Production test PC & DWDIO.DLL

Subject: Interfaces between PC Tools and dwdio.dll

Revision: 17.0
Author: Per Bøgebjerg
Last Updated: 12/12/2005 11:38
Last Updated By: PBO
Printout Date: 12/12/2005 01:21
File: N:\dwdtools\dwdio\Documentation\Production_test_pc_dll\production_test_pc_dll_v17.0.doc

Version history

Date	Revision	Who	Description
25.07.2005	8.3	PBO	The following function is added: DWD_btd_set_tx_burst_mode DWD_btd_set_tx_cw_mode DWD_btd_set_rx_cw_mode DWD_btd_get_bluemoon_firmware_version DWD_key_generic DWD_set_audio_parms DWD_set_iqrms_sample_number DWD_rtc_set_date_val DWD_rtc_get_date_val DWD_rtc_set_time_val DWD_rtc_get_time_val DWD_rtc_set_event_datetime DWD_rtc_get_event_datetime DWD_rtc_clear_event_datetime DWD_bl_set_backlight Following functions has been removed: DWDIO_set_CharDelay DWD_get_error_status

			DWD_get_pc_id ULC platform added.
01.09.2005	8.4	PBO	New function added. DWD_poll_key_matrix_result DWD_set_power_down_mode DWD_chr_get_measurement_control_status DWD_egoldradio_version_revision Function there already was released, but description missing is added DWD_start_buzzer_melody DWD_mmci_check DWD_get_handle_for_com_port Function DWD_osc_get_rx_afc_calib_data is changed.
06.10.2005	8.5	PBO	New function added. DWD_lcd_get_hw_id DWD_fs_ms_generic DWD_fs_ms_generic_timeout_delay DWD_get_product_name DWD_non_sig_use_calib_parms DWD_osc_get_xocal Function DWD_get_dynamic_parm_xx and DWD_store_dynamic_parm_xx is changed.
08.11.2005	8.6	PBO	New function added. DWD_configure_sim_simulation DWD_btd_get_hw_info Changed function: Set_current_path DWD_wrk_get_ptm DWD_osc_xocal DWD_btd_get_bluemoon_firmware_version
15.11.2005	8.7	PBO	Description for some Bluetooth functions in chapter 6.13 is updated. General aspect in chapter 2 is updated. Two new generic function added (DWD_aud_generic_longint and DWD_aud_generic_longint_timeout_delay).
06.12.2005	17.0	PBO	This document version number is now aligned with the dwdio.dll version number. New function added: DWD_get_sim_icc_id DWD_ver_generic - DWD_ver_generic_timeout_delay DWD_trap_generic – DWD_trap_generic_timeout_delay

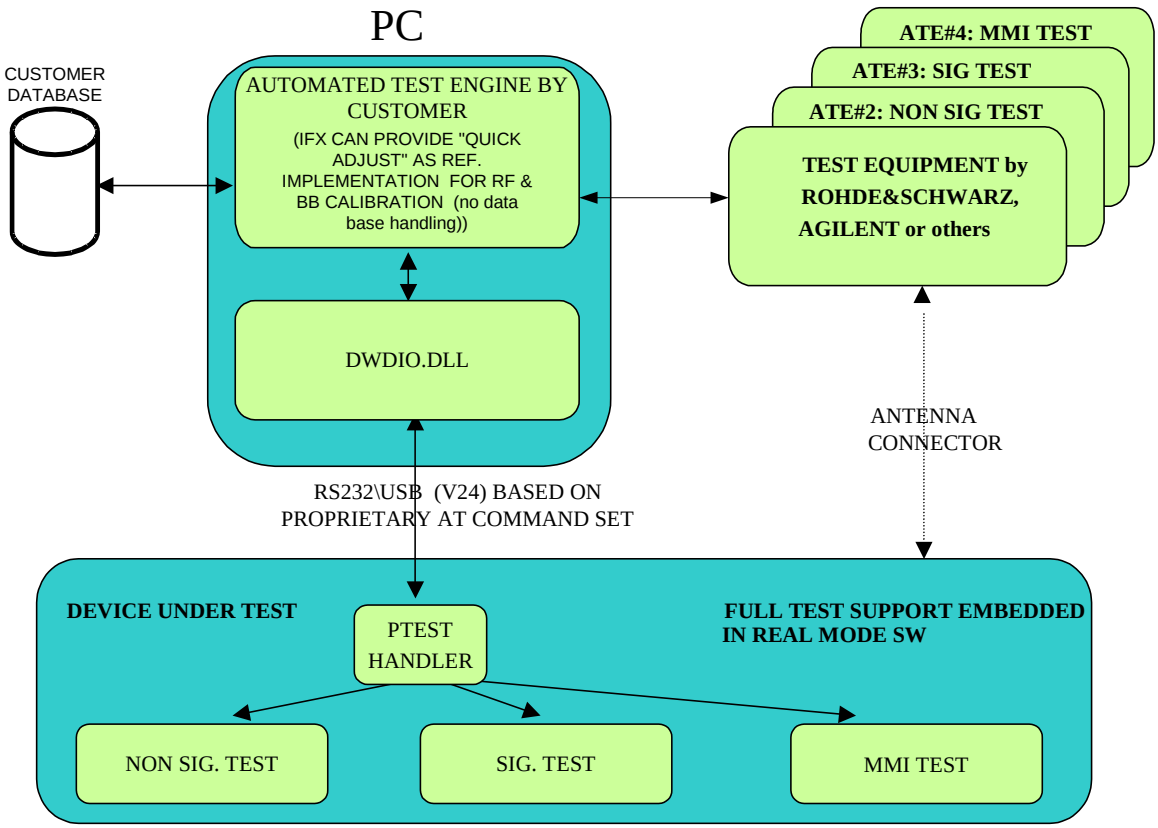
			DWD_emd_generic – DWD_emd_generic_timeout_delay DWD_rtc_generic – DWD_rtc_generic_timeout_delay DWD_iic_generic – DWD_iic_generic_timeout_delay DWD_get_rxlev_gain_offset_v2 DWD_store_rxlev_gain_offset_v2 DWD_set_rxlev_gain_offset_v2 DWD_get_edge_txcorr DWD_store_edge_txcorr DWD_btd_bmu_get_address DWD_btd_bmu_store_address DWD_btd_bmu_get_tx_power_offset DWD_btd_bmu_store_tx_power_offset DWD_btd_bmu_get_baud_rate DWD_btd_bmu_store_baud_rate DWD_get_lcd_contrast_v2 DWD_store_lcd_contrast_v2 DWD_get_hw_coding Removed function: DWD_non_sig_use_calib_parms. Renamed function: DWD_lcd_get_hw_id ->DWD_gdd_get_lcd_id

Approved/owner

SW developer
Per Bøgebjerg / PBO

1 Introduction

This document describes the interface functions included in the DWDIO DLL. The DWDIO is mainly used as communication DLL for automated production testing, such as our Quick Adjust and Phone Tool. Other specialized tools may utilize the functionality of the DLL as well. The DLL encapsulates the HW interface to the DUT(Device Under Test). Below is showed the context of DWDIO.DLL in production test mode (ATE means Automatic Test Equipment).



Details on the communication protocol between DUT and DWDIO.DLL can be found in Appendix 2.

Main features of the DLL are listed below:

- High Level API for Base Band and RF calibration
- High Level API for IMEI programming and Mobile Personalization
- Auto detection of Non Volatile Memory version
- Auto mapping of NV parameters during read and write
- Easy to integrate into various production and development tools
- DWDIO.DLL Trace easing debugging of the DLL integration

The document is divided into the following main chapters:

- General aspects
- Communication link
- Configuration of test modes

- RF testing
- Base Band functions
- Security
- Non Volatile Memory handling
- Workbench
- Generic Test Bench
- Debugging utilities
- DLL Trace
- Exception
- Miscellaneous functions
- Appendix 1: Reference DLL functions used by Quick Adjust
- Apendix2: Communication protocol between DWDIO.DLL and DUT

Details on suggested HW calibration is out of the scope of this document and must be found in other relevant documents – see list of references.
Note that the DWDIO.DLL are generic and used for all the INFINEON's platforms. Functions supported for a specific platform are stated in the description of the individual function. If a customer wants to leave out some features of a specific platform then omit using the corresponding functions.

1.1 Platform definition

Platform name	Description
P2002	EGOLD+ v2.
P2002+	EGOLD+ v3.
BP2	EGOLD-LITE.
BP30	EGOLD-RADIO
MP1	SGOLD & SGOLD-LITE
MP1(E)	SGOLD with EDGE
MP1(U)	SGOLD with UMTS
MPE	SGOLD2 + EDGE
MPEU	SGOLD2 + EDGE + UMTS
ULC	EGOLD-RADIO (Ultra low cost platform).

1.2 Reference documents

Title
Production test concept
ATE#2_test_specification (calibration for relevant platform)
ATE#3_RF_test_spec (performance check for relevant platform)
Note about changed Rx level calibration
RF-SW Interface specification (for relevant platform)
BB-SW Interface specification (for relevant platform)
Audio Interface

2 General aspect

2.1 Usage of DLL functions

The dwdio.dll can support two comports. When function DWD_set_com_port is called, will it return a handle there can be used to switch between comport. There are a blocking mechanism, so you can only have one command active, and you need to wait for confirm or dll timeout, before you can issue a command to the other mobile. Please see example below:

handle1 = DWD_set_com_port(COM1)
handle2 = DWD_set_com_port(COM2)

/*establish connection with PCB connected to COM1*/
/*DWD_check_comm_link will not return untill confirm from target PCB has been recived or alternatively timeout condition has occured*/
if(!DWD_check_comm_link(Handle1)) connection_failed do error error handling

/*After respons from target PCB connected to COM1 connection establishment with PCB connected to COM2 can be initiatated.*/
if(!DWD_check_comm_link(handle2))

General for all interface functions are that the return parameter only states if the functions has been carried out and not the result. In case the result is OK, it means that the DUT has acknowledged the command (in case the functions require target communication). In case of ERROR there might be a problem with the HW connection to the DUT, or that the given input arguments are invalid.

Furthermore, all functions that communicate with the DUT or refer to NV parameters have a *handle* as input parameter, which is used to identify the COM PORT or NV mirror target for the communication.

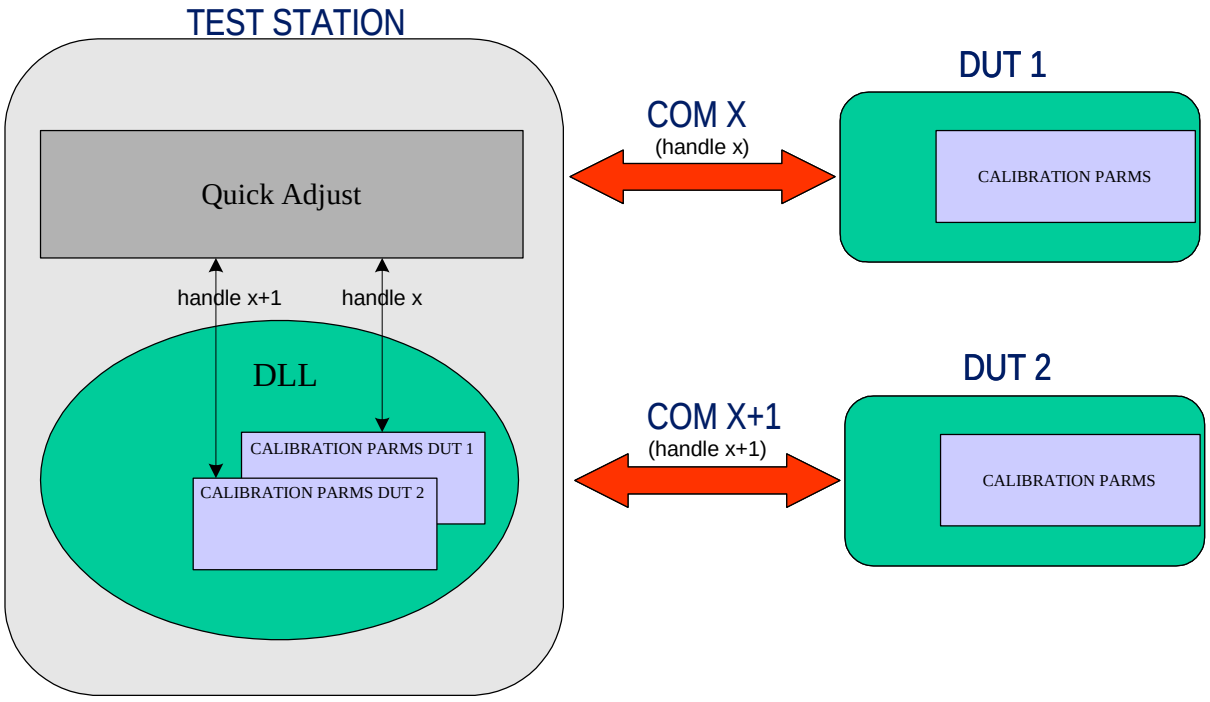
Example to be followed for all functions in order to detect communication errors on the serial line:

```
{
    if(DWD_xxxxxx(handle_comportX)==dwd_ok)
    { /* proceed with test*/
        ……
    }
    else /*dwd_error*/
    {
        /*Stop test*/
        /*This should never occur unless something is wrong on the electrical connection of the serial
        link*/
    }
}
```

2.2 Handling of NV Memory

Special care must be taken for handling of NV memory. When accessing the DUT first time after connecting, the DLL will read the complete static NV memory content into RAM mirror. Subsequent get and store access will be directed towards the mirror and calibration date will be kept here until

completion of calibration. It is possible to have two devices connected to the DLL simultaneously and the corresponding NV mirror is associated via the handle allocated when calling DWD_set_comport.
The NV mirrors are illustrated in the figure below:

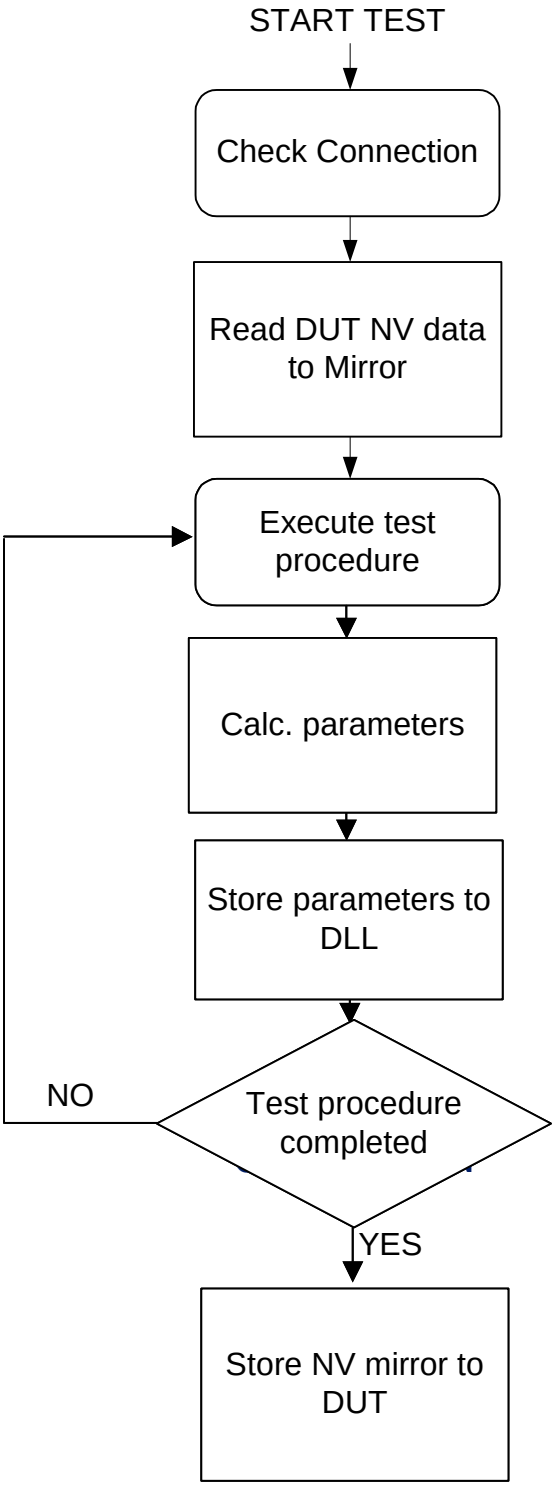


Calling ***DWD_store_to_nv_wchksum*** or ***DWD_store_to_nv_memory*** does storing the NV parameters from mirror to target. This MUST be done at the last stage of a test station just before powering off. This is necessary because of the following reasons:

- In order to save time
- The Static NV block residing in Flash will be invalid during the erasure, which might cause unexpected behavior of the various test functionality on target

The EEP configuration file must be located in the same directory as dwdio.dll except that you have changed the current directory by using the dwdio.dll function ***Set_current_directory***. The EEP prefix needs also to be set ***DWD_set_eep_cfg_filename***. EEP prefix means this name you can find in the EEP configuration filename before "_eepxxx.cfg"

Suggested handling is illustrated in the figure below:


START TEST
Check Connection
Read DUT NV data to Mirror
Execute test procedure
Calc. parameters
Store parameters to DLL
Test procedure completed
NO
YES
Store NV mirror to DUT

3 Communication

3.1 DWD_set_comport

Function name:	Bool DWD_set_com_port(unsigned char com_port, int *handle)
Parameters:	Com_port: The comport number
Returns:	Result of the operation: *OK or Error*
Description	This function is used to set the COM port connected to the DUT and here after this will be used when communicating with the target. Every time the COM port change this function must be called. If a COM port is already active when the function is called, it is automatically terminated before the new COM port is set.
Platform:	All platform
Mode:	N/A

