibly decreasing the compatibility, by using a synthesis options (see section on synthesis options).

INTERRUPTS

The UART can generate an interrupt signal as the result of six prioritized interrupt sources. The sources, listed from highest to lowest priority levels, are:

Level 1 (max.)	- Receiver Line Status
Level 2	- Received Data Ready
	- Reception Timeout
Level 3	- Transmitter Holding Reg. Empty
Level 4	- Modem Status
Level 5	- DMA Reception End of Transfer
Level 6 (min.)	- DMA Transmission End of Trans.

The different interrupt sources are enabled or disabled making use of the Interrupt Enable Register (IER), which has one bit per each of the above six interrupt classes.

The interrupt is signaled to the microprocessor with the *irq* pin. This pin gets high whenever an enabled interrupt condition appears and only returns to zero when the interrupt has been explicitly reset by the microprocessor (the exact way depends on the interrupt source, see ISR description below). The *irq_n* pin is just a complementary version of *irq*.

Besides the IER, in order to make the *irq* pin active it may be necessary to set a global interrupt enable bit located in the MCR resgister (bit 3). The behavior of this MCR bit as a global interrupt mask is an implementation option selected in the file *uart_16550_features.v* with the Verilog definition:

`define UART_USE_MCR3_MASK.

The default implementation has this behavior implemented, as can be found in other standard 1655x UARTs.

The interrupt status of the UART can be read in the Interrupt Status Register (ISR). This register always provides the code of the highest priority pending interrupt.

A description of each of the interrupt sources follows:

- Receiver Line Status

If enabled, an interrupt will be generated when an error is detected in the received data. For flag errors that are queued in the receiver FIFO, the interrupt appears when the flagged data gets the top of the FIFO (i.e. is the next to be read).

This interrupt is related with bits 1 to 4 of the LSR.

- Received Data Ready / Reception Timeout

These two interrupt sources refer to the same fact: There is available data to be read from the receiver's FIFO (or 1-word RHR).

In non-FIFO mode (16450 mode), an interrupt will be generated (if enabled) as soon as a received character passes from the RSR to the RHR. It will be cleared when the microprocessor reads the new character from RHR.

In FIFO mode (16550 mode), the interrupt will appear when the number of words in the receiver's FIFO reaches the trigger level programmed in the FCR. The interrupt signal will stay active while the number of words in the FIFO stays higher than that value and will be cleared when the microprocessor reads the necessary words to make the number of words in the FIFO less than the trigger level.

Besides, for FIFO mode operation a time out mechanism is implemented. Independently of the trigger level of the FIFO, an interrupt will be generated if there is at least one word in the FIFO and for a time equivalent to the transmission of four characters:

- no new character has been received and
- the microprocessor has not read the RHR

To compute the time out, the current total number of bits (start, data, parity and stop(s)) is used, together with the current baud rate (i.e., it depends on the contents of the LCR, DLL, DLM and PSD registers).

- Transmitter Holding Register Empty

In non-FIFO mode, an interrupt will be generated (if enabled) when the THR gets empty, so the microprocessor knows that it can write new data to be sent.

In FIFO mode the interrupt will appear when the transmission FIFO gets completely empty, so the microprocessor knows it can write 16 new characters to be sent.

This interrupt is directly related to the value of the THR Empty bit in the LSR.

- Modem Status

If enabled, an interrupt will be generated whenever a change is detected in the modem control input pins. For *ri_n* input the change must be from 0 to 1.

This interrupt is directly related to bits 0 to 3 of the Modem Status Register.

- DMA Reception End of Transfer

This is an interrupt source not present in standard 16550s. This does not prevent the module from being completely compatible with the standard 16550.

This interrupt source is intended to be used in the case that the UART is connected to a DMA controller that provides flags to indicate the end of its transfers. In this case the UART interrupt could be used instead of an independent DMA interrupt.

If enabled, the interrupt line will be set active when the input pin *dma_rxend* gets high. The status of this pin is shown in the ISR (see below).

- DMA Transmission End of Transfer

Like the previous one, this is an interrupt source not present in standard 16550s but this does not prevent the module from being completely compatible with the standard 16550.

Again its use is intended when the UART is connected to the same kind of DMA controller.

If enabled, the interrupt line will be set active whenever the input pin *dma_txend* gets high. The status of this pin is shown in the ISR (see below).

DMA SIGNALS

The UART provides two output signals for connection to a DMA controller: *rxrdy* and *txrdy*, together with their complementary versions *rxrdy_n* and *txrdy_n*.

The purpose of the signal *rxrdy* is to indicate to the DMA controller when there is new received data to be transferred to a memory reception buffer.

The purpose of the signal *txrdy* is to indicate to the DMA controller when the UART is ready to accept new characters to be transmitted.

The way in which these two signals move depends on the "DMA mode" selected:

- DMA mode 0:

> This is the single-transfer DMA mode. It is selected when the UART is working in the 16450 mode (FIFOs are disabled) and also when the FIFOs are enabled but bit 3 of FCR has been set to 0.
>
> In this mode, *rxrdy* is active (i.e. *rxrdy*=1 and *rxrdy_n*=0) whenever there is available data in the receiver's FIFO or 1-word RHR. It is inactive when the receiver's FIFO / RHR is empty.
>
> The signal *txrdy* is active (i.e. *txrdy*=1 and *txrdy_n*=0) only when the transmission FIFO (or 1-word THR) is completely empty. It is inactive when at least 1 byte is in the transmitter's FIFO / THR.

