**Technion**

*Electrical Engineering Department*

High Speed Digital System Lab

UART

Documentation

**Version**: 2.0

**Last update**: 18.03.2011

**Written By**: Beeri Schreiber and Alon Yavich

Table of content

[Table of Changes 3](#_Toc293164660)

[1 Scope 3](#_Toc293164661)

[2 Abbreviations 3](#_Toc293164662)

[3 General Description 3](#_Toc293164663)

[3.1 UART Transmitter and Receiver 3](#_Toc293164664)

[3.2 UART Characteristics 3](#_Toc293164665)

[3.3 Pinout 4](#_Toc293164666)

[3.4 UART RX Generic Parameters 4](#_Toc293164667)

[4 Basic features and function description 5](#_Toc293164668)

[4.1 FSM Diagram 6](#_Toc293164669)

[4.2 RX Operation 7](#_Toc293164670)

[5 UART TX Pinout 7](#_Toc293164671)

[5.1 UART TX Generic Parameters 7](#_Toc293164672)

[6 Basic features and function description 8](#_Toc293164673)

[6.1 FSM Diagram 9](#_Toc293164674)

[6.2 TX Operation 9](#_Toc293164675)

[6.3 Resources 10](#_Toc293164676)

[6.3.1 RX 10](#_Toc293164677)

[6.3.2 TX 10](#_Toc293164678)

[6.4 UART Simulation Tests 11](#_Toc293164679)

Table of Tables

[Table 1 – Table of changes 3](#_Toc275835884)

Table 2 – UART RX Pinout 4

Table 3 – UART RX Generic Pinout 4

[Table 4 – UART TX Pinout 7](#_Toc275835884)

Table 5 – UART TX Generic Pinout 7

Table 6 – UART Simulation test 11

Table of Figures

[Figure 1 – UART RX Pinout 4](#_Toc275754778)

[Figure 2 – Middle bit sampling 5](#_Toc275754778)

[Figure 3 – UART RX FSM Diagram 6](#_Toc275754779)

[Figure 4 – UART RX Operation 7](#_Toc275754780)

[Figure 5 – UART TX Pinout 7](#_UART_TX_Pinout)

[Figure 6 – UART TX FSM Diagram 9](#_FSM_Diagram)

[Figure 7 – UART TX Operation 9](#_TX_Operation)

# Table of Changes

|  |  |  |
| --- | --- | --- |
| Version | Date of Change | Description |
| 1.0 | 26.02.2011 | Creation of documentation |
| 2.0 | 18.03.2011 | Modifications |

Table 1 – Table of Changes

# Scope

This document aims to describe the UART (Universal Asynchronous Receiver Transmitter), together with an example environment of how to use it.

# Abbreviations

1. UART - Universal Asynchronous Receiver Transmitter
2. UART RX – UART Receiver
3. UART TX – UART Transmitter
4. TB – Test bench

# General Description

## UART Transmitter and Receiver

This component is used for asynchronous serial data channel.

The receiver converts serial start bit, data, parity and stop bit to parallel data.

The transmitter converts parallel data into serial form and automatically adds start bit, parity and stop bit.

The data word length can be 5-8 bits, according to generic parameter. Parity bit can be odd or even or if decided can be inhibited, according to generic parameters.

All inputs and outputs are synchronized with the positive edge of the clock.

Any system clock and any baud rate are supported, according to generic parameter.

## UART Characteristics

The UART is implemented with 2 components:

1. UART RX – UART Receiver
2. UART TX – UART Transmitter

Both of the components are based on the same characteristics, the parameters are generics:

1. **Start bit**: '1' or '0' according to user's choice (*uart\_idle\_g*).
2. **Data length**: 5-8 data bits (between Start and stop bit).
3. **Parity bit**: can be added after the data bits frame, include 3 generic options:
   1. Odd - a bit will be added so the total '1' bits will be odd.
   2. Even - a bit will be added so the total '1' bits will be even.
   3. Inhibited.
4. **Stop bit**: opposite of Start bit, 1 bit in length.
5. **Reset**: Reset polarity can be chosen.
6. **Baud Rate**: Transmission rate.
7. **System Clock:** 133MHzUART RX

## Pinout

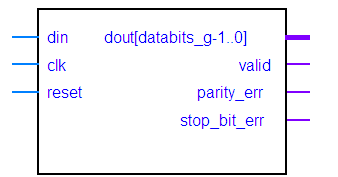


Figure 1 – UART RX Pinout

| Pin Name | Direction | Description |
| --- | --- | --- |
| Clk | In | Input clock. |
| Reset | In | Reset. Reset polarity will be set according to the generic parameter *'reset\_polarity\_g'*. |
| Din | In | Serial data in. |
| Parity\_err | Out | '1' when there is an error in parity bit. |
| Stop\_bit\_err | Out | '1' when there is an error in stop bit. |
| Valid | Out | '1' when Parallel data valid. |
| Dout[7..0] | Out | Parallel data out. |