3.2 DWD_close_comport

Function name:	Bool DWD_close_com_port(int handle);
Parameters:	None
Returns:	Result of the operation: o*k or error*
Description	This function is used to close the active COM port.
Platform	All platform
Mode:	N/A

3.3 DWD_check_comm_link

Function name:	Bool DWD_check_comm_link(int handle);
Parameters:	None
Returns:	Result of the operation: *ok or error*
Description	This function is used to setup the communication to the DUT before starting the actual test and must be called initially every time the DUT is powered up.
Platform	All platform
Mode:	N/A

3.4 DWD_set_baud_rate

Function name:	Bool DWD_set_baud_rate(int baud, int handle)
Parameters:	baud: baud rate of COM port. Valid values are 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200, 230400,460800, 921600
Returns:	Result of the operation: *ok or error*
Description	Changes the default baud rate of a COM port from 115200 to the given rate. **Please note the baud rate depends on the target software in the DUT. Not all baud rate's are supported in all platform.**

Platform	All Platform
Mode:	N/A

3.5 DWD_set_DTR

Function name:	Bool DWD_set_DTR(bool value, int handle)
Parameters:	Value: true or false
Returns:	Result of the operation: *ok or error*
Description	This function is used to set DTR true or false, by default is DTR true.
Platform	All Platform
Mode:	N/A

3.6 DWD_set_RTS

Function name:	Bool DWD_set_RTS(bool value, int handle)
Parameters:	Value: true or false
Returns:	Result of the operation: *ok or error*
Description	This function is used to set RTS true or false, by default is RTS false
Platform	All Platform
Mode:	N/A

3.7 DWD_set_v24_mode

Function name:	Bool DWD_set_v24_mode(dwd_v24_mode_type mode, int handle)
Parameters:	Mode: can be set to one of the values in the typedef enum called dwd_v24_mode_type. typedef enum{ dwd_at_hash_on, dwd_at_hash_off }dwd_v24_mode_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to enable or disable communication with AT# commands when the DUT are powered up in normal mode.
Platform	All Platform
Mode:	N/A

3.8 DWD_check_CTS

Function name:	Bool DWD_check_CTS(int handle)
Parameters:	None
Returns:	Result of the operation: *ok or error*
Description	This function is used to check the CTS signal in the open comport.
Platform	All Platform
Mode	N/A

3.9 DWD_set_ms_baudrate

Function name:	Bool DWD_set_ms_baudrate(**dwd_baud_rate_type** baudrate, int handle)
Parameters:	Baudrate: the requested baudrate. typedef enum { DWD_BAUDRATE_1200, DWD_BAUDRATE_2400, DWD_BAUDRATE_4800, DWD_BAUDRATE_9600, DWD_BAUDRATE_19200, DWD_BAUDRATE_38400, DWD_BAUDRATE_57600, DWD_BAUDRATE_115200, DWD_BAUDRATE_230400, DWD_BAUDRATE_460800, DWD_BAUDRATE_921600 }**dwd_baud_rate_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the baud rate on the target side.
Platform	P2002, P2002+, MP1
Mode:	N/A
Atcptest:	atctst_change_baud_rate_req

4 Configuration of test modes

4.1 DWD_set_test_mode

Function name:	Bool DWD_set_test_mode(unsigned int16 mode, unsigned int16 arfcn, int handle);

Parameters:	Mode: *dwd_in_line_mode* (ATE#2 or Non-signaling test mode), *dwd_calib_mode*(ATE#3 or Signaling test mode) or dwd_mmi_mode(ATE#4).
	Arfcn: The channel number is only valid in the case dwd_calib_mode is selected and states the BCCH channel target for DUT synchronization.
	
	enum dwd_test_modes
	{
	dwd_in_line_mode, // gsm
	dwd_calib_mode, // gsm
	dwd_mmi_mode
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to set up the test mode of the DUT prior to start of test. IMORTANT NOTES: • When the test mode first has been selected switching of the DUT and repeating the procedure can only change it. In case the signaling mode is selected, the BCCH channel generated by the SS must be set up before giving this command. Special means in the DUT has been implemented to shorten the synchronization time and thereby decreasing the total test time. **Please note: dwd_inline_mode and dwd_calib_mode is only valid for RF GSM /GPRS / EGPRS calibration and validation. For UMTS calibration please referrer to chapter 4.2**
Platform	All Platform
Mode:	Default test mode
Atcptest:	atctst_inline_test_mode_req, atctst_calib_test_mode_req, atctst_mmi_test_mode_req

4.2 DWD_set_umts_fdd_test_mode

Function name:	Bool DWD_set_umts_fdd_test_mode(unsigned int16 mode, unsigned int16 uarfcn, unsigned int16 primary_sc, int handle)
Parameters:	Mode: *dwd_umts_fdd_non_sig_mode* (ATE#2), *dwd_umts_fdd_sig_mode*(ATE#3). uarfcn: The uarfcn states the channel configured on the System Simulator. This is given in order to speed up the synchronization process. The parameter is only valid in case of *dwd_umts_fdd_sig_mode*. Otherwise the parameter is don't care. primary_sc: The primary scrambling code gives the primary scrambling code configured on the System Simulator. This speeds up the synchronization process. The parameter is only valid in case of *dwd_umts_fdd_sig_mode*. Otherwise the parameter is don't care. enum dwd_umts_fdd_test_mode {

	dwd_umts_fdd_non_sig_mode, dwd_umts_fdd_sig_mode };
Returns:	Result of the operation: *ok or error*
Description	This function is used to set up the test mode of the DUT prior to start of test. IMORTANT NOTES: • When the test mode first has been selected switching of the DUT and repeating the procedure can only change it. In case the signaling mode is selected, the BCCH carrier generated by the SS must be set up before giving this command. Special means in the DUT has been implemented to shorten the synchronization time and thereby decreasing the total test time.
Platform	MP1(U)
Mode:	Default test mode
Atcptest:	atctst_set_umts_fdd_test_mode_req

5 RF

5.1 GSM / GPRS / EDGE

5.1.1 Signaling test mode

Before using any of the special function for signaling test mode, the DUT must have been set into Signaling test mode. See more description in chapter 4.1 on how to do that.

The calibration and test is carried out in real mode during call. Setting up an emergency call enters the traffic channel. To shorten the call setup time it is recommended that *very early assignment* of the traffic channel is used (no SDCCH channel needed).

5.1.1.1 Synchronization to BCCH

Synchronization to BCCH carrier is started by the DWD_set_test_mode command as described in chapter 4.1.

5.1.1.2 DWD_dut_idle

Function name:	Bool DWD_dut_in_idle(bool *in_idle, int handle)
Parameters:	Idle_mode: Reference to result. If TRUE the DUT is in IDLE mode.
Returns:	Result of the operation: *ok or error*
Description	This function is used to derive if the DUT has synchronized to the arfcn given in DWD_set_test_mode. The function is polled until the DUT is in IDLE or timeout.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_poll_for_idle_mode_req

5.1.1.3 DWD_clear_rxlev

Function name:	Bool DWD_clear_rxlev(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function restarts the collection of the RXLEV.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_clear_rxlev_req

5.1.1.4 DWD_get_rxlev

Function name:	Bool DWD_get_rxlev(signed char *rxlev_ptr, int handle)
Parameters:	*rxlev_ptr: Reference to the fetched RX level.
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the RX level averaged over the period since DWD_clear_rxlev was called. The RX level returned by this function is only valid during call.

Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_get_rxlev_req

5.1.1.5 DWD_set_pa_offset

Function name:	Bool DWD_set_pa_offset(unsigned char band, unsigned int16 txpwr, signed int16 offset, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* txpwr: Actual power level. Offset: value to be added to the Dac16 value (final value in the up ramp) of the power ramp for the actual power level
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the Dac16 offset (offset to be added to the final value of the up ramp) of the power ramp during TX calibration in order to iteratively to find the right calibration offset. NOTE: The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_adjust_pa_offset_req

5.1.1.6 DWD_set_pa_timing

Function name:	Bool DWD_set_pa_timing_offset(unsigned char band, unsigned int16 txpwr, signed char rampup_offset, signed char rampdown_offset, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* txpwr: Actual power level. Rampup_offset: Specifies the time shift of the up ramp. Resolution in ½ microseconds. Ramp down: Specifies the time shift of the down ramp. Resolution in ½ microseconds.
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the position of the up ramp and the down ramp of the TX burst for a specific power level in order find the right calibration offsets iteratively. NOTE: The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_adjust_pa_timing_req

5.1.1.7 DWD_set_rxlev_ch_comp_offset

Function name:	Bool DWD_set_rxlev_ch_comp_offset(unsigned char band, **dwd_ch_comp_rxlev_offset_table_type** *rxlev_ch_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Rxlev_ch_offsets_ptr: Reference to rxlev CH comp offsets for specified band. typedef signed char **dwd_ch_comp_rxlev_offset_table_type[8].**
Returns:	Result of the operation: *ok or error*

Description	This function sets the calculated *rxlev_ch_comp_offset.* NOTE: The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_set_rxlev_ch_comp_req

5.1.1.8 DWD_set_vhome_offset

Function name:	Bool DWD_set_vhome_offset(unsigned char band, unsigned int16 txpwr, unsigned char offset, int handle);
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* txpwr: Actual power level. Offset: value to be added to the *VHOME* value (first value in the up ramp) of the power ramp for the actual power level
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the *VHOME* offset (first value in the up ramp) of the power ramp during TX calibration in order to iteratively find the right calibration offset. NOTE: The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, BP2
Mode:	Signalling test mode
Atcptest:	atctst_adjust_vhome_offset_req

5.1.1.9 DWD_set_edge_pa_ch_comp

Function name:	bool DWD_set_edge_pa_ch_comp(unsigned char band, unsigned int16 txpwr, **dwd_pa_ch_comp_elm_type** *offset, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* txpwr: Actual TX power. Offset: References to a table holding the channel compensation for the actual power level. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**;
Returns:	Result of the operation: *ok or error*
Description	This function set the table of channel compensation for a given power level for a given band. NOTE: The PA CH COMP is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	MP1(E)
Mode:	Signalling test mode
Atcptest:	atctst_set_edge_pa_ch_comp_req

5.1.1.10 DWD_set_gmsk_pa_ch_comp

Function name:	DWD_set_gmsk_pa_ch_comp(unsigned char band, unsigned int16 txpwr, **dwd_pa_ch_comp_elm_type** *offset, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* TXPWR: Actual TX power. OFFSET: References to a table holding the channel compensation for the actual power level. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**;
Returns:	Result of the operation: *ok or error*
Description	This function set the table of channel compensation for a given power level for a given band. NOTE: The PA COMP is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	MP1
Mode:	Signalling test mode
Atcptest:	atctst_set_gmsk_pa_ch_comp_req

5.1.1.11 DWD_set_max_pa_ch_comp

Function name:	Bool DWD_set_max_pa_ch_comp(unsigned char band, unsigned char ch_high_low, signed char offset, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* ch_high_low: *dwd_low_ch* for the low range of channels of the specified band and *dwd_high_ch* for the high range of channels of the specified band. Offsets: Offset to be added to the DAC16 value for the max power for the given band.
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the channel compensation offset (offset to be added to the final value of the up ramp) of the power ramp during TX calibration in order to iteratively to find the right channel compensation calibration offset. NOTE: The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_set_max_pa_ch_comp_offset_req

5.1.1.12 DWD_setup_emergency_call

Function name:	Bool DWD_setup_emergency_call(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function commands the DUT to initiate emergency call and return *ok* if the

	emergency call has been initiated. If the call is successful or failed must it be determined by the SS.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_setup_call_req

5.1.1.13 DWD_treat_1800_as_1900

Function name:	DWD_treat_1800_as_1900(unsigned int16 mode, int handle)

Parameters:	Mode: The mode can be one of the following: *dwd_treat_1800_as_1800* or *dwd_treat_1800_as_1900*.
	enum
	{
	dwd_treat_1800_as_1800 = 0,
	dwd_treat_1800_as_1900 = 1
	};

Returns:	Result of the operation: *ok or error*
Description	**This function is applicable for tri band or quad band mobiles**. Since the MS cannot tell the difference between DCS1800 and PCS1900 due to the overlapping channel numbering scheme this function is called to configure how the MS should treat the channels. Default *mode* will be *dwd_treat_1800_as_1800*. If a handover\assignment to PCS1900 should be executed the mode should be set to *dwd_treat_1800_as_1900*. Prerequisites are: 1. A Circuit Switched call has been initiated 2. The current band is GSM900 The HW is tri- or quad band enabled
Platform	P2002, P2002+, MP1, BP2
Mode:	Signalling test mode
Atcptest:	atctst_treat_1800_as_1900_req

5.1.1.14 DWD_sig_poll_gprs_attached

Function name:	DWD_sig_poll_gprs_attached(unsigned char *status, int handle)

Parameters:	Status: Reference fetched the status value.
	
	enum
	{
	dwd_sig_gprs_not_attached,
	dwd_sig_gprs_attached,
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to poll for GPRS attach
Platform	P2002, P2002+
Mode:	Signalling test mode
Atcptest:	atctst_sig_poll_gprs_attached_req

5.1.1.15 DWD_sig_gprs_attach_req

Function name:	DWD_sig_gprs_attach_req(int handle)
Parameters:	None
Returns:	Result of the operation: *ok or error*
Description	This function is used to attach the DUT to GPRS
Platform	P2002, P2002+
Mode:	Signalling test mode
Atcptest:	atctst_sig_gprs_attach_req

5.1.1.16 DWD_xbandselect

Function name:	Bool DWD_xbandselect(unsigned int16 band1, unsigned int16 band2, unsigned int16 *status, int handle)
Parameters:	Band1 & band2 is those bands you want to use ex. 900, 1800. *status: Reference with result
Returns:	Result of the operation: *ok or error*
Description	This function is used to change band in signaling test mode. **The function is not verified yet and the target is not fully implemented.**
Platform	MP1
Mode:	Signalling test mode
Atcptest:	atctst_bandselect_req

5.1.1.17 DWD_bandquery

Function name:	Bool DWD_bandquery(unsigned int32 *band_list, int handle)
Parameters:	*band_list: Reference fetched the list of selected bands.
Returns:	Result of the operation: *ok or error*
Description	This function is used to query for selected bands. **The function is not verified yet and the target is not fully implemented.**
Platform	MP1
Mode:	Signalling test mode
Atcptest:	atctst_bandquery_req

5.1.2 Non-Signaling test mode

Before using any of the special function for non-signaling test, the DUT must have been set into Non-signaling test mode. See chapter 4.1 how to set the DUT into this mode.

5.1.2.1 DWD_change_pa_level

Function name:	Bool DWD_change_pa_level(unsigned char txpwr, unsigned char txpwr2, int handle)
Parameters:	Txpwr: Wanted power level Txpwr2: Wanted power level if two TX timeslot is used.
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the power level in *tx_burst* mode. The default power

	level in *tx_burst* mode is the max power for the corresponding band. **This command will have no effect in *tx_cont* mode, where the minimum power level is always used.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_change_pa_level_req

5.1.2.2 DWD_change_pa_level_v2

Function name:	Bool DWD_change_pa_level_v2(unsigned char txpwr1, unsigned char txpwr2, unsigned char txpwr3, unsigned txpwr4, int handle)
Parameters:	Txpwr1: Power level for TX timeslot 1 Txpwr2: Power level for second TX timeslot. Txpwr3: Power level for third TX timeslot. Txpwr4: Power level for fourth TX timeslot.
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the power level in *tx_burst* mode. The default power level in *tx_burst* mode is the max power for the corresponding band. **This command will have no effect in *tx_cont* mode, where the minimum power level is always used.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Non-Signalling test mode
Atcptest:	atctst_change_pa_level_req

5.1.2.3 DWD_get_iqrms

Function name:	bool DWD_get_iqrms(unsigned int16 *iqrms, int handle)
Parameters:	*iqrms: Reference to fetched IQRMS value
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the measured IQRMS value when the *rf mode* is *rx_burst*, *rx_cont* or *mon_burst*.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_get_iqrms_req

5.1.2.4 DWD_set_iqrms_sample_number

Function name:	bool DWD_set_iqrms_sample_number(int16 nof_samples, int handle)
Parameters:	Nof_samples: fetched the numbers of samples value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set multiple samples of MON IQRMS value in the DUT
Platform	BP30
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_iqrms_sample_no_req.