- DMA mode 1:

> This is the multiple-transfer DMA mode. It is selected when working in the 16550 mode (FIFOs are enabled) and bit 3 of FCR has been set to 1.
>
> In this mode, *rxrdy* gets active when the number of characters in the receiver's FIFO is equal or greater than the trigger level programmed in the FCR. It is also set if a timeout condition has been reached (see timeout interrupt above). Once it is set, *rxrdy* will stay active until all the characters in the receiver's FIFO have been read, i.e. until the reception FIFO gets empty again.
>
> The signal *txrdy* goes from an active to an inactive state when the transmitter's FIFO gets full. After this, it is kept inactive until the transmission FIFO gets empty.

The two signals described above are found in any standard 16550 UART. Besides of those signals, this UART accepts two other input signals coming from a DMA controller: *dma_rxend* and *dma_txend*. As explained in the description above about UART interrupts, these two signals are can be used to notify the UART that a complete transfer has finished (reception or transmission transfer respectively). The UART will provide the status of these signals in the ISR and if desired will also generate an interrupt when they become active. In order to be fully compatible in the values read from the ISR, this "Dma End signaling" has to be enabled by writing bit 4 of FCR. Otherwise reading ISR will get "standard zeros" in the corresponding bit locations.

GENERAL-PURPOSE OUTPUTS

The UART has two pins that are general-purpose outputs: *out1_n* and *out2_n*. These pins are controllable from the MCR register. Actually the *out2_n* is really general-purpose only if bit 3 of MCR is not used as a global mask (this is a synthesis option, as explained above). This is because *out1_n* and *out2_n* always follow the state of bits 2 and 3 of the MCR, independently of the synthesis options.

LOOP BACK

By setting bit 4 of the MCR it is possible to put the UART in a loop back mode, useful for testing purposes.

In this mode the *txd* output pin and the modem control output pins are externally kept in the inactive state (logic 1), while internally *txd* is connected to *rxd* and the modem control and general purpose output pins are connected to the modem control input pins (See MCR bit 4 description for more details).

Except for these looping connections the UART continues working in the same way.

THE SCRATCH PAD REGISTER

A standard 16550 register is the Scratch Pad Register (SPR). This one is totally independent from the rest of the UART and its purpose is to serve only as a spare location where the programmed can write (and read) whatever he or she wants.

As this is not a very important feature, not being used very often, a synthesis option decides whether to implement it or not. By default it is implemented to get full compatibility. See the section on synthesis options.

Register Description

The UART is controlled through a set of registers addressable with the *add[2:0]* input.

The following table summarizes these registers, which are explained below:

Table 2 – Registers Summary

Address & Access type		Register		Bit number								reset value
				7	6	5	4	3	2	1	0	
General Register Set												
000	R	Receiver Holding Register	RHR	Character Received								00
000	W	Transmitter Holding Register	THR	Character to be Transmitted								00
001	R/W	Interrupt Enable Register	IER	DMA Tx End	DMA Rx End	0	0	Modem Status	Receiver Line Status	THR Empty	Data Ready	00
010	R	Interrupt Status Register	ISR	FIFOs enabled	FIFOs enabled	DMA Tx End	DMA Rx End	Interrupt Identification Code			Interrupt Status	01
010	W	FIFO Control Register	FCR	Receiver's FIFO Trigger Level		0	Enable DMA End	DMA mode	Tx FIFO Reset	Rx FIFO Reset	FIFO enable	00
011	R/W	Line Control Register	LCR	DLAB	Set Break	Force Parity	Even Parity	Parity Enable	Stop Bits	Word Length		00[1]
100	R/W	Modem Control Register	MCR	0	0	0	Loop back	Out 2 / Int. Enable[2]	Out 1	RTS	DTR	00
101	R	Line Status Register	LSR	FIFO data Error	Transmitter Empty	THR Empty	Break Interrupt	Framing Error	Parity Error	Overrun Error	Data Ready	60
110	R	Modem Status Register	MSR	CD	RI	DSR	CTS	delta CD	trailing edge RI	delta DSR	delta CTS	00[3]
111	R/W	Scratch Pad Register	SPR	User Data								00
Registers accesible only when DLAB = 1												
000	R/W	Divisor Latch, Least signif. byte	DLL	Baudrate Divisor's Constant Least Significant Byte								01[4]
001	R/W	Divisor Latch, Most signif. byte	DLM	Baudrate Divisor's Constant Most Significant Byte								01[4]
101	W	Prescaler Division	PSD	0	0	0	0	Prescaler's Division Factor				00

[1] Standard reset state for LCR is with all bits cleared, but it is possible to synthesize the module with a 03 reset value (8 bits word length upon reset)

[2] Out 2 bit may act or not as a global interrupt enable mask depending on a synthesis option. Standard option is to act as a mask (see MCR description)

[3] This reset value will be always obtained until the second cycle after reset. After that state of bits 4 to 7 depends on the state of the corresponding UART inputs, while delta bits 0 to 3 assume as initial condition the inactive situation of the input lines (i.e., logic 1 in the pins and zeroes in bits 4-7).

[4] Note that actually there is no standard value for this reset condition. Therefore the programmer must always program DLL and DLM.

Receiver Holding Register (RHR)

The user can get the data received through the serial channel (pin *rxd*) reading this read-only location. Note that DLAB bit in LCR must be 0.

