SpartanMC

Universal Asynchronous Receiver Transmitter (UART)

Table of Contents

List of Figures

List of Tables

Universal Asynchronous Receiver Transmitter (UART)

The UART is a SpartanMC peripheral device for serial communication with external systems. The UART enables a bit stream of 5-8 bits to be shifted in and out of the peripheral at a programmed bit rate. The peripheral is connected with the external system environment by the two signals Rx and Tx. If the UART is parametrized as modem, the additional signals DTR, DCD and CTS are usable. The incoming and outgoing data is written to configurable FIFO memory modules which are connected to the input and output shift registers of the UART.

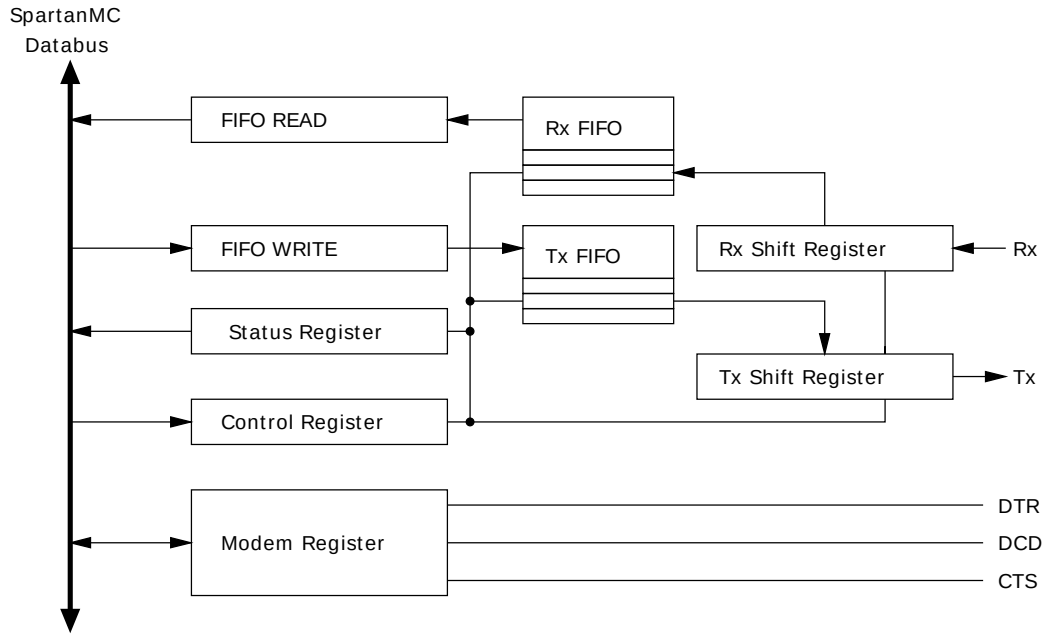

Figure 1: UART block diagram

1. Framing

Each frame starts with a logic low start bit followed by a configurable number of data bits (5-8), an optional parity bit and one or more logic hight stop bits. The parity bit can be defined as odd or even. If the parity is set to "even", the hamming weight of all data bits including the partiy bit have to be even for a valid frame. Contrariwise, the hamming weight of all data bits and the parity bit have to be odd if the parity bit is set to "odd". The transmission of the data field starts with the least significant bit (LSB).

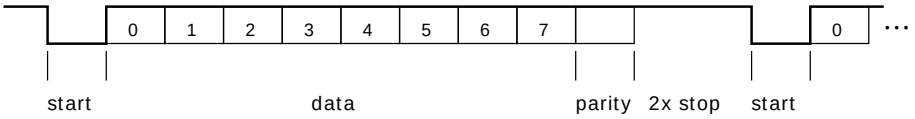

Figure 2: UART frame example

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 of the receiver FIFO memory for incoming data.
FIFO_TX_DEPTH	8	Number of 8 bit cells of the sender FIFO memory for outgoing data.
MODEM	0	Allows the usage of the UART peripheral control signals. If MODEM is set to one the control signals RTS, DCD and DTR are active. If MODEM is set to zero RTS, DTR and DTR are set to logic high.
CLKIN_FREQ	25000000	The clock frequency in Hz which the module is currently driven by. This frequency is used to calculate the prescalers for the different baud rates available in UART_CTRL.

3. Interrupts

For interrupt driven serial communication the UART provides two interrupt signals which will be generated at the following conditions:

- TX interrupt is set for each sent character. The interrupt remains one until the next read access to UART_CTRL (Register Nr. 3).
- RX interrupt is set for each received character. The interrupt remains one until the next read acces to UART_FIFO_READ (Register Nr. 1).

Note: In case the software application uses these interrupts, the SpartanMC SoC requires an interrupt controller which provides an appropriate interface for the interrupt signals.

