RTX2254

Bluetooth RF Tester

API Specification

Table of Contents

1 History

Date	Initials	Rev.	Description
23/01/2017	MHP	1.0	Initial version
03/04/2017	MVC	1.1	Layout updated. Some textual updates.
30/05/2017	MVC	1.2	Layout updated, new logo. Some textual updates.
23/08/2017	MVC	1.3	Document sub title changed to “API Specification”. Added short note about supported protocols.

2 References

- TBD

3 Terms and abbreviations

Term	Description
API	Application Programming Interface
BtTst	The Bluetooth tester expansion module. In this document simply referred to as *BtTst* or *the module*.
Call	The combination of a request followed by a confirm.
Confirm	The result of the request returned by Rtx2300 module.
Dll	Dynamic Link Library
Firmware	The software running in the target.
Generic types	RTX basic types
Global types	RTX product specific types
Instrument	The software that, along with the BtTst module and the DUT can be used for testing the DUT. This is similar to a physical desktop instrument.
Master	The software and or system controlling the module, typically an application running on a PC.
Request	A command sent to the module, e.g. measure voltage.
RTX2300	An integrated production and calibration system.
Target	The complete module, including the board containing the circuitry and the software running it.
Task	A self-contained major software component in the RTX standard software environment.

4 Introduction

This document describes the SW interface (API) between master PC running the module driver software and the tester. An overview of the system is shown in Figure 1.

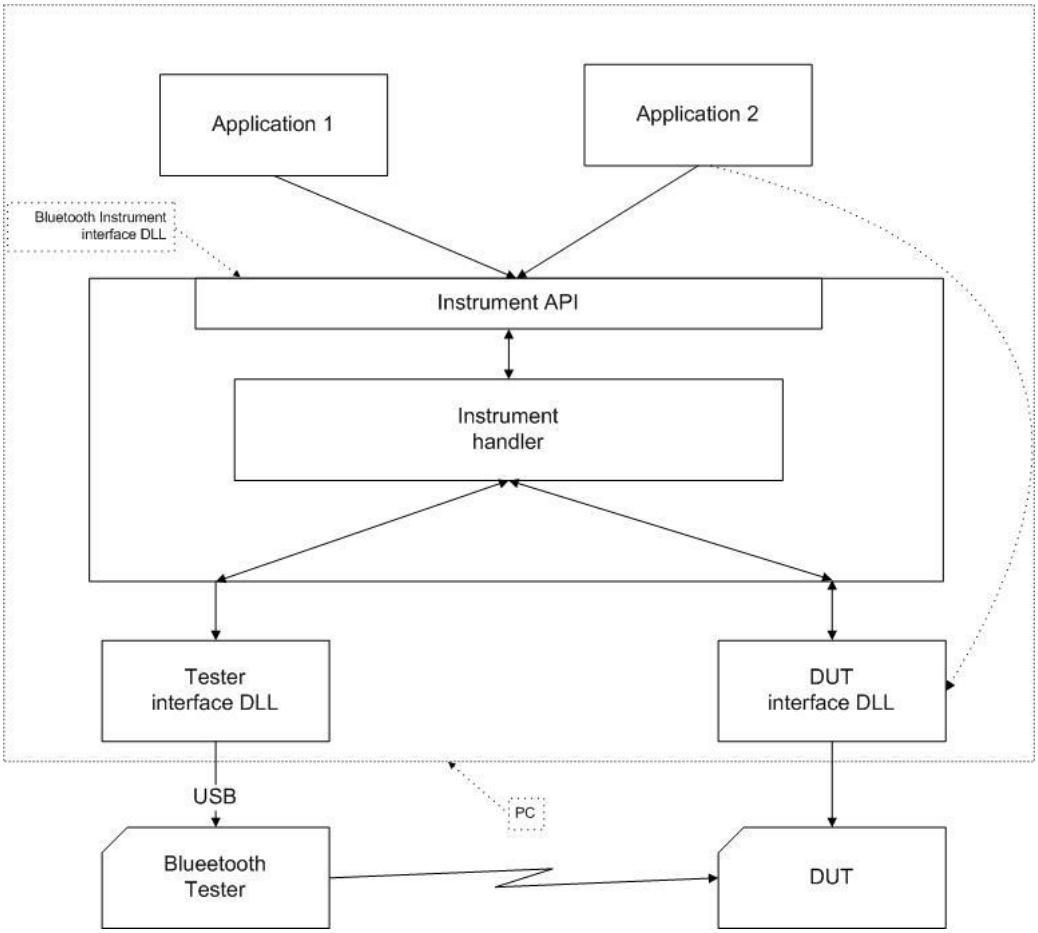

Figure 1

The Bluetooth Tester consists of the actual tester hardware and the *Tester Interface DLL*. The two communicates using three UART over USB COM-ports:

- RTX BLE Analyser
- RTX BLE Generator
- RTX BLE Tester

The interface DLL also needs a second DLL, the *DUT interface DLL*, in order to communicate with the DUT. This DLL is a simple wrapper that connects the fixed interface of the tester interface DLL and whatever interface the DUT may support:

UART interface:

- RTX BLE DUT 0
- RTX BLE DUT 1

Or

USB interface:

- The USB device driver name

It is provided as source code and must be modified by the customer to suit the DUT.

The Tester interface DLL exposes the Instrument API to applications using it, as well as the Tester API and the DUT API. For normal use only the instrument API is needed.

The DLL and LIB files are released in Visual Studio 2010 format to support as many systems as possible. However, they can be linked with projects using Visual Studio 2013 or 2015 without problems.

Bluetooth HCI and 2-wire protocols are supported.

5 Generic Types

The interfaces use RTX standard platform independent types. These types must be defined in accordance with the platform used.

Type name	Typical definition	Description
rsuint8	typedef unsigned char rsuint8;	unsigned 8 bit
rsint8	typedef signed char rsint8;	signed 8 bit
rsuint16	typedef unsigned short rsuint16;	unsigned 16 bit
rsint16	typedef signed short rsint16;	signed 16 bit
rsuint32	typedef unsigned long rsuint32;	unsigned 32 bit
rsint32	typedef signed long rsint32;	signed 32 bit
Rsbitfield	typedef unsigned char rsbitfield;	bitfield designator

6 Interfaces, Mails, Calls and Types

Communication with a device is done using an *interface*, which is a collection of mails, functions and types.

Interfaces are documented using the following format:

Interface:	**The name of the interface**
Description:	A description of the interface.

All mails, functions and types following an interface specification belong to that interface, until the end of the document or a new interface is specified.

Most communication in an interface is mail based. Please read the *Rtx2300 Interface Specification* for a detailed description of mails and primitives. A set of mails is known as a ***mail set***. A typical mail set consists of a request and a confirm, although other mail types may be found as well. All mail sets also supply functions for sending and receiving the mails (which makes the mail interfaces function based as well). This document describes the mails of any mail set only – to find the corresponding functions look up the function having the same name as the mail primitive. Example:

> To make a power measurement the BTTST_GET_POWER_REQ mail must be send. This can be done using the function SendBtTstGetPowerReq, which takes the parameters described in the mail interface for BTTST_GET_POWER_REQ. The reply will typically be received by a mail handler and delivered to the application as a BtTstGetPowerCfm structure, containing the fields described in the mail interface for the BTTST_GET_POWER_CFM. These mail sending functions are easily recognized by the word *Send* prefixed to the function name. Alternatively can the function
>
> BtTstGetPowerCfm BtTstGetPower(x,y,z)
>
> be used instead. This variant (without the *Send* prefix) is blocking, i.e. it will send the request and wait until the confirm has been received. It relieves the application from having to implement a mail handler, but precludes concurrent execution of commands.

The mail sets are documented using the following format:

MailSet:	The name of the call, e.g. SetPower	
Description:	A detailed description of what the call does.	
Request:	The name of the request part of the call. Optional.	
Description:	Overall description of the request. Optional.	
Primitive:	The primitive used by the request. The value of the primitive may be specified here as well. Optional.	
Parameters:		
Type	**Name**	**Description**
		Here the types and names of all fields in the request are specified.

Confirm:	The name of the confirm part of the call. Optional.		
Description:	Overall description of the confirm. Optional.		
Primitive:	The primitive used by the confirm. The value of the primitive may be specified here as well. Optional.		
Parameters:			
Type		**Name**	**Description**
			Here the types and names of all fields in the confirm are specified.

Note that this format also documents the functions available for sending the mails. In these the fields of the request corresponds to the arguments in the call, while the fields in the confirm corresponds to the return value of the function. If the confirm only lists a single field that field is returned by the function. If multiple fields are listed the function will return a structure containing all the fields.

Function interfaces do not use mails for communication. Typically, these interact with the DLL and do not directly communicate with the target. A function is documented like this:

Call:	**The name of the function**	
Description:	A description of what the function does	
Return value type:	The type of the return value. This may be a simple type or the name of a composite type, which is documented in the *Types* section in this document. If a composite type is only used as a call return value it may be documented immediately after the call documentation. Some types are global Rtx2300 types which are documented elsewhere.	
Return value description:	A description of the return value	
Parameters:		
Type	**Name**	**Description**
		Here the types and names of all parameters in the function is described. Types used for parameters are documented elsewhere in this document, unless it is a simple type or a type documented elsewhere in the Rtx2300 documentation.

Type definitions are documented using one of two similar formats:

TypeName :	**The name of the defined type**
Group:	The kind of type. Typical groups are enumerations, structures, unions, constants, etc.
Description:	A description of the type
Type:	The underlying type, e.g. rsuint8, int32, rsbool etc.
Value:	The value of the type (constants only)

TypeName :	The name of the defined type
Group:	The kind of type. Typical groups are enumerations, structures, unions, constants, etc.
Description:	A description of the underlying type

Code	Description
Code that defines the members of the type	A description of each member.

The following sections document the various interfaces used in this system and their mail sets, functions, and types.

7 Instrument interface

Interface:	InstrumentIntf
Description:	This interface allows applications to use and configure the Bluetooth tester system and the DUT as an instrument. All functions and types defined in this interface are prefixed with *BtTst.* Note that the interface also uses types from other interfaces which uses other prefixes.

7.1 Function interface

7.1.1 Power measurements

Call:	BtTstGetPower	
Description:	Make a power measurement and return the result.	
Return value type:	BtTstGetPowerResultType	
Return value description:		
Parameters:		
Type	**Name**	**Description**
BtTstRfMeasureModeType	RfMode	The RF mode to use (CW or burst). Note! CW is not available with all DUT devices.
BtTstOutputRFConfigurationType	DutRfConnector	The selected RF interface DUT 0 or DUT 1
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstPowerLevelType	DutPowerLvl	The power level. Unit dBm. Range depends on DUT device.
BtTstDataLengthType	Length	The payload length in bytes (0 - 37)
BtTstPayloadTypeType	PacketType	The type of the payload (0-7)

TypeName:	BtTstGetPowerResultType
Group:	Struct
Description:	The measured power and error info
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found
double MeasuredPower;	The measured power in dBm.

7.1.2 Sensitivity measurements

Call:	BtTstGetPacketErrorRate	
Description:	Make a sensitivity measurement and return the result	
Return value type:	BtTstGetPacketErrorRateResultType	
Return value description:		
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstPowerLevelType	PowerLevel	The power level. Unit dBm (-40 dBm to -100 dBm)
BtTstPacketCountType	Packets	The number of packets to use in packet error rate (PER) test

BtTstDataLengthType	Length	The payload length in bytes (0 - 37)
BtTstPayloadTypeType	PacketType	The type of the payload (0-7)

TypeName:	**BtTstGetPacketErrorRateResultType**
Group:	Struct
Description:	The measured packet error rate and error info
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found
double PacketErrorRate;	The measured error rate in percent. Result is between 0% and 100%. The result is the rate of packet failures, so 0 means no packets was lost.
BtTstPacketCountType TxCount;	The number of packets actually sent. Because of internal execution and communication time spent the actual number of packets may be slightly larger than what was requested.
BtTstPacketCountType ErrorCount;	The number of missing or erroneous packets. An integer between 0 and the *TxCount* value.

7.1.3 Frequency offset correction

Call:	**BtTstMeasureOffset**	
Description:	Measure the DUT RF frequency offset.	
Return value type:	BtTstMeasureOffsetResultType	
Return value description:		
Parameters:		
Type	**Name**	**Description**
rsbool	PowerMeasure	Whatever to do a power measurement during frequency offset measurement. Set TRUE to include.
BtTstRfMeasureModeType	RfMode	The RF mode to use (CW or burst)
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstOutputRFConfigurationType	DutRfConnector	The selected RF interface DUT 0 or DUT 1

TypeName:	**BtTstMeasureOffsetResultType**
Group:	Struct
Description:	The measured packet error rate and error info
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: a supplied parameter is out of range. This may happen if a faulty parameter is specified, or if the DUT is not able to handle the specified compensation value. Note that not all DUT's are able to provide this information.
BtTstFrequencyType OffsetHz;	The measured frequency offset in Hz.Valid range is -500.000 to +500.000 Hz.
BtTstRfOffsetType OffsetPpm;	The measured frequency offset in ppm. Valid range is -100.00 to +100.00. Resolution is 0.01
BtTstRSSIType RSSIValue;	The measured RSSI value in dBm if enabled in PowerMeasure.

