Introduction to the UART (covers material for labs 5-8)

Serial communication is an essential to computers and allows them to communicate with low speed peripheral devices, such as the keyboard, the mouse, modems etc. Thus, the UART or **U**niversal **A**synchronous **R**eceiver/ **T**ransmitter is the most important component required in serial communication.

Asynchronous communication is performed between two (or more) devices if the devices operate on independent clocks. This is because there is no guarantee that the clocks of the communicating devices will have the exact frequency and phase over extended periods. To combat this problem, asynchronous communication requires additional synchronisation bits to be added around actual data in order to maintain signal integrity. Figure 1 below illustrate the waveform of an asynchronous serial data stream.

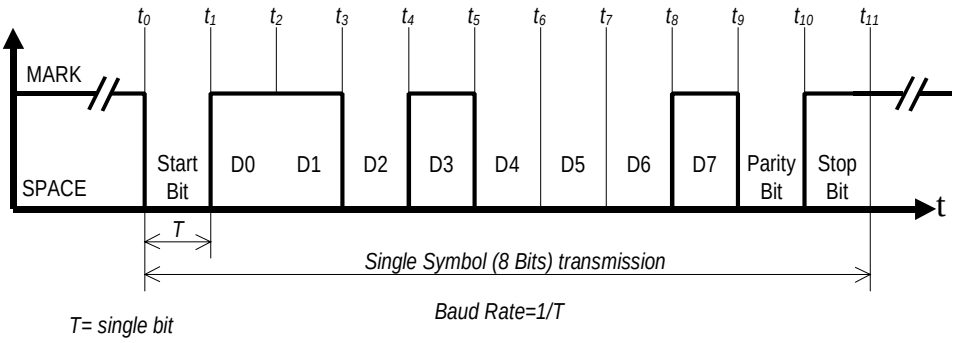

Figure 1 Serial Data Stream

In asynchronous communication, data is preceded with a start bit (Space) that indicates to the receiver that a word (a chunk of data broken up into individual bits) is about to begin. The end of a word is followed by a stop bit (Mark), which tells the receiver that the word has come to an end. Now it should begin looking for the next start bit. Any bits it receives before getting the start bit should be ignored. To insure data integrity, a parity bit is often added between the last bit of data and the stop bit. The parity bit ensures that the data received is composed of the same number of bits in the same order in which they were sent.

UART Architecture

The UART circuit enables a computing processing unit (CPU) serial access to the external peripheral. The interface between the CPU and the UART is usually byte parallel and can be synchronous (i.e. Register Map interface). The transmission properties of the UART, such as parity check, number of symbol bits, number of stop bits etc., can be programmed via a control register which is part of the UART circuitry. The CPU can configure the UART by writing the

specific control bits via the parallel interface. Figure 2 below illustrates the simplified block diagram of an UART circuit, including its interface.

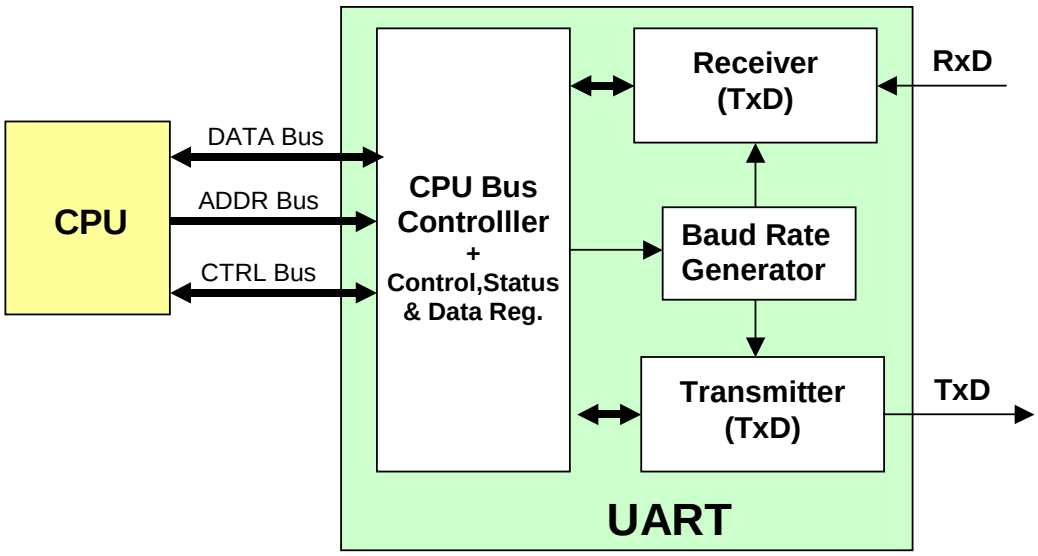

Figure 2 Simplified Block diagram of an UART

The URAT consists of the following four main function blocks:

- CPU Bus Controller
- Baud rate generator
- Receiver
- Transmitter

CPU Bus Controller

The CPU Bus Controller provides the parallel data I/O interface to the local processor bus. It generates the necessary control signal to enable the uP (or CPU) to access onto the data, status and control register of the UART circuit. Furthermore, it generates the local control signal within the UART circuit according to the control register content and updates the status register content according to the local status signal values generated by the other function blocks. It also accommodates the transmitter and receiver buffer (Hold Register or FIFO) that is essential for asynchronous transmission.