Table 2 – UART RX Pinout

## UART RX Generic Parameters

| Generic Parameter | Default Value | Description |
| --- | --- | --- |
| Parity\_en\_g | '0' | '1' when Parity bit enabled. |
| Parity\_odd\_g | False | False – parity EVEN, true – parity ODD. |
| uart\_idle\_g | '1' | IDLE state line value. |
| baudrate\_g | 115200 | UART's baudrate [Hz]. |
| Clkrate\_g | 133333333 | System clock [Hz]. |
| Databits\_g | 8 | Number of data bits. |
| Reset\_polarity\_g | '0' | '0' – reset is 'Active Low'. |

Table 3 – UART RX Generic Pinout

# Basic features and function description

UART RX is implemented with a FSM as its main process. It is synchronized with the system clock, and responds in any time to a Reset input with the return to IDLE state and zeroes all relevant signals. The FSM is based on a serial data flowing in, as the machine characterizes each bit as the protocol compels. Each bit which enters to the UART RX component is sampled 5 times in its middle (as shown in the wave) so there will not be any mistakes about its value, in case of lines spikes.

Figure 2 – Middle bit sampling

Another process takes place in the beginning and prevents metastability. It does so with the data in goes through two FF. where the first FF is probably in metastabiliy state but the chance that the metastability will pass beyond the second FF is almost zero.

The output is a parallel data implemented with a shift register.

An important internal constant helps us to count the period time for each bit:

* + - *Clk\_div\_factor* – is an internal constant in the FSM that represents the ratio between the system clock and the UART baud rate. This ratio is exactly the period length of one bit in the incoming data.
    - 

Each data bit that enters the machine is being sampled 5 times at its middle so it will be absolutely clear what is the value of the data sampled, i.e. need at least 3 samples of the same value to deiced the data value. The most important place this sampling takes place is when start bit is being detected – in this case we can avoid spikes and making the implementation noise proof.

The FSM has 5 states:

1. IDLE

The state waits for start bit (shown when inversion in *uart\_idle\_g* is being detected).

An internal counter counts until reaches to half the *clk\_div\_factor* (Middle of the bit) and then moves on to the next state.

1. STARTBIT

This state makes sure we received the start bit (with the sampler) then moves on to the next state.

* + - From now each state will advance to the next suitable state only when Clk\_div\_factor is reached.

1. RX

This state sampling each of the data bits (5-8 bits according to user's choice) while using a shift register to place the data in *dout* (represents parallel data). In case there is a parity bit (defined by the user) the FSM goes to PARITY states or straight to STOPBIT state when parity bit is inhibited.

In addition, parity value is being calculated for checking with the real parity entered later.

Calculation is according to user's choice of odd / even parity bit.

1. PARITY

The Parity that has been calculated in the RX state is now being compared with the parity bit entered, if equals going to next state (STOPBIT), if not raising a parity error flag.

1. STOPBIT

Stop bit received should be the same as *uart\_idle\_g* if not flag is rises (*stopbit\_err*).

While waiting for end of clock counter or the beginning of the next bit, valid flag is in high level to indicate that a word has been received properly.

Machine returns to IDLE state.

## FSM Diagram

Figure 3 – UART RX FSM

## RX Operation

Figure 4 – Uart RX Operation

This simulation shows the RX operation in all its stages. Detecting the start bit afterwards 8 data bits and a stop bit. Valid flag rises to high level when whole word received and *dout* has the relevant parallel data.

# UART TX Pinout

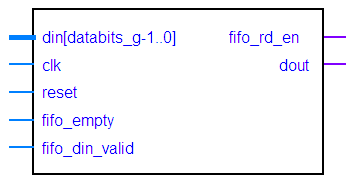


Figure 5 – UART TX Pinout

| Pin Name | Direction | Description |
| --- | --- | --- |
| Clk | In | Input clock |
| Reset | In | Reset. Reset polarity will be set according to the generic parameter 'reset\_polarity\_g'. |
| Din[7..0] | In | Parallel data in. |
| fifo\_empty | In | '1' when FIFO is empty . |
| fifo\_valid | in | '1' when FIFO data is valid. |
| fifo\_rd\_en | Out | '1' for FIFO's rd\_en line – data can be read from FIFO. |
| Dout | Out | Serial data out. |

Table 4 – UART TX Pinout

## UART TX Generic Parameters

| Generic Parameter | Default Value | Description |
| --- | --- | --- |
| Parity\_en\_g | '0' | '1' when Parity bit enabled. |
| Parity\_odd\_g | false | false – parity EVEN, true – parity ODD. |
| uart\_idle\_g | '1' | IDLE state line value. |
| baudrate\_g | 115200 | UART's baudrate [Hz]. |
| Clkrate\_g | 133333333 | System clock [Hz]. |
| Databits\_g | 8 | Number of data bits. |
| Reset\_polarity\_g | '0' | '0' – reset is 'Active Low'. |

Table 5 – UART TX Generic Pinout

# Basic features and function description

UART TX is implemented with a FSM as its main process. A secondary process is implementing and functions as a Shift register that also calculating the parity bit. The processes are synchronized with the system clock, and respond in any time to a Reset request with the return to IDLE state and zero all relevant signals.