Depending on whether the FIFOs are implemented and enabled, this location will refer to a 1-byte register which receives the contents of the Receiver Shift Register once a character has been assembled, or to the top of a 16-word FIFO.

Before reading this register the user should check LSR for possible errors. The status shown in LSR corresponds to the character on top of the FIFO, which is the one ready to be read from RHR.

If a character less than 8 bits in width is received, the extra bits are read as ‘0’.

TRANSMITTER HOLDING REGISTER (THR)

The user writes in this write-only location the data to be sent through the *txd* pin of the UART. Note that DLAB bit in LCR must be 0.

If a character less than 8 bits in width is going to be transmitted, it must be right-justified. Left bits don’t care. For example, with a word length of 5 bits, writing B5h or 15h will result in the transmission of a 15h character.

Before writing this register the user must ensure that the UART is ready to accept data for transmission, for example checking that THR Empty flag is set in the LSR (see the description of this register below).

Depending on whether the FIFOs are implemented and enabled or not, the data written to this register goes to either a 1-byte register or to the 16-byte transmission FIFO.

INTERRUPT ENABLE REGISTER (IER)

This register individually enables each of the possible interrupt sources. A logic “1” in any of these bits enables the corresponding interrupt, while a logic “0” disables it. For a detailed description of the interrupt sources, see the description of the Interrupt Status Register (ISR) below.

- bit 0: This bit enables the data ready interrupt.
- bit 1: This bit enables the THR Empty interrupt.
- bit 2: This bit enables the Receiver Line Status interrupt.
- bit 3: This bit enables the Modem Status interrupt.
- bits 4-5: These bits are not used and are always cleared.
- bit 6: This bit enables the non-standard interrupt issued when a DMA reception transfer is finished.
- bit 7: This bit enables the non-standard interrupt issued when a DMA transmission transfer is finished.

Note that besides these independent interrupt mask bits, there is a global interrupt enable bit located in bit 3 of the Modem Control Register (MCR) (see below). To actually have an interrupt indication in the UART output pins *irq*, *irq_n*, both the corresponding bit in IER and the global enable bit in MCR must be set. MCR[3] standard behavior as a global enable bit can be deactivated at synthesis time.

INTERRUPT STATUS REGISTER (ISR)

The main purpose of this register is to identify the interrupt with the highest priority that is currently pending. The UART implements a priority encoder with six levels, from higher priority to lower priority:

- Priority 1 (highest) – Receiver Line Status (error notifications).
- Priority 2 – Receiver Data Ready or Time out (data can be read)
- Priority 3 – Transmitter Holding Register Empty (data can be written)
- Priority 4 – Modem Status (changes in modem lines).
- Priority 5 – DMA Transfer End for received data.
- Priority 6 (lowest) – DMA Transfer End for transmitted data

The last two priority levels are not found in standard 16550 UART and may appear only if the DMA End signaling is enabled (bit 4 of FCR).

Besides this main function, the ISR provides other flags. Following is a description of each of the bits:

- bit 0: This bit indicates whether an interrupt is pending or not. An interrupt is pending if this bit is cleared (‘0’).
- bits 1-3: These bits identify the highest priority interrupt that is pending. Table 3 below describes the different interrupt conditions and their codes of identification, together with their reset method. Note that for an interrupt source to be considered as pending, the corresponding bit in the IER must be enabled. However the notifications in this register are independent of the state of the global interrupt enable bit in the MCR.
- bit 4: This bit reflects the state of the *dmarx_end* input pin which signals the end of a complete DMA transfer for received data. This is a non-standard flag that is enabled only if DMA End signaling has been enabled with bit 4 of FCR register. Otherwise it will always be read as ‘0’.

- bit 5: This bit reflects the state of the *dmatx_end* input pin which signals the end of a complete DMA transfer for transmitted data. This is a non-standard flag that is enabled only if DMA End signaling has been enabled with bit 4 of FCR register. Otherwise it will always be read as ‘0’.

- bits 6 and 7: These two bits are set if the FIFOs are implemented and enabled (by setting FCR bit 0). They are cleared in non-FIFO (16450) mode.