Call:	BtTstSetOffsetCompensation	
Description:	Change the XTAL frequency to compensate for frequency offset in the RF output. This call may be as part of an adjustment loop, in which the value is not written to NVS. When the compensation is satisfactory the call can be used to write the value to NVS and optionally reset the DUT.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: a supplied parameter is out of range. This may happen if a faulty parameter is specified, or if the DUT is not able to handle the specified compensation value. Note that not all DUT's are able to provide this information.	
Parameters:		
Type	**Name**	**Description**
BtTstFrequencyPPMType	CompensationValue	The amount to move the XTAL frequency. The unit is ppm and the valid range is -1000000 to +1000000. Resolution is 0.1 ppm. Note that the oscillator in the DUT is most likely not able to handle the entire range.
BtTstNativeCrystalTuneType	NativeTuneValue	The DUT native crystal tune value during frequency offset measurement.
rsbool	WriteToNvs	False: the compensation value is applied to the XTAL frequency only. Use this during the adjustment. True: write the compensation value to NVS to make it permanent.
rsbool	Reset	False: no reset is applied True: the DUT is reset after the value is written. This parameter has no effect if the WriteToNvs parameter is false.

Call:	BtTstCalculatePpm	
Description:	Calculate the difference in ppm between the two specified frequencies.	
Return value type:	BtTstRfOffsetType	
Return value description:		
Parameters:		
Type	**Name**	**Description**
Rtx2300FrequencyType	Freq1	The expected frequency in Hz
Rtx2300FrequencyType	Freq2	The actual frequency in Hz

7.1.4 Configuration

These functions are used to configure the DUT and the DUT interface DLL.

Call:	**BtTstLoadDutInterfaceDll**	
Description:	Load the DLL containing the DUT interface. It is important to load a DUT interface before accessing the DUT or using the tester. If the string specifies a full path, the function searches only that path for the module. If the string specifies a relative path or a module name without a path, the function uses the standard Windows search strategy to find the module. If the function cannot find the module, the function fails. When specifying a path, be sure to use backslashes (\), not forward slashes (/).	
Return value type:	BtTstLoadDutInterfaceDllResultType	
Return value description:		
Parameters:		
Type	**Name**	**Description**
const char*	Filename	The path to the interface DLL

TypeName:	**BtTstLoadDutInterfaceDllResultType**
Group:	Struct
Description:	The result of unloading the DUT interface DLL
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: DLL not found or could not be loaded.
rsuint32 Handle;	The windows handle to the loaded DLL

Call:	**BtTstUnloadDutInterfaceDll**	
Description:	Unload the DLL containing the DUT interface.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: DLL could not be unloaded.	
Parameters:		
Type	**Name**	**Description**

Call:	**BtTstConfigureDut**	
Description:	Make a configuration change in the DUT interface. The values in the *Cfg* parameter are passed unchanged to the DUT interface DLL, and may be used to configure the interface.	
Return value type:	BtTstDutConfigurationType	
Return value description:	Data returned from the DUT interface DLL.	
Parameters:		
Type	**Name**	**Description**
BtTstDutConfigurationType	Cfg	The configuration data

Call:	BtTstSetDutComPort	
Description:	Specify the number of the COM port to use in the DUT DLL. Note that the DLL may not support or use a COM port at all. Configuration of all other types of communication must be done using the *BtTstConfigureDut* function.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_UNSUPPORTED: the DLL does not support a COM port RTX2300_ERR_BUSY: unable to open the COM port (if trying to open a port) or to close it (if trying to close a port) RTX2300_ERR_NO_ACCESS: the specified COM port does not exist	
Parameters:		
Type	**Name**	**Description**
rsuint16	ComPortNumber	The number of the ComPort to open. Specify 0 to close an already open COM port.
rsuint32	ComBaudRate	The baud rate to use with DUT.
rsbool	EnableHwFlowCtrl	Set to enable DUT HW flow control
BtTstDutProtocolSelectType	BtTstDutProtocol	The communication protocol

Call:	BtTstSetDutCommunication	
Description:	Open or close the communication protocol between the DUT interface DLL and the DUT. Opening the protocol will try to establish communication with the DUT and initialize it.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsbool	Open	True: open the communication and establish a connection with the DUT False: close the connection. Note that this will not close the COM port.

Call:	BtTstSetTesterCommunication	
Description:	Specify the EAI port server name and number of the COM port to use for communication with the tester main module.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_UNSUPPORTED: the DLL does not support a COM port RTX2300_ERR_BUSY: unable to open the COM port (if trying to open a port) or to close it (if trying to close a port) RTX2300_ERR_NO_ACCESS: the specified COM port does not exist	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType*	InstNo	Pointer to destinaton that will receive the instance number of this instance. This instance number must be specified in all following calls to API functions operating on this instance. If the returned instance number is RTX2300INTF_ERROR_NONE the port server instance could not be found or connected to.
const char*	PortServerName	The name of the port server to use.
rsuint16	ComPortNumber	The number of the ComPort to open. Specify 0xFF to log on to an existing port server without changing the COM port.

Call:	BtTstSetGeneratorComPort	
Description:	Opens the COM port specified for generator module.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	ComPortNumber	The number of the COM port to open. Specify 0 to close an already open COM.

Call:	BtTstSetAnalyzerComPort	
Description:	Opens the COM port specified for RF measurements.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	ComPortNumber	The number of the COM port to open. Specify 0 to close an already open COM.

Call:	BtTstSetIoExt	
Description:	DUT test interface to use for test.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	**Name**	**Description**
BtTstOutputRFConfigurationType	RfOutputConfiguration	Setup of front RF port
BtTstOutputConfigurationType	OutputConfiguration	Setup of front communication port (UART/USB)

8 DUT interface

Interface:	DutIntf
Description:	This interface allows applications to use and configure the DUT using the *DUT interface DLL*. It is intended for debugging only. All messages and types in this interface are prefixed with *Dut.* Note: this interface cannot be used unless a DUT interface has been loaded, see 0

8.1 Function interface

8.1.1 Power measurements

Call:	DutStartTx	
Description:	Start a packet transmission from the DUT to the tester. Starts BLE Bluetooth Low Energy transmitter test mode (equivalent to the HCI_LE_Transmitter_Test command). Starts packet transmission on selected channel.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found	
Parameters:		
Type	Name	Description
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstDataLengthType	Length	The payload length in bytes (a number between 1 and 37)
BtTstPayloadTypeType	Type	The type of the data payload. Available types depend on used protocol HCI or 2-Wire, refer to Bluetooth specification: HCI: (0-7) 0 = PRBS9 1 = 11110000 2 = 01010101 3 = PRBS15 4 = 11111111 5 = 00000000 6 = 00001111 7 = 01010101 2-Wire: (0-3) 0 = PRBS9 1 = 11110000 2 = 0101010 3 = Vendor specific

Call:	DutStopTx	
Description:	Stop an ongoing packet transmission from the DUT (equivalent to the HCI_LE_Test_End command).	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	Name	Description

Call:	DutStartContinuousTx	
Description:	Start a continuous carrier from the DUT to the tester. Note that the frequency is selected as a channel number. Note! May not be supported on all devices. This call is for debugging only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.	
Parameters:		
Type	Name	Description
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	DutStopContinuousTx	
Description:	Stop an ongoing continuous transmission from the DUT. Note! On some devices the transmission must be terminated by a device reset. This call is for debugging only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	Name	Description

8.1.2 Sensitivity measurements

Call:	DutStartRx	
Description:	Start a packet reception in the DUT. Starts BLE Bluetooth Low Energy receive test mode (equivalent to the HCI_LE_Receiver_Test command). Starts packet reception on a selected. Note: reception must be stopped by sending DutStopRx. If not done so within appr. 42 seconds after starting, the packet count may be invalid because the internal 16 bit packet counter in the DUT will overflow!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found	
Parameters:		
Type	Name	Description
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	DutStopRx	
Description:	Stop an ongoing packet reception in the DUT and return the number of packets received (equivalent to the HCI_LE_Test_End command).	
Return value type:	DutStopRxResultType	
Return value description:		
Parameters:		
Type	Name	Description

TypeName:	DutStopRxResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no reception was ongoing. The packet count is invalid.
BtTstPacketCountType Count;	The number of ok packets received.

Call:	DutStartContinuousRx	
Description:	Start receive of a continuous carrier from the Tester. Note that the frequency is selected as a channel number. Note! May not be supported on all devices. This call is for debugging only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.	
Parameters:		
Type	Name	Description
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	DutStopContinuousRx	
Description:	Stop an ongoing receive of a continuous transmission from the Tester. Note! On some devices the transmission must be terminated by a device reset. This call is for debugging only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	Name	Description

Call:	DutReadRSSI		
Description:	Get the RSSI value of the signal from the tester measured by DUT. This call is for debugging only!		
Return value type:	DutReadRSSIResultType		
Return value description:			
Parameters:			
Type	**Name**	**Description**	

TypeName:	DutReadRSSIResultType
Group:	Struct
Description:	

Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstRSSIType RSSIValue;	The RSSI value in dBm

8.1.1 Frequency offset correction

Call:	DutReadFreqEst		
Description:	Get the frequency offset of the tester measured by DUT. This call is for debugging only!		
Return value type:	DutReadFreqEstResultType		
Return value description:			
Parameters:			
Type	**Name**	**Description**	

TypeName:	DutReadFreqEstResultType
Group:	Struct
Description:	

Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstFrequencyType FreqEstValue;	The frequency offset in hertz (Hz)

Call:	DutSetOffsetCompensation
Description:	Change the XTAL frequency to compensate for frequency offset in the RF output. This call may be as part of an adjustment loop, in which the value is not written to NVS. When the compensation is satisfactory the call can be used to write the value to NVS and optionally reset the DUT. This call is for debugging only!
Return value type:	Rtx2300ErrorType
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: a supplied parameter is out of range. This may happen if a faulty parameter is specified, or if the DUT is not able to handle the specified compensation value. Note that not all DUT's are able to provide this information.

Parameters:		
Type	**Name**	**Description**
BtTstFrequencyPPMType	CompensationValue	The amount to move the XTAL frequency. The unit is ppm and the valid range is -1000000 to +1000000. Resolution is 0.1 ppm. Note that the oscillator in the DUT is most likely not able to handle the entire range.
BtTstNativeCrystalTuneType	NativeTuneValue	The DUT native crystal tune value during frequency offset measurement.
rsbool	WriteToNvs	False: the compensation value is applied to the XTAL frequency only. Use this during the adjustment. True: write the compensation value to NVS to make it permanent.
rsbool	Reset	False: no reset is applied True: the DUT is reset after the value is written. This parameter has no effect if the WriteToNvs parameter is false.

Call:	DutGetOffsetCompensation
Description:	Get the current XTAL frequency compensation This call is for debugging only!
Return value type:	DutGetOffsetCompensationResultType
Return value description:	

Parameters:		
Type	**Name**	**Description**
BtTstGetOffsetType	GetOffset	Where to get the offset from
BtTstFrequencyType	FreqOffset	The frequency offset to get compensation for

TypeName:	DutGetOffsetCompensationResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstRfOffsetType CompensationValue;	The current compensation value in ppm
BtTstRfOffsetType CompensationValueNative;	The current compensation value in whatever unit the DUT reports it. This value is for debugging only, and is not guaranteed to always be valid.

8.1.2 Configuration

Call:	DutConfigure
Description:	Make a configuration change in the DUT interface. The values in the *Cfg* parameter are passed unchanged to the DUT interface DLL, and may be used to configure the interface.
Return value type:	BtTstDutConfigurationType
Return value description:	Data returned from the DUT interface DLL.

Parameters:		
Type	Name	Description
BtTstDutConfigurationType	Cfg	The configuration data

Call:	DutSetComPort
Description:	Specify the number of the COM port to use in the DUT DLL. Note that the DLL may not support or use a COM port at all. Configuration of all other types of communication must be done using the *DutConfigure* function.
Return value type:	Rtx2300ErrorType
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_UNSUPPORTED: the DLL does not support a COM port RTX2300_ERR_BUSY: unable to open the COM port (if trying to open a port) or to close it (if trying to close a port) RTX2300_ERR_NO_ACCESS: the specified COM port does not exist

Parameters:		
Type	Name	Description
rsuint16	ComPortNumber	The number of the ComPort to open. Specify 0 to close an already open COM port.
rsuint32	ComBaudRate	The baud rate to use with DUT.
rsbool	EnableHwFlowCtrl	Set to enable DUT HW flow control
BtTstDutProtocolSelectType	BtTstDutProtocol	The communication protocol

Call:	DutSetCommunication	
Description:	Open or close the communication protocol between the DUT interface DLL and the DUT. Opening the protocol will try to establish communication with the DUT and initialize it.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsbool	Open	True: open the communication and establish a connection with the DUT False: close the connection. Note that this will not close the COM port.

