UART BFM – Quick Reference

uart_transmit (data_value, msg, clk, tx, [config, [scope, [msg_id_panel]]])

Example: uart_transmit(x"AA", "Sending data to Peripheral 1", clk, tx);

***Suggested usage**: uart_transmit(C_ASCII_A, "Transmitting ASCII A to DUT"); -- Suggested usage requires local overload (see section 5)*

uart_receive (data_value, msg, clk, rx, terminate_loop, [config, [scope, [msg_id_panel, [proc_name]]]])

Example: uart_receive(v_data_out, "Receive from Peripheral 1", clk, rx, terminate_signal);

***Suggested usage**: uart_receive(v_data_out, "Receive from Peripheral 1"); -- Suggested usage requires local overload (see section 5)*

uart_expect (data_exp, max_receptions, timeout, alert_level, msg, clk, rx, terminate_loop, [config, [msg_id_panel, [scope]]])

Example: uart_expect(x"3B",1, 0 ns, ERROR, "Expecting data on UART RX", clk, rx, terminate_signal);

***Suggested usage**: uart_expect(C_CR_BYTE, C_TIMEOUT, C_MAX_RECEPTIONS, ERROR, "Expecting carriage return"); -- Suggested usage requires local overload (see section 5)*

BFM

uart_bfm_pkg.vhd

BFM Configuration record ´**t_uart_bfm_config**´

Name	Type	C_UART_BFM_CONFIG_DEFAULT
clock_period	time	10 ns
clocks_per_bit	natural	16
num_data_bits	natural	8
start_bit	std_logic	'0'
stop_bit	std_logic	'1'
num_stop_bits	t_stop_bits	STOP_BITS_ONE
parity	t_parity	PARITY_ODD
max_wait_cycles	natural	50
max_wait_cycles_severity	t_alert_level	FAILURE
received_data_to_log_before_expected_data	natural	10
id_for_bfm	t_msg_id	ID_BFM
id_for_bfm_wait	t_msg_id	ID_BFM_WAIT
id_for_bfm_poll	t_msg_id	ID_BFM_POLL

UVVM™

VHDL 2008 only

BFM non-signal parameters

Name	Type	Example(s)	Description
data_value	std_logic_vector	x"D3"	The data value to be transmitted to the DUT
data_exp	std_logic_vector	x"0D"	The data value to expect when receiving the addressed register. A mismatch results in an alert 'alert_level'
max_receptions	natural	1	The maximum number of bytes received before the expected data must be received. Exceeding this limit results in an alert with severity 'alert_level'.
timeout	time	100 ns	The maximum time to pass before the expected data must be received. Exceeding this limit results in an alert with severity 'alert_level'.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the method.
msg	string	"Receiving data"	A custom message to be appended in the log/alert.
scope	string	"UART BFM"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "UART BFM". In a verification component typically "UART_VVC".
msg_id_panel	t_msg_id_panel	shared_msg_id_panel	Optional msg_id_panel, controlling verbosity within a specified scope. Defaults to a common ID panel defined in the adaptations package.
config	t_uart_bfm_config	C_UART_BFM_CONFIG_DEFAULT	Configuration of BFM behaviour and restrictions. See section 2 for details.

BFM signal parameters

Name	Type	Description
clk	std_logic	The clock signal used to receive and transmit data in/out of the UART BFM.
terminate_loop	std_logic	External control of loop termination to e.g. stop expect procedure prematurely
tx	std_logic	The UART BFM transmission signal. Must be connected to the UART DUT 'rx' port.
rx	std_logic	The UART BFM reception signal. Must be connected to the UART DUT 'tx' port.

Note: All signals are active high.