Table 3 – Interrupt sources and their identification codes in the ISR

Interrupt Status Register (ISR) Code				**Priority Level**	**Interrupt Type**	**Interrupt Source Description**	**Interrupt Reset Method**
bit 3	**bit 2**	**bit 1**	**bit 0**				
0	0	0	1	None	No interrupt	There is no interrupt pending	N.A.
0	1	1	0	1	Receiver Line Status	There is an overrun error, parity error, framing error or break interrupt indication corresponding to the received data on top of the receiver’s FIFO. Note that the FIFO error flag in LSR does not influence this interrupt, which is related only to the data on top of the Rx FIFO. This is directly related to the presence of a 1 in any of the LSR bits 1 to 4.	Read the Line Status Register (LSR)
0	1	0	0	2	Received Data Ready	In non-FIFO mode, there is received data available in the RHR register. In FIFO-mode, the number of characters in the reception FIFO is equal or greater than the trigger level programmed in FCR. Note that this is not directly related to LSR bit 0, which always indicates that there is at least one word ready.	Read the Receiver Holding Register (RHR).
1	1	0	0	2	Reception Timeout	There is at least one character in the receiver’s FIFO and during a time corresponding to four characters at the selected baud rate no new character has been received and no reading has been executed on the receiver’s FIFO.	Read the Receiver Holding Register (RHR). Reading it once is enough to reset the condition.
0	0	1	0	3	Transmitter Holding Register Empty	In non-FIFO mode, the 1-byte THR is empty. In FIFO mode, the complete 16-byte transmitter’s FIFO is empty, so 1 to 16 characters can be written to THR. That is to say, THR Empty bit in LSR is one.	Write the Transmitter Holding Register (THR). Alternatively, reading the Interrupt Status Register (ISR) will also clear the interrupt if this is the interrupt type being currently indicated (this will not clear the flag in the LSR).
0	0	0	0	4	Modem Status	A change has been detected in the Clear To Send (CTS), Data Set Ready (DSR) or Carrier Detect (CD) input lines or a trailing edge in the Ring Indicator (RI) input line. That is to say, at least one of MSR bits 0 to 3 is one.	Read the Modem Status Register (MSR)
1	1	1	0	5	DMA Reception End of Transfer	A ‘1’ has been detected in the *dmarx_end* input pin. This is supposed to imply the end of a complete DMA transfer for received data, executed by a DMA controller that provides this signal.	Read the Interrupt Status Register (ISR) (return of *dmarx_end* to zero does not reset the interrupt)
1	0	1	0	6	DMA Transmission End of Transfer	A ‘1’ has been detected in the *dmatx_end* input pin. This is supposed to imply the end of a complete DMA transfer for received data, executed by a DMA controller that provides this signal.	Read the Interrupt Status Register (ISR) (return of *dmatx_end* to zero does not reset the interrupt)

FIFO Control Register (FCR)

This register controls the FIFO buffers used by the receiver and the transmitter. It also selects some options for DMA signaling.

- bit 0: When set ('1') this bits enables both the transmitter and receiver FIFOs. In any writing to FCR, this bit must be set in order to affect the rest of the bits, except for bit 4. Changing this bit automatically resets both FIFOs.

- bit 1: Writing a one to this bit resets the receiver's FIFO (the pointers are reset and all the words are cleared). The Receiver Shift Register is not cleared, so any reception active will continue. The bit will automatically return to zero.

- bit 2: Writing a one to this bit resets the transmitter's FIFO (the pointers are reset). The Transmitter Shift Register is not cleared, so any transmission active will continue. The bit will automatically return to zero.

- bit 3: This bit selects the DMA mode. The DMA mode affects the way in which the DMA signaling outputs pins (*txrdy*, *rxrdy* and their inverted versions) behave. See the *DMA signals* explanation in the *Operation* section above for details.

  Mode 0 is intended to transfer one character at a time. Mode 1 is intended to transfer a set of characters at a time.

- bit 4: This bit enables the DMA End signaling. This non-standard feature is useful when the UART is connected to a DMA controller which provides signals to indicate when a complete DMA transfer has been completed, either for reception or transmission (*dmaend_rx* and *dmaend_tx* input pins). In this case, these events are processed in the same way as other UART interrupt sources. When DMA End signaling is enabled, the user has a pair of flags in the ISR register and also a couple of interrupt conditions indicated in this same register and capable of activating the *irq* pin.

- bit 5: Reserved for future use.

- bits 6 and 7: These bits define the trigger level for the receiver's FIFO. In FIFO mode an interrupt will be generated (if enabled) when the number of words in the receiver's FIFO is equal or greater than this trigger level. Besides, the DMA pin *rxrdy* will be set in DMA mode 1 when the same condition appears. The following Table 4 shows the available trigger levels.

Table 4 – Trigger levels

FCR code		Receiver's FIFO
bit 7	bit 6	Trigger Level
0	0	1 character
0	1	4 characters
1	0	8 characters
1	1	14 characters

Line Control Register (LCR)

This register controls the way in which transmitted characters are serialized and received characters are assembled and checked.

- bits 0 and 1: These bits define the word length of the data being transmitted and received. See Table 5 below for the possible selections.

- bit 2: This bit selects the number of stop bits to be transmitted. If cleared, only one stop bit will be transmitted. If set, two stop bits (1.5 with 5-bit data) will be transmitted before the start bit of the next character. The receiver always checks only one stop bit.

- bits 3 to 5: These bits select the way in which parity control is performed. Bit 3 is an enable bit: it selects whether a parity bit is used or not. Bit 4 selects the polarity of this control bit. Bit 5 forces a value for this bit, independent of the data being transmitted or received. See Table 6 below for the meaning of each of the combinations of these bits.

- bit 6: When this bit is set a break condition is forced in the transmission line. The serial output pin (*txd*) is forced to the spacing state (zero). When this bit is cleared, the break state is removed. The break state has no effect on the transmitter's logic, so if several characters are stored in the transmitter's FIFO they will be removed from this FIFO and passed sequentially to the Transmitter Shift Register which serializes them. This fact can be useful to establish the break time making use of the THR Empty and Transmitter Empty flags of the LSR.

  In order to force a break condition without producing erroneous characters, the user should wait until the current transmission is finished and ensure that the break time situation is held at least for a character time.