5.1.2.5 DWD_set_gmsk_mode

Function name:	Bool DWD_set_gmsk_mode(unsigned int16 mode, unsigned int16 tsc, int handle);

Parameters:	Mode: Bit content of the TX burst: *dwd_high*, *dwd_random* or *dwd_normal*, *dwd_low*. (For EDGE is the value dwd_edge_high, dwd_edge_random, dwd_edge_normal, dwd_edge_low)
	Tsc: Training sequence to use in case of mode is set to *dwd_normal*.
	
	enum dwd_gmsk_modes
	{
	dwd_high /* all ones*/,
	dwd_random /* all random*/,
	dwd_normal /*normal burst including training sequence*/,
	dwd_edge_high,
	dwd_edge_random,
	dwd_edge_normal,
	dwd_edge_low,
	dwd_low
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to select the content of the TX bursts. *Dwd_high* is the default setting.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_gmsk_mode_req

5.1.2.6 DWD_set_rf_channel

Function name:	Bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle)

Parameters:	Arfcn_tp: *dwd_rx_arfcn_tp*, *dwd_tx_arfcn_tp* and *dwd_mon_arfcn_tp* indicate what the channel is used to.
	Arfcn: Indicates the actual channel number.
	
	enum dwd_arfcn_tps
	{
	dwd_rx_arfcn_tp,
	dwd_tx_arfcn_tp,
	dwd_mon_arfcn_tp
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to set the channel number for *rx*, *tx* and *mon* in the Non-signaling test mode. The band is determined from the ARFCN.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_rf_channel_req

5.1.2.7 DWD_set_rf_mode

Function name:	Bool DWD_set_rf_mode(unsigned int16 mode, int handle)
Parameters:	Mode: Could be one of the mode from enum *dwd_rf_modes*

	enum dwd_rf_modes
	{
	dwd_rx_cont,
	dwd_tx_cont,
	dwd_rx_burst,
	dwd_tx_burst,
	dwd_mon_burst,
	dwd_rxtxmon,
	dwd_stop,
	dwd_tx_calib_gmsk,
	dwd_tx_calib_edge,
	dwd_rx_calib
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to configure the various RF modes needed to support Non-signaling test mode. When the mode is changed, the old mode is automatically terminated. The default mode is OFF /**dwd_stop*/ **REMARK!!! Not all modes are supported in all platforms.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_rf_mode_req

5.1.2.8 DWD_set_rf_gain

Function name:	Bool DWD_set_rf_gain(unsigned int16 gain_tp, signed char gain, int handle)
Parameters:	Gain_tp: *dwd_rx_gain_tp*, and *dwd_mon_gain_tp* indicates what gain is used for. Gain: Actual gain to use. The range is as defined in the RF SW interface specs. See Other document for more detail.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the gain *rx* and *mon* in the Non-signaling test mode. **The requested gain is only used if the rf mode is set to *dwd_rx_cont* or *dwd_rx_burst*.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_rf_gain_req

5.1.2.9 DWD_set_rf_gain_v2

Function name:	Bool DWD_set_rf_gain_v2(unsigned int16 gain_tp, signed int16 gain, int handle)
Parameters:	Gain_tp: *dwd_rx_gain_tp*, and *dwd_mon_gain_tp* indicates what gain is used for. Gain: Actual gain to use. The range is as defined in the RF SW interface specs. See Other document for more detail.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the gain *rx* and *mon* in the Non-signaling test mode. **The requested gain is only used if the rf mode is set to *dwd_rx_cont* or *dwd_rx_burst*.** **This function must be used when the gain setting value use more than 8 bit in the**

	telegram, Smarti PM is one example.
Platform	MP1
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_rf_gain_req

5.1.2.10 DWD_non_sig_set_calib_parm

Function name:	Bool DWD_non_sig_set_calib_parm(unsigned char setting, int handle)

Parameters:	Setting: Should be set to one of the dwd_inline_opcode
	
	enum dwd_inline_opcode
	{
	dwd_non_sig_dont_use_calib_parm,
	dwd_non_sig_use_calib_parm
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to enable or disable PA Offset calibration parameters.
Platform	P2002, P2002+, MP1
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_generic_req

5.1.3 Temperature calibration

5.1.3.1 DWD_get_rf_temp

Function name:	Bool DWD_get_rf_temp(signed char *rf_temp, int handle)
Parameters:	Rf_temp: reference to fetched RF temperature
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the actual RF temperature.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode & Signalling test mode
Atcptest:	atctst_rf_temp_req

5.1.3.2 DWD_set_pmb_temp_offset

Function name:	Bool DWD_set_pmb_temp_offset(signed int16 rf_temp_offset, int handle)
Parameters:	Rf_temp_offset: Reference to fetched rf_temp_offset.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the actual *pmb temperature compensation offset* in the DUT, which is stored in RAM. The offset is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode & Signalling test mode
Atcptest:	atctst_set_pmb_temp_offset_req

5.1.4 AFC Calibration

5.1.4.1 DWD_set_afc

Function name:	Bool DWD_set_afc(unsigned int16 dac_value, int handle)
Parameters:	Dac_value: The resolution of the AFC DAC is 11 bit giving the range 0 to 2047.
Returns:	Result of the operation: *ok or error*
Description	This function is used when deriving AFC calibration values (*default_dac* and *step_size_in_hz*). Note: The value is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, MP1
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_afc_req

5.1.4.2 DWD_set_afc_v2

Function name:	Bool DWD_set_afc_v2(unsigned int16 dac_xocal, unsigned int16 dac_xotune, int handle)
Parameters:	Dac_xocal: 3 bit xocal value [0…7] Dac_xotune: 13 bit xotune value [0…8191]
Returns:	Result of the operation: *ok or error*
Description	Note: The value is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_set_afc_req

5.1.5 Additional functions

5.1.5.1 DWD_set_power_saving

Function name:	Bool DWD_set_power_saving(bool on, int handle)
Parameters:	On: Set the power saving to on or Off
Returns:	Result of the operation: *ok or error*
Description	N/A
Platform	P2002, P2002+, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_set_power_saving_req

5.1.5.2 DWD_get_pa_dac16_range

Function name:	Bool DWD_get_pa_dac16_range(unsigned int16 band, unsigned char high_txpwr, unsigned char low_txpwr, unsigned int16 *result, unsigned int16 *data, int handle)
Parameters:	Band: *dwd_gsm_900*, *dwd_dcs_1800*, *dwd_pcs_1900*, *dwd_gsm_850*. high_txpwr: Highest power level how_txpwr: Lowest power level

	*result: *dwd_error* or *dwd_ok*. *data: Reference to where the requested dac16 values should place.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read the default dac16 (final ramp values) from Non-Volatile memory of the DUT for all the power levels in the requested range *high_txpwr* to *low_txpwr*. These are needed by the calibration routines. As an example: for GSM power class *4 high_txpwr* is 5 and *low_txpwr* is 19. This function substitutes DWD_get_default_pa_dac16, which is now removed from the DLL.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_get_pa_dac16_range_req

5.1.5.3 DWD_get_edge_pa_dac16_range

Function name:	Bool DWD_get_edge_pa_dac16_range(unsigned int16 band, unsigned char high_txpwr, unsigned char low_txpwr, unsigned int16 *result, unsigned int16 *data, int handle)
Parameters:	Band: *dwd_gsm_900*, *dwd_dcs_1800*, *dwd_pcs_1900*, *dwd_gsm_850*. high_txpwr: Highest power level how_txpwr: Lowest power level *result: *dwd_error* or *dwd_ok*. *data: Reference to where the requested dac16 values should place.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read the default dac16 (final ramp values) from Non-Volatile memory of the DUT for all the power levels in the requested range *high_txpwr* to *low_txpwr*. These are needed by the calibration routines. As an example: for GSM power class *4 high_txpwr* is 5 and *low_txpwr* is 19.
Platform	MP1(E)
Mode:	Default test mode
Atcptest:	atctst_get_edge_pa_dac16_range_req

5.1.5.4 DWD_get_max_pa_offsets

Function name:	Bool DWD_get_max_pa_offsets(unsigned char band, unsigned int16 *max_pa_levels_ptr, int handle)
Parameters:	Band: Reference fetched the band *dwd_gsm_900*, *dwd_dcs_1800*, *dwd_pcs_1900*, *dwd_gsm_850*. *Max_pa_levels_ptr: Reference fetched the max PA offsets value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the max PA offsets for a specified band.
Platform	P2002, P2002+, BP2
Mode:	Non-Signalling test mode & Signalling test mode
Atcptest:	atctst_get_max_pa_offsets_req

5.1.5.5 DWD_set_serial_interface_mode

Function name:	Bool DWD_set_serial_interface_mode(**dwd_mode_type** mode, int handle)
Parameters:	Mode: Serial interface mode. typedef enum { dwd_normal_mode, dwd_not_applicable, dwd_ptest_mode }**dwd_mode_type**;
Returns:	Result of operation: *ok* or *error*
Description:	This function is used to set the serial interface mode startup condition in the static NV: *dwd_normal_mode*: When this mode is set in Static NV the MS powers up in *normal mode*. This mode must be set when the MS is leaving the factory. In this mode 'AT#' can not be received and the dwdio.dll communication will not work. *dwd_ptest_mode*: When this mode is set in Static NV the MS powers up in *test mode*. This is the mode which must be set during ATE#2, ATE#3, ATE#4 testing and for the repair station. This mode is usually also set in the download station ATE#1
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

5.1.5.6 DWD_get_sw_version

Function name:	Bool DWD_get_sw_version(char *sw_version, int handle)
Parameters:	sw_version: Reference to a string holding the SW version
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the SW version of the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_get_sw_version_req

5.1.5.7 DWD_rf_bands_supported

Function name:	Bool DWD_rf_bands_supported(unsigned char *band, int handle)

Parameters:	Band: Reference to which band is supported.
	enum dwd_rf_bands_support
	{
	dwd_900_1800 = 0,
	dwd_850_1900,
	dwd_900_1800_1900,
	dwd_850_900_1800_1900
	};

Returns:	Result of the operation: *ok or error*
Description	This function fetches the supported RF band in the target hardware. The tester for automatically determination of which bands to calibrate can use this.
Platform	P2002, P2002+, MP1, BP2, ULC

Mode:	Default test mode
Atcptest:	atctst_rf_bands_supported_req

5.1.5.8 DWD_sw_reset

Function name:	DWD_sw_reset(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to reset the MS in the production test. This is very useful if Non-signaling test station and Signaling test station are combined. If *supply voltage* and *charger voltage* is present the mobile will restart automatically in production test mode when calling *DWD_sw_reset.* In order to open the connection to the MS *DWD_check_comm_link* must be called. After this, the test mode *(dwd_in_line_mode (ATE#2), dwd_calib_mode(ATE#3) or dwd_mmi_mode)* must be configured. **REMARK!!! For projects using SM-POWER should supply voltage just be present to restart automatically in production test mode when calling DWD_sw_reset.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_sw_reset_req

5.1.5.9 DWD_get_edge_power_ramp

Function name:	Bool DWD_get_edge_power_ramp(unsigned char band, unsigned int16 pa_level, unsigned int16 *ramp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900*, *dwd_dcs_1800*, *dwd_gsm850* or *dwd_pcs_1900*. Pa_level: Reference to selected power level. *ramp_ptr: Reference to fetched power ramp.
Returns:	Result of the operation: *ok or error*
Description	This function gets the EDGE power ramp values for a given band and power level.
Platform	MP1
Mode:	Non-Signalling test mode, Signalling test mode.
Atcptest:	atctst_get_power_ramp_edge_req

5.1.5.10 DWD_get_power_ramp

Function name:	Bool DWD_get_power_ramp(unsigned char band, unsigned int16 pa_level, unsigned int16 *ramp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900*, *dwd_dcs_1800*, *dwd_gsm850* or *dwd_pcs_1900*. Pa_level: Reference to selected power level. *ramp_ptr: Reference to fetched power ramp.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the power ramp values for a given band and power level.
Platform	P2002, P2002+, MP1, BP2, ULC

Mode:	Non-Signalling test mode, Signalling test mode
Atcptest:	atctst_get_power_ramp_req

5.1.5.11 DWD_inl_get_mult_rx_iqrms

Function name:	Bool DWD_inl_get_mult_rx_iqrms(unsigned int16 *iqrms, int handle)
Parameters:	*iqrms: Reference to fetched IQRMS value.
Returns:	Result of the operation: *ok or error*
Description	This function gets the measured IQRMS value for 1…4 RX Slot when the *rf_mode is set to rx_burst.*
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_get_mult_rx_iqrms_req

5.1.5.12 DWD_inl_set_nof_rx_slot

Function name:	Bool DWD_inl_set_nof_rx_slot(unsigned char nof_rx_slot, int handle)
Parameters:	Nof_rx_slot: Set the number of used RX slot (1..4)
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the number of RX timeslot. Default RX slot is 1 and the maximum is 4.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_set_nof_rx_slot_req

5.1.5.13 DWD_inl_set_nof_slot

Function name:	Bool DWD_inl_set_nof_slot(unsigned char nof_tx_slot, int handle)
Parameters:	Nof_tx_slot: Set the number of TX slot (1 or 2).
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the number of TX timeslot. Default TX slot is 1 and maximum is 2.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_set_nof_slot_req

5.1.5.14 DWD_inl_set_power_ramp_mode

Function name:	Bool DWD_inl_set_power_ramp_mode(unsigned char mode, int handle);

Parameters:	Mode: *dwd_external_pa_ramp* or *dwd_internal_pa_ramp*.
	
	enum dwd_ramp_mode
	{
	dwd_external_pa_ramp,
	dwd_internal_pa_ramp,
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to set the power ramp mode (external or internal mode).

Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_set_power_ramp_mode_req

5.1.5.15 DWD_inl_set_power_ramps

Function name:	Bool DWD_inl_set_power_ramps(unsigned int16 *power_ramp1_ptr, unsigned int16 *power_ramp2_ptr, int handle)
Parameters:	*Power_ramp_ptr1 & 2: Reference to fetched power ramp.
Returns:	Result of the operation: *ok or error*
Description	This function is used set the power ramp in the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_inl_set_power_ramps_req

5.1.5.16 DWD_get_rxlev_gain_offset

Function name:	Bool DWD_get_rxlev_gain_offset(unsigned char band, **dwd_rxlev_offset_table_type** *rxlev_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900*,*dwd_dcs_1800*,*dwd_gsm_850* or *dwd_pcs_1900*. Rxlev_offsets_ptr: Reference to rxlev liniarization offsets fetched for the specified band. typedef signed char dwd_rxlev_offset_table_type[70];
Returns:	Result of the operation: *ok or error*
Description	This function fetches for the specified band the *rxlev gain offsets* stored in EEPROM
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode, Signalling test mode.

5.1.5.17 DWD_get_rxlev_gain_offset_v2

Function name:	bool CALL_CONVENTION DWD_get_rxlev_gain_offset_v2(unsigned char band, **dwd_rxlev_offset_table_v2_type** *rxlev_offsets_ptr, int handle);
Parameters:	Band: *dwd_gsm_900*,*dwd_dcs_1800*,*dwd_gsm_850* or *dwd_pcs_1900*. Rxlev_offsets_ptr: Reference to rxlev liniarization offsets fetched for the specified band. typedef signed char dwd_rxlev_offset_table_v2_type[5];
Returns:	Result of the operation: *ok or error*
Description	This function fetches for the specified band the *rxlev gain offsets* stored in EEPROM
Platform	MPE, MPEU
Mode:	Non-Signalling test mode, Signalling test mode.

5.1.5.18 DWD_set_rxlev_gain_offset

Function name:	Bool DWD_set_rxlev_gain_offset(unsigned char band, **dwd_rxlev_offset_table_type** *rxlev_offsets_ptr, int handle)

Parameters:	Band: *dwd_gsm_900,dwd_dcs_1800,dwd_gsm_850 or dwd_pcs_1900.* Rxlev_offsets_ptr: Reference to rxlev linearization offsets fetched for the specified band. typedef signed char dwd_rxlev_offset_table_type[70];
Returns:	Result of the operation: *ok or error*
Description	This function sets the calculated *rxlev_gain_offset* directly in the DUT, which then uses the parameters immediately. This is possible since the offsets are in stored in RAM. Offset must be given for all RX LEV indexes from 0 to 69.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Signalling test mode
Atcptest:	atctst_set_rxlev_gain_req

5.1.5.19 DWD_set_rxlev_gain_offset_v2

Function name:	Bool DWD_set_rxlev_gain_offset_v2(unsigned char band, **dwd_rxlev_offset_table_v2_type** *rxlev_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900, dwd_dcs_1800, dwd_gsm_850 or dwd_pcs_1900.* Rxlev_offsets_ptr: Reference to rxlev linearization offsets fetched for the specified band. typedef signed char dwd_rxlev_offset_table_v2_type[5];
Returns:	Result of the operation: *ok or error*
Description	This function sets the calculated *rxlev_gain_offset* directly in the DUT, which then uses the parameters immediately. This is possible since the offsets are in stored in RAM. Offset must be given for all RX LEV indexes from 0 to 69.
Platform	MPE, MPEU
Mode:	Signalling test mode
Atcptest:	atctst_set_rxlev_gain_req

5.1.5.20 DWD_store_rxlev_gain_offset

Function name:	Bool DWD_store_rxlev_gain_offset(unsigned char band, **dwd_rxlev_offset_table_type** *rxlev_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900,dwd_dcs_1800,dwd_gsm_850 or dwd_pcs_1900.* Rxlev_offsets_ptr: Reference to rxlev linearization offsets fetched for the specified band. typedef signed char dwd_rxlev_offset_table_type[70];
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived RX LEV offset for the specified band.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Non-Signalling test mode, Signalling test mode

5.1.5.21 DWD_store_rxlev_gain_offset_v2

Function name:	Bool DWD_store_rxlev_gain_offset_v2(unsigned char band, **dwd_rxlev_offset_table_v2_type** *rxlev_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900 ,dwd_dcs_1800, dwd_gsm_850 or dwd_pcs_1900.* Rxlev_offsets_ptr: Reference to rxlev linearization offsets fetched for the specified

	band. typedef signed char dwd_rxlev_offset_table_v2_type[5];
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived RX LEV offset for the specified band.
Platform	MPE, MPEU
Mode:	Non-Signalling test mode, Signalling test mode

5.1.5.22 DWD_get_edge_txcorr

Function name:	bool DWD_get_edge_txcorr(unsigned char band, **dwd_edge_txcorr_type** *txcorr,int handle)
Parameters:	Band: *dwd_gsm_900 , dwd_dcs_1800, dwd_gsm_850 or dwd_pcs_1900.* txcorr: Reference to txcorr parameter for the specified band. typedef signed char dwd_edge_txcorr_type[32];
Returns:	Result of the operation: *ok or error*
Description	This function gets the txcorr parameters for the specified band stored in the EEPROM.
Platform	MPE, MPEU
Mode:	Non-Signalling test mode, Signalling test mode

