VERIFICATION PLAN DOCUMENT for WISHBONE2UART

# EVOLUTION OF DOCUMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Modifications** |
| 0.1 | 14.08.  2019 | Aleksandar Komazec | Initial release |
| 0.2 | 15.08 | Aleksandar Komazec |  |
|  |  |  |  |
|  |  |  |  |

# REFERENCES

**DUT documents:**

DUT’s specification documents:

* UART\_spec.pdf
* UART\_Protocol.pdf

DUT’s other documents:

* 16550.pdf

Protocol specification documents:

* WISHBONE\_B4\_Protocol.pdf

Protocol other documents:

* 8251-08-serial.pdf
* serial-communication-uart.pdf

# TABLE OF CONTENTS

[EVOLUTION OF DOCUMENT 2](#_Toc16745187)

[REFERENCES 3](#_Toc16745188)

[TABLE OF CONTENTS 4](#_Toc16745189)

[1. INTERFACES 5](#_Toc16745190)

[1.1 UART interface 5](#_Toc16745191)

[1.2 WISHBONE Interface 6](#_Toc16745192)

[1.2.1 Checker list 6](#_Toc16745193)

[1.2.2 Coverage list 8](#_Toc16745194)

[2. FUNCTIONAL FEATURES 9](#_Toc16745195)

[Master operation 10](#_Toc16745196)

[2.1.1 Checker list 10](#_Toc16745197)

[2.1.2 Coverage list 10](#_Toc16745198)

[3. REGISTER ACCESS 11](#_Toc16745199)

[3.1 Receiver Buffer Register (RBR) 11](#_Toc16745200)

[3.2 Transmitter Holding Register (THR) 12](#_Toc16745201)

[3.3 Interrupt Enable 12](#_Toc16745202)

[3.4 Interrupt Identification 13](#_Toc16745203)

[3.5 FIFO Control Register 13](#_Toc16745204)

[3.6 Line Control Register 14](#_Toc16745205)

[3.7 Modem Control Register 14](#_Toc16745206)

[3.8 Line Status Register 14](#_Toc16745207)

[3.9 Divisor latch register (byte 1) 15](#_Toc16745208)

[3.10 Divisor latch register (byte 2) 15](#_Toc16745209)

[3.11 Debug register 1 16](#_Toc16745210)

[3.12 Debug register 2 16](#_Toc16745211)

[3.13 Modem Status Register 16](#_Toc16745212)

[4. RESET AND CLOCK 17](#_Toc16745213)

[5. DEBUG AND TEST 18](#_Toc16745214)

[6. TEST LIST 18](#_Toc16745215)

[6.1 Wishbone control register write/read 18](#_Toc16745216)

[6.2 Wishbone fifo register write/read 18](#_Toc16745217)

[6.3 Wishbone fifo register write/read 19](#_Toc16745218)

## INTERFACES

### UART interface

Checker

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | All\_12\_bits\_are\_valid | All bits in a word must be valid  1 bit for start bit  8 bits for data  1 bit for parity  2 bits for stop bit | Protocol checker in monitor |
| 2 | All\_9\_bits\_are\_valid | All bits in a word must be valid  1 bit for start bit  5 bits for data  1 bit for parity  1 or 1,5 bit for stop bit | Protocol checker in monitor |
| 3 | start\_9\_stop | After start bit the value after 9 events must be 1 (It is stop bit)  NOTE: for 8 data bits | Protocol checker in monitor |
| 3 | start\_6\_stop | After start bit the value after 6 events must be 1 (It is stop bit)  NOTE: for 8 data bits | Protocol checker in monitor |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_start\_bit | It covers starting bit value (0) | Protocol coverage in monitor |
| 2 | cov\_stop\_bit | It covers stopping bit value (1) | Protocol coverage in monitor |
| 3 | cross\_start\_stop\_bit | It covers whether the transaction  Is regular (VALID start bit, data and parity VALID stop bit) | Protocol coverage in monitor |