The FSM is based on a parallel data flowing from a FIFO and then into the Shift register. Start bit and stop bit is being positioned in the LSB and MSB palaces in the Shift register. Every bit transmission the register is being shifted to the Right (UART's standard is LSB first) so the next bit will be ready for transmission, this way we receive Serial data in the output.

TX FSM has 3 states:

* + - Each state will advance to the next suitable state only when *Clk\_div\_factor* is reached.

1. IDLE

In this state, the machine waits for FIFO not to be empty for going to the next state, when this happens *fifo\_rd\_en* flag rises and indicates that TX unit is ready to receive data from the FIFO.

1. REGDATA

Intermediate state that waits for FIFO validation so the parallel data from FIFO could be pulled into the shift register in TX\_ST.

Start and Stop bits are being positioned inside the shift register according to LSB first configuration, Parity bit is also being calculated according to user's odd/even choice.

The data itself from the FIFO is being positioned in the shift register.

1. TX

Data bits are being pulled out from Shift register that is shifting right after each bit is transmitted. In case there is a parity bit (defined by the user) the calculated parity bit will also be transmitted. After all bits where transmitted, FSM goes back to IDLE state.

* Each step in the shift register will take place only when *Clk\_div\_factor* is reached.

## FSM Diagram

Figure 6 – UART TX FSM

## TX Operation

Figure 7 – Uart TX Operation

This simulation shows the TX operation in all its stages. Generating the start bit and after FIFO Valid flag is in a high level transmitting the data itself, At the end generating parity bit ( if needed ) and stop bit.

## Resources

### RX

* 6 AND gates
* 6 OR gate
* 41 DFF
* 123 MUX
* 1 State machine
* 4 additional operators
* 7 'Less Then' operators
* 24 selector operators
* 4 Equal operators

Maximum Frequency: 160MHz

### TX

* 1 AND gates
* 28 DFF
* 66 MUX
* 1 State machine
* 2 additional operators
* 2 Equal operators

 Maximum Frequency: 300 MHz

## UART Simulation Tests

Macro ( Do ) files had been created so multiple simulations could be execute more easily.

The variable parameters include TX and RX baud rate, Parity bit and Odd/Even parity.

Validation against expected data indicates if test had passed or not.

One Kbyte of data transfer has been simulated.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pass/Fail** | **Description** | | | | | | **Test Num** |
|  | ODD/EVEN | PARITY | Deviation % | RX BAUD. | TX BAUD. | CLK MHz |  |
| Pass | x | x | 0 | 115200 | 115200 | 133 | 1 |
| Pass | x | x | -4 | 115200 | 110592 | 133 | 2 |
| Pass | x | x | +5 | 115200 | 120960 | 133 | 3 |
| Fail | x | x | -5 | 115200 | 109440 | 133 | 4 |
| Pass | ODD | V | 0 | 115200 | 115200 | 133 | 5 |
| Pass | ODD | V | -5 | 115200 | 109440 | 133 | 6 |
| Pass | ODD | V | +5 | 115200 | 120960 | 133 | 7 |
| Pass | EVEN | V | 0 | 115200 | 115200 | 133 | 8 |
| Pass | EVEN | V | -4 | 115200 | 110592 | 133 | 9 |
| Pass | EVEN | V | +5 | 115200 | 120960 | 133 | 10 |
| Pass | x | x | 0 | 115200 | 115200 | 33 | 11 |
| Pass | x | x | -5 | 115200 | 109440 | 33 | 12 |
| Pass | x | x | +5 | 115200 | 120960 | 33 | 13 |
| Fail | x | x | -6 | 115200 | 108288 | 33 | 14 |
| Pass | ODD | V | 0 | 115200 | 115200 | 33 | 15 |
| Pass | ODD | V | -5 | 115200 | 109440 | 33 | 16 |
| Pass | ODD | V | +5 | 115200 | 120960 | 33 | 17 |
| Pass | EVEN | V | 0 | 115200 | 115200 | 33 | 18 |
| Pass | EVEN | V | -5 | 115200 | 109440 | 33 | 19 |
| Pass | EVEN | V | +5 | 115200 | 120960 | 33 | 20 |

Table 6 – UART Simulation Test