5.1.5.23 DWD_store_edge_txcorr

Function name:	bool DWD_store_edge_txcorr(unsigned char band, dwd_edge_txcorr_type *txcorr, int handle)
Parameters:	Band: *dwd_gsm_900 , dwd_dcs_1800, dwd_gsm_850 or dwd_pcs_1900.* txcorr: Reference to txcorr parameter for the specified band. typedef signed char dwd_edge_txcorr_type[32];
Returns:	Result of the operation: *ok or error*
Description	This function stores the txcorr parameters for the specified band.
Platform	MPE, MPEU
Mode:	Non-Signalling test mode, Signalling test mode

5.1.5.24 DWD_get_product_name

Function name:	Bool DWD_get_product_name(char *product_name, int handle);
Parameters:	Product_name: Reference to a string fetched the product name
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the product_name from the DUT.
Platform	BP30
Mode:	Default test mode
Atcptest:	atctst_get_product_name_req

5.1.6 OneSecondCalibration

5.1.6.1 DWD_osc_set_tx_calib_channel

Function name:	Bool DWD_osc_set_tx_calib_channel(unsigned int16 nof_ch, **dwd_tx_calib_ch_type** *ch, int handle)
Parameters:	Nof_ch: Should indication how many channel there are set. *ch: Pointer holding all the channel numbers. typedef struct { unsigned int16 channel_no[50]; }dwd_tx_calib_ch_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to setup those channels there should be used by OneSecondCalibration in the DUT. The way to set a 1900 channel number is to “or” the channel number with 0x8000 ex. ch512 | 0x8000 = 33280(decimal)
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_set_tx_calib_channel_req

5.1.6.2 DWD_osc_set_tx_calib_power_levels

Function name:	bool DWD_osc_set_tx_calib_power_levels(**dwd_tx_calib_power_level_type** *ptr, int handle);

Parameters:	*ptr: pointer to a structure holding the power levels
	
	typedef struct
	{
	unsigned char p_lowband[7]; //means 850 and 900
	unsigned char p_highband[7]; // means 1800 and 1900
	}dwd_tx_calib_power_level_type;

Returns:	Result of the operation: *ok or error*
Description	This function is used to setup those power levels there should be used by OneSecondCalibration in the DUT.
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_set_tx_calib_power_level_req

5.1.6.3 DWD_osc_set_rx_calib_channel

Function name:	bool DWD_osc_set_rx_calib_channel(unsigned int16 nof_ch, **dwd_tx_rx_calib_ch_type** *ch, unsigned int16 NofAfcDelta, unsigned int16 AfcUpdateDist, **dwd_afc_delta_type** *afc_delta, unsigned int16 FchDist, **dwd_rx_center_ch_type** *CenterCh, int handle)
Parameters:	Nof_ch: Indicate how many channel are set Ch: fetched all the channel number typedef struct {

	unsigned int16 channel_no[50]; }**dwd_tx_rx_calib_ch_type**; NofafcDelta : indicate the numbers of afc delta's AfcUpdateDist: ? *afc_delta: fetched all the afc_delta data. typedef struct { unsigned int16 afc_delta_list[10]; }**dwd_afc_delta_type**; FchDist: ? *CenterCh: Fetched all the center channels typedef struct { unsigned int16 center_ch[4]; }**dwd_rx_center_ch_type**;
Returns:	Result of the operation: *ok or error*
Description	Function used to setup RX calibration in One Second calibration concept. See the document *OneSecCalib_DWD_SW_spec_x_x* for more detail.
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_set_rx_calib_channel_req

5.1.6.4 DWD_osc_set_rx_calib_gain

Function name:	bool DWD_osc_set_rx_calib_gain(**dwd_rx_gain_setting_type** *rxgain, int handle)
Parameters:	*rxgain: fetched all the RX gain setting values. typedef struct { unsigned int16 rx_gain[7]; }**dwd_rx_gain_setting_type**;
Returns:	Result of the operation: *ok or error*
Description	Function to setup RX calibration gain values
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_set_rx_calib_gain_req

5.1.6.5 DWD_osc_get_rx_afc_calib_status

Function name:	bool DWD_osc_get_rx_afc_calib_status(unsigned int16 *status, int handle)
Parameters:	*status: fetched the status value
Returns:	Result of the operation: *ok or error*

Description	Function used to get the RX calibration status
Platform	MP!, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_get_rx_afc_calib_status_req

5.1.6.6 DWD_osc_get_rx_afc_calib_result

Function name:	bool DWD_osc_get_rx_afc_calib_result(unsigned char band, **dwd_ch_comp_rxlev_offset_table_type** *ChCompOffset,
Parameters:	Band: *dwd_gsm_900,dwd_dcs_1800,dwd_gsm_850 or dwd_pcs_1900.* *ChCompOffset: Fetched the Channel compensations value for a given band typedef signed char **dwd_ch_comp_rxlev_offset_table_type**[8];
Returns:	Result of the operation: *ok or error*
Description	Function to get the result value for RX AFC calibration
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_get_rx_afc_calib_result_req

5.1.6.7 DWD_osc_get_rx_afc_calib_data

Function name:	bool DWD_osc_get_rx_afc_calib_data(unsigned int16 channel, **dwd_rx_afc_calib_data_type** *RxMeasData, int handle)
Parameters:	Channel: Indicated the channel you want to get afc calib data *RxMeasData: Fetched all the RX measured data typedef struct { unsigned int16 rms[7]; signed int16 freqoffset; }**dwd_rx_afc_calib_data_type**;
Returns:	Result of the operation: *ok or error*
Description	Function to get RX AFC calibrated data
Platform	MP1, P2002+, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_get_rx_afc_calib_data_req

5.1.6.8 DWD_osc_set_adc_calib

Function name:	bool DWD_osc_set_adc_calib(unsigned int16 NofSamples, unsigned int16 NofVolt**,dwd_adc_calib_voltagelist_type** *VoltageList, int handle)
Parameters:	NofSamples: Nofvolt:

	Voltagelist: typedef struct { unsigned int16 voltageList[2]; }**dwd_adc_calib_voltagelist_type**;
Returns:	Result of the operation: *ok or error*
Description	Function to get set ADC calibration data **NOT IMPLEMENTED IN TARGET SOFTWARE.**
Platform	?
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_set_adc_calib_req

5.1.6.9 DWD_osc_get_adc_calib_status

Function name:	bool DWD_osc_get_adc_calib_status(unsigned int16 *status, int handle)
Parameters:	Status: fetchec the adc status.
Returns:	Result of the operation: *ok or error*
Description	Function to get the adc calibration status **NOT IMPLEMENTED IN TARGET SOFTWARE.**
Platform	?
Mode:	Non-Signalling test mode
Atcptest:	atctst_osc_get_adc_calib_status_req

5.1.6.10 DWD_osc_get_adc_calib_result

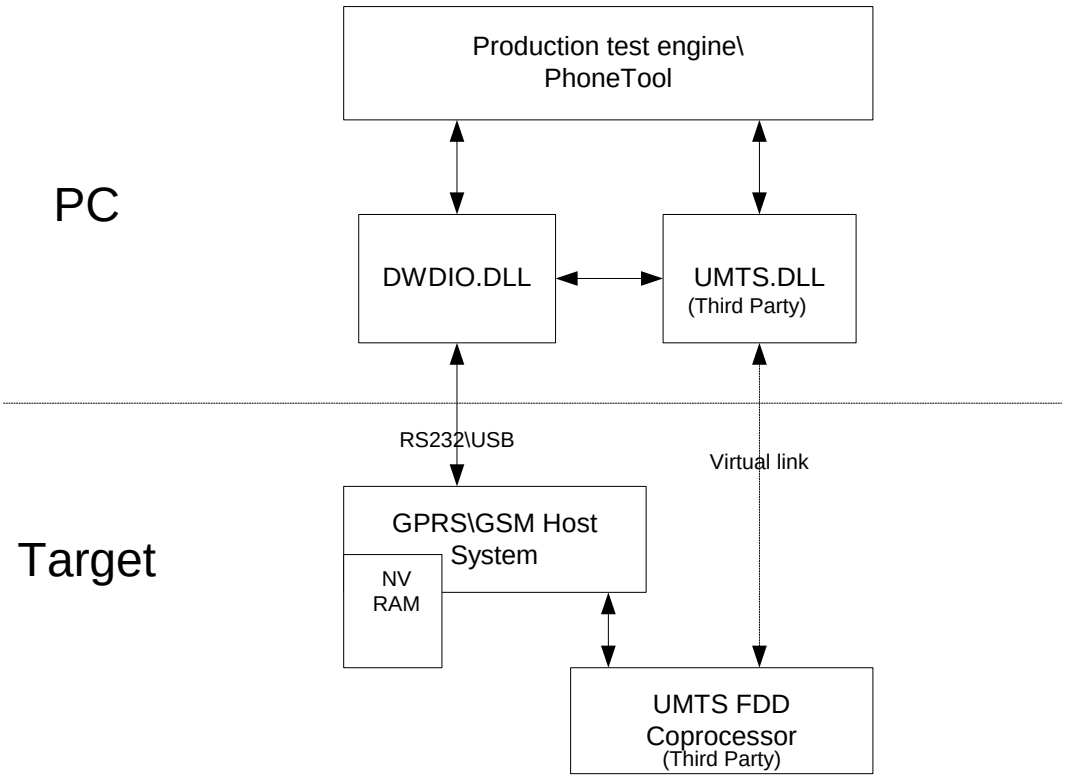
Function name:	bool DWD_osc_get_adc_calib_result(**dwd_adc_adjusted_comp_parms_type** *comp_ptr, int handle)
Parameters:	Com_ptr: fetched the ADC calib result typedef struct { signed int16 vbat_gain; signed int16 vbat_offset; signed int16 tbat_gain; signed int16 tbat_offset; signed int16 tenv_gain; signed int16 tenv_offset; signed int16 btec_gain; signed int16 btec_offset; signed int16 tvco_gain; signed int16 tvco_offset; } **dwd_adc_adjusted_comp_parms_type**;
Returns:	Result of the operation: *ok or error*
Description	Function to get the adc calibration result **NOT IMPLEMENTED IN TARGET SOFTWARE.**
Platform	?
Mode:	Non-Signalling test mode

5.1.6.11 DWD_osc_get_xocal

Function name:	DWD_osc_get_xocal(unsigned int16 *xocal, unsigned int16 *xotune, unsigned int16 *dac_step, int handle)
Parameters:	Xocal: fetched the xocal value. Xotune: fetched the xotune value, Dac_step: fetched the dac step value
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the xocal value from DUT
Platform	MP1, ULC
Mode:	Non-Signalling test mode

5.2 UMTS

Below is seen the system diagram supporting production testing of the UMTS FDD sub system:



System block diagram supporting production test of UMTS FDD sub system.

In order to support production testing of the UMTS FDD subsystem a DLL is provided by the Third party vendor of the UMTS coprocessor (in the figure named umts.dll). The UMTS.DLL provides all the

interface function to test and calibrate the UMTS part; these will be described in a separate document provided by third party.
The DWDIO.DLL provides interface function to configure the UMTS test modes and serves as transport channel for the UMTS.DLL test commands. Furthermore, the DWDIO.DLL provides to the UMTS.DLL interface functions for reading, writing and storing NV RAM parameters. As seen for the drawing the NV RAM parameters are physically placed in the GPRS\GSM host system and transferred at boot time to the UMTS sub system.

5.2.1 DWD_send_umts_fdd_coprocessor_cmd

Function name:	Bool DWD_send_umts_fdd_coprocessor_cmd(unsigned int16 nof_send_bytes, unsigned char *psend, unsigned int16 *nof_receive_bytes, unsigned char *preceive, int handle)
Parameters:	Nof_send_bytes: Numbers of bytes you want to send. *psend: Pointer to input data buffer (Max 200 bytes). *nof_send_bytes: Pointer to the numbers of received bytes *preceive: Pointer to the receive buffer (Max 200 bytes).
Returns:	Result of the operation: *ok or error*
Description	This function is used to transparently send and receive commands to and from the UMTS FDD Coprocessor via main Base Band (as SGOLD).
Platform	MP1(U)
Mode:	dwd_umts_fdd_non_sig_mode or dwd_umts_fdd_sig_mode
Atcptest:	atctst_send_umts_fdd_coprocessor_cmd_req

6 Base Band functions

6.1 ADC Calibration

6.1.1 DWD_meas_all_adc

Function name:	Bool DWD_meas_all_adc(**dwd_adc_meas_type** *meas_ptr, int handle)
Parameters:	Meas_ptr: Reference to where the ADC measurements should be delivered. typedef struct { signed int16 vbat_adc_value; /* Used for battery voltage */ signed int16 tbat_adc_value; /* Used for detection of external equipment */ signed int16 tenv_adc_value; /* Used for RF temperature */ signed int16 btec_adc_value; /* Used for battery ID */ signed int16 tvco_adc_value; } **dwd_adc_meas_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used show all there measured ADC value
Platform	P2002, P2002+, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_meas_all_adc_req

6.1.2 DWD_meas_all_adc_direct

Function name:	Bool DWD_meas_all_adc_direct(signed int16 *vbat, signed int16 *tbat, signed int16 *tenv, signed int16 *btec, signed int16 *tvco, int handle)
Parameters:	Vbat, tbat, tenv, btec, and tvco fetch the measured ADC value.
Returns:	Result of the operation: *ok or error*
Description	This function is the same as the function DWD_meas_all_adc, but it is easier to use in the Lab View program.
Platform	P2002, P2002+, BP2, ULC
Mode:	Non-Signalling test mode
Atcptest:	atctst_meas_all_adc_req

6.1.3 DWD_meas_all_adc_v2

Function name:	Bool DWD_meas_all_adc_v2(**dwd_adc_meas_v2_type** *meas_ptr, int handle)
Parameters:	Meas_ptr: Reference to where the ADC measurements should be delivered. typedef struct { signed int16 vbat_adc_value; /* Used for battery voltage */ signed int16 tbat_adc_value; /* Used for detection of external equipment */ signed int16 tenv_adc_value; /* Used for RF temperature */

	signed int16 btec_adc_value; /* Used for battery ID */ signed int16 tvco_adc_value; signed int16 curr_adc_value; } **dwd_adc_meas_v2_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used show all the measured ADC value.
Platform	MP1
Mode:	Non-Signalling test mode
Atcptest:	atctst_meas_all_adc_req

6.1.4 DWD_meas_all_adc_direct_v2

Function name:	Bool DWD_meas_all_adc_direct_v2(signed int16 *vbat, signed int16 *tbat, signed int16 *tenv, signed int16 *btec, signed int16 *tvco, signed int16 *curr, int handle)
Parameters:	Vbat, tbat, tenv, btec, tvco and curr fetched the measured ADC value.
Returns:	Result of the operation: *ok or error*
Description	This function is the same as the function DWD_meas_all_adc_v2, but it is easier to use in the Lab View program.
Platform	MP1
Mode:	Non-Signalling test mode
Atcptest:	atctst_meas_all_adc_req

6.2 IrDa calibration

6.2.1 DWD_enter_irda_test_mode

Function name:	Bool DWD_enter_irda_test_mode(unsigned int16 *result, int handle)
Parameters:	Result: Result fetched the result from the operation.
Returns:	Result of the operation: *ok or error*
Description	This function is used to enter the IrDA test mode.
Platform	P2002+, MP1, BP2
Mode:	Default test mode, MMI Test mode
Atcptest:	atctst_enter_irda_test_mode_req

6.2.2 DWD_exit_irda_test_mode

Function name:	Bool DWD_exit_irda_test_mode(unsigned int16 *result, int handle)
Parameters:	Result: Result fetched the result from the operation.
Returns:	Result of the operation: *ok or error*
Description	This function is used to exit from IrDA test mode.
Platform	P2002+, MP1, BP2
Mode:	Default test mode, MMI test mode
Atcotest:	atctst_exit_irda_test_mode_req

6.2.3 DWD_receive_irda_data

Function name:	Bool DWD_receive_irda_data(unsigned int16 pattern, unsigned int16 timeout,

	unsigned int32 baudrate, unsigned int16 *result, int handle);
Parameters:	Pattern: Predefined data to received Timeout: The time in seconds, which the function is looking for the specified pattern. Baudrate: baud rate you want to receive the data (2400,9600,19200,38400,57600,115200) Result: Result fetched the result from the operation.
Returns:	Result of the operation: *ok or error*
Description	This function is used to receive a data stream from the onboard IrDa diode.
Platform	P2002+, BP2
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_receive_irda_data_req

6.2.4 DWD_receive_irda_data_v2

Function name:	bool DWD_receive_irda_data_v2(unsigned int16 pattern, unsigned int16 timeout, unsigned int32 baudrate, int handle)
Parameters:	Pattern: Predefined data to received Timeout: The time in milliseconds, which the function is looking for the specified pattern. Baudrate: baud rate you want to receive the data (2400,9600,19200,38400,57600,115200)
Returns:	Result of the operation: *ok or error*
Description	This function is used to receive a data stream from the onboard IrDa diode.
Platform	MP1
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_receive_irda_data_v2_req

6.2.5 DWD_send_irda_data

Function name:	Bool DWD_send_irda_data(unsigned int16 pattern, unsigned int32 baudrate, unsigned int16 *result, int handle)
Parameters:	Pattern: Predefined data to send. Baudrate: Baud rate you want to send the data 2400,9600,19200,38400,57600,115200) Result: Result fetched the result from the operation.
Returns:	Result of the operation: *ok or error*
Description	This function is used to transmit a data stream from the onboard IrDA diode.
Platform	P2002+, BP2
Mode:	Default test mode, MMI test mode.
Atcptest:	atctst_send_irda_data_req

6.2.6 DWD_send_irda_data_v2

Function name:	bool DWD_send_irda_data_v2(unsigned int16 pattern, unsigned int32 baudrate, unsigned int16 timeout, int handle)
Parameters:	Pattern: Predefined data to send. Baudrate: Baud rate you want to send the data 2400,9600,19200,38400,57600,115200) Timeout: time out time in milliseconds.

