SpartanMC

Simple Universal Asynchronous Receiver Transmitter (UART Light)

Table of Contents

List of Figures

List of Tables

Simple Universal Asynchronous Receiver Transmitter (UART Light)

The UART Light is a SpartanMC peripheral device for serial communication with external systems. Compared to the standard UART the UART Light uses a lightwight interface which enables a smaller resource footprint. To provide the minimum interface the peripheral is primarily configured via pre synthesis parameters.

The UART Light bitstream is fixed to 8 databits per frame, the datarate and FIFO buffer depth is configurable via module paramater. The signals Rx and Tx are used as connection to the external system environment. The incoming and outgoing data is written to FIFO memory modules which are connected to the input and output shift registers of the UART.

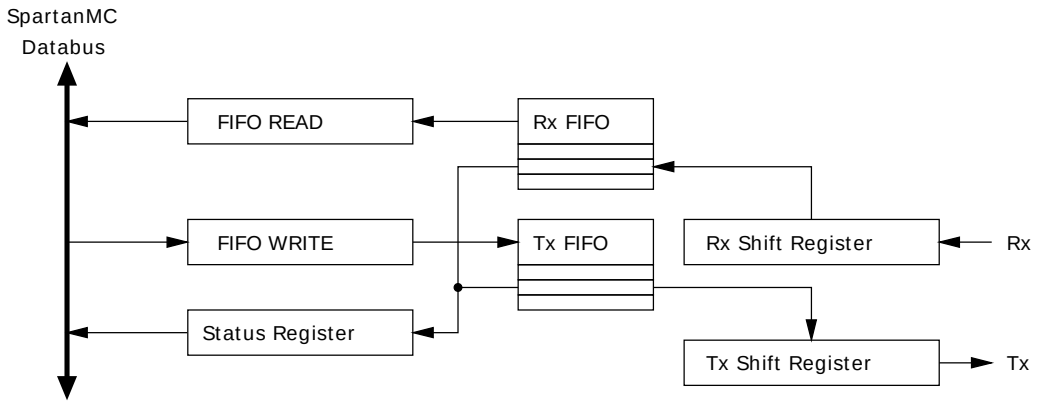

Figure 1: UART Light block diagram

1. Framing

Each frame starts with a logic low start bit followed by a fixed number of 8 databits. Parity bits and additional stop bits are not supported. The transmission of the data field starts with the least significant bit (LSB).

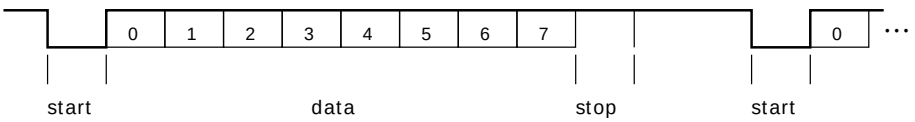

Figure 2: UART Light frame

2. Module parameters

Table 1: UART module parameters

Parameter	Default Value	Descripton
BASE_ADR	0x10	Start address of the memory mapped peripheral registers. The value is taken as offset to the start address of the peripheral memory space. **This parameter is set by jConfig automatically.**
FIFO_RX_DEPTH	8	Number of 8 bit cells for incoming data of the receiver FIFO memory. (maximum 2048)
FIFO_TX_DEPTH	8	Number of 8 bit cells for outgoing data of the sender FIFO memory. (maximum 2048)
BAUDRATE	115200	Defines the required baud rate: 921600 Baud 576000 Baud 460800 Baud 230400 Baud 115200 Baud 57600 Baud 28000 Baud 14400 Baud 7200 Baud 4800 Baud 2400 Baud 1200 Baud 600 Baud 300 Baud
INTERRUPT_SUPPORTED	FALSE	If this parameter is set to FALSE, the content of the FIFO memories must be determined via polling. If this parameter is set to TRUE, each received or sent frame can generates an interrupt.
CLOCK_FREQUENCY	25000000	The clock frequency in Hz which the module is currently driven by. This frequency is used to calculate the prescalers for the configured baudrate.

3. Interrupts

In case the UART Light is synthezied for interrupt mode it provides two interrupt signals. Both interrupt signals must be enabled in the UART STATUS register. The interrupts will be generated at the following conditions:

- The Tx interrupt (intr_tx) is set to one if the Tx FIFO status not full and write in last time to Tx-Data. The interrupt indicates that one data in the Tx FIFO memory were sent.
- The Rx interrupt (intr_rx) is set if the first frame is written to the Rx FIFO memory.