- bit 7: This is Divisor Latch Access Bit (DLAB). This bit must be set in order to access the DLL, DLM and PSD registers which program the division constants for the baud rate divider and the prescaler. As these registers occupy the same locations as the THR, RHR, and IER, DLAB must be zero to access these other registers. (Note that the PSD is in the same address as the LSR but write-only)

Table 5 – Word lengths

LCR code		Character's
bit 1	bit 0	Word Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

Table 6 – Parity codifications

LCR code			Parity type	Description
bit 5	bit 4	bit 3		
X	X	0	Disabled	No parity bit is transmitted nor expected
0	0	1	Odd	The number of bits including the parity bit must be odd
0	1	1	Even	The number of bits including the parity bit must be even
1	0	1	Forced 1	The parity bit is sent as/checked to be 1
1	1	1	Forced 0	The parity bit is sent as/checked to be 0

Modem Control Register (MCR)

By writing this register the user can set the modem control outputs (DTR and CTS). But it also controls the loop back mode, provides general purpose outputs and has a global interrupt mask bit.

- bit 0: This bit (DTR) controls the "data terminal ready" active low output (*dtr_n*). A 1 in this bit makes *dtr_n* output a 0. When the bit is cleared, *dtr_n* outputs a 1.
- bit 1: This bit (RTS) controls the "request to send" active low output (*rts_n*) in the same way as bit 0 controls *dtr_n*.
- bit 2: This bit (Out 1) controls the general purpose, active low, output *out1_n* in the same way as bit 0 controls *dtr_n*.
- bit 3: This bit (Out 2/Int. Enable) controls the general purpose, active low, output *out2_n* in the same way as bit 0 controls *dtr_n*.

  Besides, this bit may act as a global interrupt enable bit. In this case, the complementary interrupt lines *irq* and *irq_n* will become active (1 and 0 respectively) only if this bit is 1 (and an interrupt condition is taken place).

  The behavior of this bit as a global interrupt mask is decided at synthesis time, by using in the file *uart_16550_features.v* the Verilog definition:

  `define UART_USE_MCR3_MASK.

  The default implementation has this behavior implemented, as can be found in other standard 1655x UARTs.
- bit 4: This is the loop back mode control bit. Loop back mode is intended to test the UART communication. When this bit is set to 1, the following occurs:
  - The serial output is connected internally to the serial input, so every character sent is looped back and received.
  - The input pin *rxd* is not used and the output pin *txd* is set to 1 (inactive state).
  - The four modem control inputs are internally connected to the two modem control outputs plus the general purpose outputs. This way, *cts_n* is internally controlled by *rts_n*, *dsr_n* by *dtr_n*, *ri_n* by *out1_n* and *cd_n* by *out2_n*. It is to say; there is a not ordered correspondence between the four least significant bits of the MCR and the four most significant bits of the MSR.
  - The four modem control input pins *cts_n*, *dsr_n*, *ri_n* and *cd_n* are not used. The two modem control output pins *dtr_n* and *rts_n* and the two user outputs *out1_n* and *out2_n* are set to 1 (inactive state).
  - MCR bit 3 continues acting as a global interrupt mask if synthesized to do so.
- bits 5 – 7: These bits are permanently set to zero.

LINE STATUS REGISTER (LSR)

This register informs the user about the status of the transmitter and the receiver. In order to get information about a received character, LSR must be read before reading that received character from RHR.

Four interrupts are somewhat associated to the status reported by this register: the Received Data Ready, Reception Time-out, Receiver Line Status and THR Empty interrupts. Refer to the description of the ISR for details.

- bit 0: This is the Data Ready bit. It is set if one of more characters have been received and are waiting in the receiver's FIFO for the user to read them. It is zero if there is no available data in the receiver's FIFO. In case that the 16-character FIFO is not active the same description holds for the 1-character RHR register.

- bit 1: This is the Overrun Error flag. When it is set, a character has been completely assembled in the Receiver Shift Register without having free space to put it in the receiver's FIFO or holding register. When an overrun condition appears, the result is different depending on whether the 16-byte FIFO is active or not:

  - o If the FIFO is not active, so that only a 1-character Receiver Holding Register is available, the unread data in this RHR is overwritten with the new character just received.

  - o If the FIFO is implemented and active, the character just received in the Receiver Shift Register will be overwritten, but the data already present in the FIFO is not changed.

  The Overrun Error flag is set as soon as the overrun condition appears. It is not queued in the FIFO if this is active.

  This bit is cleared as soon as the LSR is read.

- bit 2: This is the Parity Error flag. When it is set, it indicates that the parity of the received character is wrong according to the current setting in LCR.

  This bit is cleared as soon as the LSR is read.

  This bit is queued in the receiver's FIFO, so it is associated to the particular character that had the error. Therefore, LSR must be read before RHR: each time a character is read from RHR the next character passes to the top of the FIFO and LSR is loaded with the queued error flags corresponding to this top-of-the-FIFO character.

- bit 3: This is the Framing Error flag. It indicates that the received character did not have a valid stop bit (i.e., a 0 was detected in the (first) stop bit position instead of a 1).

  This bit is cleared as soon as the LSR is read.

  This bit is queued in the receiver's FIFO in the same way as the Parity Error bit.

  When a frame error is detected, the receiver tries to resynchronize: if the next sample is again a zero it will be taken as the beginning of a possible new start bit.

- bit 4: This is the Break Interrupt indicator. It is set to 1 if the receiver's line input *rxd* was held at zero for a complete character time. It is to say, the positions corresponding to the start bit, the data, the parity bit (if any) and the (first) stop bit were all detected as zeroes. Note that a Frame Error flag always accompanies this flag.

  This bit is cleared as soon as the LSR is read.

  This bit is by default queued in the receiver's FIFO in the same way as the Parity Error bit (standard behavior). Alternatively, the Break Interrupt bit can be generated from the top-of-FIFO data and error flags. This allows the use of a 10-bit wide receiver's FIFO instead of an 11-bit one, saving some area. This behavior is selected by making active the Verilog definition *`define UART_GENERATE_BI*. The user will perceive a different behavior only if the LCR register is changed in the middle of a reception.