Returns:	Result of the operation: *ok or error*
Description	This function is used to transmit a data stream from the onboard IrDA diode.
Platform	MP1
Mode:	Default test mode, MMI test mode.
Atcptest:	atctst_send_irda_data_v2_req

6.2.7 DWD_get_irda_test_result

Function name:	bool DWD_get_irda_test_result(bool *result, int handle)
Parameters:	*result: test result.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the IrDA test result.
Platform	MP1
Mode:	Default test mode, MMI test mode.
Atcptest:	atctst_get_irda_test_result_req

6.3 LED test

6.3.1 DWD_led_change_intensity

Function name:	Bool DWD_led_change_intensity(unsigned char color, unsigned char intensity, int handle);
Parameters:	Color: The actual color you want to change the intensity. Intensity: the value of intensity (1 to 7)
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the intensity of and actual color. **Can not be used if #define LED_MULTICOLOR_NEW_INTERFACE is used.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_led_change_intensity_req

6.3.2 DWD_led_color_off

Function name:	Bool DWD_led_color_off(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to switch off the LED. **Can not be used if #define LED_MULTICOLOR_NEW_INTERFACE is used.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_led_color_off_req

6.3.3 DWD_led_set_color

Function name:	Bool DWD_led_set_color(unsigned int16 color, unsigned int16 time_on, unsigned int16 total_time, unsigned int16 action, int handle);

Parameters:	Color: dwd_led_blue, dwd_led_red, dwd_led_green, dwd_led_blue_red, dwd_led_blue_green, dwd_led_red_green, dwd_led_red_blue_green, Action: LED_OFF = 0, LED_ON = 1, LED_FLASH = 2, LED_COLOR_STREAM = 3 On_time: time (in ms) for the selected led to be turned on. Total_time: on_time + off_time. (ms) typedef enum { dwd_led_blue = 0, dwd_led_red, dwd_led_green, dwd_led_blue_red, dwd_led_blue_green, dwd_led_red_green, dwd_led_red_blue_green, dwd_led_nof_colors } **dwd_led_color_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the color for the selected LED. **If LED_FLASH is selected, then the LED will flash with the defined color and the selected frequency.** **Can not be used if #define LED_MULTICOLOR_NEW_INTERFACE is used.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_led_set_color_req

6.3.4 DWD_led_set_colorstream

Function name:	Bool DWD_led_set_colorstream(unsigned char nof_color, **dwd_colorstream_type** *color , int handle)
Parameters:	Nof_color: Means how many new colors are set? *color: Reference fetched the color stream. Its possible to set 10 colors and the color value could be from 0-7. total_time and on_time must also be set for each color. typedef struct { unsigned char color[10]; unsigned int16 total_time[10]; unsigned int16 on_time[10]; } **dwd_colorstream_type;**
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the default color stream in the DUT. **Can not be used if #define LED_MULTICOLOR_NEW_INTERFACE is used.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_led_set_colorstream_req

6.3.5 DWD_led_set_photolight

Function name:	Bool DWD_led_set_photolight(unsigned int16 action, unsigned int16 intensity, unsigned int16 total_time, unsigned int16 on_time, int handle)
Parameters:	Action: Used to set the photo light to one of the operations Off(0), Flash Once(1), Flash Forever(2), On(3). Intensity: Used to specify intensity levels betweens 1 to 64. Total time: Used to specify the total time in ms, when action is Flash Forever. The values that can be specified are in multiples of 10, the minimum value is 10 and maximum is 4000. On Time: Used to specify the on time in ms, when action is Flash Once or Flash Forever. The values that can be specified are in multiples of 10, the minimum value is 10 and maximum is 2000
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the photo light in DUT. **The #define LED_MULTICOLOR_NEW_INTERFACE must no have been set in target SW.**
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_led_set_photolight_req

There have been created a new generic multi color interface to LED. It means it has been necesarry to create some new production test functions to test this new functionality. The new interface has been wrapped up in the driver by a #define called LED_MULTICOLOR_NEW_INTERFACE. You must ensure this #define are enabled in the target system before you can use the following production test function to test the LED, you must also be aware there are a dependency to the hardware.

6.3.6 DWD_led_on

Function name:	bool DWD_led_on(**dwd_led_on_type** *data, signed int32 *status, int handle)
Parameters:	*data: The color intensity is specified as a percentage of available intensity levels. E.g 0% = led off and 100% is led on with max intensity. Repeat = 0 will continue forever. Repeat = Total time = on time = 0 will swich on the leds without flashing. typedef struct { unsigned int32 led_on_red; unsigned int32 led_on_green; unsigned int32 led_on_blue; unsigned int32 led_on_white; unsigned int32 total_time; unsigned int32 on_time; unsigned int32 repeat; unsigned int32 led_type; }dwd_led_on_type; *status: has the result value from the LED driver. See the definition in led.h (led_result_type)
Returns:	Result of the operation: *ok or error*

Description	This function is used to turn on the LED
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_led_generic_req

6.3.7 DWD_led_off

Function name:	bool DWD_led_off(**dwd_led_off_type** *data, signed int32 *status, int handle)
Parameters:	*status: has the result value from the LED driver. See the definition in led.h (led_result_type) typedef struct { unsigned int32 fade; unsigned int32 led_type; }dwd_led_off_type;
Returns:	Result of the operation: *ok or error*
Description	Ths function is used to turn off the LED
Platform	MP1
Mode	Default test mode
Atcptest:	Atctst_led_generic_req

6.4 Backlight test

6.4.1 DWD_bl_get_intensity

Function name:	Bool DWD_bl_get_intensity(unsigned char lcd_tp, unsigned int16 *intensity, int handle)
Parameters:	Lcd_tp: BL_LCD_MAIN(0) or BL_LCD_SUB(1) or BL_KEY(2) *Intensity: Reference fetched the intensity.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the backlight intensity from the DUT depend on which LCD_TP there are selected.
Platform	P2002, P2002+, MP1, BP2
Mode	Default test mode
Atcptest:	atctst_bl_get_intensity_req

6.4.2 DWD_bl_get_status

Function name:	Bool DWD_bl_get_status(unsigned int16 *status, int handle);
Parameters:	Status: Reference fetched the status from the DUT The bits indicate whether a backlight is off(0) or on(1) bit 0: main LCD backlight bit 1: sub LCD backlight bit 2: key backlight bit 3-7: not used.

Returns:	Result of the operation: *ok or error*
Description	This function is used to get the status of the backlight (On or Off)
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_bl_get_status_req

6.4.3 DWD_keypad_backlight_init

Function name:	Bool DWD_keypad_backlight_init(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to initialize the keypad backlight
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_bl_keypad_backlight_init_req

6.4.4 DWD_lcd_backlight_init

Function name:	Bool DWD_lcd_backlight_init(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to initialize the LCD backlight
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_bl_lcd_backlight_init_req

6.4.5 DWD_set_keypad_backlight

Function name:	Bool DWD_set_keypad_backlight(unsigned char action, unsigned int16 fade, unsigned int16 intensity, int handle)
Parameters:	Action: BL_OFF (0) or BL_ON (1) Fade: could be *true* or *false*. **If not supported by the HW platform this parameter is "don't care"**. Intensity: intensity value. **If not supported by the HW platform this parameter is "don't care".**
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the keypad backlight *on* or *off*. It is possible to change the intensity and to set the fade *on* or *off*.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_bl_set_keypad_backlight_req

6.4.6 DWD_set_lcd_backlight

Function name:	Bool DWD_set_lcd_backlight(unsigned char lcd_tp, unsigned char action, unsigned int16 fade, unsigned int16 intensity, int handle)
Parameters:	Lcd_tp: BL_LCD_MAIN (0) or BL_LCD_SUB (1)

	Action: BL_OFF (0) or BL_ON (1) Fade: could be *true* or *false*. **If not supported by the HW platform this parameter is "don't care".** Intensity: Reference fetched the intensity. **If not supported by the HW platform this parameter is "don't care".**
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the LCD backlight *on* or *off*. It is possible to change the intensity and to set the fade *on* and *off*.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_bl_set_lcd_backlight_req

6.4.7 DWD_switch_off_backlight

Function name:	Bool DWD_switch_off_back_light(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function switches off the backlight. **This function can only be used by a target SW where #define DWD_BL_ENABLE_OLD_INTERFACE is enabled and #define DWD_MODEM_ONLY not are defined.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_switch_off_back_light_req

6.4.8 DWD_switch_on_backlight

Function name:	Bool DWD_switch_on_back_light(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function switches off the backlight. **This function can only be used by a target SW where #define DWD_BL_ENABLE_OLD_INTERFACE is enabled and #define DWD_MODEM_ONLY not are defined.**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_switch_on_back_light_req

6.4.9 DWD_bl_set_backlight

Function name:	bool DWD_bl_set_backlight(unsigned char action, unsigned int16 *result, int handle)
Parameters:	Action: Should bet set to 0 (Means Off) or 1 (Means On). *Result: Fetched the result of the action operation.
Returns:	Result of the operation: *ok or error*
Description	This function is used to turn On or OFF the Backlight for both Keyboard and LCD.
Platform	ULC
Mode	Default test mode
Atcptest:	atctst_bl_set_backlight_req

6.5 Vibrator test

6.5.1 DWD_switch_vibrator_on

Function name:	Bool DWD_switch_vibrator_on(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to switch *on* the vibrator.
Platform	P2002, P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_switch_vibrator_on_req

6.5.2 DWD_switch_vibrator_on_continues

Function name:	Bool DWD_switch_vibrator_on(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to switch *on* the vibrator continues. **This function can only be used if the #define NEW_AUDIO_DRIVER_KONCEPT are enabled in the DUT.**
Platform	P2002, P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_switch_vibrator_on_continus_req

6.5.3 DWD_switch_vibrator_off

Function name:	Bool DWD_switch_vibrator_off(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to switch *off* the vibrator.
Platform	P2002, P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_switch_vibrator_off_req

6.5.4 DWD_vibrator_on

Function name:	bool DWD_vibrator_on(unsigned int16 intensity, unsigned int16 mode, unsigned int16 channel,unsigned int16 *result, int handle)
Parameters:	Intensity: Set the level of the vibrator. The value can be set from 0 -> 12. “0” means the default level, “1” the weakest vibration level and “12” strongest vibration level. Mode: Set which mode the vibrator should working in. The modes are defined as followed in target header file “vib.h”. enum { VIB_ASYNC_ON, // = aud_vibrator_on,

	VIB_ASYNC_BLINKING1, // = aud_vibrator_async_blinking1, VIB_ASYNC_BLINKING2, // = aud_vibrator_async_blinking2, VIB_ASYNC_BLINKING3, // = aud_vibrator_async_blinking3, VIB_ASYNC_BLINKING4, // = aud_vibrator_async_blinking4, VIB_ASYNC_BLINKING5, // = aud_vibrator_async_blinking5, VIB_SYNC_MA_CONTROL = 0x11, // = aud_vibrator_sync1, VIB_SYNC_FW_CONTROL = 0x81, VIB_MODE_END }; Channel: The parameter defines the channel that the vibrator should be synchronized with. *Result: Reflect the error indication from the vibrator driver.
Returns:	Result of the operation: *ok or error*
Description	This function is used to start the vibrator in different level, mode and channel by using the vip generic interface. Be aware there is a hardware dependency by using all functionality in the function.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_vib_generic_func_req

6.5.5 DWD_vibrator_off

Function name:	bool DWD_vibrator_off(unsigned int16 *result, int handle);
Parameters:	*Result: Reflect the error indication from the vibrator driver.
Returns:	Result of the operation: *ok or error*
Description	This function is used to stop the vibrator by using the generic vibrator interface
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_vib_generic_func_req

6.6 RTC (Real Time Clock test)

For verification of RTC two sets of functions exists:

1. *DWD_check_rtc*, which is used to check the *32 KHz Oscillator* connections in DUT where no *backup battery* is mounted.
2. *DWD_program_rtc_date_time* and *DWD_get_rtc_date_time*, which is used to verify the RTC including the *Backup battery*. This is used as follows:
   - STEP1: At the beginning of the production flow the time is programmed by calling *DWD_program_rtc_date_time*
   - Later in the production flow after power has been removed one or several times the function *DWD_get_rtc_date*_time is called to read out the date & time. If the RTC date & time is still correct at this stages the RTC including backup battery is working.

6.6.1 DWD_check_rtc

Function name:	Bool DWD_check_rtc(bool *result, int handle)
Parameters:	Result: Reference fetched the result. The result could be *true* or *false*.
Returns:	Result of the operation: *ok or error*
Description	This function is used to check the RTC is running (32 KHz oscillator soldering OK). **The DUT must be in "Non-signaling test mode" before using this command.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_check_rtc_req

6.6.2 DWD_get_rtc_data_time

Function name:	Bool DWD_get_rtc_date_time(**dwd_rtc_date_type** *date, **dwd_rtc_time_type** *time, int handle)
Parameters:	Date: Reference fetched to the date string in which the MS date is returned. It has the following format: yyyy.mm.dd like "2002.05.16". Time: Reference fetched to the time string in which the MS time is returned. **It has following 24 hour format: hh:mm:ss like: "23:35:10"** **#define DWD_RTC_TIME_STRING_LENGTH 8** **#define DWD_RTC_DATE_STRING_LENGTH 10** **typedef unsigned char dwd_rtc_time_type[DWD_RTC_TIME_STRING_LENGTH + 1];** **typedef unsigned char dwd_rtc_date_type[DWD_RTC_DATE_STRING_LENGTH + 1];**
Returns:	Result of the operation: *ok or error*
Description	This function fetches the RTC date and time from the DUT.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode, Non-Signalling test mode, MMI test mode
Atcptest:	atctst_get_rtc_date_time_req

6.6.3 DWD_program_rtc_date_time

Function name:	Bool DWD_program_rtc_date_time(**dwd_rtc_date_type** *date, **dwd_rtc_time_type** *time, **dwd_rtc_result_type** *result, int handle);
Parameters:	Date: Reference fetched the date string with following format: **yyyy.mm.dd** like "**2002.05.16**". Time: Reference fetched the time string, which has following 24 hour format: **hh:mm:ss** like: "**23:35:10**" Result: Reference fetched the Result of the time Programming: *dwd_rtc_setting_ok* if the setting was okay or *dwd_rtc_setting_invalid* if the date and time setting failed . **#define DWD_RTC_TIME_STRING_LENGTH 8** **#define DWD_RTC_DATE_STRING_LENGTH 10** **typedef unsigned char dwd_rtc_time_type[DWD_RTC_TIME_STRING_LENGTH + 1];**

	typedef unsigned char dwd_rtc_date_type[DWD_RTC_DATE_STRING_LENGTH + 1]; typedef enum { dwd_rtc_setting_ok, dwd_rtc_setting_invalid, }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the RTC date and time in the DUT.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode, Non-Signalling test mode, MMI test mode
Atcptest:	atctst_set_rtc_date_time_req

6.6.4 DWD_rtc_set_date_val

Function name:	bool DWD_rtc_set_date_val(**dwd_rtc_date_val_type** *date_val, **dwd_rtc_result_type** *result, int handle)
Parameters:	Date_val: Fetched the date value you want to set in the DUT. *result: Fetched the result of the date value programming typedef struct { unsigned int16 year; unsigned char month; unsigned char day; }**dwd_rtc_date_val_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the RTC date value in the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_set_date_val_req

6.6.5 DWD_rtc_get_date_val

Function name:	bool DWD_rtc_get_date_val(dwd_rtc_date_val_type *date_val, dwd_rtc_result_type *result, int handle)
Parameters:	Date_val: Fetched the date value received from the DUT. *result: Fetched the result of getting the date value from DUT. typedef struct { unsigned int16 year; unsigned char month; unsigned char day; }**dwd_rtc_date_val_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the RTC date value from the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_get_date_val_req

6.6.6 DWD_rtc_set_time_val

Function name:	bool DWD_rtc_set_time_val(**dwd_rtc_time_val_type** *time_val, **dwd_rtc_result_type** *result, int handle)
Parameters:	Time_val: Fetched the time value you want to set in the DUT. *result: Fetched the result of the time value programming typedef struct { unsigned char hour; unsigned char minute; unsigned char second; }**dwd_rtc_time_val_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed,

	dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the RTC time value in the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_set_time_val_req

6.6.7 DWD_rtc_get_time_val

Function name:	bool DWD_rtc_get_time_val(**dwd_rtc_time_val_type** *time_val, **dwd_rtc_result_type** *result, int handle)
Parameters:	Date_val: Fetched the time value received from the DUT. *result: Fetched the result of getting the time value from DUT. typedef struct { unsigned int16 year; unsigned char month; unsigned char day; }**dwd_rtc_date_val_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the RTC time value from the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_get_time_val_req

6.6.8 DWD_rtc_set_event_date_time

Function name:	bool DWD_rtc_set_event_date_time(**dwd_rtc_event_datetime_type** *event,

	dwd_rtc_result_type *result, int handle)
Parameters:	Event: Fetched the event value you want to set in the DUT. *result: Fetched the result of the event programming in the DUT. typedef struct { dwd_rtc_date_val_type date; dwd_rtc_time_val_type time; }**dwd_rtc_event_datetime_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the RTC event value in the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_set_event_date_time_req