Call:	DutSetTxPower	
Description:	Set Tx power of the DUT. Note! might not be supported by all devices Set Tx power of the DUT. Note! DUT Tx power is controlled by vendor specific HCI or 2-wire commands and is different from manufacture to manufacture and might not be supported by all devices. The function is by default empty, i.e. DUT will use default Tx power setting.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsint8	TxPower	The Tx power is a value in dBm. E.g. 0 for 0 dBm Note! the dBm value must be mapped to vendor specific setup value, e.g. 0 dBm value maps to register value 2 for TI CC254x DUT

Call:	DutWriteHWReg	
Description:	Write value to specified hardware register in DUT. This call is for debugging only!	
Return value type:	DutWriteHWRegResultType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	RegAddress	The physical address of the hardware register to write
rsuint8	RegValue	The value to write hardware register

TypeName:	DutWriteHWRegResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
rsuint8 HWregValue;	The value of the written hardware register, i.e. readback of just written.

Call:	DutReadBdAddress
Description:	Read the DUT BD address. Note! might not be supported by all devices
Return value type:	DutReadBdAddressResultType
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).

Parameters:		
Type	Name	Description

TypeName:	DutReadBdAddressResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstBdAddressType BdAddress;	The device address from DUT.

Call:	DutReadBdAddressCS
Description:	Read the DUT BD address. Note! might not be supported by all devices
Return value type:	void
Return value description:	

Parameters:		
Type	Name	Description
DutReadBdAddressTypeCS*	BDAddressPtr	A pointer to store read Bluetooth Device address

TypeName:	DutReadBdAddressTypeCS	
Group:	Struct	
Description:		
Code		**Description**
Rtx2300ErrorType ErrorCode;		RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).
rsuint8 B0;		The device address[0] from DUT
rsuint8 B1;		The device address[1] from DUT
rsuint8 B2;		The device address[2] from DUT
rsuint8 B3;		The device address[3] from DUT
rsuint8 B4;		The device address[4] from DUT
rsuint8 B5;		The device address[5] from DUT

Call:	DutReset	
Description:	Resets the DUT. Note! might not be supported by all devices	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**

8.2 Types

TypeName:	BtTstFileNameType
Group:	Array
Description:	This type specifies a file name
Type:	rsuint8
Size:	MAX_PATH

TypeName:	BtTstGetOffsetType
Group:	Enumeration
Description:	This type to get the offset from
Code	**Description**
RTX2300_RADIO_INTERFACE = 0	The compensation value is read directly from the radio interface, i.e. the current value from radio interface – compensation is 0.
RTX2300_NVM = 1	The compensation value is read from NVS, i.e. the last stored compensation value – compensation is 0.
RTX2300_CALCULATE_PPM = 2	The frequency offset calculated as a ppm value.
RTX2300_CALCULATE = 3	The compensation value is calculated from the frequency offset called, i.e. new updated value.

TypeName:	BtTstFrequencyType
Group:	Simple
Description:	This type specifies a frequency. e.g. a frequency offset
Type:	rsint32

TypeName:	BtTstBdAddressType
Group:	Array
Description:	This type specifies a Bluetooth Device Address
Type:	rsuint8
Size:	6

TypeName:	BtTstDutProtocolSelectType
Group:	Enumeration
Description:	This type specifies the communication protocol to use with the DUT as stated in the Bluetooth specification

Code	Description
BTTST_DUT_PROTOCOL_HCI = 0	Tester uses HCI protocol to DUT
BTTST_DUT_PROTOCOL_2WIRE = 1	Tester uses 2-Wire protocol to DUT

9 Tester module interface

Interface:	TesterIntf
Description:	This interface allows applications to use and configure the tester module. It is intended for debugging only.

9.1 Mail interface

All messages and types in this interface are prefixed with *Tm.*

9.1.1 Tester RF output

MailSet:	TmSetupTx		
Description:	Set up a packet transmission to the DUT. Starts Bluetooth Low Energy (BLE) transmit test mode (equivalent to the HCI_LE_Transmitter_Test command). Starts packet transmission on a fixed Channel, packet payload Length, and payload Bit pattern. Transmission ends after the specified time or when TmStopTx is sent. Note! This function must be called twice. First time with "TxSetupInit" = TRUE to setup internal interrupt. Call TesterStartTx() to start the transmission and then call this function the second time with TxSetupInit = FALSE and it will return when the specified number of packets have been transmitted.		
Request:			
Description:			
Primitive:	BTTST_TM_SETUP_TX_REQ = 0x7600		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		TxSetupInit	TRUE = 1 to setup interrupts (TRUE for first call) FALSE = 0 for Tx measurements (FALSE in next calls)
BtTstPowerLevelType		PowerLvl	The power level. Unit dBm.
BtTstPacketCountType		Packets	The number of packets to send. Value 1 to 65.535. – if set to 0 it will continuously send until stopped
BtTstChannelNumberType		Channel	The channel number (0 – 39)
BtTstDataLengthType		Length	The payload length in bytes (0 - 37)
BtTstPayloadTypeType		PayloadType	The type of the payload (0-7)
Confirm:			
Description:	The packet transmission has been started		
Primitive:	BTTST_TM_SETUP_TX_CFM = 0x7601		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found
BtTstPacketCountType		Count	The number of packets actually sent. Note that the response time of the bluetooth module may cause it to send a few more packets than requested. Use this actual number for calculations – not the requested count.

MailSet:	TmStopTx		
Description:	Stop packet transmission in the tester module. Obsolete - DO NOT USE		
Request:			
Description:			
Primitive:	BTTST_TM_STOP_TX_REQ = 0x7602		
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:			
Description:	The packet transmission has been stopped		
Primitive:	BTTST_TM_STOP_TX_CFM = 0x7603		
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.
BtTstPacketCountType		Count	The number of packets sent.

MailSet:	TmSetAttenuation		
Description:	Set the attenuation of the RF signal from the tester to the DUT. Range is 0 to 93 dB.		
Request:			
Description:			
Primitive:	BTTST_TM_SET_ATTENUATION_REQ = 0x7604		
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		Attenuation	The attenuation to set. Range is 0-93, corresponding to 0-93 dB attenuation.
Confirm:			
Description:	The attenuator has been set		
Primitive:	BTTST_TM_SET_ATTENUATION_CFM = 0x7605		
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: illegal attenuation specified

MailSet:	TmSetRfOutputLevel		
Description:	Set the level of the RF signal from the tester to the DUT. Setting the level using this command works like the *PowerLvl* parameter in TmSetupTx, but may be used while a tx is ongoing.		
Request:			
Description:			
Primitive:	BTTST_TM_SET_RF_OUTPUT_LVL_REQ = 0x7606		
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The instance number.

BtTstPowerLevelType	PowerLvl	The power level to set. Unit dBm. Range is -40 to -100 dBm.
Confirm:		
Description:	The level has been set	
Primitive:	BTTST_TM_SET_RF_OUTPUT_LVL_CFM = 0x7607	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: illegal level specified

9.1.2 Tester RF input

MailSet:	**TmMeasureNtp**	
Description:	Measure the DUT transmitter power. The DUT must be configured to transmit prior to sending this command.	
Request:		
Description:		
Primitive:	BTTST_TM_MEASURE_NTP_REQ = 0x7620	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Confirm:		
Description:	The NTP has been measured. Note that the result consists of two parts: one holding the integer part and one holding the fractional part. Both are signed, so a result of -12.3 will be returned as -12 in the integer part and -3 in the fractional part	
Primitive:	BTTST_TM_MEASURE_NTP_CFM = 0x7621	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.
BtTstPowerLevelType	PowerLvl_Int	The integer part of the power level. Unit dBm.
BtTstPowerLevelType	PowerLvl_Frac	The fractional part of the power level. Resolution 0.1 dBm. Range -9 to +9

MailSet:	**TmReadAdc**	
Description:	Read the ADC. The DUT must be configured to transmit prior to sending this command.	
Request:		
Description:		
Primitive:	BTTST_TM_READ_ADC_REQ = 0x7622	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
rsuint8	Gain	The gain of the PGA, see the ADS7870 datasheet for details. Valid range is 0 – 7.

Confirm:			
Description:	The ADC value has been read.		
Primitive:	BTTST_TM_READ_ADC_CFM = 0x7623		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: invalid parameter found
rsuint16		Reading	The value of the ADC. The 4 most significant bits are always 0.
rsbool		Overload	If true the ADC input was overloaded while the ADC was read, and the reading is invalid.
rsuint8		PgaStatus	If *Overload* is true this parameters describes the nature of the overload. See the ADS7870 datasheet for details. If *Overload* if false the value of this parameter is indeterminate.

MailSet:	**TmMeasureOffset**		
Description:	Measure the RF frequency offset. The DUT must be configured to transmit prior to sending this command. Obsolete - DO NOT USE		
Request:			
Description:			
Primitive:	BTTST_TM_MEASURE_OFFSET_REQ = 0x7624		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		AvgCount	The number of measurements to make. The returned offset is the resulting average of all measurements. Valid range: 1 - 255.
Confirm:			
Description:	The offset value has been measured. Note that the result consists of two parts: one holding the integer part and one holding the fractional part. Both are signed, so a result of -12.3 will be returned as -12 in the integer part and -3 in the fractional part		
Primitive:	BTTST_TM_MEASURE_OFFSET_CFM = 0x7625		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: invalid parameter found RTX2300_ERR_BUSY: the module is busy.
BtTstRfOffsetIntegerType		Offset_Int	The integer part of the offset. Unit ppm.
BtTstRfOffsetIntegerType		Offset_Frac	The fractional part of the offset. Unit ppm. Resolution 0.1 ppm. Range -9 to +9

9.1.3 Indications

The following indications may be sent from the module to the master at any time after initialization.

MailSet:	TmResetIndication		
Description:	Reset indication. The module has finished it's reset handling and is now ready to accept requests.		
Request:			
Description:	This request is a dummy, i.e. it is never used and exists only to satisfy the interface spec parser.		
Primitive:	BTTST_TM_RESET_INDICATION_DUMMY = 0x7630		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:	TmResetInd		
Description:			
Primitive:	BTTST_TM_RESET_IND = 0x7631		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.

MailSet:	TmBcspEventIndication		
Description:	BCSP event indication. The BCSP protocol has generated an event. Obsolete - DO NOT USE		
Request:			
Description:	This request is a dummy, i.e. it is never used and exists only to satisfy the interface spec parser.		
Primitive:	BTTST_TM_BCSP_EVENT_IND_DUMMY = 0x7632		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:	TmBcspEventInd		
Description:			
Primitive:	BTTST_TM_BCSP_EVENT_IND = 0x7633		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		EventNo	The event.

9.1.4 Send BlueCore command

MailSet:	TmSendBcCmd		
Description:	Send a standard CSR BlueCore command to the tester and return the reply. Obsolete - DO NOT USE		
Request:			
Description:			
Primitive:	BTTST_TM_SEND_BCCMD_REQ = 0x7634		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
TmBcCmdType		BcCmd	The command to send
rsuint8		PayloadSize	The number of 16 bit payload parameters in the command to send. Note that at least 7 parameters will always be sent (see TmBcCmdType) regardless of the value specified here. If more than 7 is specified the additional parameters will be filled with 0x0000 before being sent to the Bluetooth module.
Confirm:			
Description:	The Bluetooth module has replied.		
Primitive:	BTTST_TM_SEND_BCCMD_CFM = 0x7635		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR
TmBcCmdType		BcCmd	The BlueCore response to the command

9.1.5 General housekeeping

9.1.5.1 Initializing the system

MailSet:	TmInit		
Description:	Initialize the module. Caution: this command **must** be sent before using the tester module. It will perform a lengthy initialization procedure of the onboard BlueTooth module, so please allow for extended execution time. If the module has already been initialized sending this command will have no effect.		
Request:			
Description:			
Primitive:	BTTST_TM_INIT_REQ = 0x7640		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsbool		SkipBtInit	If true the firmware in the tester module will not initialize the Bluetooth module. Instead the DLL must handle this.
Confirm:			
Description:	The initialization has finished		
Primitive:	BTTST_TM_INIT_CFM = 0x7641		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.

Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: error initializing the Bluetooth module.
TmStatusType	Status	The module status prior to executing the TmInit command.

9.1.5.2 Getting the Manufacturer Information

The Manufacturer Information is a set of information that describes the system. The information is stored during manufacturing and cannot be changed.

MailSet:	**TmGetManufacturerInfo**	
Description:	Get the Manufacturer Information	
Request:		
Description:		
Primitive:	BTTST_TM_GET_MANUFACTURER_INFO_REQ = 0x7642	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Confirm:		
Description:		
Primitive:	BTTST_TM_GET_MANUFACTURER_INFO_CFM = 0x7643	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR
TmManufacturerInfoType	Info	The Manufacturer Information.

9.1.5.3 User data handling

These requests allow the client to access the user area of the on-board EEPROM. The area consists of 100 bytes and may be used by the customer for any purpose.

If the flag TM_GLOBAL_ACCESS_FLAG is OR'ed to the address, it is considered an absolute EEPROM address, capable of reaching the entire EEPROM. This is only possible in *Manufacturer mode*.