### WISHBONE Interface

#### Checker list

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | cyco\_stbo\_same\_time | It checks whether WB\_CYC and WB\_STB are asserted/ deasserted at the same time, on the same rising edge.  NOTE: It works for SINGLE READ and WRITE | Protocol checker in interface |
| 2 | stable\_info\_signals\_stb | It checks whether info signals are stable during WB\_STB is active  NOTE: Info signals:  WB\_ADDR,  WB\_DAT,  WB\_SEL,  WB\_WE,  WB\_TGD,  WB\_TGA | Protocol checker in interface |
| 3 | stable\_info\_signals\_cyc | It checks whether info signals are stable during WB\_CYC is active  NOTE: Info signals:  WB\_TGC | Protocol checker in interface |
| 4 | check\_stbo\_if\_cyco | It checks whether WB\_STB is active after WB\_CYC is active.  The order is very important. | Protocol checker in interface |
| 5 | check\_cyco\_if\_stbo | WB\_CYC can be deaserted if  WB\_STB is deaserted | Protocol checker in interface |
| 6 | stable\_stbo\_cyco\_tl\_ack | Check whether WB\_CYC and WB\_STB are asserted until WB\_ACK is inactive | Protocol checker in interface |
| 7 | chk\_ack\_active\_stb\_cyc | It checks whether is WB\_ACK is asserted. It is possible when “WB\_CYC and WB\_STB = 1” | Protocol checker in interface |
| 8 | chk\_ack\_inactive\_stb\_cyc | It checks whether is WB\_ACK is inactive. It is possible when “WB\_CYC, WB\_STB are changed from 1 to 0 | Protocol checker in interface |
| 9 | chk\_ack\_data | It checks if WB\_ACK is active then WB\_DATA\_O must be valid | Protocol checker in interface |
| 10 | ack\_active\_other\_inactive | If WB\_ACK is active other must be inactive.  NOTE: Others are WB\_ERR and WB\_RTY | Protocol checker in interface |
| 10 | err\_active\_other\_inactive | If WB\_ERR is active other must be inactive.  NOTE: Others are WB\_ACK and WB\_RTY | Protocol checker in interface |
| 11 | rty\_active\_other\_inactive | If WB\_RTY is active other must be inactive.  NOTE: Others are WB\_ACK and WB\_ERR | Protocol checker in interface |
| 12 | PIP\_stall\_high | It checks if WB\_STALL is high in pipline mode, request signal is low.  NOTE: Request signals:  WB\_STB  WB\_CYC | Protocol checker in interface |
| 13 | PIP\_ack\_high\_others\_low | When WB\_ACK is high OTHERS have to be deaserted.  NOTE: OTHERS are WB\_ADDR, WB\_SEL, WB\_TGA, WB\_TGC  and WB\_DATA\_O and WB\_TGD\_O in WRITING | Protocol checker in interface |

#### Coverage list

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_WB\_STB | It covers all states of WB\_STB | Functional coverage in monitor |
| 2 | cov\_WB\_CYC | It coverls all states of WB\_CYC | Functional coverage in monitor |
| 3 | cross\_cov\_STB\_CYC | It mixes them:  cov\_WB\_STB,  cov\_WB\_CYC | Functional coverage in monitor |
| 4 | cov\_DATA\_O | (8 bit mode) [32 bit mode]  HIGHEST targets are values (7:6) [31:23]  HIGH targets are values (5:4) [22:14]  LOW targets are values (3:2) [13:8]  LOWEST targets are values (1:0) [7:0] | Functional coverage in monitor |
| 5 | cov\_WE | It covers all states of WE | Functional coverage in monitor |
| 6 | cross\_WE\_DATA\_O | It mixes cov\_WE[1] and cov\_DATA\_O | Functional coverage in monitor |
| 7 | cov\_DATA\_I | (8 bit mode) [32 bit mode]  HIGHEST targets are values (7:6) [31:23]  HIGH targets are values (5:4) [22:14]  LOW targets are values (3:2) [13:8]  LOWEST targets are values (1:0) [7:0] | Functional coverage in monitor |
| 8 | cross\_WE\_DATA\_I | It mixes cov\_WE[0] and cov\_DATA\_I | Functional coverage in monitor |