- bit 5: This is the Transmit Holding Register Empty flag. In non-FIFO mode, this bit is set whenever the 1-byte THR is empty. If the THR holds data to be transmitted, THR is immediately set when this data is passed to the TSR (Transmitter Shift Register). In FIFO mode, this bit is set when the transmitter's FIFO is completely empty, being 0 if there is at least one byte in the FIFO waiting to be passed to the TSR for transmission.

  This bit is cleared when the microprocessor writes new data in the THR.

- bit 6: This is the Transmitter Empty flag. It is 1 when both the THR (or transmitter's FIFO) and the TSR are empty. Reading this bit as 1 means that no transmission is currently taking place in the *txd* output pin, the transmission line is idle.

  As soon as new data is written in the THR, this bit will be cleared.

- bit 7: This the FIFO data error bit. If the FIFO is not implemented or disabled (16450 mode), this bit is always zero. If the FIFO is active, this bit will be set as soon as any data character in the receiver's FIFO has parity or framing error or the break indication active.

The bit is cleared when the microprocessor reads the LSR and the rest of the data in the receiver's FIFO do not have any of these three associated flags on.

MODEM STATUS REGISTER (MSR)

This register provides information about the status of the four modem control input pins. The four most significant bits provide directly the status of the pin, while the four least significant give information about changes in these pins (always with a 2-cycle delay).

The four least significant bits can generate an interrupt (Modem Status interrupt) if enabled by the corresponding bit in the IER. The interrupt will be generated as soon as any of them is 1. Refer to the ISR description for more information.

- bit 0: This is the delta-CTS flag. If set, it means that the *cts_n* input has changed since the last time the microprocessor read this register.
- bit 1: This is the delta-DSR flag. If set, it means that the *dsr_n* input has changed since the last time the microprocessor read this register.
- bit 2: This bit is set when a trailing edge is detected in the *ri_n* input pin, it is to say, when *ri_n* changes from 0 to 1.
- bit 3: This is the delta-CD flag. If set, it means that the *cd_n* input has changed since the last time the microprocessor read this register.
- bit 4: This bit, Clear To Send (CTS), is the complement of the *cts_n* input
- bit 5: This bit, Data Set Ready (DSR), is the complement of the *dsr_n* input
- bit 6: This bit, Ring Indicator (RI), is the complement of the *ri_n* input
- bit 7: This bit, Carrier Detect (CD), is the complement of the *cd_n* input

Notice that in loop back mode (MCR[4]=1), the four modem control inputs are internally connected to the modem control outputs and general purpose outputs (see MCR bit 4 description for details).

SCRATCHPAD REGISTER (SPR)

This is an 8-bit read/write register intended for the programmer's use for any purpose. It does not affect in any way the operation of the UART. This register is provided for complete software compatibility with industry standard 16550, but its implementation can be avoided at synthesis time, saving some area, by commenting the Verilog definition:

`define UART_IMPLEMENT_SCRATCHPAD

in the file *uart_16550_features.v*. By default this definition is active and so the scratchpad register is implemented.

DIVISOR LATCH (DLL, DLM)

These two registers, together with the Prescaler Division (PSD) select the speed at which the communication will occur. This is the baud rate at which characters will be transmitted and the expected baud rate for the characters that will be received. Only one baud rate is defined for both transmission and reception.

The Divisor Latch is a 16-bit register, whose most significant byte is hold in DLM and its least significant byte is hold in DLL. The access to these two registers, located at addresses 1 and 0 respectively, is conditioned to the value of the DLAB bit in LCR register. Only if this bit is 1 the two registers can be written and read. Otherwise the IER, RHR, and THR will be accessed instead.

The baud rate is defined as the system clock (*clk*) frequency divided by 16, divided by the contents of the PSD register plus one, divided by the contents of the Divisor Latch.

A value of 0 in the Divisor Latch has the meaning of dividing by 65,536, which is the maximum divisor factor programmable in this register.

PRESCALER DIVISION (PSD)

This four-bit register (PSD[3:0]) adds a second programmable division factor to obtain the desired baud rate (see Divisor Latch description above). The division factor is the value hold in this register plus one, so the maximum factor is 16 and the minimum is 1. Bits 4 to 7 are always zero.

This is a non-standard register (i.e., it is not present in the industry standard 16550 UART). Its purpose is to provide a second division factor that could be useful in systems which are driven by a clock multiple of one of the typical frequencies used with this UART. For example, the most typical 16550 clock frequency is 1.8432 MHz. By using a PSD value of 15 (division = 16), a 29.4912 MHz clock can be used while keeping the usual values for the DLM,DLL pair to obtain the different baud rates.

This register is only accessible in write mode and only when the DLAB bit in LCR is set. Otherwise the Line Status Register will be accessed. (In a read operation, LSR will be always accessed, independently of the state of DLAB).

Synthesis options

A number of implementation options can be defined at synthesis time. These options are defined by enabling or commenting different *`define* Verilog sentences in the file *uart_16550_features.v*, which can be found in the *header* subdirectory (see directory structure below).

The following table gives a description of each of these options. The *Default* column shows the state of the options in the file *uart_16550_features.v* that is delivered. These default options correspond to the most 16550-standard implementation.