MailSet:	**TmWriteUserData**	
Description:	Write user data to non-volatile storage. Required access rights: *Admin*.	
Request:		
Description:		
Primitive:	BTTST_TM_WRITE_USERDATA_REQ = 0x7644	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
rsuint16	Addr	The user data address
rsuint8	ByteCount	The number of bytes to write, max 16 bytes
TmUserDataType	Data	The data to write
Confirm:		
Description:	The data has been written	
Primitive:	BTTST_TM_WRITE_USERDATA_CFM = 0x7645	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.

Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: the user does not have the required privilege to do this. RTX2300_ERR_RANGE: attempt to access outside the user area, or more than 16 bytes specified.

MailSet:	**TmReadUserData**	
Description:	Read data from the NVS	
Request:		
Description:		
Primitive:	BTTST_TM_READ_USERDATA_REQ = 0x7646	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
rsuint16	Addr	The user data address
rsuint8	ByteCount	The number of bytes to read, max 16 bytes
Confirm:		
Description:	The data has been read	
Primitive:	BTTST_TM_READ_USERDATA_CFM = 0x7647	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: attempt to access outside the user area, or more than 16 bytes specified
rsuint8	ByteCount	The number of bytes read
TmUserDataType	Data	The data to write

9.1.5.4 Requesting system status

MailSet:	**TmGetStatus**	
Description:	Get the current status of the module.	
Request:		
Description:		
Primitive:	BTTST_TM_GET_STATUS_REQ = 0x7648	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Confirm:		
Description:	Return the current status	
Primitive:	BTTST_TM_GET_STATUS_CFM = 0x7649	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR
TmStatusType	Status	The module status

9.1.5.5 Requesting firmware version

MailSet:	TmGetVersion	
Description:	Get version info for installed firmware. The info consists of a firmware defined NULL terminated string, and a 16 bit version number.	
Request:		
Description:		
Primitive:	BTTST_TM_GET_VERSION_REQ = 0x764A	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Confirm:		
Description:	Return the version info	
Primitive:	BTTST_TM_GET_VERSION_CFM = 0x764B	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_UNSUPPORTED: firmware not found.
Rtx2300VersionInfoType	VersionInfo	The version info.

9.1.5.6 Requesting firmware information

MailSet:	TmGetFirmwareInfo	
Description:	Get additional firmware info.	
Request:		
Description:		
Primitive:	BTTST_TM_GET_FIRMWARE_INFO_REQ = 0x764C	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Confirm:		
Description:	Return the firmware info	
Primitive:	BTTST_TM_GET_FIRMWARE_INFO_CFM = 0x764D	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR
Rtx2300DateType	LinkDate	The link date
Rtx2300VersionLabelType	VersionLabel	This field contains the version label as a zero terminated string.

9.1.5.7 Setting access mode

MailSet:	TmSetAccessMode		
Description:	Set the access mode. Some requests needs a priviledged access mode to execute. Please note that 2 failed attemps to set the access mode are accepted. If the third attempt fails the system enters an internal loop and must be restarted.		
Request:			
Description:			
Primitive:	BTTST_TM_SET_ACCESS_MODE_REQ = 0x764E		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300AccessModeType		AccessMode	The required access mode
Rtx2300PasswordType		Password	The password required to enable the mode. No password is required to enable user mode, use 0.
Confirm:			
Description:	Access mode has been enabled		
Primitive:	BTTST_TM_SET_ACCESS_MODE_CFM = 0x764F		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: wrong password RTX2300_ERR_RANGE: unknown mode

MailSet:	TmGetAccessMode		
Description:	Get the access mode		
Request:			
Description:			
Primitive:	BTTST_TM_GET_ACCESS_MODE_REQ = 0x7650		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:			
Description:	The current access mode		
Primitive:	BTTST_TM_GET_ACCESS_MODE_CFM = 0x7651		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: wrong password
Rtx2300AccessModeType		AccessMode	The current access mode

9.1.5.8 Getting/setting serial number

MailSet:	TmSetSerialNo		
Description:	Set serial number information. The serial number information is not used by the firmware. The primary serial number is a number that uniquely identifies this particular RTX2300 system. The secondary serial number may be used for any purpose. It requires *Manufacturer* access rights to change the primary serial number, while the secondary serial number requires *Admin* access rights.		
Request:			
Description:			
Primitive:	BTTST_TM_SET_SERIALNO_REQ = 0x7652		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsbool		SetPrimary	True: set the primary serial number False: set the secondary serial number
Rtx2300SerialNumberType		SerialNo	The serial number.
Confirm:			
Description:	The serial number has been set		
Primitive:	BTTST_TM_SET_SERIALNO_CFM = 0x7653		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: the user does not have the required privilege to do this.

MailSet:	TmGetSerialNo		
Description:	Get the serial number		
Request:			
Description:			
Primitive:	BTTST_TM_GET_SERIALNO_REQ = 0x7654		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:			
Description:			
Primitive:	BTTST_TM_GET_SERIALNO_CFM = 0x7655		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR
Rtx2300SerialNumberType		PrimSerialNo	The primary serial number.
Rtx2300SerialNumberType		SecSerialNo	The secondary serial number.

9.1.5.9 Preset settings to default values

MailSet:	TmSetNvsDefault		
Description:	Preset some or all system settings in Non Volatile Storage to their default values, according to the specified mode. Required access rights: *Manufacturer*.		
Request:			
Description:			
Primitive:	BTTST_TM_SET_NVS_DEFAULT_REQ = 0x7656		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300NvsDefaultModeType		Mode	The mode to use when presetting the settings.
Confirm:			
Description:	The settings has been set		
Primitive:	BTTST_TM_SET_NVS_DEFAULT_CFM = 0x7657		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: the user does not have the required privilege to do this.

9.1.5.10 Getting the current temperature

MailSet:	TmGetTemperature		
Description:	Request current temperature from the device		
Request:			
Description:			
Primitive:	BTTST_TM_GET_TEMPERATURE_REQ = 0x7658		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:			
Description:	The temperature info has been returned from the device		
Primitive:	BTTST_TM_GET_TEMPERATURE_CFM = 0x7659		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR
Rtx2300TemperatureType		Temperature	The current temperature in degrees Celsius. Accuray is +/- 10 degrees.
rsuint16		TemperatureRaw	The current temperature as raw ADC value.

9.1.5.11 Getting internal debug info

MailSet:	TmGetInfo		
Description:	Request debug info. This call is for internal use only!		
Request:			
Description:			
Primitive:	BTTST_TM_GET_INFO_REQ = 0x765A		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		InfoType	The type of info to get
Confirm:			
Description:	The info has been returned from the device		
Primitive:	BTTST_TM_GET_INFO_CFM = 0x765B		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR
rsuint8		InfoType	The type of info
rsuint16		Info0	
rsuint16		Info1	
rsuint16		Info2	
float		Info3	

9.1.5.12 Debug mode

MailSet:	TmSetDebugMode		
Description:	Set debug mode. This call is for internal use only!		
Request:			
Description:			
Primitive:	BTTST_TM_SET_DEBUG_MODE_REQ = 0x765C		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
TmDebugModeType		DebugMode	The debug mode to set
Confirm:			
Description:	The mode has been set in the device		
Primitive:	BTTST_TM_SET_DEBUG_MODE_CFM = 0x765D		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR

TypeName:	TmDebugModeType
Group:	Enumeration
Description:	This type defines the available message types, see CSR BCCMD documentation
Code	**Description**
DEBUGMODE_NONE = 0	Disable debug mode.

DEBUGMODE_UART_B2B = 1	Enable UART back-to-back mode. This mode will never return a confirm, and any further communication with the tester is not possible until it is reset.

9.1.6 Calibration

MailSet:	**TmCalibrate**		
Description:	Write calibration data to the module Obsolete - DO NOT USE		
Request:			
Description:			
Primitive:	BTTST_TM_CALIBRATE_REQ = 0x7660		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Confirm:			
Description:			
Primitive:	BTTST_TM_CALIBRATE_CFM = 0x7661		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR

MailSet:	**TmSetupCwTx**		
Description:	Set up a continuous wave output to the DUT. Note: the output will remain active until the tester is reset. Obsolete - DO NOT USE		
Request:			
Description:			
Primitive:	BTTST_TM_SETUP_CW_TX_REQ = 0x7662		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		PowerLvl	The power level (0-63)
rsuint16		Frequency	Transmitter frequency in MHz (2402-2495)
Confirm:			
Description:	The cw transmission has been started		
Primitive:	BTTST_TM_SETUP_CW_TX_CFM = 0x7663		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found

9.1.7 Other

MailSet:	TmSetTxMode	
Description:	Set the TxMode control signal	
Request:		
Description:		
Primitive:	BTTST_TM_SET_TXMODE_REQ = 0x7670	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
rsbool	On	The new state of the TxMode signal
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstOutputRFConfigurationType	DutRfConnector	The selected RF interface DUT 0 or DUT 1
Confirm:		
Description:		
Primitive:	BTTST_TM_SET_TXMODE_CFM = 0x7671	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR

MailSet:	TmReset	
Description:	Reset the entire module. Notice that doing so triggers initialization of the internals, which may require some time. Also note that the confirm may not always reach the client before the module resets itself.	
Request:		
Description:		
Primitive:	BTTST_TM_RESET_REQ = 0x7672	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
TmResetType	ResetModule	The module/modules to reset
Confirm:		
Description:		
Primitive:	BTTST_TM_RESET_CFM = 0x7673	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The instance number.
Rtx2300ErrorType	ErrorCode	RTX2300_ERR_NO_ERROR

MailSet:	TmTestSetAttenuation		
Description:	Test interface to set the attenuation of the RF signal from the tester to the DUT. This call is for internal use only!		
Request:			
Description:			
Primitive:	BTTST_TM_TEST_SET_ATTENUATION_REQ = 0x7674		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint8		Address	The attenuator module address.
rsuint8		Attenuation	The attenuation to set.
Confirm:			
Description:	The attenuator has been set		
Primitive:	BTTST_TM_TEST_SET_ATTENUATION_CFM = 0x7675		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: illegal attenuation specified

MailSet:	TmTestSetClockDAC		
Description:	Test interface to set the DAC output voltage for internal clock control. This call is for internal use only!		
Request:			
Description:			
Primitive:	BTTST_TM_TEST_SET_CLOCK_DAC_REQ = 0x7676		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint16		DAC_Setting	The DAC value to set.
Confirm:			
Description:	The DAC setting has been set		
Primitive:	BTTST_TM_TEST_SET_CLOCK_DAC_CFM = 0x7677		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: illegal DAC setting specified

MailSet:	TmTestSetIoExt		
Description:	Test interface to set the I/O extender on carrier board. This call is for internal use only!		
Request:			
Description:			
Primitive:	BTTST_TM_TEST_SET_IO_EXT_REQ = 0x7678		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint16		IO_Ext_Setting	The IO Ext. value to set.
Confirm:			
Description:	The IO ext. setting has been set		
Primitive:	BTTST_TM_TEST_SET_IO_EXT_CFM = 0x7679		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: illegal DAC setting specified

MailSet:	TmWriteNVMData		
Description:	Write NVM data to non-volatile storage. Required access rights: *Admin*.		
Request:			
Description:			
Primitive:	BTTST_TM_WRITE_NVMDATA_REQ = 0x767A		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint16		Addr	The user data address
rsuint8		ByteCount	The number of bytes to write, max 16 bytes
TmNVMDataType		Data	The data to write
Confirm:			
Description:	The data has been written		
Primitive:	BTTST_TM_WRITE_NVMDATA_CFM = 0x767B		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_AUTHENTICATION: the user does not have the required privilege to do this. RTX2300_ERR_RANGE: attempt to access outside the user area, or more than 16 bytes specified.