The Tx interrupt reset is perfomed by reading the UART_STATUS register. The Rx interrupt reset is perfomed by reading data from the Rx FIFO memory.

Note: In case the software application uses this interrupt, the SpartanMC SoC requires an interrupt controller which provides an appropriate inferface for the interrupt signal.

4. Peripheral Registers

4.1. UART Register Description

The UART peripheral provides three 18 bit registers which are mapped to the Spartan-MC address space located at 0x1A000 + BASE_ADR + Offset.

Table 2: UART registers

Offset	Name	Access	Description
0	UART_STATUS	read/ (write)	Contains the current Rx/Tx FIFO status. For resetting the tx interrupt this register should be write.
1	UART_FIFO_READ	read	Register for incoming data. The LSB of the data word is written to $UART\_FIFO\_READ_0$. For resetting the rx interrupt this register should be read.
2	UART_FIFO_WRITE	write	Register for outgoing data. The LSB of the data word is written $UART\_FIFO\_WRITE_0$.

4.2. UART_STATUS Register

Table 3: UART status register layout

Bit	Name	Access	Default	Description
0	RX_EMPTY	read	1	Set to one if the Rx FIFO is empty otherwise the value is zero.
1	RX_FULL	read	0	Set to one if the Rx FIFO is full otherwise the value is zero.

Bit	Name	Access	Default	Description
2	TX_EMPTY	read	1	Set to one if the Tx FIFO is empty otherwise the value is zero.
3	TX_FULL	read	0	Set to one if the Tx FIFO is full otherwise the value is zero.
4	TX_IRQ_PRE	read	1	Set to one while data were witten to the Tx FIFO. The value is reset to zero through the next TX access to the Tx FIFO memory.
5	TX_IRQ_FLAG	read	0	Set to one while data were sent. The value is reset to zero through the next write access to TX_IRQ_FLAG.
6-8	x	read	0	Not used.
9	RX_IRQ_ENABLE	read/ write	0/0	To enable rx interrupt set this bit should be set to otherwise set to zero. This bit is only in not polling mode available.
10	TX_IRQ_ENABLE	read/ write	0/0	To enable tx interrupt this bit should be set to one otherwise set to zero. This bit is only in not polling mode available.
11-17	x	read	0	Not used.

Table 3: UART status register layout

4.3. UART_FIFO_READ Register

Table 4: UART status register layout

Bit	Name	Access	Default	Description
0-7	RX	read	0	Register for received data.
8-17	x	read	0	Not used.

4.4. UART_FIFO_WRITE Register

Table 5: UART status register layout

Bit	Name	Access	Default	Description
0-7	TX	write	0	Register for data to send.
8-17	x	write	x	Not used.

4.5. UART C-Header for Register Description

```
#ifndef UART_LIGHT_H_
#define UART_LIGHT_H_

#include ‹stdint.h›

// Status Signale
#define UART_LIGHT_RX_EMPTY (1 « 0)
#define UART_LIGHT_RX_FULL (1 « 1)
#define UART_LIGHT_TX_EMPTY (1 « 2)
#define UART_LIGHT_TX_FULL (1 « 3)
#define UART_LIGHT_TX_IRQ_PRE (1 « 4)
#define UART_LIGHT_TX_IRQ_FLAG (1 « 5)

// Steuersignale
#define UART_LIGHT_RX_IE (1 « 13)
#define UART_LIGHT_TX_IE (1 « 14)

// Rueckgabewerte fuer non blocking read
#define UART_LIGHT_OK 0
#define UART_LIGHT_NO_DATA 1

typedef struct {
   volatile uint18_t status; // r/w w = Reset Tx Interrupt
   volatile uint18_t rx_data; // r r = Reset Rx Interrupt
   volatile uint18_t tx_data; // w
} uart_light_regs_t;

void uart_light_send (uart_light_regs_t *uart, unsigned char
value);
unsigned char uart_light_receive (uart_light_regs_t *uart);
int uart_light_receive_nb (uart_light_regs_t *uart, unsigned
char *value);

#endif /*UART_LIGHT_H_*/
```