## FUNCTIONAL FEATURES

### Master operation

#### Checker list

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_THR\_buffer | Data loaded to transmit holding register will be transmitted over RS232 | Implemented in scoreboard |
| 2 | chk\_RHR\_buffer | Data loaded to receiveholding register will be transmitted over Wishbone protocol. | Implemented in scoreboard |
| 3 | chk\_DLR\_vvalue | DLR’s (Divisor Latch Register) value must be between 2 and 65520 for optimal work | Implemented in scoreboard |
| 4 | chk\_LCR\_vparity | If parity (3rd bit OF LCR) is disabled stick parity bit (5th bit of LCR) should be disabled | Implemented in scoreboard |
| 5 | chk\_LCR\_vbreak | It is not recommended to enable break control bit (6th bit of LCR) during the transaction. | Implemented in scoreboard |
| 6 | chk\_LCR\_DLR\_io\_op | The register is set to the default value of 0 on reset, which disables all serial I/O operations in order to ensure explicit setup of the register in the software.  Therfore, before transactions DLR register must be configured by LCR. | Implemented in scoreboard |
| 7 | chk\_MCR\_vinput | Shouldn’t trying to activate DSR and CTS at the same time. | Implemented in scoreboard |

#### Coverage list

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_en\_every\_int | Cover every control interrupt field by enabling each of available interrupt. | Implemented in monitor |
| 2 | cov\_every\_int | Cover every interrupt going to the certain triggering condition.  Go out from the interrupts respecting certain conditions. | Implemented in monitor |
| 3 | cov\_DLR\_on | It will be covered by entering value different from reset value | Implemented in monitor |
| 4 | cov\_parity\_ |  | Implemented in monitor |

## REGISTER ACCESS

### Receiver Buffer Register (RBR)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_RBR\_read\_only | It is read only register. Therefore WB\_WRITE can’t be high | Implemented in scoreboard |
| 2 | chk\_RBR\_valid\_address | The address of the register is 0. For accessing this register the WB\_ADDR must be 0 | Implemented in scoreboard |
| 3 | chk\_RBR\_reset\_value | Data value should be 0x00 when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_RBR \_read | It covers the value (0) of the WB\_WRITE | Implemented in monitor |
| 2 | cov\_RBR \_address | It covers the value (0) of the WB\_ADDR | Implemented in  monitor |
| 3 | cov\_RBR\_value | All values different from 0x00 will cover it. | Implemented in  monitor |

### Transmitter Holding Register (THR)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_THR \_write\_only | It is write only register. Therefore iWB\_WRITE can’t be low | Implemented in scoreboard |
| 2 | chk\_THR \_valid\_address | The address of the register is 0. For accessing this register the WB\_ADDR must be 0 | Implemented in scoreboard |
| 3 | chk\_THR\_reset\_value | Data value should be 0xFF when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_THR \_write | It covers the value (1) of the WB\_WRITE | Implemented in  monitor |
| 2 | cov\_THR \_address | It covers the value (0) of the WB\_ADDR | Implemented in  monitor |
| 3 | cov\_THR\_value | All values different from 0xFF will cover it. | Implemented in  monitor |

### Interrupt Enable Register (IER)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_IE\_reset\_value | Reset value of Interrupt Enable register should be 0x00. | Implemented in monitor |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_IE\_value | The address range can be 0x01 up to 0x0F. These values will cover all scnearios. | Implemented in monitor |

### Interrupt Identification Register (IIR)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_II\_read\_only | It is read only register. | Implemented in scoreboard |
| 2 | chk\_II\_reset\_value | Value should be 0x00 when the reset is active | Implemented in scoreboard |
| 3 | chk\_II\_valid\_value | The valid value should be in range 11000000 to 11001101 | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_II\_value | All values that are different from reset value (0x00) | Implemented in monitor |

### FIFO Control Register (FCR)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_FCR \_write\_only | It is write only register. Therefore WB\_WRITE can’t be low | Implemented in scoreboard |
| 2 | chk\_FCR\_reset\_value | Value should be 11000000b when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_FCR \_write | It covers the value (1) of the WB\_WRITE | Implemented in monitor |
| 2 | cov\_FCR\_value | All values different from the reset values will cover it. | Implemented in  monitor |

### Line Control Register

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_LCR\_reset\_value | Reset value of Interrupt Enable register should be 00000011b | Implemented in  scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_LCR\_value | Address that that is different from 00000011b can covers it. | Implemented in monitor |