MailSet:	TmReadNVMData		
Description:	Read data from the NVM		
Request:			
Description:			
Primitive:	BTTST_TM_READ_NVMDATA_REQ = 0x767C		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
rsuint16		Addr	The user data address
rsuint8		ByteCount	The number of bytes to read, max 16 bytes
Confirm:			
Description:	The data has been read		
Primitive:	BTTST_TM_READ_NVMDATA_CFM = 0x767D		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The instance number.
Rtx2300ErrorType		ErrorCode	RTX2300_ERR_NO_ERROR RTX2300_ERR_RANGE: attempt to access outside the user area, or more than 16 bytes specified
rsuint8		ByteCount	The number of bytes read
TmNVMDataType		Data	The data to write

9.2 Types

9.2.1 StatusType

TypeName:	TmStatusType
Group:	NonStandard
Description:	This type is used to return module status information to the PC

Code	Description
typedef union TmStatusType	
{	
struct	
{	
rsbitfield InitDone : 1;	The module has been initialized and is ready to accept commands.
rsbitfield SafeMode : 1;	The firmware is in safemode
rsbitfield BtInitialized : 1;	The onboard Bluetooth module has been initialized.
rsbitfield BtInitFailed : 1;	Initialization of the Bluetooth module has failed.
rsbitfield Reserved1 : 4;	
rsbitfield Reserved2 : 8;	
} Bits;	
rsuint16 Data;	
} TmStatusType;	

9.2.2 ResetType

TypeName:	TmResetType
Group:	NonStandard
Description:	This type is used to reset different modules in the tester

Code	Description
typedef union TmResetType	

{	
struct	
{	
rsbitfield ResetAll : 1;	Reset the Tester, Generator and Analyzer and USB modules (except for Analyzer module)
rsbitfield ResetGeneratorModule : 1;	Reset the Generator module
rsbitfield ResetAnalyzer : 1;	Reset the Analyzer and USB module
rsbitfield ResetUSBModules : 1;	Reset the USB modules, except for Analyzer module
rsbitfield Reserved1 : 4;	
} Bits;	
rsuint8 Data;	
} TmResetType;	

TypeName:	**TmResetEnumType**
Group:	Enumeration
Description:	This type is used to reset different modules in the tester. This is a bit field and must match definition for TmResetType

Code	**Description**
RESET_NONE = 0	Disable debug mode.
RESET_ALL=0x01	Reset the Tester, Generator and Analyzer and USB modules (except for Analyzer module)
RESET_GENERATOR_MODULE =0x02	Reset the Generator module
RESET_ANALYZER_MODULE =0x04	Reset the Analyzer and USB module
RESET_USB_MODULES =0x08	Reset the USB modules, except for Analyzer module
RESET_RESERVED1 =0x10	Reserved
RESET_RESERVED2 =0x20	Reserved
RESET_RESERVED4 =0x40	Reserved
RESET_RESERVED8 =0x80	Reserved

9.2.3 User Data Type

TypeName:	**TmUserDataType**
Group:	Struct
Description:	This type contains data transmitted to or from the EEPROM

Code	**Description**
rsuint8 Data[16];	

TypeName:	**TmNVMDataType**
Group:	Struct
Description:	This type contains data transmitted to or from the NVM

Code	**Description**
rsuint8 Data[16];	

9.2.4 User data constants

TypeName:	TmUserDataSize
Group:	Constant
Description:	The number of bytes in the user data area
Type:	rsuint32
Value:	100

TypeName:	TmNVMDataSize
Group:	Constant
Description:	The number of bytes in the NVM data area
Type:	rsuint32
Value:	1024

TypeName:	TmGlobalDataFlag
Group:	Constant
Description:	For internal use only
Type:	rsuint16
Value:	0x8000

9.2.5 Manufacturer Info Type

TypeName:	TmManufacturerInfoType
Group:	Struct
Description:	Bluetooth tester manufacturer information type

Code	Description
Rtx2300DateType ProdDate;	The date of production
Rtx2300SerialNumberType MainboardSerial;	The mainboard serial number, 0 if not applicable
Rtx2300VersionNoType HwVersion;	The hardware version
Rtx2300VersionNoType TestVersion;	The test version

9.3 Function interface

This section contains the functions to start and stop transmit Tx and receive Rx used for different measurements.

9.3.1 Init

Call:	TesterInit	
Description:	This function must be called at init with the Instance number.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR	
Parameters:		
Type	Name	Description
Rtx2300InstanceNoType	InstNo	The instance number.

9.3.2 Transmit

Call:	TesterStartTx	
Description:	Start a packet transmission from the Tester to the DUT	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found	
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstPacketCountType	Packets	The number of packets to send. Value 1 to 65.535. – if set to 0 it will continuously send until stopped
BtTstDataLengthType	Length	The payload length in bytes (a number between 0 and 37)
BtTstPayloadTypeType	Type	The type of the payload

Call:	TesterStopTx	
Description:	Stop an ongoing packet transmission from the Tester.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	**Name**	**Description**

Call:	TesterStartContinuousTx	
Description:	Start a continuous carrier from the Tester to the DUT. Note that the frequency is selected as a channel number. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.	
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)
BtTstPowerLevelType	TxPower	The Tx power level. Unit depends on Tester type. Could be dBm or a register value.

Call:	TesterStopContinuousTx	
Description:	Stop an ongoing continuous transmission from the Tester. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	**Name**	**Description**

9.3.3 Receive

Call:	TesterStartRx	
Description:	Start a packet reception in the Tester. Starts BLE Bluetooth Low Energy receive test mode (equivalent to the HCI_LE_Receiver_Test command). Starts packet reception on a fixed Channel. Note: reception must be stopped by sending TesterStopRx. If not done so within appr. 42 seconds after starting, the packet count may be invalid because the internal 16 bit packet counter in the Tester will overflow!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy. RTX2300_ERR_RANGE: illegal parameter(s) found	
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	TesterStopRx	
Description:	Stop an ongoing packet reception in the Tester and return the number of packets received.	
Return value type:	TesterStopRxResultType	
Return value description:		
Parameters:		
Type	**Name**	**Description**

TypeName:	TesterStopRxResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no reception was ongoing. The packet count is invalid.
BtTstPacketCountType Count;	The number of ok packets received.

Call:	TesterStartContinuousRx	
Description:	Start a receive of a continuous carrier from the DUT to the Tester. Note that the frequency is selected as a channel number. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.	
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	TesterStopContinuousRx	
Description:	Stop an ongoing receive of a continuous transmission from the DUT. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	Name	Description

9.3.4 Power measurement

Call:	TesterReadRSSI	
Description:	Get the RSSI value of the signal from the DUT measured by Tester. This call is for internal use only!	
Return value type:	TesterReadRSSIResultType	
Return value description:		
Parameters:		
Type	Name	Description

TypeName:	TesterReadRSSIResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstRSSIType RSSIValue;	The RSSI value in dBm

Call:	TesterSetTxPower	
Description:	Set Tx power of the Tester.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	Name	Description
rsint8	TxPower	The Tx power is a value in dBm. E.g. 0 for 0 dBm Note! the dBm value must be mapped to vendor specific setup value, e.g. 0 dBm value maps to register value 2 for DUT

9.3.5 Internal Tester registers

Call:	TesterReadHWReg	
Description:	Read value from specified hardware register in Tester. Internal use only	
Return value type:	TesterReadHWRegResultType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	RegAddress	The physical address of the hardware register to read from

TypeName:	TesterReadHWRegResultType
Group:	Struct
Description:	
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
rsuint8 HWregValue;	The value of the read hardware register.

Call:	TesterWriteHWReg	
Description:	Write value to specified hardware register in Tester.	
Return value type:	TesterWriteHWRegResultType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	RegAddress	The physical address of the hardware register to write
rsuint8	RegValue	The value to write hardware register

TypeName:	TesterWriteHWRegResultType
Group:	Struct
Description:	
Code	**Description**
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
rsuint8 HWregValue;	The value of the written hardware register, i.e. readback of just written.

9.3.1 Frequency offset correction

Call:	TesterStartFreqEst	
Description:	Start measurement of frequency estimate of DUT.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	Name	Description

Call:	TesterReadFreqEst	
Description:	Get the frequency offset of the DUT measured by Tester. The result = 999999 Hz is returned if frequency estimate is not ready.	
Return value type:	TesterReadFreqEstResultType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	Name	Description

TypeName:	TesterReadFreqEstResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstFrequencyType FreqEstValue;	The frequency offset in hertz (Hz)

TypeName:	TesterGetOffsetCompensationResultType
Group:	Struct
Description:	

Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR
BtTstRfOffsetType CompensationValue;	The current compensation value in ppm
BtTstRfOffsetType CompensationValueNative;	The current compensation value in whatever unit the DUT reports it. This value is for debugging only, and is not guaranteed to always be valid.

9.3.2 Communication

Call:	TesterSetCommunication	
Description:	Write value to specified hardware register in Tester.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the DUT did not respond).	
Parameters:		
Type	**Name**	**Description**
rsbool	Open	True: open the communication and establish a connection with the Tester False: close the connection. Note that this will not close the COM port.

Call:	TesterSetGeneratorComPort	
Description:	Opens the COM port specified for generator module.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	ComPortNumber	The number of the COM port to open. Specify 0 to close an already open COM.

Call:	TesterGeneratorComPortStatus	
Description:	Returns the status of the COM port.	
Return value type:	rsbool	
Return value description:	FALSE: The COM port is closed TRUE: The COM port is opened	
Parameters:		
Type	**Name**	**Description**

Call:	TesterSetAnalyzerComPort	
Description:	Opens the COM port specified for RF measurements.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	**Name**	**Description**
rsuint16	ComPortNumber	The number of the COM port to open. Specify 0 to close an already open COM.

Call:	TesterAnalyzerComPortStatus	
Description:	Returns the status of the COM port.	
Return value type:	rsbool	
Return value description:	FALSE: The COM port is closed TRUE: The COM port is opened	
Parameters:		
Type	Name	Description

9.3.3 Configuration

Call:	TesterReadConfigMemory	
Description:	Reads from internal configuration memory – the read/write NVM version. Tester has 1 KB of internal configuration memory (NVM). The configuration memory is partitioned into a fixed structure (NVM layout). Address offset is between 0x0000 and 0x07FF. A max. of 128 bytes can be read. This call is for internal use only!	
Return value type:	TesterReadConfigMemoryResultType	
Return value description:		
Parameters:		
Type	Name	Description
rsuint16	ReadAddress	The address to read configuration from. Note! Address offset is between 0x0000 and 0x07FF
rsuint8	ReadLen	The number of data to read from selected address Note! Valid value is max. 128 bytes.

TypeName:	TesterReadConfigMemoryResultType
Group:	Struct
Description:	
Code	Description
Rtx2300ErrorType ErrorCode;	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: read failed.
rsuint8 ConfigData[128];	The read configuration memory. Max. 128 bytes.

Call:	TesterReadRuntimeConfigMemory		
Description:	Reads from internal configuration memory – the run-time version used by system. A copy of the read/write NVM version. Copied on power-up. Tester has 1 KB of internal configuration memory (NVM). The configuration memory is partitioned into a fixed structure (NVM layout). Address offset is between 0x0000 and 0x07FF. A max. of 128 bytes can be read. This call is for internal use only!		
Return value type:	TesterReadConfigMemoryResultType		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
rsuint16	ReadAddress	The address to read configuration from. Note! Address offset is between 0x0000 and 0x07FF	
rsuint8	ReadLen	The number of data to read from selected address Note! Valid value is max. 128 bytes.	

Call:	TesterReadDefaultConfigMemory		
Description:	Reads from internal configuration memory – the default read-only version. It contains some default values following the firmware release. Tester has 1 KB of internal configuration memory (NVM). The configuration memory is partitioned into a fixed structure (NVM layout). Address offset is between 0x0000 and 0x07FF. A max. of 128 bytes can be read. This call is for internal use only!		
Return value type:	TesterReadConfigMemoryResultType		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
rsuint16	ReadAddress	The address to read configuration from. Note! Address offset is between 0x0000 and 0x07FF	
rsuint8	ReadLen	The number of data to read from selected address Note! Valid value is max. 128 bytes.	

Call:	TesterWriteConfigMemory		
Description:	Writes new data to internal configuration memory – the read/write NVM version. Tester has 1 KB of internal configuration memory (NVM). The configuration memory is partitioned into a fixed structure (NVM layout). Address offset is between 0x0000 and 0x07FF. A min. of 4 byte blocks (e.g. 4, 8,12) and max. of 128 bytes can be written. **Caution!!** **The configuration memory must be erased before update.** This call is for internal use only!		
Return value type:	Rtx2300ErrorType		
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: read failed.		
Parameters:			
Type	**Name**	**Description**	
rsuint16	WriteAddress	The address to write configuration to. Note! Address is between 0x0000 and 0x07FF	
rsuint8	WriteLen	The number of data to write to selected address. Note! Valid value is min. 4 byte blocks and max. 128 bytes.	
rsuint8 *	WriteDataPtr	A pointer to the new data to write to configuration memory.	

TypeName:	TesterWriteConfigMemoryType
Group:	Struct
Description:	

Code	Description
rsuint8 ConfigData[128];	Type for writing configuration memory. Max. 128 bytes.

Call:	TesterEraseConfigMemory		
Description:	Erases the internal configuration memory – the read/write NVM version. Tester has 1 KB of internal configuration memory (NVM). The configuration memory is partitioned into a fixed structure (NVM layout). **Caution!!** **All configuration memory is completely erase and must be updated.** This call is for internal use only!		
Return value type:	Rtx2300ErrorType		
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: read failed.		
Parameters:			
Type	**Name**	**Description**	

9.3.4 House keeping

Call:	TesterSystemReset	
Description:	Resets the tester system board.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	Name	Description

Call:	TesterBcCmdReset	
Description:	Resets the tester generator system board.	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	Name	Description

Call:	TesterTestSetClockDAC	
Description:	Test interface to set the DAC output voltage for internal clock control. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: the communication failed (the Tester did not respond).	
Parameters:		
Type	Name	Description
rsint16	DAC_Setting	The DAC value to set.