4. Peripheral Registers

4.1. UART Register Description

The UART peripheral provides four 18 bit registers which are mapped to the SpartanMC address space e.g. 0x1A000 + BASE_ADR + Offset.

Table 2: UART registers

Offset	Name	Access	Description
0	UART_STATUS	read	Contains the current Rx/Tx FIFO status, current framing/parity errors, the modem signals if available and the the busy signals for the Rx/Tx shift registers.
1	UART_FIFO_READ	read	Register for incoming data. The LSB of the data word is written to $\texttt{UART\_FIFO\_READ}_0$.
2	UART_FIFO_WRITE	write	Register for outgoing data. The LSB of the data word is written $\texttt{UART\_FIFO\_WRITE}_0$.
3	UART_CTRL	read/ write	Contains the current UART setting e.g. baud rate, data field length and parity and interrupt settings.
4	UART_MODEM	read/ write	Contains the input/output setting for the modem signals.

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.
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	RX_P_ERR	read	0	Set to one until the next frame if a parity error occurs.
7	RX_F_ERR	read	0	Set to one until the next frame if a frameing error (break character or high impetance input) occurs.
8	RX_D_ERR	read	0	Set to one if the Rx FIFO is full while receiving a frame. The value remains one until the status of the Rx FIFO is not full.
9	M_DCD	read	0	The current value of the modem signal DCD if the parameter MODEM is set to one.
10	M_CTS	read	0	The current value of the modem signal CTS if the parameter MODEM is set to one.
11	M_DSR	read	0	The current value of the modem signal DSR if the parameter MODEM is set to one.
12	RX_CLK	read	0	Current clock of the Rx shift register. (For debugging purposes)
13	RX_STOP	read	0	It is set to one while the Rx shift register is not busy.
14	TX_CLK	read	0	Current clock of the Tx shift register. (For debugging purposes)
15	RST_UART	read	0	Is set to one during a UART reset. Typically, the reset is started with the SpartanMC processor core reset signal. The reset remains one for one UART bit period. This is equivalent to 8.6 us at 115200 Baud.
16	TX_STOP	read	0	It is set to one while the Tx shift register is not busy.
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	x	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	x	Register for data to send.
8-17	x	write	x	Not used.

4.5. UART_CTRL Register

Note: Befor writing UART_CTRL it has to be assured that RST_UART are set to zero and RX_EMPTY, TX_EMPTY, RX_STOP and TX_STOP are set to one.

Table 6: UART control register layout

Bit	Name	Access	Default	Description
0	RX_EN	write	0	Turns the Rx shift register off if set to zero.
1	TX_EN	write	0	Turns the Tx shift register off if set to zero.
2	PARI_EN	write	0	If set to one the usage of parity bits in frames will be enabled.
3	PARI_EVEN	write	0	If set to one the parity is even otherwise the parity is odd.
4	TWO_STOP	write	0	Enables the usage of a second stop bit.
5-7	DATA_LEN	write	000	Sets the length of the data field : 111 = 8 bit data 110 = 7 bit data 101 = 6 bit data 100 = 5 bit data
8	x	write	x	Not used.
9-12	BPS	write	0000	Sets the baud rate: 0000 = 115200 Baud 0001 = 57600 Baud 0010 = 38400 Baud 0011 = 31250 Baud (MIDI data rate) 0100 = 19200 Baud 0101 = 9600 Baud 0110 = 4800 Baud 0111 = 2400 Baud 1000 = 1200 Baud 1001 = 600 Baud 1010 = 300 Baud 1011 = 150 Baud 1100 = 75 Baud 1101 = 50 Baud All other values are mapped to 7812.5 Baud.
13	RX_IE	write	0	If set to one the Rx interrupt will be enabled.

Bit	Name	Access	Default	Description
14	TX_IE	write	0	If set to one the Tx interrupt will be enabled.
15	TX_BREAK	write	0	If set to one the UART will sent a break signal. (Tx logic low) The duration of the break signal must be longer than a complete frame (start, data, stop and parity). To identify breaks the framing error detection can be used.
16-17	x	write	x	Not used.

Table 6: UART control register layout

4.6. UART_MODEM Register

The UART peripheral provides three control lines for hardware handshaking and flow control: Data Carrier Detect (DCD), Data Set Ready (DSR)/Data Terminal Ready (DTR) and Request To Send (RTS)/Clear TO Send (CTS). The signal name and the i/o-direction depends on the RS-232 device class - Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). For more information cf. the EIA-232 or RS-232 standard.