### Modem Control Register

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_MCR \_write\_only | It is write only register. Therefore iWB\_WRITE can’t be low | Implemented in scoreboard |
| 2 | chk\_MCR \_valid\_address | The value should be in range 00000000b to xxx11111b | Implemented in scoreboard |
| 3 | chk\_MCR\_reset\_value | Data value should be 0x00 when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_ MCR \_write | It covers the value (1) of the WB\_WRITE | Implemented in monitor |
| 2 | cov\_MCR\_value | All values different from 0x00 will cover it. | Implemented in  monitor |

### Line Status Register

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_LSR\_read\_only | It is read only register. | Implemented in scoreboard |
| 2 | chk\_LSR\_reset\_value | Value should be 01100000 when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_LSR \_read | It covers the value (0) of the WB\_WRITE | Implemented in monitor |
| 2 | cov\_RBR\_value | All values different from 01100000 will cover it. | Implemented in  monitor |

### Divisor latch register (byte 1)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_DLC1\_vvalue | Valid value should be different from 0 (reset value) and not above 1111 0000 | Implemented in Scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_DLC1\_vvalue | Values different from 0 will cover it | Implemented in monitor |

### Divisor latch register (byte 2)

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_DLC2\_vvalue | Valid value should be different from 0 (reset value) and not above 1111 1111 | Implemented in Scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_DLC2\_vvalue | Values different from 0 will cover it | Implemented in monitor |

### Debug register 1

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
|  |  |  |  |

### Debug register 2

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
|  |  |  |  |

### Modem Status Register

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_MSR\_read\_only | It is read only register. | Implemented in scoreboard |
| 2 | chk\_MSR\_reset\_value | Value should be xxxx0000 when the reset is active | Implemented in scoreboard |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_MSR \_read | It covers the value (0) of the WB\_WRITE | Implemented in monitor |
| 2 | cov\_MSR\_value | All values different from xxxx0000 will cover it. | Implemented in  monitor |

## RESET AND CLOCK

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Checker name | Description | Implementation Details |
| 1 | chk\_reset\_data\_stable | It checks whether all relative signals (WB\_STB , WB\_CYC )are stable ( There is no change ) during WB\_RST\_I is active ( WB\_RST\_I = 1 ) | Protocol checker in interface |
| 2 | reset\_at\_least | It checks whether WB\_RST\_I us active at least 1 clock cycle | Protocol checker in interface |
| 3 | r\_edge\_signal\_clk | All signals have to be sensitive on a rising edge | Protocol checker in interface |
| 4 | pow\_condition\_reset | It checks whether WB\_RST\_I is activated at the beginning of the simulation (During a power-up condition) | Protocol checker in interface |
| 5 | check\_clk\_isnt\_working | It checks that the interface isn’t working when | Protocol checker in interface |

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Coverage name | Description | Implementation Details |
| 1 | cov\_clk | It covers all values of clock | Coverage in monitor |
| 2 | cov\_rst | It covers all values of reset | Coverage in monitor |

## DEBUG AND TEST

*This chapter has to contain list of checkers and coverage items related to debug and test features.*

## TEST LIST

6.1 Wishbone control register write/read

6.2 Wishbone fifo register write/read

6.3 RS232 fifo register write/read

### Wishbone control register write/read

1. Read control registers value before the first cycle (No writing before)

2. Writing into control registers before configuring baud rate by DLR.

3. Writing into DLR using value that exceeds regular baud data range (min 2, max 65 520)

### Wishbone fifo register write/read

There are a few scenarios:

1. Read from fifo register value before the first cycle (No writing before)

2. Regular fifo reading (a few reading cycles in a row)

3. Regular fifo writing (a few reading cycles in a row)

4. After a few regular fifo writing, clear the fifo (by FCR) and then try to read.

5. Try to write into full fifo.

6. Try to write during the interrupt (Receiver Data available )

### Wishbone fifo register write/read

There are a few scenarios:

1. Read from fifo register value before the first cycle (No writing before)

2. Regular fifo reading (a few reading cycles in a row)

3. Regular fifo writing (a few reading cycles in a row)

4. After a few regular fifo writing, clear the fifo (by FCR) and then try to read.

5. Try to write into full fifo.

6. Try to write during the interrupt (Receiver Data available )