Call:	TesterGetAnalyzerBuildInfo	
Description:	Test interface to get the build info from the analyzer module.	
Return value type:	TesterGetAnalyzerBuildInfoType	
Return value description:		
Parameters:		
Type	Name	Description

TypeName:	TesterGetAnalyzerBuildInfoType	
Group:	Struct	
Description:	This type contains the build info returned from the analyzer module.	
Code		**Description**
Rtx2300ErrorType ErrorCode;		RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: read failed.
Rtx2300VersionStrType FirmwareVersion;		Array with the firmware version number E.g. “RTX BTLE V2.4.0 ”
rsuint8 BuildDate[12];		Array with the firmware build date E.g. “Apr 4 2017 ”
rsuint8 BuildTime[9];		Array with the firmware build time E.g. “12:43:31 ”
Rtx2300VersionNoType ApiVersion;		Support API version number

Call:	TesterTestStartContinuousTx	
Description:	Start a continuous carrier from the Tester to calibrate internal RX module. Note that the frequency is selected as a channel number. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_BUSY: the module is busy.	
Parameters:		
Type	**Name**	**Description**
BtTstChannelNumberType	Channel	The channel number (0 – 39)

Call:	TesterTestStopContinuousTx	
Description:	Stop an ongoing continuous transmission from the Tester. This call is for internal use only!	
Return value type:	Rtx2300ErrorType	
Return value description:	RTX2300_ERR_NO_ERROR RTX2300_ERR_NO_ACCESS: no transmission was ongoing.	
Parameters:		
Type	**Name**	**Description**

10 Rtx2300 Common Interface

Interface:	Rtx2300Common
Description:	This interface contains types and constants that are shared by all Rtx2300 systems and components.

10.1 Instance number constants/types

TypeName:	RTX2300_INSTNO_NONE
Group:	Constant
Description:	Indicates that no instance number exists
Type:	rsuint8
Value:	0

TypeName:	RTX2300_INSTNO_FIRST
Group:	Constant
Description:	The first valid instance number
Type:	rsuint8
Value:	1

TypeName:	RTX2300_INSTNO_LAST
Group:	Constant
Description:	The last valid instance number
Type:	rsuint8
Value:	0xFD

TypeName:	RTX2300_INSTNO_BROADCAST
Group:	Constant
Description:	Indicates that all instances are receivers.
Type:	rsuint8
Value:	0xFE

TypeName:	RTX2300_INSTNO_COUNT
Group:	Constant
Description:	The total number of instance numbers
Type:	rsuint16
Value:	0x100

TypeName:	Rtx2300InstanceNoType
Group:	Simple
Description:	The Rtx2300 instance number. The instance number is a handle that identifies the application/DLL connection.
Type:	rsuint8

10.2 Error type

TypeName:	Rtx2300ErrorType
Group:	Enumeration
Description:	

Code	Description
RTX2300_ERR_NO_ERROR	Operation successful.
RTX2300_ERR_UNSUPPORTED	The operation is not supported.
RTX2300_ERR_BUSY	The request was rejected by the protocol manager as the Rtx2300 is busy and not able to accept the request.
RTX2300_ERR_TIMEOUT	The operation timed out.
RTX2300_ERR_RANGE	A parameter was outside the legal range.
RTX2300_ERR_NO_ACCESS	The request is not allowed in the current access mode.
RTX2300_ERR_AUTHENTICATION	The firmware did not pass the authentication check.
RTX2300_ERR_VERSION	Firmware version inconsistency! One or more software in the system are having different version numbers. The firmware must be updated before the system can be used.
RTX2300_ERR_SYSINT_FAULT	System Integrity Fault. The integrity of the system has been compromised, please contact RTX Telecom.

10.3 Value types

TypeName:	Rtx2300SignalLvlType
Group:	Simple
Description:	Measured voltage in millivolts.
Type:	rsint32

TypeName:	Rtx2300DistortionLvlType
Group:	Simple
Description:	Measured distortion in per mille (i.e. 1/10 percent).
Type:	rsint16

TypeName:	Rtx2300FrequencyType
Group:	Simple
Description:	Frequency in Hertz
Type:	rsuint32

TypeName:	Rtx2300VoltageType
Group:	Simple
Description:	Voltage in mV
Type:	rsint16

TypeName:	Rtx2300CurrentType
Group:	Simple
Description:	Current in mA
Type:	rsint16

TypeName:	Rtx2300TemperatureType
Group:	Simple
Description:	Temperature in degrees Celsius
Type:	rsint8

TypeName:	Rtx2300AudioAttenuationType
Group:	Simple
Description:	Attenuation in dB. Special values: RTX2300_ATT_MUTE: attenuator is muted
Type:	rsuint8

TypeName:	Rtx2300RealTimeType
Group:	Simple
Description:	Time in seconds since 00:00:00 January 1 1970.
Type:	rsuint32

TypeName:	Rtx2300TimeSpanType
Group:	Simple
Description:	Time span in milliseconds
Type:	rsuint32

TypeName:	Rtx2300SerialNumberType
Group:	Simple
Description:	Serial number stored in the Rtx2300
Type:	rsuint32

10.4 Firmware types

TypeName:	Rtx2300FirmwareType
Group:	Enumeration
Description:	Defines the possible firmware's in the system.

Code	Description
RTX2300_FIRMWARE_TARGET	The main Rtx2300 firmware
RTX2300_FIRMWARE_COPROCESSOR	The Rtx2300 coprocessor firmware
RTX2300_FIRMWARE_POWERSUPPLY	The power supply module firmware
RTX2300_FIRMWARE_EXPANSION_1A	The firmware found in expansion module 1
RTX2300_FIRMWARE_EXPANSION_1B	The additional firmware found in expansion module 1
RTX2300_FIRMWARE_EXPANSION_2A	The firmware found in expansion module 2
RTX2300_FIRMWARE_EXPANSION_2B	The additional firmware found in expansion module 2
RTX2300_FIRMWARE_EXPANSION_3A	The firmware found in expansion module 3
RTX2300_FIRMWARE_EXPANSION_3B	The additional firmware found in expansion module 3
RTX2300_FIRMWARE_EXPANSION_4A	The firmware found in expansion module 4
RTX2300_FIRMWARE_EXPANSION_4B	The additional firmware found in expansion module 4
RTX2300_FIRMWARE_FREQCNT	The frequency counter firmware
RTX2300_FIRMWARE_DLL	The DLL
RTX2300_FIRMWARE_BTTST	The Bluetooth tester firmware
RTX2300_FIRMWARE_LIDCTRL	The Rtx2300 Lid Controller firmware
RTX2300_FIRMWARE_COUNT	

TypeName:	Rtx2300VersionNoType
Group:	Simple
Description:	Rtx2300 version number type. Low byte: minor version High byte: major version Please note that this type may be used as a single hexadecimal value, e.g. if the version is v1.5 the type will contain 0x0105.
Type:	rsuint16

TypeName:	Rtx2300VersionStrType
Group:	Array
Description:	Rtx2300 version string type. Contains a NULL terminated string.
Type:	rsuint8
Size:	16

TypeName:	Rtx2300VersionInfoType
Group:	Struct
Description:	Rtx2300 version info type

Code	Description
Rtx2300VersionNoType VersionNo;	The version number
Rtx2300VersionStrType VersionStr;	The version string

TypeName:	Rtx2300VersionLabelType
Group:	NonStandard
Description:	This type contains the label used to identify the firmware in the VCS system

Code	Description
typedef struct {	
rsuint8 Label[64];	A zero terminated string containing the VCS label.
} Rtx2300VersionLabelType;	

10.5 System types

TypeName:	Rtx2300DateType
Group:	Struct
Description:	This type is used to return the time and date, e.g. of linking the firmware. All fields are BCD coded numeric values as returned by the C standard function time().

Code	Description
rsuint8 Year;	Years since 2000
rsuint8 Month;	Month of year, range 1-12, 1=January
rsuint8 Day;	Day of month, range 1-31
rsuint8 Hour;	Hour of day, range 0-23
rsuint8 Minute;	Minute of hour, range 0-59

TypeName:	Rtx2300NvsDefaultModeType
Group:	Enumeration
Description:	Defines the possible NVS default modes. Preset settings to their defaultvalues. See system NVS documentation for more info.

Code	Description
RTX2300_NVS_DEFAULTMODE_USER	Preset only some of the settings to their default values.
RTX2300_NVS_DEFAULTMODE_FACTORY	Preset all NVS settings to their default values.

RTX2300_NVS_DEFAULTMODE_COUNT	

TypeName:	**Rtx2300PersonalityType**
Group:	Simple
Description:	Personality info stored in the Rtx2300
Type:	rsuint8

TypeName:	**Rtx2300AccessModeType**
Group:	Enumeration
Description:	Defines the possible access modes in the system.

Code	**Description**
RTX2300_ACCESS_MODE_USER	
RTX2300_ACCESS_MODE_ADMIN	
RTX2300_ACCESS_MODE_MANUFACTURER	
RTX2300_ACCESS_MODE_COUNT	

TypeName:	**Rtx2300PasswordType**
Group:	NonStandard
Description:	Rtx2300 standard password type

Code	**Description**
typedef struct Rtx2300PasswordType {	
rsuint8 Password[8];	The password
} Rtx2300PasswordType;	

TypeName:	**Rtx2300SimCfgDataType**
Group:	Struct
Description:	This type is used to hold general data for simulation configuration.

Code	**Description**
rsuint8 U8[4];	8 bit data
rsuint8 U16[2];	16 bit data
rsuint8 U32;	32 bit data

11 DLL Interface

Interface:	TmIntf_DllIntf
Description:	This interface allows applications to configure the Rtx2300 interface DLL.

This interface provides a set of DLL functions for initializing, using and terminating the DLL instance.

Call:	TmIntf_Init		
Description:	Create an instance and init the interface		
Return value type:	Rtx2300Intf_ErrorType		
Return value description:	RTX2300_ERR_NO_ERROR: no problems. RTX2300_ERR_TIMEOUT: no contact to the target RTX2300_ERR_VERSION: version inconsistency detected. A firmware update is required, see Rtx2300Intf_StartFwu()		
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType*	InstNo	Pointer to destination that will receive the instance number of this instance. This instance number must be specified in all following calls to API functions operating on this instance. If the returned instance number is RTX2300INTF_ERROR_NONE the port server instance could not be found or connected to.	
const char*	InstName	The name of this instance. The name is not used by the DLL and may be returned by GetInstanceInfo().	
const char*	PortServerName	The name of the REPS port server to connect to.	
rsuint32	UartComPort	The number of the COM port to use. If REPS_USE_DEFAULT_UART is specified the port is not changed. All other UART and transport layer setting will be set by the DLL.	
rsuint32	MaxBlockTime	The maximum time in ms to wait for a confirm when using the blocking interface. Default is 1000ms. A value of 0 zero means no timeout	

Call:	TmIntf_GetDllVersion		
Description:	Return the version of the DLL		
Return value type:	Rtx2300VersionNoType		
Return value description:			
Parameters:			
Type	**Name**	**Description**	

Call:	TmIntf_GetInstanceInfo		
Description:	Get instance info. Call this function repeatedly until it returns false, i.e. there are no more instances. The first must call have the Init parameter = true, all following call must have Init = false. All remaining parameters are pointers to destination variables for information elements of the instance. If a particular element is not needed the pointer may be set to NULL.		
Return value type:	rsbool		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
rsbool	Init		
Rtx2300InstanceNoType*	IntfInstNoPtr		
Rtx2300InstanceNoType*	MailInstNoPtr		
char*	InstNamePtr		
char*	PortServerNamePtr		
rsuint32*	UartComPortPtr		
rsuint32*	UartBaudRatePtr		
rsuint8*	RepsProgramIdPtr		

Call:	TmIntf_GetThisInstanceInfo		
Description:	Get instance info for the specified instance. All parameters are pointers to destination variables for information elements of the instance. If a particular element is not needed the pointer may be set to NULL.		
Return value type:	rsbool		
Return value description:	false if the instance does not exist		
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	IntfInstNo		
Rtx2300InstanceNoType*	MailInstNoPtr		
char*	InstNamePtr		
char*	PortServerNamePtr		
rsuint32*	UartComPortPtr		
rsuint32*	UartBaudRatePtr		
rsuint8*	RepsProgramIdPtr		

Call:	TmIntf_DeleteInstance		
Description:	Delete the specified instance. All internal threads and queues will be destructed. Please note that this function must be called when terminating the application, before destructing mail handlers, error handlers etc. Otherwise there is a risk that an incoming mail will call the applications handlers after they have been destructed. Do not delete instances you did not create!		
Return value type:	rsbool		
Return value description:	Returns false if the instance is being used by someone else		
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	instance number of the instance to terminate	
rsbool	ClosePortserver	flag, true if the EAP portserver should be closed before terminating the instance.	