6.6.9 DWD_rtc_get_event_date_time

Function name:	bool DWD_rtc_get_event_date_time(**dwd_rtc_event_datetime_type** *event, **dwd_rtc_result_type** *result, int handle)
Parameters:	Event: Fetched the event data received from the DUT. *result: Fetched the result of getting the event data from DUT. typedef struct { dwd_rtc_date_val_type date; dwd_rtc_time_val_type time; }**dwd_rtc_event_datetime_type**; typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid,

	dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the RTC event data from the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_get_event_date_time_req

6.6.10 DWD_rtc_clear_event_date_time

Function name:	bool DWD_rtc_clear_event_date_time(**dwd_rtc_result_type** *result, int handle)
Parameters:	*result: Fetched the result of clearing the event data in the DUT. typedef enum { dwd_rtc_setting_ok, dwd_rtc_register_access_failed, dwd_rtc_nvram_access_failed, dwd_rtc_limits_exceeded, dwd_rtc_alarm_program_failed, dwd_rtc_no_alarm, dwd_rtc_setting_invalid, dwd_rtc_default_error }**dwd_rtc_result_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to clear RTC event data in the DUT
Platform	ULC
Mode:	Default test mode
Atcptest:	atctst_rtc_clear_event_date_time_req

6.7 SIM Test

6.7.1 DWD_verify_sim_connection

Function name:	Bool DWD_verify_sim_connection(bool *sim_ok, int handle)
Parameters:	Sim_ok: *true* if SIM connection is *OK* else *false*
Returns:	Result of the operation: *ok or error*
Description	This function is used to determine if the SIM connection is OK. **NOTE: The connection to the SIM is ONLY checked once when executing** **If the SIM connection at this point is OK then *DWD_verify_sim_connection* will always set *sim_ok* to TRUE regardless of the SIM is removed afterwards.** **If the SIM connection at this point failed then *DWD_verify_sim_connection* will always set *sim_ok* to FALSE even if the SIM at a later stage is reinserted.**
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode, Non-Signalling test mode, MMI test mode, Signalling test mode
Atcptest:	atctst_verify_sim_connection_req

6.7.2 DWD_configure_sim_simulation

Function name:	bool DWD_configure_sim_simulation(unsigned int16 enable_simulation,unsigned int16 *result,int handle);
Parameters:	Enable_simulation: 0 means disable and 1 means enable. Result: indicate the result of disable / enable sim simulation, It is not a check for the serial line.
Returns:	Result of the operation: *ok or error*
Description	This function is used to enable or disable sim simulation **Please note: No real SIM card must be connected at the same time, otherwise this function will not work.**
Platform	BP30
Mode:	Default test mode, Non-Signalling test mode, MMI test mode, Signalling test mode
Atcptest:	atctst_sim_generic_req

6.7.3 DWD_get_sim_icc_id

Function name:	bool DWD_get_sim_icc_id(unsigned int16 *result, unsigned char *icc_id, int handle)
Parameters:	*result: Fetched the result code from the SIM driver. *icc_id: Fetched the icc_id code for the present SIM card in BCD format.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the ICC ID for the present SIM card.
Platform	?
Mode:	Default test mode, Non-Signalling test mode, MMI test mode, Signalling test mode
Atcptest:	atctst_sim_generic_req

6.8 Charger test

6.8.1 DWD_charger_inserted

Function name:	Bool DWD_charger_inserted(bool *result, int handle)
Parameters:	Result: Reference fetched the result. The result will be true if charger is connected.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to test a charger is connected.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_charger_inserted_req

6.8.2 DWD_set_charge_mode

Function name:	Bool DWD_set_charge_mode(unsigned char on, int handle)
Parameters:	On: If *true* charging is enabled. If *false* charging is disabled.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the charge mode.

	The DUT must be in “Non-signaling” test mode before using this command.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_set_charging_req

6.9 GDD (Graphic Device)

Function descriped in chapter **6.9.1** and **6.9.2** is using old CAM or LCD generic interfaces except *DWD_set_lcd_contrast* and *DWD_lcd_test_image*.

Function descriped in chapter **6.9.3** ad **6.9.4** are both using the GDD generic which is the new interface.

6.9.1 LCD test function (Using LCD generic)

6.9.1.1 DWD_set_lcd_contrast

Function name:	Bool DWD_set_lcd_contrast(unsigned int16 contrast, int handle);
Parameters:	Contrast: Reference fetched the new contrast value
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the contrast directly in the DUT until the right value has been found. Note: The contrast value is stored temporarily in RAM and is lost in case the DUT is powered off.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_lcd_set_contrast_req

6.9.1.2 DWD_set_pixel

Function name:	Bool DWD_set_pixel(unsigned char x, unsigned char y, unsigned int16 nof_input_bytes, unsigned char *data, int handle)
Parameters:	X: The value of x coordinate. Y: The value of y coordinate. Nof_input_bytes: The numbers of input bytes you want to send to the DUT. data: Reference fetched the input to the data buffer.
Returns:	Result of the operation: *ok or error*
Description	This function is used to write the specified data into the pixel specified by the coordinates X, Y (Upper left pixel (X;Y) = (0,0)
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_lcd_generic_func_req

6.9.1.3 DWD_lcd_test_image

Function name:	Bool DWD_lcd_test_image(unsigned int16 pattern, int handle)
Parameters:	Pattern: Is the number of this test image you want to start, the test image is predefined in DUT.

	Pattern No. Description *1 4x4 dot Ichimatsu (1)* *2 4x4 dot Ichimatsu (2)* *3 All White* *4 All Black* *5 Graduation (4 steps)* *6 1 dot width black frame* *7 TBD.* *8 All Blue* *9 7 Color bar* *10 All Red* *11 All Green* *12 RGB Ichimatsu (1)* *13 RGB Ichimatsu (2)* *14 Graduation (9 steps) – 1* *15 Graduation (9 steps) – 2* *16 Cross talk checker* *17 Calibration frame*
Returns:	Result of the operation: *ok or error*
Description	This function is used to start displaying a test image.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_start_test_image_req

6.9.1.4 DWD_read_lcd_register

Function name:	Bool DWD_read_lcd_register(unsigned char address, unsigned char *data, int handle)
Parameters:	Address: Reference fetched the input address. Data: Reference fetched the return data.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read the LCD register at the given address.
Platform	P2002, P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_lcd_generic_func_req

6.9.1.5 DWD_write_lcd_register

Function name:	Bool DWD_write_lcd_register(unsigned char address, unsigned char data, int handle)
Parameters:	Address: Reference fetched the input address. Data: Reference fetched the input data.
Returns:	Result of the operation: *ok or error*
Description	This function is used to write data to the LCD register at the given address.
Platform	P2002, P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_lcd_generic_func_req

6.9.2 Camera test function (Using CAM generic)

6.9.2.1 DWD_get_camera_status

Function name:	Bool DWD_get_camera_status(unsigned int16 *status, int handle)
Parameters:	Status: Reference fetched the status code from the DUT.
Returns:	Result of the operation: *ok or error*
Description	This function gets the reads the status from the camera (sleep, preview, picture, etc).
Platform	P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_cam_generic_func_req

6.9.2.2 DWD_sleep_camera

Function name:	Bool DWD_sleep_camera(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function bring the camera in sleep mode
Platform	P2002+, BP2
Mode:	Default test mode.
Atcptest:	Atctst_cam_generic_func_req

6.9.2.3 DWD_start_preview

Function name:	Bool DWD_start_preview(unsigned int16 mode, unsigned int16 framerate, int handle)
Parameters:	Mode: *dwd_preview_mode1* or *dwd_preview_mode2* or *dwd_preview_mode3* Framerate: Fetched the frame rate
Returns:	Result of the operation: *ok or error*
Description	This function is used to startup a camera preview mode.
Platform	P2002+, BP2
Mode:	Default test mode
Atcptest:	atctst_cam_generic_func_req

6.9.2.4 DWD_stop_preview

Function name:	Bool DWD_stop_preview(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to stop the camera preview mode
Platform	P2002+, BP2
Mode:	Default test mode.
Atcptest:	atc_cam_generic_func_req

6.9.2.5 DWD_take_picture

Function name:	Bool DWD_take_picture(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function makes a snapshot, which will be shown in the display

Platform	P2002+, BP2
Mode:	Default mode
Atcptest:	atc_cam_generic_func_req

6.9.2.6 DWD_put_jpeg_picture

Function name:	Bool DWD_put_jpeg_picture(char *filename, unsigned int16 left, unsigned int16 top, unsigned int16 right, unsigned int16 bottom, int handle)
Parameters:	Filename: Fetched the input filename. Left, right, top & bottom fetched the input to the view area coordinates, where you want to see the picture in the DUT display.
Returns:	Result of the operation: *ok or error*
Description	This function makes a snapshot, which will be shown in the display.
Platform	P2002+, BP2
Mode:	Default test mode.
Atcptest:	atctst_cam_generic_func_req

6.9.2.7 DWD_get_picture

Function name:	Bool DWD_get_picture(char *filename, int handle)
Parameters:	Filename: Reference fetched the filename.
Returns:	Result of the operation: *ok or error*
Description	This function gets picture from the mobile and saved it in a file on the pc.
Platform	P2002+, BP2
Mode:	Default test mode.
Atcptest:	atctst_cam_generic_func_req

6.9.3 LCD test function (Using GDD generic)

6.9.3.1 DWD_lcd_set_contrast_v2

Function name:	bool DWD_set_lcd_contrast_v2(unsigned int32 hw_id, unsigned int32 contrast,unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to set the contrast to. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; Contrast: Reflect the new contrast value. You must be aware the limits for this contrast

	value depend on the hardware. The function: **Error! Reference source not found.** can get the contrast limits: *status: Reflect the errorcode indication from the GDD driver.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set a new contrast value. The value is changed in RAM which means the value is lost in this case the target is powered off.
Platform	MP1
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_gdd_generic_req

6.9.3.2 DWD_lcd_set_pixel

Function name:	bool DWD_lcd_set_pixel(unsigned int32 hw_id,unsigned int32 x, unsigned int32 y, unsigned char data, unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to set the contrast to. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; X: Reflect the x coordinate. Y: Reflect the y coordinate. Data: Reflect the data to be set at the specified coordinate. *status: Reflect the errorcode indication from the GDD driver.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set a pixel at a given coordinate. Uppper left corner x = 0, y =0
Platform	MP1
Mode:	Default test mode.
Atcptest:	atctst_gdd_generic_req

6.9.3.3 DWD_lcd_test_image_v2

Function name:	bool DWD_lcd_test_image_v2(unsigned int32 hw_id, unsigned int32 pattern, unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to set the contrast to. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. typedef enum

	{ /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; Pattern: Is the number of this test image you want to start, the test image is predefined in DUT. *Pattern No. Description* *1 4x4 dot Ichimatsu (1)* *2 4x4 dot Ichimatsu (2)* *3 All White* *4 All Black* *5 Graduation (4 steps)* *6 1 dot width black frame* *7 TBD.* *8 All Blue* *9 7 Color bar* *10 All Red* *11 All Green* *12 RGB Ichimatsu (1)* *13 RGB Ichimatsu (2)* *14 Graduation (9 steps) – 1* *15 Graduation (9 steps) – 2* *16 Cross talk checker* *17 Calibration frame* *status: Reflect the errorcode indication from the GDD driver.
Returns:	Result of the operation: *ok or error*
Description	This function is used to start displaying a test image.
Platform	MP1
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_gdd_generic_req

6.9.3.4 DWD_get_lcd_contrast_limits

Function name:	bool DWD_get_lcd_contrast_limits(unsigned int32 hw_id, unsigned int32 *min_value, unsigned int32 *def_value, unsigned int32 *max_value, unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to set the contrast to. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. typedef enum

	{ /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; *min_value: Reflect the minimum contrast value; *def_value: Refelect the default contrast value; *max_value: Reflect the maximum contrast value there can be set by using the function: **Error! Reference source not found.** *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the contrast limits
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4 Camera test function (Using GDD generic)

6.9.4.1 DWD_gdd_cam_enable

Function name:	bool DWD_gdd_cam_enable(unsigned int32 left, unsigned int32 top, unsigned int32 right, unsigned int32 bottom, unsigned int32 *status, int handle)
Parameters:	Left: Reflect the viewfinder left coordinate. Top: Reflect the viewfinder top coordinate. Right: Reflect the viewfinder right coordinate. Bottom: Reflect the viewfinder bottom coordinate. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	The function is used to start the viewfinder at a given coordinate in the MAIN LCD
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.2 DWD_gdd_cam_disable

Function name:	bool DWD_gdd_cam_disable(unsigned int32 *status, int handle)
Parameters:	*status: Reflect the error code from the GDD driver after execution of this function.
Returns:	Result of the operation: *ok or error*
Description	The function is used to stop the viewfinder.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.3 DWD_gdd_cam_capture

Function name:	bool DWD_gdd_cam_capture(unsigned int32 hw_id, unsigned int32 *status, int handle)
Parameters:	Hw_id: Reflect the hardware ID no of the camera. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function is used to capture a picture. **This function has not been verified yet.**
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.4 DWD_gdd_set_cam_compression

Function name:	bool DWD_gdd_set_cam_compression(unsigned int32 compression, unsigned int32 *status, int handle)

Parameters:	Compression: Reflect a value from the gdd_cam_compression_type
	
	typedef enum
	{
	/* Do not change the order */
	GDD_CAM_COMP_HIGHEST = 0x00,
	GDD_CAM_COMP_HIGH,
	GDD_CAM_COMP_MEDIUM,
	GDD_CAM_COMP_LOW,
	GDD_CAM_COMP_ALLOWED_LIMITS /* Should be the last value */
	} gdd_cam_compression_type; // can be found in gdd.h source file
	
	*status: Reflect the error code from the GDD driver after execution this function.

Returns:	Result of the operation: *ok or error*
Description	This function is used to set the image compression format. Be aware this function is hardware depended and you need to now which compression format your hardware support. You can get the capability for your hardware to see it. **This function has not been verified yet.**
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.5 DWD_gdd_set_cam_brightness

Function name:	bool DWD_gdd_set_cam_brightness(unsigned int32 brightness, unsigned int32 *status, int handle)
Parameters:	Brightness: Reflect the brightness value *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to set camera brightness, but it hardware depended. See the capability for the camera if this feature is supported? If yes then you can call the

	function: **DWD_gdd_get_cam_brightness_limits**
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.6 DWD_gdd_get_cam_brightness_limits

Function name:	bool DWD_gdd_get_cam_brightness_limits(unsigned int32 hw_id, unsigned int32 *min_value, unsigned int32 *max_value, unsigned int32 *def_value, unsigned int32 *status, int handle)
Parameters:	Hw_id: Must be set to the camera hardware ID. *min_value: Reflect the minimum brightness value. *max_value: Reflect the maximum brightness value. *def_value: Reflect the default value. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to get the limits of legal brightness values.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.7 DWD_gdd_set_cam_antiflicker_mode

Function name:	bool DWD_gdd_set_cam_antiflicker_mode(unsigned int32 antiflicker_mode, unsigned int32 *status, int handle)
Parameters:	Antiflicker_mode: Should be set to one of the option from the following: typedef enum { GDD_CAM_ANTI_FLICKER_50Hz, GDD_CAM_ANTI_FLICKER_60Hz, GDD_CAM_ANTI_FLICKER_OFF } gdd_cam_flicker_type; *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the camera anti flicker mode. Be aware this function is hardware depended.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.8 DWD_gdd_set_cam_twilight_mode

Function name:	bool DWD_gdd_set_cam_twilight_mode(unsigned int32 twilight_mode, unsigned int32 *status, int handle)
Parameters:	Twilight_mode: Should be set to one of the option from the following structure: typedef enum

	{ GDD_CAM_TWILIGHT_MODE_ENABLE, GDD_CAM_TWILIGHT_MODE_DISABLE } gdd_cam_twilight_type; *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function is used to enable or disable camera twilight mode. Be aware this function can only be used if it supported by the hardware.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.9 DWD_gdd_set_cam_image_format

Function name:	bool DWD_gdd_set_cam_image_format(unsigned int32 image_format, unsigned int32 *status, int handle)

Parameters:	Image_format: Should be set to one of the following type.
	
	typedef enum
	{
	GDD_CAM_CIF_FORMAT = 0, // Default format - eeprom.
	GDD_CAM_QCIF_FORMAT = 1,
	GDD_CAM_QQCIF_FORMAT = 2,
	GDD_CAM_VGA_FORMAT = 3,
	GDD_CAM_QVGA_FORMAT = 4,
	GDD_CAM_QQVGA_FORMAT = 5,
	GDD_CAM_SQCIF_FORMAT = 6,
	GDD_CAM_QSIF_FORMAT = 7,
	GDD_CAM_QSIFSP_FORMAT = 8,
	GDD_CAM_SXGA_FORMAT = 9,
	GDD_CAM_1_2MPIX_FORMAT = 10,
	GDD_CAM_XGA_FORMAT = 11,
	GDD_CAM_UXGA_FORMAT = 12,
	GDD_CAM_NOF_IMAGE_FORMATS = 13
	} gdd_cam_image_format_type;
	
	*status: Reflect the error code from the GDD driver after execution this function.