Baud Rate Generator

The Baud Rate Generator is a programmable transmit and receive bit timing device. Given the programmed value, it generates a periodic pulse, which determines the baud rate of the UART transmission. This pulse is used by the receiver and transmitter circuit to generate a sampling pulse for sampling the received serial data and to determine the bit width of the transmit data.

Receiver

The Receiver block detects the start bit of an incoming serial data and samples the data bits, bit by bit, according to the baud clock of the baud rate

generator. It completes the receive process of a single symbol of 6,7 or 8 bits with the detection of the stop bit (the stop bit can be 1, 1.5 or 2 bits width). Parity check of the received symbol ensures that the data has been received correctly. In the case of invalid stop bit or parity check error, the status signals parity error or frame error will be set. Finally the receiver writes the received symbol data onto the local data bus, which connected to the CPU Bus controller, and sets a signal to indicate "Receiver data Write". This signal initiates that the CPU Bus controller informs the CPU via an interrupt about the data arrival.

Transmitter

The transmitter block is responsible for the serial transmitting of the data, which is written by the uP (CPU) onto the TxD Hold register (or FIFO) at the CPU Bus controller block. First the transmitter detects whether the UART transmitter buffer (FIFO or TxD Hold Register) contains data for transmission. If it does, it loads the data onto the transmit register at the transmitter circuit via the local data bus (which connects the CPU Bus controller and the transmitter) and sets a signal to indicate "Transmit data Read". This signal initiates a sequence where the CPU Bus controller informs the CPU via an interrupt about the transmitted data, so that the CPU can load a new value. Synchronous to the baud clock which is generated by the baud rate generator, the transmitter sets the start bit on the TxD signal line to initiate the start of a frame and then bit by bit the symbol data. It finally completes the transmission by sending a parity bit that represents the parity of transmitted data and completes the frame with the final stop bit. The procedure will be repeated for another symbol, if the transmitter buffer contains another symbol, else the transmitter goes to an idle mode, transmitting "Marks".

UART Design specification

As part of this laboratory, only the receiver baud rate generator and the transmitter circuit shall be designed. The CPU bus control and register block is not a part of the scope of this laboratory.

The targeted UART circuit shall supports following features:

- 7 bit and 8 bit symbol word size.
- Programmability parity check (EVEN and ODD parity)
- Frame Error Indication
- Parity Error Indication
- Baud rats according to the table below.

BAUD_SEL	Baud Rate
000	300 bits per second
001	1200 bits per second
010	4800 bits per second
011	9600 bits per second
100	19200 bits per second
101	38400 bits per second
110	57600 bits per second
111	115200 bits per second

Figure 3 illustrates the block diagram and the external interface of the UART circuit that shall be design as part of this laboratory.