Call:	TmIntf_CheckConnection	
Description:	Check the connection to target by sending a mail and waiting for the reply	
Return value type:	rsbool	
Return value description:	If no reply has been received after 'timeout' milliseconds false is returned	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	
rsuint16	Timeout	

Call:	TmIntf_EnableUartHdlcProtocol	
Description:	Set the protocol used to communicate between PC and Rtx2300. Older versions (up to v0050) uses no HDLC protocol, while newer version incorporate a HDLC protocol to handle lost packets in noisy environments. Note that this function is only to be used in special cases – under normal circumstances the DLL and system firmware will come from the same release, and the HDLC setting will automatically be correct. Note: this function must be called before calling the Init() function – otherwise will the call to this function have no effect!	
Return value type:	rsbool	
Return value description:	The default HDLC setting: if true the DLL expects to use the HDLC protocol.	
Parameters:		
Type	**Name**	**Description**
rsbool	UseHdlc	if true the HDLC protocol will be enabled, if false no retransmission protocol will be used.

11.1 Mail, log and error handling

Call:	TmIntf_InstallMailHandler		
Description:	Install a mail handler. Multiple handlers may be installed. Please note that the handlers will be called in context of an internal thread, i.e. the application data modified by the handlers must be properly protected! If parameter Primitive is anything but RTX2300_PRIMITIVE_NONE the handler is specific-mail handler and will only be called with mails containing that particular primitive. Otherwise the handler is a general mail handler, and it will be called for all mail types. If no mail handler is found to handle an incoming mail, it will be stored in the mail queue.		
Return value type:	void		
Return value description:			
Parameters:			
Type	Name	Description	
Rtx2300InstanceNoType	InstNo		
Rtx2300Intf_MailHdlPtrType	Hdl		
BtTstPrimitiveType	Primitive		

Call:	TmIntf_UninstallMailHandler	
Description:	Uninstall the specified handler	
Return value type:	void	
Return value description:		
Parameters:		
Type	Name	Description
Rtx2300InstanceNoType	InstNo	The handle to the instance
Rtx2300Intf_MailHdlPtrType	Hdl	

Call:	TmIntf_ReadQueuedMail	
Description:	Read the specified mail. The mail remains on the queue. If the mail could not be found NULL is returned. If at least one mail handler is found to handle the incoming mail it will NOT be stored in the queue!	
Return value type:	const BtTstMailType*	
Return value description:		
Parameters:		
Type	Name	Description
Rtx2300InstanceNoType	InstNo	The handle to the instance
Rtx2300Intf_MailIndexType	Index	

Call:	TmIntf_RemoveQueuedMail		
Description:	Remove the specified mail from the queue.		
Return value type:	void		
Return value description:			
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The handle to the instance
Rtx2300Intf_MailIndexType		Index	

Call:	TmIntf_GetQueuedMailCount		
Description:	return the number of queued mails		
Return value type:	Rtx2300Intf_MailIndexType		
Return value description:			
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The handle to the instance

Call:	TmIntf_ClearMailQueue		
Description:	Clear the entire mailqueue		
Return value type:	void		
Return value description:			
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The handle to the instance

Call:	TmIntf_InstallLogHandler		
Description:	Install a log handler. Only one handler may be installed! If no handler is installed logs will be discarded. Please note that the handlers will be called in context of an internal thread, i.e. the application data modified by the handlers must be properly protected!		
Return value type:	void		
Return value description:			
Parameters:			
Type		Name	Description
Rtx2300InstanceNoType		InstNo	The handle to the instance
Rtx2300Intf_LogHdlPtrType		Hdl	

Call:	TmIntf_InstallErrorHandler		
Description:	Install an error handler. Only one handler may be installed! If no handler is installed error messages will be discarded (not recommended!) Please note that the handlers will be called in context of an internal thread, i.e. the application data modified by the handlers must be properly protected!		
Return value type:	void		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	The handle to the instance	
Rtx2300Intf_ErrorHdlPtrType	Hdl		

Call:	TmIntf_InstallDefaultErrorHandler		
Description:	Install a default error handler. This handler is used if an error occurs but the error is not related to a specific instance. Only one handler may be installed! If no handler is installed, error messages will be shown and the application terminated. (not recommended!) Please note that the handlers will be called in context of an internal thread, i.e. the application data modified by the handlers must be properly protected! A special problem may occur during initialization of the DLL: if the COM port does not exist the DLL terminates the application because it cannot communicate with the target. The default error handler may be used to remedy this situation: if the COM port could not be opened the DDL will try to disable the transport layer. If successful it will call the default error handler (if installed) with the error code RTX2300INTF_ERROR_PORTSERVER. The error handler may choose to ignore this error, fix the port server now (or even later), and let the application continue.		
Return value type:	void		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
Rtx2300Intf_DefaultErrorHdlPtrType	Hdl		

Call:	TmIntf_SetConfiguration		
Description:	Enable or disable various facilities in the DLL.		
Return value type:	void		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	The handle to the instance	
Rtx2300Intf_CfgType	Cfg		

Call:	TmIntf_GetConfiguration		
Description:	Get configuration settings in the DLL.		
Return value type:	Rtx2300Intf_CfgType		
Return value description:			
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The handle to the instance

Call:	TmIntf_SetMaxWaitingTime		
Description:	Set the new max time to wait in the blocking interface and return the old time.		
Return value type:	rsuint32		
Return value description:	The old waiting time		
Parameters:			
Type		**Name**	**Description**
Rtx2300InstanceNoType		InstNo	The handle to the instance
rsuint32		NewTime	

11.2 Data decoding

Call:	TmIntf_DecodeMail		
Description:	Decode the specified mail and store the resulting string in DestBuf. If ColorDest is not NULL the color of the mail is stored in the variable pointed to by ColorDest.		
Return value type:	void		
Return value description:			
Parameters:			
Type		**Name**	**Description**
const BtTstMailType*		MailPtr	
char*		DestBuf	
rsuint32*		ColorDest	

Call:	TmIntf_DecodePrimitive		
Description:	Decode the specified primitive and store the resulting string in DestBuf.		
Return value type:	void		
Return value description:			
Parameters:			
Type		**Name**	**Description**
BtTstPrimitiveType		Primitive	
char*		DestBuf	

Call:	TmIntf_DecodeErrorCode		
Description:	Decode the specified error code and store the resulting string in DestBuf.		
Return value type:	void		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
Rtx2300ErrorType	Error		
char*	DestBuf		

11.3 Firmware update

Call:	TmIntf_CheckTesterFwu	
Description:	Check if a firmware update is available for Tester module	
Return value type:	rsbool	
Return value description:	Returns true if a new firmware is available, false if not.	
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The handle to the instance
Rtx2300VersionNoType*	CurVer	pointer to Rtx2300VersionInfoType that will receive the version of the firmware currently running on the system
Rtx2300VersionNoType*	NewVer	pointer to Rtx2300VersionInfoType that will receive the version of the new firmware

Call:	TmIntf_StartTesterFwu	
Description:	Start a firmware update. The update will always use the newest version available, and the entire Rtx2300 basic system will be updated. When the update has finished, the user is notified to make a system reset and restart the application. It is possible to specify a path to an executable which will be started when the firmware update is done.	
Return value type:	void	
Return value description:		
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The handle to the instance
const char*	App	The path to an application that will be started after the firmware update. Specify NULL if this is not required.

Call:	TmIntf_CheckAnalyzerFwu		
Description:	Check if a firmware update is available for Analyzer module		
Return value type:	rsbool		
Return value description:	Returns true if a new firmware is available, false if not.		
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	The handle to the instance	
Rtx2300VersionNoType*	CurVer	pointer to Rtx2300VersionInfoType that will receive the version of the firmware currently running on the system	
Rtx2300VersionNoType*	NewVer	pointer to Rtx2300VersionInfoType that will receive the version of the new firmware	

Call:	TmIntf_StartAnalyzerFwu		
Description:	Start a firmware update. This function will start another dedicated bootloader tool. The analyzer COM port is closed and reopen when finished. The COM port and firmware file must be selected - Click “Port Settings” and setup the COM-port for the ‘Analyzer’ interface and select Baud=57600, Flow=None, Parity=None, Stop Bits=One and Data Bits=8 - Click “Select File” and select the file “RTX_BTLE_Analyzer_V<version no.>” - Click “Erase Image” and then “Load Image” - Wait for firmware to download and click ‘OK’ - Exit RTX BTLE Serial Bootloader It is possible to specify a path to an executable which will be started when the firmware update is done.		
Return value type:	void		
Return value description:			
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	The handle to the instance	
const char*	App	The path to an application that will be started after the firmware update. Specify NULL if this is not required.	

Call:	TmIntf_CheckGeneratorFwu		
Description:	Check if a firmware update is available for Generator module For internal use only!!		
Return value type:	rsbool		
Return value description:	Returns true if a new firmware is available, false if not.		
Parameters:			
Type	**Name**	**Description**	
Rtx2300InstanceNoType	InstNo	The handle to the instance	

Rtx2300VersionNoType*	CurVer	pointer to Rtx2300VersionInfoType that will receive the version of the firmware currently running on the system
Rtx2300VersionNoType*	NewVer	pointer to Rtx2300VersionInfoType that will receive the version of the new firmware

Call:	**TmIntf_StartGeneratorFwu**	
Description:	Start a firmware update. This function will start another dedicated bootloader tool. For internal use only!!	
Return value type:	void	
Return value description:		
Parameters:		
Type	**Name**	**Description**
Rtx2300InstanceNoType	InstNo	The handle to the instance
const char*	App	The path to an application that will be started after the firmware update. Specify NULL if this is not required.

11.4 Debug functions

These functions are intended for testing the interface between the DLL and the application. They have no effects in the DLL and may be used without connection to the Rtx2300 system. If problems with the calling convention are suspected these function may be helpful.

Call:	**TmIntf_SetDbgCfg**	
Description:	Set the debug configuration. Please note that enabling debugging will seriously affect performance. Debugging features are only available in the debug version of the DLL.	
Return value type:	void	
Return value description:		
Parameters:		
Type	**Name**	**Description**
Rtx2300Intf_DbgCfgType	Cfg	

Call:	**TmIntf_DbgTestFuncCall0**	
Description:	Test a call with no parameters.	
Return value type:	void	
Return value description:		
Parameters:		
Type	**Name**	**Description**

Call:	TmIntf_DbgTestFuncCall8		
Description:	Test a call with one 8 bit parameter.		
Return value type:	rsuint8		
Return value description:	the value supplied in the parameter		
Parameters:			
Type	Name	Description	
rsuint8	rsuint8data		

Call:	TmIntf_DbgTestFuncCall16		
Description:	Test a call with one 16 bit parameter.		
Return value type:	rsuint16		
Return value description:	the value supplied in the parameter		
Parameters:			
Type	Name	Description	
rsuint16	rsuint16data		

Call:	TmIntf_DbgTestFuncCall32		
Description:	Test a call with one 32 bit parameter.		
Return value type:	rsuint32		
Return value description:	the value supplied in the parameter		
Parameters:			
Type	Name	Description	
rsuint32	rsuint32data		

Call:	TmIntf_DbgGetIntfTestData1		
Description:	Fill the structure pointed to by the TestdataPtr with known data, to test data alignment. Each member of the structure is initialized with the number of bits in the member, i.e. the Rsuint8 member is initialized with 8.		
Return value type:	void		
Return value description:			
Parameters:			
Type	Name	Description	
Rtx2300Intf_DbgDataType*	TestDataPtr		

Call:	TmIntf_DbgMakeError		
Description:	Call the installed error handler for the specified instance		
Return value type:	void		
Return value description:			
Parameters:			
Type	Name	Description	
Rtx2300InstanceNoType	InstNo	The handle to the instance	

Call:	TmIntf_DbgMakeErrorDefault		
Description:	Call the installed default error handler		
Return value type:	void		
Return value description:			
Parameters:			
Type	Name	Description	

11.5 Types

TypeName:	REPS_USE_DEFAULT_UART
Group:	Constant
Description:	If specified as COM port number in the call to Intf_Init() the instance will reuse the COM port currently used by the port server.
Type:	rsuint8
Value:	0xFF

TypeName:	TRAFFICLOG_FILENAME
Group:	Constant
Description:	Name of the debug log file for logging traffic
Type:	String
Value:	"DbgTrafficLog.txt"

TypeName:	CALLLOG_FILENAME
Group:	Constant
Description:	Name of the debug log file for logging calls
Type:	String
Value:	"DbgCallLog.txt"

TypeName:	Rtx2300Intf_ErrorType
Group:	Enumeration
Description:	The possible error codes returned from calls to the DLL interface.

Code	Description
RTX2300INTF_ERROR_NONE	
RTX2300INTF_ERROR_INIT	The interface was not initialized or initialized twice!
RTX2300INTF_ERROR_PORTSERVER	Error communicating with the port server
RTX2300INTF_ERROR_UNKNOWN_INST	The specified instance number is unknown
RTX2300INTF_ERROR_INST_OVERFLOW	Too many instances in use!

TypeName:	Rtx2300Intf_MailIndexType
Group:	Simple
Description:	Mail index type. Mails are indexed in chronological order of reception, i.e. index 0 is the oldest mail.
Type:	rsuint16

TypeName:	Rtx2300Intf_CfgType
Group:	Struct
Description:	This type is used to configure the DLL interface.