BFM details

1 BFM procedure details and examples

Procedure	Description
uart_transmit()	**uart_transmit (data_value, msg, clk, tx, [config, [scope, [msg_id_panel]]])** The uart_transmit() procedure transmits the data in 'data_value' to the DUT, using the UART protocol. For protocol details, see the UART specification. - The start bit, stop bit, parity, number of stop bits and number of data bits per transmission is defined in the 'config' parameter. - The default value of scope is C_SCOPE ("UART BFM") - The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util. - The default value of config is C_UART_BFM_CONFIG_DEFAULT, see table on the first page. - A log message is written if ID_BFM ID is enabled for the specified message ID panel. Examples: - uart_transmit(x"AA", "Transmitting data to peripheral 1", clk, tx); - uart_transmit(x"AA", "Transmitting data to peripheral 1", clk, tx, C_UART_BFM_CONFIG_DEFAULT, C_SCOPE, shared_msg_id_panel); Suggested usage (requires local overload, see section 5): - uart_transmit(C_ASCII_A, "Transmitting ASCII A to DUT");
uart_receive()	**uart_receive (data_value, msg, clk, rx, terminate_loop, [config, [scope, [msg_id_panel, [proc_name]]]])** The uart_receive() procedure receives data from the DUT at the given address, using the UART protocol. For protocol details, see the UART specification. When called, the uart_receive procedure will wait for the start bit to be present on the rx line. The initial wait for the start bit will be terminated if one of the following occurs: 1. The start bit is present on the rx line. 2. The terminate_loop flag is set to '1'. 3. The number of clock cycles waited for the start bit exceeds 'config.max_wait_cycles' clock cycles. Once all the bits have been received according to the UART specification, the parity and stop bit are checked. If correct, the read data is placed on the output 'data_value' and the procedure returns. - The default value of scope is C_SCOPE ("UART BFM") - The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util. - The default value of config is C_UART_BFM_CONFIG_DEFAULT, see table on the first page. - The default value of proc_name is "uart_receive". This argument is intended to be used internally, when procedure is called by uart_expect(). - A log message is written if ID_BFM ID is enabled for the specified message ID panel. This will only occur if the argument proc_name is left unchanged. The procedure reports an alert if: - timeout occurs, i.e. start bit does not occur within 'config.max_wait_cycles' clock cycles (alert level: 'config.max_wait_cycles_severity') - terminate_loop is set to '1' (alert level: WARNING) - expected stop_bit does not match received stop bit(s) (alert level: ERROR) - Calculated parity 'config.parity' does not match received parity (alert level: ERROR) Example - uart_receive(v_data_out, "Receive from Peripheral 1", clk, rx, terminate_signal); - uart_receive(v_data_out, "Receive from Peripheral 1", clk, rx, terminate_signal, C_UART_BFM_CONFIG_DEFAULT, C_SCOPE, shared_msg_id_panel);

Suggested usage (requires local overload, see section 5):
- uart_receive(v_data_out, “Receive from Peripheral 1”);

uart_expect()

uart_expect (data_exp, max_receptions, timeout, alert_level, msg, clk, rx, terminate_loop, [config, [msg_id_panel, [scope]]])

The uart_expect() procedure receives data from the DUT on the BFM rx line, using the receive procedure as described in the uart_receive() procedure. After receiving data from the UART rx line, the data is compared with the expected data, ‘data_exp’. If the received data does not match the expected data, another uart_receive() procedure will be initiated. This process will repeat until one of the following occurs:

1. The received data matches the expected data.
2. A timeout occurs.
3. The process has repeated ‘max_receptions’ number of times.
4. The ‘terminate_loop’ signal is set to ‘1’.

- The default value of scope is C_SCOPE (“UART BFM”)
- The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util.
- The default value of config is C_UART_BFM_CONFIG_DEFAULT, see table on the first page.
- A log message with ID ID_BFM is issued when the uart_expect procedure starts
- If the data was received successfully, and the received data matches the expected data, a log message is written with ID ID_BFM (if this ID has been enabled).
- If the received data did not match the expected data, an alert with severity ‘alert_level’ will be reported.

This procedure reports an alert if:
- ‘max_receptions’ and ‘timeout’ are set to 0, which will result in a possible infinite loop (alert_level: ERROR)
- the expected data is not received within the time set in ‘timeout’ (alert_level: ‘alert_level’)
- the expected data is not received within the number of received packets set in ‘max_receptions’ (alert_level: ‘alert_level’)
- ‘terminate_loop’ is set to ‘1’ (alert_level: WARNING)

The procedure will also report alerts for the same conditions as the uart_receive() procedure.