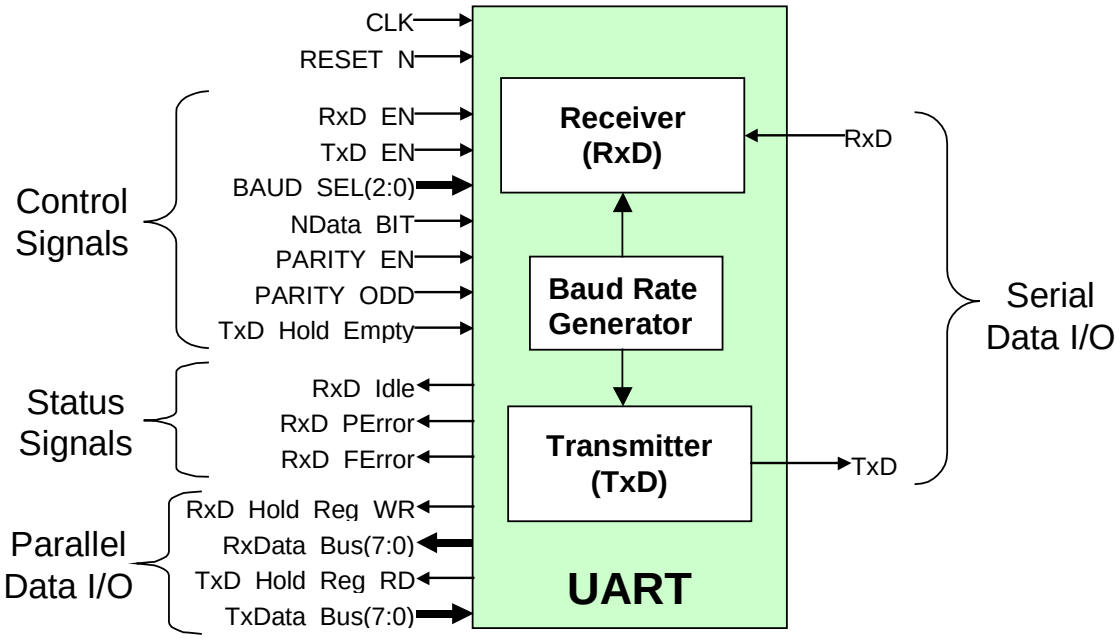

Figure 3 UART specification

The signal definitions are given overleaf and some timing information in figure 4.

Signal Name	Function	I/O	Activity
CLK	System Clock (connected to 14.7456 MHz)	Input	-- --
RESET_N	System Reset	Input	Low
RxD_EN	Receiver Enable	Input	High
TxD_EN	Transmitter Enable	Input	High
BAUD_SEL(2:0)	Determines the baud rate of the UART	Input	-- --
NData_BIT	Determines if transmit and receive frame contains 7 or 8 bits. '1' => 8bits, '0'=> 7 bits	Input	'0'=8bits '1'=7bits
PARITY_EN	Determines parity check '1'= Parity Enable	Input	High
PARITY_ODD	'0'= EVEN Parity Check, '1' =ODD Parity Check	Input	'0'=EVEN '1'=ODD
TxD_Hold_Empty	Indicate that the TX Hold Register is empty, therefore the CPU has no data for transmit. If set to '1' , the transmitter goes to idle state, else it reads next transmit data.	Input	High
RxD_Idle	Indicates RX is idle i.e. does not receive data	Output	High
RxD_Perror	Receiver parity error. Indicates that the last received frame contained a parity error. This will be set after the parity check until a new frame arrives, and should be evaluated during active pulse of RxD_Hold_Reg_WR.	Output	High
RxD_Ferror	Receiver framing error which indicates the last received frame contained a framing error (missing stop bit). This is set after detecting the stop bit until a new frame arrives, and should be evaluated during active pulse of RxD_Hold_Reg_WR .	Output	High
RxD_Hold_Reg_WR	Write strobe signal which indicates the RX has written new data into the RX Hold Register via the RxData_Bus(7:0).	Output	High
RxData_Bus	Receiver Data Bus (7:0)	Output	-- --
TxD_Hold_Reg_RD	Read strobe signal that indicates the TX has read current data at the TX Hold Register via the TxData_Bus(7:0).	Output	High
TxData_Bus	TX Data Bus (7:0)	Input	-- --
RxD	Serial Receiver Input	Input	-- --
TxD	Serial Transmitter Output	Output	-- --

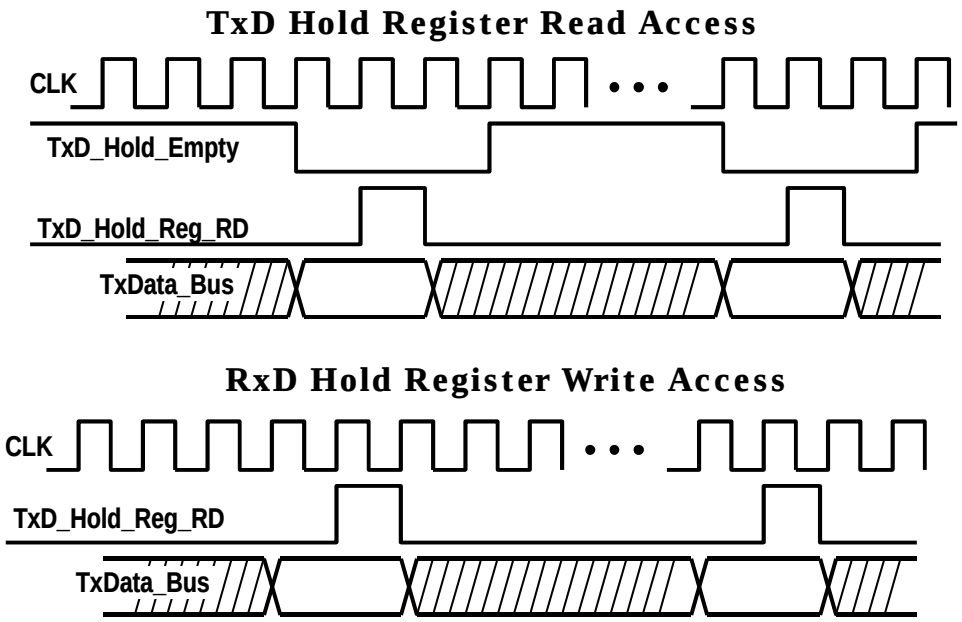

Figure 4 Timing diagram of the Parallel I/O interface

Notes: