UART BFM – Quick Reference

NOTE: As of UVVM v3.x, all shared variables have been made protected. This means that any access to shared variables must be done using get- and set-methods. This documentation has not yet been updated with the methods for accessing these variables, but will be very soon.
CAUTION: shaded code/description is preliminary.

BFM

uart_bfm_pkg.vhd

uart_transmit (data_value, msg, tx, [config, [scope, [msg_id_panel]]]) [1]

Example: uart_transmit(x"AA", "Sending data to DUT UART instance 1", tx);

Suggested usage: *uart_transmit(C_ASCII_A, "Transmitting ASCII A to DUT UART instance 1"); -- Suggested usage requires local overload (see section 5)*

uart_receive (data_value, msg, rx, terminate_loop, [config, [scope, [msg_id_panel, [proc_name]]]]) [1]

Example: uart_receive(v_data_out, "Receive from DUT UART instance 1", rx, terminate_signal);

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

uart_expect (data_exp, msg, rx, terminate_loop, [max_receptions, [timeout, [alert_level, [config, [msg_id_panel, [scope]]]]]]) [1]

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

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

BFM Configuration record ´**t_uart_bfm_config**´

Record element	Type	C_UART_BFM_CONFIG_DEFAULT
bit_time	time	-1 ns
num_data_bits	natural	8
idle_state	std_logic	'1'
num_stop_bits	t_stop_bits	STOP_BITS_ONE
parity	t_parity	PARITY_ODD
timeout	time	0 ns
timeout_severity	t_alert_level	ERROR
received_data_to_log_before_expected_data	natural	10
match_strictness	t_match_strictness	MATCH_EXACT
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
id_for_bfm_poll_summary	t_msg_id	ID_BFM_POLL_SUMMARY
error_injection	t_bfm_error_injection	C_ERROR_INJECTION_INACTIVE

Note 1: the BFM configuration has to be defined and used when calling the UART BFM procedures. See section 5 for an example of how to define a local BFM config.

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'. Default value is 1.
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
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 Error injection record (inside the BFM configuration record)

Field name	Type	Default value	Description
parity_bit_error	Boolean	False	Will invert the parity bit in a transmission if TRUE, and thus generate a parity error.
stop_bit_error	Boolean	False	Will invert the first stop bit in a transmission if TRUE. Note that the following UART frame may be misinterpreted if there is no Idle period or additional stop bits after the error injection. Hence a stop_bit_error may lead to multiple following UART frame errors.

Error injection in general is explained in 'UVVM Essential Mechanisms' located in uvvm_vvc_framework/doc.
Note: The error_injection_config in the VVC config will override any error injection specified in the BFM config when using VVCs.

BFM details

1 BFM procedure details and examples

Procedure	Description

uart_transmit()	**uart_transmit (data_value, msg, 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.
	Errors may be injected – depending on the error_injection_config sub-record within the bfm_config
	
	Examples:
	`uart_transmit(x"AA", "Transmitting data to DUT UART instance 1", tx);`
	`uart_transmit(x"AA", "Transmitting data to DUT UART instance 1", 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 instance 1");`

uart_receive()	**uart_receive (data_value, msg, 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)
	
	Examples:
	`uart_receive(v_data_out, "Receive from DUT UART instance 1", clk, terminate_signal);`
	`uart_receive(v_data_out, "Receive from DUT UART instance 1", clk, 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 DUT UART instance 1");`

uart_expect()

uart_expect (data_exp, msg, rx, terminate_loop, [max_receptions, [timeout, [alert_level, [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 alert_level is ERROR
- 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 `max_receptions` is 1.
- 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:

```
uart_expect(x"3B", "Expect data on UART RX", rx, terminate_signal, 1, 0 ns);
```

Suggested usage (requires local overload, see section 5):

```
uart_expect(C_CR_BYTE, "Expecting carriage return");
uart_expect(C_CR_BYTE, "Expecting carriage return", C_TIMEOUT, C_MAX_RECEPTIONS);
```

2 BFM Configuration record

Type name: t_uart_bfm_config

Record element	Type	C_UART_BFM_CONFIG_DEFAULT	Description
bit_time	time	-1 ns	The time it takes to transfer one bit. Will raise an error if not set.
num_data_bits	natural	8	Number of data bits to send per transmission
idle_state	std_logic	'1'	Bit value when line is idle
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}
timeout	time	0 ns	The maximum time to wait for the UART start bit on the RX line before timeout
timeout_severity	t_alert_level	error	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.
match_strictness	t_match_strictness	MATCH_EXACT	Matching strictness for std_logic values in check procedures. MATCH_EXACT requires both values to be the same. Note that the expected value can contain the don't care operator '-'. MATCH_STD allows comparisons between 'H' and '1', 'L' and '0' and '-' in both values.
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
id_for_bfm_poll_summary	t_msg_id	ID_BFM_POLL_SUMMARY	The message ID used for logging polling summary in the UART BFM
error_injection	t_bfm_error_injection	C_ERROR_INJECTION_INACTIVE	See error injection record on page 2. Error injection in general is explained in 'UVVM Essential Mechanisms' located in uvvm_vvc_framework/doc.

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.
See UVVM Essential Mechanisms located in uvvm_vvc_framework/doc for information about compile scripts

4.1 Simulator compatibility and setup

See README.md for a list of supported simulators.

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", tx, C_UART_CONFIG_LOCAL, 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
                 tx,                            -- Signals must be visible in local process scope
                 C_UART_CONFIG_LOCAL,           -- Use locally defined configuration
                 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

See section 6 for defining a BFM configuration to use with the local overload and when calling the BFM procedures.

6 Local BFM configuration

The UART BFM requires that a local configuration is declared in the testbench and used in the BFM procedure calls. The default BFM configuration is defined with a clock period of -1 ns so that the BFM can detect and alert the user that the configuration has not been set. See page 1 for the UART BFM configuration record fields.

Defining a local UART BFM configuration:

```
constant C_UART_CONFIG_LOCAL : t_uart_bfm_config := (
    bit_time                                => C_UART_BIT_TIME,
    num_data_bits                           => 8,
    idle_state                              => '1',
    num_stop_bits                           => STOP_BITS_ONE,
    parity                                  => PARITY_ODD,
    timeout                                 => 0 ns,
    timeout_severity                        => error,
    num_bytes_to_log_before_expected_data   => 10,
    match_strictness                        => MATCH_EXACT,
    id_for_bfm                              => ID_BFM,
    id_for_bfm_wait                         => ID_BFM_WAIT,
    id_for_bfm_poll                         => ID_BFM_POLL,
    id_for_bfm_poll_summary                 => ID_BFM_POLL_SUMMARY,
    error_injection                         => C_BFM_ERROR_INJECTION_INACTIVE
);
```

See section 5 for how to define a local overload procedure and how to use a BFM config with the procedure call.

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