Table 7 – Synthesis options

Synthesis option name	Default	Description
UART_IMPLEMENT_FIFO	Active	If the provided register-based FIFO buffer modules are used (*uart_16550_fifo16x11.v* and *uart_16550_fifo16x8.v* in *fifo_reg* subdirectory), this option will decide whether they will describe a 16 word FIFO for transmission and reception (active condition) or they will describe a 1 word holding register for transmission and reception (inactive condition). If inactive, the resulting UART will be 16450 compatible instead of 16550 compatible, since the 16550 implies having FIFOs. No other different in the behavior will be obtained apart of those points related to the presence of FIFOs.
UART_IMPLEMENT_SCRATCHPAD	Active	If active, the scratch pad register will be implemented. Otherwise the area for this register will be saved. Standard 16450 and 16550 UARTs have this register, although it has nothing to do with the rest of the UART.
UART_USE_MCR3_MASK	Active	If active, bit 3 of the Modem Control Register (MCR) will act as a global interrupt enable bit. Normally 16550 modules have this behavior while 16450 modules do not (but anyway personal computer drivers always manage this bit). If not active, the generation of a high level in the *irq* line will be masked only by the Interrupt Enable Register (IER).
UART_8BITS_RESET	Inactive	If active, the Line Control Register will reset with a non-standard 03h value. This configures the UART upon reset to use 8 bits characters instead of 5 bits ones.
UART_GENERATE_BI	Inactive	If active, the Break Interrupt indication flag read in the Line Status Register (LSR) will not be really queued in the reception FIFO. Instead it will be generated from the Frame Error, Parity Error and the data itself located in the top of this FIFO. Although standard modules use to queue the BI bit, the user will notice a difference in the behavior only if the line parameters are changed while there is data in the reception FIFO. On the other hand, activating this option will make possible the use of a 10-bit wide reception FIFO instead of an 11-bit wide one. Notice that the provided register-based FIFO module has its width fixed to 11 bits, so using it makes the use of this option non-sense.

IP directory structure

The UART 16550 IP delivered as Verilog RTL source code has the directory structure shown in Figure 2:

The *source* directory has all the Verilog source code needed to implement the UART (nothing but verilog files are found here). Inside, the following directories are found:

- *design* : it has files with a set of *`include* statements defining the necessary files to implement top-level modules, as the UART core or the complete UART built with register-based FIFOs

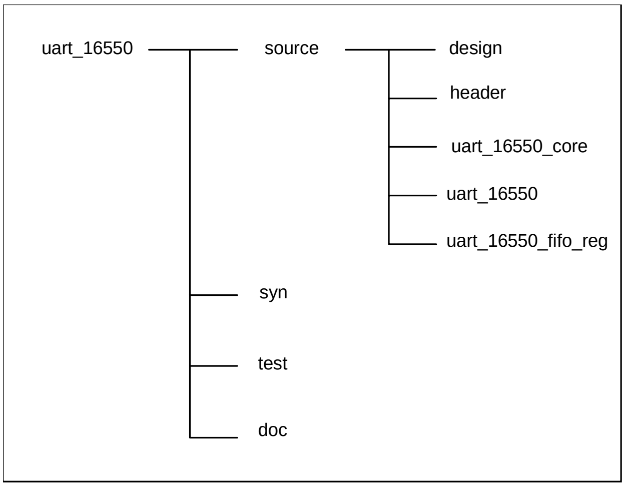


Figure 2 – Directory structure

- *header* : this subdirectory holds the *uart_16550_features.v* file defining the different synthesis options.
- *uart_16550_core* : this one has all the modules that build the UART core. The top of the core is *uart_16550_core.v*.
- *uart_16550* : this one holds the top level module for the UART (*uart_16550.v*).
- *uart_16550_fifo_reg:* this one holds the modules which implement the register-based transmission and reception FIFOs.

The *syn* directory includes a Synopsys© Design Compiler© script file to synthesize the UART.

The *test* directory holds a test bench for the UART. The simulation of the test bench performs checks of every UART functionality giving detailed information of each test step.

The *doc* directory holds this datasheet.

The core's FIFO interface

The UART core, described in the Verilog file *uart_16550_core.v*, provides a generic interface for the transmission and reception FIFOs. This allows the designer to substitute the provided FIFOs by other ones if desired (even of a different size).

Table 8 below describes each of the signals in this interface. The column "Direction" refers to the signal direction from the UART core's point of view:

