Specification of UART with AMBA AXI-Lite

Version 0 Revision 0
Aug. 10, 2024 (Aug. 10, 2024)

Future Design Systems
(http://www.future-ds.com)

504 TBC (Daedeok Tech Biz Center), 593 DaeDeokDae-Ro, Yuseong-Gu,
Daejeon 34112, Korea

Copyright notice

1 Introduction

RS-232-C (Recommended Standards) known as serial prtocol is an EIA (Electronic Industry Association) standard that defines a commonly used serial communications scheme. It is widely used to transfer data between computers or other devices using an asynchronous serial link at speeds ranging from 110 to 115,200 baud.

UART (Universal Asynchronous Receiver and Transmitter) is a controller providing the RS-232-C asynchronous serial communication protocol. One side of UART is an interface to processor and the other side is the serial port.

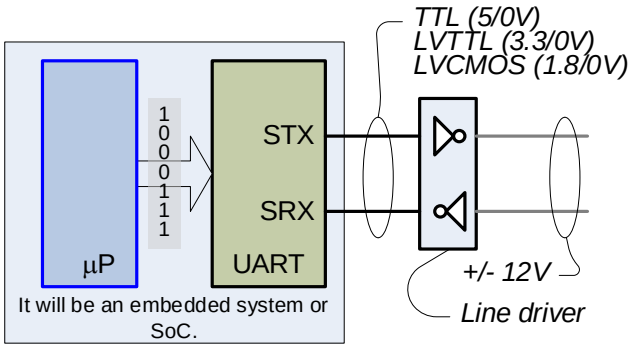

Figure 1: UART in the SoC

This document describes an implementation of UART that interfaces with the AMBA AXI-Lite protocol.

There are some highlights as follows.

- AMBA AXI-Lite interface
- Full duplex, i.e., one transmit and one receive channel
- Configurable data width 7-8 in a character
- Configurable parity bit (none, odd, or even)
- Configurable baud rate
- Macros to set clock frequency and baud rate
- It supports 8-bit, no-parity, 1-stop bit by default.

There are some limitations as follows.

- No hardware flow control
- No signals to control MODEM

2 Block diagram

'uart_axi_lite' consists of 'uar_axi_lite_if' and 'uart_core' sub-block, in which 'uart_core' consists of 'uart_csr', 'uart_rx', and 'uart_tx'.

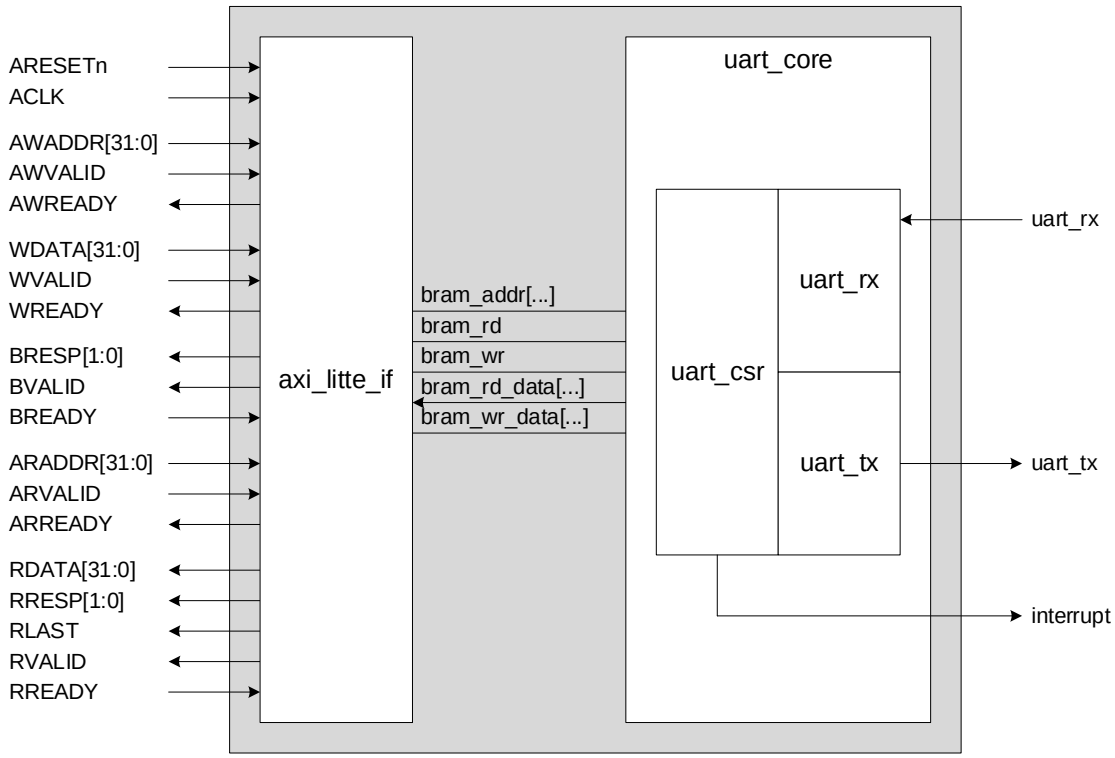

Figure 2: UART block

- AXI-Lite Interface Module: The AXI-Lite Interface Module provides the interface to the AXI-Lite and implements AXI-Lite protocol logic. It controls the core through BRAM-style interface.
- UART16550: The UART16550 Module consists of register interface, Receiver, Transmitter.