Example usage:
- uart_expect(x"3B",1, 0 ns, ERROR, “Expect data on UART RX”, clk, rx, terminate_signal);

Suggested usage (requires local overload, see section 5):
- uart_expect(C_CR_BYTE, “Expecting carriage return”);
- uart_expect(C_CR_BYTE, C_TIMEOUT, C_MAX_RECEPTIONS, ERROR, “Expecting carriage return”);

2 BFM Configuration record

Type name: t_uart_bfm_config

Name	Type	C_UART_BFM_CONFIG_DEFAULT	Description
clock_period	time	10 ns	Period of the clock signal.
clocks_per_bit	natural	16	Number of clock cycles per bit
num_data_bits	natural	8	Number of data bits to send per transmission
start_bit	std_logic	'0'	Bit indicating start of the UART transmission
stop_bit	std_logic	'1'	Bit indicating stop of the UART transmission
num_stop_bits	t_stop_bits	STOP_BITS_ONE	Number of stop-bits to use per transmission {STOP_BITS_ONE, STOP_BITS_ONE_AND_HALF, STOP_BITS_TWO}
parity	t_parity	PARITY_ODD	Transmission parity bit {PARITY_NONE, PARITY_ODD, PARITY_EVEN}
max_wait_cycles	natural	50	The maximum number of clock cycles to wait for the UART start bit on the RX line before timeout
max_wait_cycles_severity	t_alert_level	failure	The above timeout will have this severity
received_data_to_log_before_expected_data	natural	10	Maximum number of bytes to save ahead of the expected data in the receive buffer. The bytes in the receive buffer will be logged.
id_for_bfm	t_msg_id	ID_BFM	The message ID used as a general message ID in the UART BFM
id_for_bfm_wait	t_msg_id	ID_BFM_WAIT	The message ID used for logging waits in the UART BFM
id_for_bfm_poll	t_msg_id	ID_BFM_POLL	The message ID used for logging polling in the UART BFM

3 Additional Documentation

For additional documentation on the UART protocol, please see the UART specification.

4 Compilation

The UART BFM may only be compiled with VHDL 2008. It is dependent on the UVVM Utility Library (UVVM-Util), which is only compatible with VHDL 2008.
See the separate UVVM-Util documentation for more info. After UVVM-Util has been compiled, the uart_bfm_pkg.vhd BFM can be compiled into any desired library.

4.1 Simulator compatibility and setup

This BFM has been compiled and tested with Modelsim version 10.3d and Riviera-PRO version 2015.10.85.

For required simulator setup see UVVM-Util Quick reference.

5 Local BFM overloads

A good approach for better readability and maintainability is to make simple, local overloads for the BFM procedures in the TB process.
This allows calling the BFM procedures with the key parameters only
e.g.

uart_transmit(C_ASCII_A, “Transmitting ASCII A”);

rather than

uart_transmit(C_ASCII_A, “Transmitting ASCII A”, clk, tx, terminate_loop,
C_CLK_PERIOD, C_UART_BFM_CONFIG_DEFAULT, C_SCOPE, shared_msg_id_panel);

By defining the local overload as e.g.:

```
procedure uart_transmit(
   constant data_value   : in std_logic_vector;
   constant msg          : in string) is
begin
   uart_transmit(data_value,                    -- keep as is
                 msg,                           -- keep as is
                 clk,                           -- Signals must be visible in local process scope
                 tx,                            -- Signals must be visible in local process scope
                 C_UART_CONFIG_LOCAL,           -- Use locally defined configuration or C_UART_CONFIG_DEFAULT
                 C_SCOPE,                       -- Just use the default
                 shared_msg_id_panel);          -- Use global, shared msg_id_panel
end;
```

Using a local overload like this also allows the following – if wanted:

- Have address value as natural – and convert in the overload
- Set up defaults for constants. May be different for two overloads of the same BFM
- Apply dedicated message ID panel to allow dedicated verbosity control

IMPORTANT
This is a simplified Bus Functional Model for UART TX and RX.
The given BFM complies with the basic UART protocol and thus allows a normal access towards a UART interface. This BFM is not a UART protocol checker.
For a more advanced BFM please contact Bitvis AS at support@bitvis.no