Returns:	Result of the operation: *ok or error*
Description	This function is used to set the image format. Support for the different format depends on the hardware. The capability list for the camera can give you some information about which format there is supported.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.10 DWD_gdd_set_cam_contrast

Function name:	bool DWD_gdd_set_cam_contrast(unsigned int32 contrast, unsigned int32 *status, int

	handle)
Parameters:	Contrast: Reflect the new contrast value *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to set the contrast setting for the camera. It is hardware depended. See the capability for the camera if this feature is supported? If yes then you can call the function: **DWD_gdd_get_cam_contrast_limits**
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.4.11 DWD_gdd_get_cam_contrast_limits

Function name:	bool DWD_gdd_get_cam_contrast_limits(unsigned int32 hw_id, unsigned int32 *min_value, unsigned int32 *max_value, unsigned int32 *def_value, unsigned int32 *status, int handle)
Parameters:	Hw_id: Must be set to the camera hardware ID. *min_value: Reflect the minimum brightness value. *max_value: Reflect the maximum brightness value. *def_value: Reflect the default value. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to get the limits of legal contrast values.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.5 Other function using GDD generic

6.9.5.1 DWD_read_gdd_llc_register

Function name:	bool DWD_read_gdd_llc_register(unsigned int32 hw_id, unsigned int32 Register, unsigned int32 *value, unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to read register value from. The value could be one of the enumeration from gdd_hw_id_type typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; Register: Reference to this register address you want to read.

	*value: Reflect the register value. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to read register value at a given address from different devices, such as LCD (main or sub), Companion chip and camera.
Platform	P2002+, MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.5.2 DWD_write_gdd_llc_register

Function name:	bool DWD_write_gdd_llc_register(unsigned int32 hw_id, unsigned int32 Register, unsigned int32 value, unsigned int32 *status, int handle)
Parameters:	Hw_id: indicate which hardware you want to write new register settings. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type; Register: Reference to this register address you want to write. value: The new register value. *status: Reflect the error code from the GDD driver after execution this function.
Returns:	Result of the operation: *ok or error*
Description	This function can be used to write new register value at a given address to different devices, such as LCD (main or sub), Companion chip and camera.
Platform	P2002+, MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.5.3 DWD_get_gdd_data_stream

Function name:	bool DWD_get_gdd_data_stream(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned char *poutput, unsigned int16 timeout_delay, int handle)
Parameters:	Opcode: Must be to "*dwd_gdd_dump_cam_picture*". The opcode can be found in dwdptest.h. Nof_input_bytes: Should be set to zero, since this opcode doesn't have any input data. *pinput: Should also be set to zero. *poutput: Fetched the picture. Timeout_delay: dwdio.dll time out delay 1000 = 1 sec.

Returns:	Result of the operation: *ok or error*
Description	This function can be used to reading a picture from the mobile to PC in a fast way.
Platform	MP1
Mode:	Default mode
Atcptest:	atctst_gdd_generic_req

6.9.5.4 DWD_gdd_get_lcd_id

Function name:	bool DWD_gdd_get_lcd_id(unsigned int16 hw_id, unsigned int32 *lcd_id, unsigned int32 *result, int handle);
Parameters:	Hw_id: Must be set to 250 (GDD_MAIN_LCD) to get the LCD ID for the main LCD. Lcd_id: Fetched the value received from the mobile. Result: Fetched the result code from the GDD driver. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to the the LCD ID for a given display.
Platform	BP30
Mode:	Default mode
Atcptest:	Atctst_gdd_generic_req

6.10 Key test

6.10.1 DWD_get_current_key_matrix

Function name:	Bool DWD_get_current_key_matrix(unsigned int32 *keymatrix, int handle)
Parameters:	Keymatrix: Reference fetched the key matrix value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read out the current key matrix value.
Platform	P2002, P2002+,BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_poll_key_board_result_req

6.10.2 DWD_get_current_key_matrix_64bit

Function name:	bool DWD_get_current_key_matrix_64bit(unsigned int32 *key_matrix_bit0_31,

	unsigned int32 *key_matrix_bit32_63 , int handle)
Parameters:	Keymatrix0_31: Reference fetched the key matrix value for the first 32 bit. Keymatrix32_63: Reference fetched the key matrix value for the last 32bit.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read out the current key matrix value.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_poll_64bit_key_board_result_req

6.10.3 DWD_get_keyflip_status

Function name:	Bool DWD_get_keyflip_status(unsigned int16 *status, int handle)
Parameters:	Status: Reference fetched the status of the key flip.
Returns:	Result of the operation: *ok or error*
Description	This function is used the get the status of the key flip
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_get_keyflip_status_req

6.10.4 DWD_get_keypad_reference_matrix

Function name:	Long DWD_get_keypad_reference_matrix(void)
Parameters:	Return current “reference matrix”.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read out the “reference matrix” in use.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	N/A

6.10.5 DWD_get_keypad_reference_matrix_64

Function name:	long DWD_get_keypad_reference_matrix_64(void)
Parameters:	Return current “reference matrix”.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read out the “reference matrix” in use.
Platform	MP1
Mode:	Default test mode
Atcptest:	N/A

6.10.6 DWD_set_keypad_reference_matrix

Function name:	Bool DWD_set_keypad_reference_matrix(unsigned int32 matrix)
Parameters:	Matrix: Reference matrix to test against
Returns:	Result of the operation: *ok or error*
Description	This function is used to configure the “reference matrix” of the keypad. This function will allow the flexibility of testing different keypad layouts.

	For SGOLD testing is it the first 32bit in the matrix there are set.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	N/A

6.10.7 DWD_set_keypad_reference_matrix_64

Function name:	bool DWD_set_keypad_reference_matrix_64(unsigned int32 matrix)
Parameters:	Matrix: Reference matrix to test against
Returns:	Result of the operation: *ok or error*
Description	This function is used to configure the “reference matrix” of the keypad. This function will allow the flexibility of testing different keypad layouts. **Set the last 32bit (bit32-63) in the reference matrix**
Platform	MP1
Mode:	Default test mode
Atcptest:	N/A

6.10.8 DWD_poll_keyboard_result

Function name:	Bool DWD_poll_keyboard_result(bool *result, unsigned int32 *keymatrix, int handle)
Parameters:	Result: *true* if the “pressed key matrix” matches the “reference matrix” otherwise *false*. keymatrix: fetched the actual matrix value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to readout the “pressed key matrix” of the MS and compares it to the “reference matrix”.
Platform	P2002, P2002+, BP2
Mode:	Default test mode, ULC
Atcptest:	atctst_poll_key_board_result_req

6.10.9 DWD_poll_keyboard_result_64bit

Function name:	bool DWD_poll_keyboard_result_64bit(bool *result0_31, bool *result32_63, unsigned int32 *keymatrix0_31, unsigned int32 *keymatrix32_63, int handle)
Parameters:	*result0_31: Fetched the result for the first 32 bit *resutl32_63: Fetched the result for the last 32bit *keymatrix0_31: Fetched the actual key matrix value for the first 32bit. *keymatrix32_63: Fetched the actual key matrix value for the last 32 bit.
Returns:	Result of the operation: *ok or error*
Description	This function is used to readout the “pressed key matrix” of the MS and compares it to the “reference matrix”.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_poll_key_board_result_req

6.10.10 DWD_start_key_board_test

Function name:	Bool DWD_start_key_board_test(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to initiate the keyboard test. It resets the "pressed key matrix" of the MS. When a key is pressed the corresponding bit in the "pressed key matrix" will be set if the key is working. This means that all keys pressed after the call of *DWD_start_key_board_test* will be registered in the "pressed key matrix"
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_start_key_board_test_req

6.10.11 DWD_get_on_key_status

Function name:	Bool DWD_get_on_key_status(unsigned int16 *status, int handle)
Parameters:	Status: fetched the status of the on key.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the on key status.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_get_on_key_status_req

6.10.12 DWD_poll_key_matrix_result

Function name:	bool DWD_poll_key_matrix_result(unsigned int32 *key_matrix, int handle)
Parameters:	Key_matrix: fetched the key_matrix value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the result of the key matrix.
Platform	BP30
Mode:	Default test mode
Atcptest:	Atctst_poll_key_matrix_result_req

6.11 Audio test

Below are listed the audio specific dwdio.dll functions. In case the required functionality is not include please refer to Chapter 13.1.11 regarding DWD_aud_generic and relevant audio documents.

6.11.1 DWD_get_audio_data

Function name:	DWD_get_audio_data(**DWD_get_audio_data_con_type** *data, int handle)
Parameters:	Data: Reference fetched the audio data typedef struct { signed int16 filter_in_1[5]; signed int16 filter_in_2[5]; signed int16 filter_out_1[5];

	signed int16 filter_out_2[5]; signed int16 scal_in; signed int16 scal_out; signed int16 scal_mic; signed int16 scal_rec; signed int16 gain_out; signed int16 mic_mute_on; signed int16 side_tone_fact; signed int16 side_tone_val; signed int16 fill_1[4]; signed int16 hf_on; signed int16 hf_analog_gain; signed int16 fill_2[3]; signed int16 dtx_on; signed int16 mix_fact_speech; signed int16 mix_fact_tone; signed int16 hf_step_width; signed int16 hf_lms_length; signed int16 hf_lms_offset; signed int16 hf_block_length; signed int16 hf_rxtx_relation; signed int16 fill_3[3]; signed int16 hf_add_atten; signed int16 hf_min_atten; signed int16 hf_max_atten; signed int16 vgtx; signed int16 mic_1_2_on; /* 0 = mic 1, otherwise mic 2 */ signed int16 vg1_on; signed int16 vg2_on; signed int16 vg1rx; signed int16 vg2rx; unsigned char aud_main_state; unsigned char aud_buz_state; unsigned char aud_ear_state; unsigned char aud_mic_state; unsigned char audio_mode; }**DWD_get_audio_data_con_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the audio data from the DUT.
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_get_audio_data_req

6.11.2 DWD_set_audio

Function name:	DWD_set_audio(bool on, unsigned char volume, int handle); // Volume must be in the interval [0;8]
Parameters:	On: If *true* switch on the audio path. If *false* switch off the audio path. Volume: Must be in the interval [0…8]
Returns:	Result of the operation: *ok or error*
Description	This function is used to *switch on* or *switch off* the audio. The DUT must be in "Non-signaling" test mode before using this command.
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_set_audio_req

6.11.3 DWD_set_audiomode

Function name:	Bool DWD_set_audiomode(unsigned char mode, int handle)
Parameters:	Mode: audio mode
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the DUT in different audio mode. For the definition of mode please refer to the DWD document
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_aud_set_audiomode_req

6.11.4 DWD_set_earpiece

Function name:	Bool DWD_set_ear_piece(**dwd_audio_mode_type** mode, unsigned char volume, int handle)
Parameters:	Mode: Normal, headset or hands free. Volume: Must be in the interval [0…8] typedef enum { dwd_audio_mode_normal, dwd_audio_mode_headset, dwd_audio_mode_handsfree }**dwd_audio_mode_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to select internal or external ear piece. The DUT must be in "Non-signaling" test mode before using this command. **(Used by the old audio concept)**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_set_earpiece_req

6.11.5 DWD_set_mic

Function name:	Bool DWD_set_mic(**dwd_audio_mode_type** mode, int handle)
Parameters:	Mode: Normal,headset or handsfree

	typedef enum { dwd_audio_mode_normal, dwd_audio_mode_headset, dwd_audio_mode_handsfree }**dwd_audio_mode_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to select internal or external microphone. The DUT must be in "Non-Signaling" test mode before using this command. **(Used by the old audio concept)**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_set_mic_req

6.11.6 DWD_set_speakeramp_handsfree

Function name:	Bool DWD_set_speakeramp_handsfree(unsigned char mode, int handle)
Parameters:	Mode: Normal =0, Headset =1 or handsfree = 2.
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the speakeramp in different modes
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_set_speakeramp_handsfree_req

6.11.7 DWD_set_volumestep

Function name:	Bool DWD_set_volumestep(unsigned char step, int handle)
Parameters:	Step: Volume step value between 0…8
Returns:	Result of the operation: *ok or error*
Description	This function is used to set the volume in the DUT.
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_aud_set_volumestep_req

6.11.8 DWD_start_buzzer_tune

Function name:	Bool DWD_start_buzzer_tune(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to start the buzzer. **This function will only work if #define NEW_AUDIO_DRIVER_KONCEPT NOT has been used in the target SW.**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_start_buzzer_tune_req

6.11.9 DWD_stop_buzzer_tune

Function name:	Bool DWD_stop_buzzer_tune(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to stop the buzzer. **This function will only work if #define NEW_AUDIO_DRIVER_KONCEPT NOT has been used in the target SW.**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_stop_buzzer_tune_req

6.11.10 DWD_start_melody_tune

Function name:	Bool DWD_start_melody_tune(unsigned int16 tune, unsigned char volume, int handle)
Parameters:	Tune: melody number. Volume: 0…7. (Value 0 -> 5 is normal ringer volume steps, value 6 & 7, is special volume steps.
Returns:	Result of the operation: *ok or error*
Description	This function is used to start the melody ringer.
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_start_melody_tune_req

6.11.11 DWD_start_melody_tune_hp

Function name:	Bool DWD_start_melody_tune_hp(unsigned int16 tune, unsigned int16 volume, unsigned int16 timeout, int handle)
Parameters:	Tune: melody number. Volume: 0…7. (Value 0 -> 5 is normal ringer volume steps, value 6 & 7, is special volume steps. Timeout: time in millisecond.
Returns:	Result of the operation: *ok or error*
Description	This function is used to start the melody ringer, when a headset is connected. **This function can only be used if MA2 or MA3 ringer is present in the DUT.**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_start_melody_tune_hp_req

6.11.12 DWD_stop_melody_tune

Function name:	Bool DWD_stop_melody_tune(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to stop the melody ringer.
Platform	P2002, P2002+

Mode:	Default test mode
Atcptest:	atctst_stop_melody_tune_req

6.11.13 DWD_play_buzzer_tone

Function name:	Bool DWD_play_buzzer_tone(unsigned int16 freq, unsigned char duty_cycle, int handle)
Parameters:	Freq: Reference fetched the frequency. Duty_cycle: Reference fetched the duty cycle value
Returns:	Result of the operation: *ok or error*
Description	This function is used to play a tone on a given frequency and duty cycle. **The function can only be used together with a target SW where #define NEW_AUDIO_DRIVER_KONCEPT not are used.**
Platform	P2002, P2002+
Mode:	Default test mode
Atcptest:	atctst_play_buzzer_tone_req

6.11.14 DWD_start_intern_poly_ringer

Function name:	Bool DWD_start_intern_poly_ringer(unsigned char tone_id, unsigned int16 nof_repeats,int handle)
Parameters:	Tone_id: Melody number. Nof_repeats: Indicate how many times the melody should be played. If the value is zero, the melody will be played until DWD_stop_intern_poly_ringer is called.
Returns:	Result of the operation: *ok or error*
Description	This function is used to start the intern poly ringer.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.15 DWD_stop_intern_poly_ringer

Function name:	bool DWD_stop_intern_poly_ringer(int handle)
Parameters:	No input except the comport handle.
Returns:	Result of the operation: *ok or error*
Description	This function should be used to stop playing the intern poly ringer.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.16 DWD_start_extern_poly_ringer

Function name:	bool DWD_start_extern_poly_ringer(unsigned char tone_id, unsigned int16 nof_repeats, int handle)
Parameters:	Tone_id: The melody number you want to play. Nof_repeats: Indicate how many times the melody should be played. If the value is zero, the tone will be played until DWD_stop_extern_poly_ringer is called.
Returns:	Result of the operation: *ok or error*

Description	This function is used to start the extern poly ringer.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.17 DWD_stop_extern_poly_ringer

Function name:	bool DWD_stop_extern_poly_ringer(int handle)
Parameters:	No input except the comport handle.
Returns:	Result of the operation: *ok or error*
Description	This function should be used to stop playing the extern poly ringer.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.18 DWD_set_audio_uplink_path

Function name:	bool DWD_set_audio_uplink_path(unsigned int16 path, unsigned int16 enable, unsigned int16 *result, int handle)
Parameters:	Path: Indicate with path you want to setup (0-3) Enable: Indicate if it should be enabled (1) or disabled (0) *Result: Give you the result of the operation. Uplink path: 0 = Handset mic. 1 = Headset mic. 2 = I2S1_rx (Bluetooth mic.) 3 = TTY **Please note the uplink path number is project specific.**
Returns:	Result of the operation: *ok or error*
Description	This function is used to setup a uplink path in target. **The function DWD_update_audio_path shall be called after this command before it takes action on target side.**
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.19 DWD_set_audio_downlink_path

Function name:	bool DWD_set_audio_downlink_path(unsigned int16 path, unsigned int16 enable, unsigned int16 *result, int handle)
Parameters:	Path: Indicate with path you want to setup (0-9) Enable: Indicate if it should be enabled (1) or disabled (0) *Result: Give you the result of the operation. Downlink path:

	0 = Normal earpiece (Handset) 1 = Mono headset 2 = Stereo headset 3 = Back Speaker (Hands free speaker) 4 = I2S1 (Bluetooth earpiece) 5 = TTY 6 = Mono headset external ringer 7 = Stereo headet external ringer 8 = Back Speaker 9 = Earpiece 10 = Backspeaker external ringer 11 = HF carkit speaker 12 = I2S1 inband (Special path used for routing inband audio (like ringtones) to the Bluetooth headset. **Please note the downlink path number is project specific.**
Returns:	Result of the operation: *ok or error*
Description	This function is used to setup a downlink path in target. **The function DWD_update_audio_path shall be called after this command before it takes action on target side.**
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.20 DWD_update_audio_path

Function name:	Bool DWD_update_audio_path(int handle)
Parameters:	No input except comport handle
Platform	MP1
Description:	This function is used to update the target, so the new uplink or downlink path setting takes action.
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.11.21 DWD_set_audio_parms