Table 8 – FIFO interface of the UART core

Signal	Size	Direction	Active	Description
Common signals				
fifo_implemented	1	I	High	Must be set to 1 if FIFOs are actually implemented. If only 1-word registers are available, it must be set to 0.
fifo_enable	1	O	High	If 1, FIFO mode (16550 mode) is selected. If 0, non-FIFO (16450) mode is selected. This should be used by the FIFOs to use its full capacity or only 1 position. If *fifo_implemented* is 0 this signal will always be 0.
Reception FIFO signals				
fifo_rx_in[10:0]	11	O	-	Data to be written in the FIFO. Note that if *UART_GENERATE_BI* option is activated, only bits [9:0] are used. This data must be registered into the FIFO in the rising edge of *clk* if *fifo_rx_push* is 1.
fifo_rx_out[10:0]	11	I	-	Data read from the FIFO. Note that if *UART_GENERATE_BI* option is activated, only bits [9:0] are used. This data will be registered by the UART in a rising edge of *clk* if *fifo_rx_pop* is 1.
fifo_rx_push	1	O	High	Push order to the FIFO. When 1, the FIFO must register *fifo_rx_in* at next *clk* rising edge and modify its pointers accordingly.
fifo_rx_pop	1	O	High	Pop order to the FIFO. When 1, the UART will register *fifo_rx_out* at next *clk* rising edge and the FIFO must modify its pointers accordingly.
fifo_rx_reset	1	O	High	When this signal is 1 at a *clk* rising edge, the FIFO must reset its pointers to indicate an empty condition. Resetting the data to 0 is desirable but not mandatory.
fifo_rx_trig_level[1:0]	2	O	-	Trigger level code as defined for the FIFO Control Register (FCR). Note that there is no restriction to use the standard values corresponding to these codes described in the FCR explanation above. In fact, FIFOs with more that 16 words can be implemented making convenient the redefinition of these codes.
fifo_rx_empty	1	I	High	If 1 the FIFO is empty.
fifo_rx_triggered	1	I	High	If 1 the number of characters in the FIFO is equal to or greater that the value corresponding to the trigger code in *fifo_rx_trig_level*.
fifo_rx_full	1	I	High	If 1 the FIFO is full
fifo_rx_error	1	I	High	This bit is used to provide the FIFO data error bit of the Line Status Register. This bit must be 1 if *fifo_enable* is 1 and there is any character in the FIFO with bit 8 or bit 9 set at 1.
Transmission FIFO signals				
fifo_tx_in[7:0]	8	O	-	Data to be written in the FIFO. This data must be registered into the FIFO in the rising edge of *clk* if *fifo_tx_push* is 1.
fifo_tx_out[7:0]	8	I	-	Data read from the FIFO. This data will be registered by the UART in a rising edge of *clk* if *fifo_tx_pop* is 1.
fifo_tx_push	1	O	High	Push order to the FIFO. When 1, the FIFO must register *fifo_tx_in* at next *clk* rising edge and modify its pointers accordingly.
fifo_tx_pop	1	O	High	Pop order to the FIFO. When 1, the UART will register *fifo_tx_out* at next *clk* rising edge and the FIFO must modify its pointers accordingly.
fifo_tx_reset	1	O	High	When this signal is 1 at a *clk* rising edge, the FIFO must reset its pointers to indicate an empty condition.
fifo_tx_empty	1	I	High	If 1 the FIFO is empty.
fifo_tx_full	1	I	High	If 1 the FIFO is full

System integration examples

Following two examples are provided of how the UART 16550 IP can be connected to different microcontroller systems.

ARM© SYSTEM

In an ARM© 32-bit microprocessor core system, the UART module typically will be connected to the Advanced Peripheral Bus (APB) defined in the Advanced Microcontroller Bus Architecture (AMBA) specification. This connection can be implemented as in Figure 3 below.

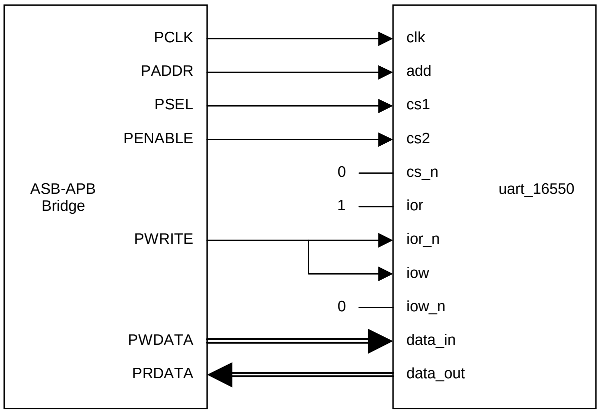
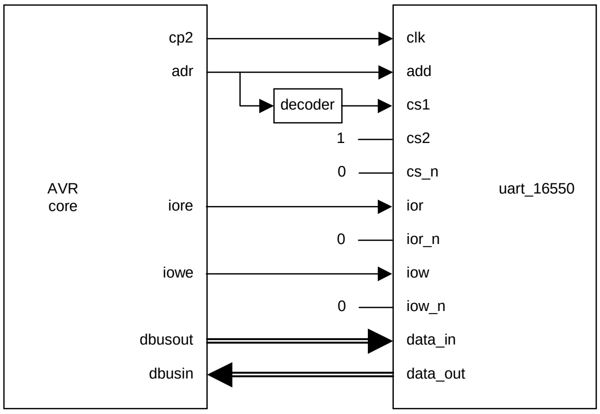


Figure 3 – Connection with ARM© through APB

AVR© SYSTEM

In an AVR© 8-bit microprocessor core system, the UART module normally will be connected to the I/O bus. This connection can be implemented as in Figure 4 below.



Note that the *adr* bus from the AVR© has a width of 6 bits while the UART needs 3 bits to decode its own registers. There is no need to implement the external address decoding if the UART is located in an area corresponding to an *adr[5:3]* value with two 1s and one 0. In that case, these most significant bits of *adr* could be connected directly to *cs1*, *cs2* and *cs_n*.

SIDSA
Semiconductor Design Solutions

European Headquarters
SIDSA
Parque Tecnológico de Madrid
Torres Quevedo, 1
28760 Tres Cantos (Madrid) SPAIN
Phone 34 91 803 5052
Fax 34 91 803 9557

info@sidsa.com
http://www.sidsa.com/

SIDSA U.S.
166 Geary Street
Suite 1307
San Francisco, CA 94108
USA

us.info@sidsa.com