Code	Description
rsuint16 LogMailsToFile : 1;	Log all mail traffic to the file DbgTrafficLog.txt. Slows execution considerably!
rsuint16 LogMailsToLog : 1;	Log all mail traffic to the log handler
rsuint16 CollectUnhandledMails : 1;	All mails that are not processed by mail handlers are collected in the mail queue
rsuint16 AssertOnTimeouts : 1;	Assert if a timeout occurs in the blocking API. Debugversion of the DLL only!
rsuint16 ErrorOnTimeouts : 1;	Call error handler if a timeout occurs in the blocking API
rsuint16 Reserved : 11;	

TypeName:	Rtx2300Intf_DbgCfgType
Group:	Struct
Description:	This type is used to configure the DLL interface for debugging

Code	Description
rsuint16 LogDllCalls : 1;	Log all calls to DLL-only functions. Slows execution considerably!
rsuint16 LogStdApiCalls : 1;	Log all calls to standard API functions. Slows execution considerably!
rsuint16 LogBlockingApiCalls : 1;	Log all calls to blocking API functions. Slows execution considerably!
rsuint16 LogInternalCalls : 1;	Log all calls to internal functions. Slows execution considerably!
rsuint16 Reserved : 12;	

TypeName:	Rtx2300Intf_DbgDataType
Group:	Struct
Description:	Data returned by debug function

Code	Description
rsuint8 rsuint8;	
rsuint16 rsuint16;	
rsuint32 rsuint32;	
rsbool rsbool;	

TypeName:	Rtx2300Intf_LogEntryType
Group:	Enumeration
Description:	Log entrytype

Code	Description
RTX2300INTF_LOGENTRY_INFO	The logentry contains general system info
RTX2300INTF_LOGENTRY_WARNING	The logentry contains a system warning
RTX2300INTF_LOGENTRY_ERROR	The logentry contains a system error
RTX2300INTF_LOGENTRY_MAILTRACE	The logentry contains a decoded mail

TypeName:	Rtx2300Intf_LogHdlPtrType
Group:	NonStandard
Description:	Pointer to log handler. This is the function that the user must provide in order to get callbacks when the system wants to log something. See XXXX_InstallLogHandler() InstNo: the instance number EntryType: the type of entry LogStr: the actual text to log Color: an rsuint32 that contains optional info on the color of the entry. This is only relevant if the logentry is a mail trace. The values used are the same as defined in conio.h

Code	Description
typedef void (__stdcall *Rtx2300Intf_LogHdlPtrType)(Rtx2300InstanceNoType InstNo, Rtx2300Intf_LogEntryType EntryType, const char* LogStr, rsuint32 Color);	

TypeName:	Rtx2300Intf_ErrorHdlPtrType
Group:	NonStandard
Description:	Pointer to error handler. This is the function that the user must provide in order to get callbacks when the system wants to log an error. See XXXX_InstallErrorHandler() InstNo: the instance number ErrNo: the error number. InstName: pointer to the instance name. FuncName: pointer to the name of the function in which the error occurred. FileName: pointer to the name of the file in which the error occurred. LineNo: the number of the line in the file in which the error occurred. ErrMsg: the actual error message.

Code	Description
typedef void (__stdcall *Rtx2300Intf_ErrorHdlPtrType)(Rtx2300InstanceNoType InstNo, Rtx2300Intf_ErrorType ErrNo, const char* InstName, const char* FuncName, const char* FileName, rsuint32 LineNo, const char* ErrMsg);	

TypeName:	Rtx2300Intf_DefaultErrorHdlPtrType
Group:	NonStandard
Description:	Pointer to the default error handler. This is the function that the user must provide in order to get callbacks when the system wants to log an error but no instances exists. See XXXX_InstallErrorHandler() ErrNo: the error number. InstName: pointer to the instance name. FuncName: pointer to the name of the function in which the error occurred. FileName: pointer to the name of the file in which the error occurred. LineNo: the number of the line in the file in which the error occurred. ErrMsg: the actual error message.

Code	Description
typedef void (__stdcall *Rtx2300Intf_DefaultErrorHdlPtrType)(Rtx2300Intf_ErrorType ErrNo, const char* FuncName, const char* FileName, rsuint32 LineNo, const char* ErrMsg);	

TypeName:	Rtx2300Intf_MailHdlPtrType	
Group:	NonStandard	
Description:	Pointer to mail handler. This is the function that the user must provide in order to get callbacks when a mail is received. See Rtx2300Intf_InstallMailHandler() instno: the instance number mail: pointer to the mail mailsize: the size of the mail	
Code		**Description**
typedef void (__stdcall *Rtx2300Intf_MailHdlPtrType)(Rtx2300InstanceNoType instno, const BtTstMailType* mail, rsuint16 mailsize);		

12 Global types

Global types are used by multiple interfaces simultaneously. They are all defined in the pseudo-interface *Global*.

Interface:	Global
Description:	This interface groups types used by multiple interfaces.

12.1 Rtx2300 System Types

The following types are standard Rtx2300 types, see *Rtx2300 System InterfaceSpecification* for a detailed description of these types. They are not defined in this document.

- Rtx2300ErrorType
- Rtx2300VersionInfoType
- Rtx2300VersionInfoStrType
- Rtx2300DateType
- Rtx2300FrequencyType
- Rtx2300AccessModeType
- Rtx2300PasswordType
- Rtx2300FwuErrorType

12.2 BtTstPowerLevelType

TypeName:	**BtTstPowerLevelType**
Group:	Simple
Description:	This type holds the result of a power measurement.
Type:	rsint16

12.3 BtTstNativeCrystalTuneType

TypeName:	**BtTstNativeCrystalTuneType**
Group:	Simple
Description:	This type holds the native crystal tune value. Integer value positive or negative.
Type:	rsint16

12.4 BtTstFrequencyType

TypeName:	**BtTstFrequencyType**
Group:	Simple
Description:	This type holds a positive or negative frequency value in hertz (Hz).
Type:	rsint32

12.5 BtTstFrequencyPPMType

TypeName:	BtTstFrequencyPPMType
Group:	Simple
Description:	This type holds a positive or negative frequency value in ppm.
Type:	double

12.6 BtTstRSSIType

TypeName:	BtTstRSSIType
Group:	Simple
Description:	This type holds the RSSI value in dBm. It's a negative number and RSSI value is received as x100, so it must be divided by 100 to give correct RSSI level with 2 decimals.
Type:	rsint16

12.7 BtTstRfOffsetType

TypeName:	BtTstRfOffsetIntegerType
Group:	Simple
Description:	This type holds the result of a RF offset measurement.
Type:	rsint16

TypeName:	BtTstRfOffsetType
Group:	Simple
Description:	This type holds the result of a RF offset measurement.
Type:	double

12.8 BtMeasurementTimeType

TypeName:	BtTstMeasurementTimeType
Group:	Simple
Description:	This type holds the duration of a measurement.
Type:	rsuint16

12.9 BtTstSensitivityType

TypeName:	BtTstSensitivityType
Group:	Simple
Description:	This type holds the result of a sensitivity measurement
Type:	rsuint32

12.10 BtTstPayloadTypeType

TypeName:	BtTstPayloadTypeType
Group:	Enumeration
Description:	This type defines the available payload types

Code	Description
BTTST_PAYLOADTYPE_RANDOM9 = 0	Pseudo random bit sequence 9
BTTST_PAYLOADTYPE_ALTBITS_11110000 = 1	Pattern of alternating bits 11110000
BTTST_PAYLOADTYPE_ALTBITS_10101010 = 2	Pattern of alternating bits 10101010
BTTST_PAYLOADTYPE_RANDOM15 = 3	Pseudo random bit sequence 15
BTTST_PAYLOADTYPE_ALL_ONES = 4	Pattern of all 1's
BTTST_PAYLOADTYPE_ALL_ZEROS = 5	Pattern of all 0's
BTTST_PAYLOADTYPE_ALTBITS_00001111 = 6	Pattern of alternating bits 00001111
BTTST_PAYLOADTYPE_ALTBITS_0101 = 7	Pattern of alternating bits 0101
BTTST_PAYLOADTYPE_COUNT	

12.11 BtTstChannelNumberType

TypeName:	BtTstChannelNumberType
Group:	Simple
Description:	This type holds a Bluetooth channel number, 0 to 39 (BLE channel = (freqMHz - 2402) / 2).
Type:	rsuint8

12.12 BtTstRfMeasureModeType

TypeName:	BtTstRfMeasureModeType
Group:	Enumeration
Description:	This type defines the selected tester RF measuring mode – burst or CW (continuous wave) signals

Code	Description
BTTST_RF_MEASURE_MODE_CW = 0	Tester uses CW
BTTST_RF_MEASURE_MODE_BURST = 1	Tester uses burst

12.13 BtTstPacketCountType

TypeName:	BtTstPacketCountType
Group:	Simple
Description:	This type holds the number of packets. Value 1 to 65.535
Type:	rsuint32

12.14 BtTstDataLengthType

TypeName:	BtTstDataLengthType
Group:	Simple
Description:	This type holds the number of payload bytes in a packet 0-37
Type:	rsuint8

12.15 BtTstDutConfigurationType

TypeName:	BtTstDutConfigurationType
Group:	Struct
Description:	DUT interface configuration data. The fields in this type have no specific use, and may be used to transport any type of configuration data to or from the application to the DUT interface DLL.

Code	Description
rsuint8 Arg1;	
rsuint32 Arg2;	
rsuint32 Arg3;	
rsuint8 Data0;	Arrays do not autogenerate easily in C# so individual fields are used here instead.
rsuint8 Data1;	
rsuint8 Data2;	
rsuint8 Data3;	
rsuint8 Data4;	
rsuint8 Data5;	
rsuint8 Data6;	
rsuint8 Data7;	
rsuint8 Data8;	
rsuint8 Data9;	
rsuint8 Data10;	
rsuint8 Data11;	
rsuint8 Data12;	
rsuint8 Data13;	
rsuint8 Data14;	
rsuint8 Data15;	

TypeName:	BtTstOutputRFConfigurationType
Group:	Enumeration
Description:	This type defines the available output configurations on the front of the tester

Code	Description
BtTstOutputRfDUT0Sel = 0	Select output DUT0 RF on the front
BtTstOutputRfDUT1Sel = 1	Select output DUT1 RF on the front

TypeName:	BtTstOutputConfigurationType
Group:	Enumeration
Description:	This type defines the available output configurations on the front of the tester

Code	Description
BtTstOutputDUT0UartSel = 0	Select output DUT0 UART on the front
BtTstOutputDUT1UartSel = 1	Select output DUT1 UART on the front
BtTstOutputDUT0Usb = 2	Select output DUT0 USB on the front
BtTstOutputDUT1Usb = 3	Select output DUT1 USB on the front

12.15.1 BcCmd Message Type Type

TypeName:	TmBcCmdMessageTypeType
Group:	Enumeration
Description:	This type defines the available message types, see CSR BCCMD documentation

Code	Description
BCCMD_MESSAGETYPE_GETREQ = 0	Client to server
BCCMD_MESSAGETYPE_GETRESP = 1	Server to client
BCCMD_MESSAGETYPE_SETREQ = 2	Client to server

12.15.2 BcCmd Message Status Type

TypeName:	TmBcCmdStatusType
Group:	Enumeration
Description:	This type defines the available status types, see CSR BCCMD documentation

Code	Description
BCCMD_STATUS_OK = 0	No problem found
BCCMD_STATUS_NO_SUCH_VARID	Variable identifier not recognized
BCCMD_STATUS_TOO_BIG	Data exceeded message capacity
BCCMD_STATUS_NO_VALUE	Variable has no value
BCCMD_STATUS_BAD_REQUEST	GETREQ or SETREQ held an error
BCCMD_STATUS_NO_ACCESS	Value of variable is inaccessible
BCCMD_STATUS_READ_ONLY	Value of variable is unwritable
BCCMD_STATUS_WRITE_ONLY	Value of variable is unreadable
BCCMD_STATUS_ERROR	Other error
BCCMD_STATUS_PERMISSION_DENIED	Request not allowed
BCCMD_STATUS_TIMEOUT	Timeout during server processing

12.15.3 BcCmd Command Type

TypeName:	TmBcCmdType
Group:	Struct
Description:	Holds a CSR BlueCore Command Type to the tester module. This type is used for sending commands to the tester module, as well as for receiving replies to commands. See CSR documentation for BCCMD commands for details.

Code	Description
TmBcCmdMessageTypeType MsgType;	The message type
rsuint16 SeqNo;	The sequence number
rsuint16 Cmd;	The command. this is also known as the VarId
rsuint16 P0;	Parameter for the command. Specify 0 if no value is required.
rsuint16 P1;	Parameter for the command. Specify 0 if no value is required.
rsuint16 P2;	Parameter for the command. Specify 0 if no value is required.
rsuint16 P3;	Parameter for the command. Specify 0 if no value is required.
rsuint16 P4;	Parameter for the command. Specify 0 if no value is required.
rsuint16 P5;	Parameter for the command. Specify 0 if no value is required.

rsuint16 P6;	Parameter for the command. Specify 0 if no value is required.
TmBcCmdStatusType Status;	The status of the reply. Set to BCCMD_STATUS_OK for commands.