Table 7: UART modem register layout

Bit	Name	Access	Default	Description
0	DTR_DSR	read/ write	0	If DTR is used as output DTR tells DCE that DTE is ready to be connected. If DTR is used as input DSR tells DTE that DCE is ready to receive commands or data.
1	RTS_CTS	write	0	If RTS is used as input RTS tells DCE to prepare to accept data from DTE. If RTS is used as output CTS acknowledges RTS and allows DTE to transmit.
2	DCD	write	0	Tells the DTE that DCE is connected to the carrier line.
3	DCD_OUT	write	0	If set to one DCD works as output which indicates the peripheral is used as DCE. If set to zero DCD works as input and the peripheral is used as DTE.
4	RTS_OUT	write	0	If set to one RTS works as output which indicates the peripheral is used as DTE. If set to zero RTS works as input and the peripheral is used as DCE.
5	DTR_OUT	write	0	If set to one DTR works as output which indicates the peripheral is used as DTE. If set to zero DTR works as input and the peripheral is used as DCE.
6-17	x	write	x	Not used.

Table 7: UART modem register layout

4.7. UART C-Header for Register Description

```
#ifndef __UART_H
#define __UART_H

#include ‹stdint.h›

// Status Signale
#define UART_RX_EMPTY (1 « 0)
#define UART_RX_FULL (1 « 1)
#define UART_TX_EMPTY (1 « 2)
#define UART_TX_FULL (1 « 3)
#define UART_TX_IRQ_PRE (1 « 4)
#define UART_TX_IRQ_FLAG (1 « 5)
#define UART_RX_P_ERR (1 « 6)
#define UART_RX_F_ERR (1 « 7)
#define UART_RX_D_ERR (1 « 8)
#define UART_M_DCD (1 « 9)
#define UART_M_CTS (1 « 10)
#define UART_M_DSR (1 « 11)
#define UART_RX_CLK (1 « 12)
#define UART_RX_STOP (1 « 13)
#define UART_TX_CLK (1 « 14)
#define UART_RST_UART (1 « 15) // UART noch im RESET, wenn = 1
#define UART_TX_STOP (1 « 16)

// Steuersignale
#define UART_RX_EN (1 « 0)
#define UART_TX_EN (1 « 1)
#define UART_PARI_EN (1 « 2)
#define UART_PARI_EVEN (1 « 3) // 0 = ungerade / 1 = gerade
#define UART_TWO_STOP (1 « 4) // 0 = ein / 1 = zwei Stopbits

#define UART_DATA_LEN_5 (4 « 5) // 0x00080
#define UART_DATA_LEN_6 (5 « 5) // 0x000A0
#define UART_DATA_LEN_7 (6 « 5) // 0x000C0
#define UART_DATA_LEN_8 (7 « 5) // 0x000E0

#define UART_BPS_115200 (0 « 9) // 0x00000
#define UART_BPS_57600 (1 « 9) // 0x00200
#define UART_BPS_38400 (2 « 9) // 0x00400
#define UART_BPS_31250 (3 « 9) // 0x00600 MIDI Datenrate
#define UART_BPS_19200 (4 « 9) // 0x00800
#define UART_BPS_9600 (5 « 9) // 0x00A00
#define UART_BPS_4800 (6 « 9) // 0x00C00
#define UART_BPS_2400 (7 « 9) // 0x00E00
#define UART_BPS_1200 (8 « 9) // 0x01000
```

```
#define UART_BPS_600 (9 « 9) // 0x01200
#define UART_BPS_300 (10 « 9) // 0x01400
#define UART_BPS_150 (11 « 9) // 0x01600
#define UART_BPS_75 (12 « 9) // 0x01800
#define UART_BPS_50 (13 « 9) // 0x01A00
#define UART_BPS_7812 (14 « 9) // 0x01C00 Boot 68hc11
       // Boot 68hc11 mit 7812,5 Baud

#define UART_RX_IE (1 « 13)
#define UART_TX_IE (1 « 14)
#define UART_TX_BREAK (1 « 15)

// Modem Outputs und Richtung (optional)
#define UART_DTR_DSR (1 « 0)
#define UART_RTS_CTS (1 « 1)
#define UART_DCD (1 « 2)
#define UART_DCD_OUT (1 « 3)
#define UART_RTS_OUT (1 « 4)
#define UART_DTR_OUT (1 « 5)

// Rueckgabewerte fuer non blocking read
#define UART_OK 0
#define UART_NO_DATA 1

typedef struct {
   volatile uint18_t status;   // read
   volatile uint18_t rx_data;   // read (Reset Rx Interrupt)
   volatile uint18_t tx_data;   // write
   volatile uint18_t ctrl_stat;   // write (or read)
           // write (or read = status and Reset Tx Interrupt)
   volatile uint18_t modem;   // write (optional)
} uart_regs_t;

#endif
```