3 Control and Status Registers

Name	Address offset		Bit#	description
VERSION	+00h		RO	Version (0x2024_0810)
NAME	+04h		RO	“UART”
RESERVED	+08h			Reserved
	+0Ch			Reserved
CONTROL	+10h		RW	CONTROL register (default: 0x0000_01B2)
			31:23	Reserved
			22	RXCLR: RX-FIFO clear when 1 • return to 0
			21	IERX: RX interrupt is enabled when 1 • received character makes interrupt
			20	IETX: TX interrupt is enabled when 1 • transmit-done makes interrupt
			19	STOP: stop-bit • 0: 1-bit • 1: 2-bit
			18	EVEN: even parity when PARITY is 1 • 0: odd-parity • 1: even-parity[1]
			17	PARITY: parity enabled • 0: disabled • 1: enabled
			16	WID: data width • 0: 8-bit • 1: 7-bit
			15:0	DIV: clock division • div = clock_frequency/baud_rate+0.5 • by default 434 for 115200 baud
STATUS	+14h			STATUS
			31:5	Reserved
			4	PERR: Perry error
			3	RX-READY: RX is ready • there are data in the FIFO.
			2	TX-DONE: TX is done • the data has been transmitted
			1	IPRX: RX interrupt pending • writing 1 to this bit makes clear
			0	IPTX: TX interrupt pending • writing 1 to this bit makes clear
TXBUFF	+18h			TX Buffer
			31	valid flag that makes transmit when 1
			30:8	Reserved

[1] even parity: make even the number of 1 in data and parity.

			7:0	data to transmit
RXBUFF	+1Ch			RX Buffer
			31	valid flag
			30	parity error if any
			29:8	Reserved
			7:0	data received
CLK_FREQ	+20h			CLOCK frequency
			31:0	clock frequency

4 Operation

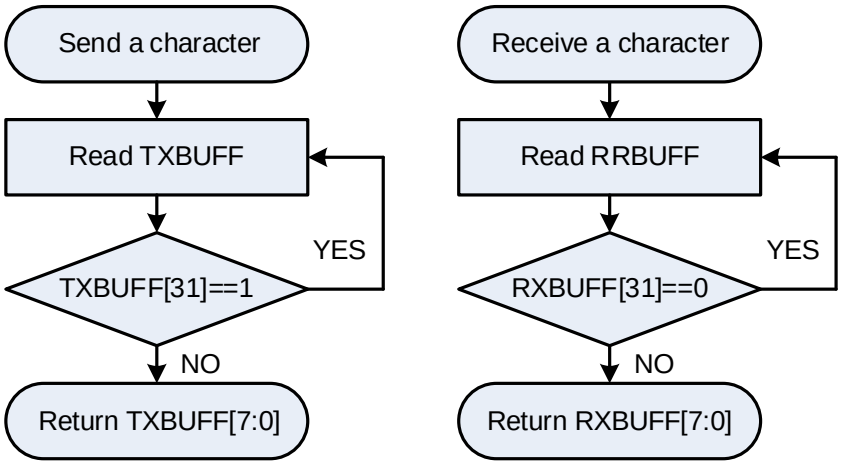

Figure 3: Send or receive a character

5 API

All API returns ‘0’ when completes successfully. Otherwise, it returns non-zero value.

```
void uart_init( unsigned int freq     // clock frequency in Hz (use CSR when 0)
              , unsigned int baudrate // baud rate
              , unsigned int width    // 8: 8-bit, 7: 7-bit
              , unsigned int parity   // 0: disable, 1: odd, 2 even
              , unsigned int stop     // 1: 1-bit, 2: 2-bit
              ) ;
```

```
unsigned int uart_put_char(char d)
```

- Returns the value of the character has been sent

```
unsigned int uart_get_char(void);
```

- Returns the value of received character

```
int dma_ahb_clear  ( void );
```

- interrupt clear

```
int uart_put_string(char* s)
```

- Returns the number of characters has been sent.

5.1 Typical usage

Following code shows a typical usage of API.

```
#include “uart_api.h”

int main() {
   uart_init( 0 // unsigned int freq     // clock frequency in Hz (use CSR when 0)
              , 115200 // unsigned int baudrate // baud rate
              , 8 // unsigned int width    // 8: 8-bit, 7: 7-bit
              , 0 // unsigned int parity   // 0: disable, 1: odd, 2 even
              , 1 // unsigned int stop     // 1: 1-bit, 2: 2-bit
              ) ;
   uart_put_string(“Hello world!”);
   return 0;
}
```

Wish list

References

[1] AMBA Specification, Rev. 3.0, ARM, www.arm.com.

Revision history

☐ 2024.08.10: Started by Ando Ki.