Function name:	bool DWD_set_audio_parms(unsigned int16 parm_id, signed int16 *data, unsigned int16 nof_bytes, int handle)
Parameters:	
Platform	All platform
Description:	This function is used to set some audio parameter depending on which PARM ID is used.
Mode:	Default test mode
Atcptest:	Atctst_set_audio_parm_req

6.11.22 DWD_start_buzzer_melody

Function name:	bool DWD_start_buzzer_melody(unsigned char ring_tune, unsigned char duty_cycle, unsigned int16 repeats, int handle)
Parameters:	Ring_tune: Duty_cycle: Repeats:
Platform	P2002
Description:	The function is used to start the buzzer melody
Mode:	Default test mode
Atcptest:	atctst_start_buzzer_melody_req

6.12 USB

6.12.1 DWD_usb_attach_status

Function name:	bool DWD_usb_attach_status(unsigned int16 *status, int handle);
Parameters:	*status: Reference to the result
Platform	MP1
Description:	This function is used to check for USB cable are attached or not to target.
Mode:	Default test mode
Atcptest:	atctst_usb_attach_status_req

6.13 Bluetooth

6.13.1 DWD_btd_enter_rf_test_mode

Function name:	bool DWD_btd_enter_rf_test_mode(bool *result, int handle)
Parameters:	*result: Reference to the result
Returns:	Result of the operation: *ok or error*
Description:	This function configures the RF test mode of the BTD. After this further control is done via the Blue Tooth System Simulator. DWD_btd_sw_reset should be given to exit the RF test mode.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.2 DWD_btd_sw_reset

Function name:	bool DWD_btd_sw_reset(bool *result, int handle)
Parameters:	*result: Reference to the result
Returns:	Result of the operation: *ok or error*
Description:	Initiates a SW reset of the blue tooth device. This command will bring the Blue Tooth Device back to initial condition ready to receive HCI command. NOTE: any change of bd data in NV memory will not take effect when calling DWD_btd_sw_reset. The values loaded at power On of the DUT will still be valid.
Platform	MP1

Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.3 DWD_btd_setup_pcm_loopback

Function name:	bool DWD_btd_setup_pcm_loopback(bool *turn_on_off, int handle)
Parameters:	*turn_on_off: Enable or disable pcm loopback mode
Returns:	Result of the operation: *ok or error*
Description:	This function is used to setup the PCM loopback mode
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.4 DWD_btd_check_host_interface

Function name:	bool DWD_btd_check_host_interface(bool *result, int handle)
Parameters:	*result: Fetched the result of checking the host interface
Returns:	Result of the operation: *ok or error*
Description:	This function checks the HOST interface of the BTD (UARTRX, UARTTX, UARTRTS and UARTCTS). DWD_btd_sw_reset should be given to reset the host interface and the BTD is ready for the next test step like check of PCM interface.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.5 DWD_btd_check_pcm_interface

Function name:	bool DWD_btd_check_pcm_interface(bool *error, unsigned char period, int handle)
Parameters:	*error: Fetched the error cause of this operation Period: Period in seconds to collect the PCM signal looped back from the Blue tooth device. The range are1 to 30 seconds.
Returns:	Result of the operation: *ok or error*
Description:	This function 1) configures the BTD in PCM loop back mode 2) requests the host to start sending PCM data to the BTD 3) routes the PCM signal to the appropriate audio speaker. This function will automatically take care of the associated audio path settings. DWD_btd_sw_reset should be given to exit the PCM loopback and the BTD is ready to enter next test step like RF test mode.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_aud_generic_func_req

6.13.6 DWD_btd_set_tx_burst_mode

Function name:	bool DWD_btd_set_tx_burst_mode(unsigned char packet, unsigned char tx_interval, unsigned char tx_channel, unsigned char tx_coarse, unsigned char tx_fine, unsigned char pattern, int handle)
Parameters:	Packet: Set the packet type (DH1 = 1, DH3 = 3, -DH5 = 5) Tx_interval: Set the TX interval in millisecond (0 = 1.25ms -> 0xEA = 293.75 ms

	which is the maximum value). Tx_channel: Set the TX channel (Value between 0 -> 78 is legal value). Tx_coarse: Set the TX power coarse (Values between 0 to 3 is legal values). Tx_fine: Set the TX power fine (Values between 0 to 3 is legal values). Pattern: Set the pattern (values between 0 to 3 is legal values) Pattern 0: None Pattern 1: 01010101 Pattern 2: 00001111 Pattern 3: PRBS 9
Returns:	Result of the operation: *ok or error*
Description:	This function is used to set up TX burst mode.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.7 DWD_btd_set_tx_cw_mode

Function name:	bool DWD_btd_set_tx_cw_mode(unsigned char tx_channel, unsigned char tx_coarse, unsigned char tx_fine, unsigned char pattern, int handle)
Parameters:	Tx_channel: Set the TX channel (Value between 0 -> 78 is legal value). Tx_coarse: Set the TX power coarse (Values between 0 to 3 is legal values). Tx_fine: Set the TX power fine (Values between 0 to 3 is legal values). Pattern: Set the pattern (values between 0 to 3 is legal values) Pattern 0: None Pattern 1: 01010101 Pattern 2: 00001111 Pattern 3: PRBS 9
Returns:	Result of the operation: *ok or error*
Description:	This function is used to set up TX continuous (cw) mode.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.8 DWD_btd_set_rx_cw_mode

Function name:	bool DWD_btd_set_rx_cw_mode(unsigned char rx_channel , int handle)
Parameters:	Rx_channel: Set the RX channel (Value between 0 -> 78 is legal value).
Returns:	Result of the operation: *ok or error*
Description:	This function is used to set up RX continuous (cw) mode.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.9 DWD_btd_get_bluemoon_firmware_version

Function name:	Bool DWD_btd_get_bluemoon_firmware_version(void *version, int handle)

Parameters:	*version: Fetched the Blue Moon firmware version.
	
	typedef struct
	{
	unsigned char lm_chip;
	unsigned char lm_x;
	unsigned char lm_y;
	unsigned char lm_z;
	unsigned char bb_chip;
	unsigned char bb_y;
	unsigned char bb_x;
	}**dwd_btd_bms_firmware_version_type**; // Bluemoon single cellular
	
	typedef struct
	{
	unsigned char hw_variant;
	unsigned char hw_revision;
	unsigned char hw_sub_revision;
	unsigned char fw_variant;
	unsigned char fw_revision;
	unsigned char fw_sub_revision;
	unsigned char fw_build_number1;
	unsigned char fw_build_number2;
	unsigned char fw_build_number3;
	unsigned char fw_patch_number;
	}**dwd_btd_bmu_firmware_version_type**; // Bluemoon universal

Returns:	Result of the operation: *ok or error*
Description:	This function is used to get the Blue Moon firmware version. Only one of the structure above shold be used to get the version information. It depends on the variant of the bluemoon chip.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.13.10 DWD_btd_get_hw_info

Function name:	bool DWD_btd_get_hw_info(unsigned int16 *bluemoon_type, unsigned int16 *serial_interface_type, int handle);
Parameters:	Bluemoon_type: fetched the bluemoon chip variant. Serial_interface: fetched the serial interface type. Bluemoon_type: 5 means Bluemoon single cellular Bluemoon_type: 3 means Bluemoon universal Serial_interface_type: 1 means UART Serial_interface_type: 2 measn USIF

Returns:	Result of the operation: *ok or error*
Description:	This function is used to get some hardware info for the Bluemoon chip
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_bt_generic_func_req

6.14Additional functions

6.14.1 DWD_battery_status

Function name:	Bool DWD_battery_status(unsigned char *dest, int handle)
Parameters:	*dest: Reference to a structure fetched all parameters. The structure can be found in the atctstif.h which is a part of DWD driver and the name of the structure is atctst_chr_debug_info_type.
Platform	P2002, P2002+, MP1, BP2
Description:	**The function can only be used if #define BATTERY_WITH_FUEL_GAUGE not are used in the target SW.**
Mode:	Default test mode
Atcptest:	atctst_get_capacity_status_req

6.14.2 DWD_charger_status

Function name:	Bool DWD_charger_status(unsigned char *dest, int handle)
Platform	N/A
Description:	**Will not work on a MODEM build and the #define BATTERY_WITH_FUEL_GAUGE should also have be set.**
Mode:	Default test mode
Atcptest:	atctst_get_charger_status_req

6.14.3 DWD_capacity_status_direct

Function name:	bool DWD_capacity_status_direct(unsigned int16 *capacity_state, unsigned int16 *level, unsigned int16 *availcap, unsigned int16 *lastdchg, unsigned int16 *compcap, unsigned int16 *ccr, unsigned int16 *dcr, int handle)
Platform	N/A
Description	**The #define BATTERY_WITH_FUEL_GAUGE must be set before this function works.**
Mode:	Default test mode.
Atcptest:	atctst_get_capacity_status_req

6.14.4 DWD_charger_status_direct

Function name:	bool DWD_charger_status_direct(unsigned int16 *battery_id, unsigned int16 battery_voltage, unsigned int16 *battery_low_warning, signed char *battery_temperature, signed char *environment_temperature, unsigned int16

	*charger_on,unsigned int16 *gl1_mode, int handle)
Platform	N/A
Description	**Will not work on a MODEM build and the #define BATTERY_WITH_FUEL_GAUGE should also have be set.**
Mode:	Default test mode
Atcptest:	atctst_get_charger_status_req

6.14.5 DWD_check_gg

Function name:	Bool DWD_check_gg(bool *ok, int handle)
Parameters:	ok: *true* if connection works and *false* if it fails.
Returns:	Result of the operation: *ok or error*
Description	This function checks the communication link to the Gas Gauge. **Will not work on a MODEM build and the #define BATTERY_WITH_FUEL_GAUGE should also have be set.**
Platform	N/A
Mode:	Default test mode
Atcptest:	atctst_check_gg_req

6.14.6 DWD_get_battery_info

Function name:	Bool DWD_get_battery_info(unsigned char *dest, int handle)
Platform	N/A
Description:	**Will not work on a MODEM build and the #define BATTERY_WITH_FUEL_GAUGE should also have be set.**
Mode:	Default test mode
Atcptest:	atctst_get_battery_info_req

6.14.7 DWD_external_keypress

Function name:	Bool DWD_external_keypress(signed int16 key, int handle)
Parameters:	Key: fetched the key number you want to simulate.
Returns:	Result of the operation: *ok or error*
Description	This function is used to simulate key presses over the serial communication.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_external_keypress_req

6.14.8 DWD_external_flip_key_simulation

Function name:	bool DWD_extern_flip_key_simulation(signed int16 keyflip_state, int handle)
Parameters:	Keyflip_state: 1 = Open key flip 0 = Closed key flip
Returns:	Result of the operation: *ok or error*
Description	This function is used to simulate key flip over the serial communication.
Platform	MP1

Mode:	Default test mode
Atcptest:	atctst_ext_keyflip_req

6.14.9 DWD_external_long_keypress

Function name:	bool DWD_external_longkeypress(signed int16 key_value, unsigned int32 delay, int handle)
Parameters:	Key_value: The key number you want to simulate. Delay: time delay indication how long time the key should be hold down (1 = 10 milisecond)
Returns:	Result of the operation: *ok or error*
Description	This function is used to simulate long key press over the serial communication.
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_ext_long_key_press_req

6.14.10 DWD_get_bb_version_revision

Function name:	Bool DWD_get_bb_version_revision(dwd_bb_ver_enum_type *bb_version, dwd_bb_rev_enum_type *bb_revision, int handle)
Parameters:	*bb_version: Reference fetched the base band chip version *bb_revision: Reference fetched the base band chip revision. typedef enum { dwd_bb_ver_invalid, dwd_bb_ver_sgold, dwd_bb_ver_sgoldlite, dwd_bb_ver_sgold2 }dwd_bb_ver_enum_type; typedef enum { dwd_bb_rev_invalid, dwd_bb_rev10, dwd_bb_rev11, dwd_bb_rev11a, dwd_bb_rev11b, dwd_bb_rev12 }dwd_bb_rev_enum_type;
Returns:	Result of the operation: *ok or error*
Description	This function can be used to get the base band chip version and revision. **It can only be used together with SGOLD family based platforms.**
Platform	MP1
Mode:	Default test mode
Atcptest:	atctst_get_bb_version_revision_req

6.14.11 DWD_mmci_check

Function name:	bool DWD_mmci_hw_check(signed int16 *result, int handle)
Parameters:	Result: fetched the result for this check
Returns:	Result of the operation: *ok or error*
Description	This function is used to check for MMC card is connected or not
Platform	P2002
Mode:	Default test mode
Atcptest:	atctst_mmci_generic_func_req

6.14.12 DWD_set_power_down_mode

Function name:	bool DWD_set_power_down_mode(unsigned int16 mode, unsigned int32 nof_frames, int handle)
Parameters:	Mode: Nof_frames:
Returns:	Result of the operation: *ok or error*
Description	This function is used set the the DUT in power down mode
Platform	BP30
Mode:	Default test mode
Atcptest:	atctst_set_power_down_mode_req

6.14.13 DWD_get_egoldradio_version_revision

Function name:	bool DWD_get_egoldradio_version_revision(**dwd_bb_ver_enum_type** *bb_version, **dwd_bb_EGR_rev_enum_type** *bb_revision, int handle)
Parameters:	*bb_version: Reference fetched the base band chip version *bb_revision: Reference fetched the base band chip revision. typedef enum { dwd_bb_ver_invalid, dwd_bb_ver_sgold, dwd_bb_ver_sgoldlite, dwd_bb_ver_sgold2 }**dwd_bb_ver_enum_type**; typedef enum { dwd_bb_rev_unknown, dwd_bb_rev22f, dwd_bb_rev22g, dwd_bb_rev22h } **dwd_bb_EGR_rev_enum_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used get the EGOLDRADIO chio version and revision
Platform	BP30
Mode:	Default test mode
Atcptest:	atctst_get_bb_version_revision_req

6.14.14 DWD_chr_get_measurement_control_status

Function name:	bool DWD_chr_get_measurement_control_status(unsigned int16 *state, unsigned int16 *battery_id, unsigned int16 *mean_vbat_idle, int handle)
Parameters:	State: fetched the state of measurement control state machine Battery_id: fethed the battery ID Mean_vbat_idle: fetched the result of ADC measurement after compensation and averaging.
Returns:	Result of the operation: *ok or error*
Description	This function is used set the the DUT in power down mode
Platform	BP30
Mode:	Default test mode
Atcptest:	atctst_set_power_down_mode_req

6.14.15 DWD_get_hw_coding

Function name:	bool DWD_get_hw_coding(unsigned int16 *hw_coding, unsigned int16 *result, int handle)
Parameters:	Hw_coding: fetched the hw_coding read from the DUT. Result: Keep information about the hw_coding is valid or not.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the hw_coding from the DUT.
Platform	BP30
Mode:	Default test mode
Atcptest:	atctst_hw_coding_req

7 Security

7.1 DWD_get_ms_id

Function name:	Bool DWD_get_ms_id(**dwd_ms_id_type** *id_ptr, int handle)
Parameters:	Id_ptr: Reference fetched the IMEI and serial number typedef struct { signed char imei[16]; unsigned int32 serial_number; } **dwd_ms_id_type**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the IMEI number and the serial number from the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_ms_id_req

7.2 DWD_Program_pers_code

Function name:	Bool DWD_program_pers_code(unsigned char nof_input_bytes, unsigned char *pers_code, unsigned char *nof_output_bytes, unsigned char *poutput, int handle)
Parameters:	Nof_input_byte: Reference fetched the number of input byte in the input buffer. Pers_code: Reference fetched the input data. Nof_output_bytes: Reference fetched the number of bytes in output buffer. Poutput: Reference fetched the output data buffer.
Returns:	Result of the operation: *ok or error*
Description	This function is used to program personalization lock to the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode, MMI test mode
Atcptest:	atctst_mp_transfer_pers_data_req

7.3 DWD_sec_user_cmd

Function name:	Bool DWD_sec_user_cmd(unsigned char *data, unsigned char length, void (*rxdone)(unsigned char *rxdata,int16 rx_length),int handle)
Parameters:	Data: reference fetched the data to be transmitted. Length: number of bytes to sent (*rxdone)(unsigned char *rxdata, unsigned char rx_length): reference to function to be called when the MS confirms the user command. In case 'NULL' is given as call back function parameter no call back function is called.
Returns:	Result of the operation: *ok or error*
Description	This function is used to program the security library, if the is supported in the project.
Platform	P2002+ , MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_sec_user_cmd_req

7.4 DWD_set_cust_key

Function name:	Bool DWD_set_cust_key(**dwd_cust_key_type** *key)
Parameters:	Key: Reference fetched the customer key. typedef struct { unsigned char key[16]; } **dwd_cust_key_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to set other customer keys. The function overwrites the input from the cust_key.sap if the input not is set to zero. **It not a platform depend function**
Platform	N/A
Mode:	N/A

7.5 DWD_store_imei

Function name:	Bool DWD_store_imei(signed char imei[14], bool *already_programmed, int handle)
Parameters:	imei: ASCII string with the IMEI to be stored in DUT. Example: “12345678901234”. The IMEI consists of 15 digits, where the last digit it the Check Digit. This digit is calculated by the DLL as specified in GSM 02.16 v. 7.2.0 release 1998, and is automatically appended to the given 14 digits, thereby yielding the complete 15-digit sequence. Only the first 14 digits of the input string are read. already_programmed: Indicates if an IMEI has already been programmed. True, means that the operation has failed because an IMEI is already present, false means that no IMEI is present.
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the IMEI in the DUT. IMPORTANT NOTE: The IMEI can only be programmed once.
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_program_imei_req

7.6 DWD_get_sec_version

Function name:	Bool DWD_get_sec_version(unsigned char *sec_version, int handle)
Parameters:	sec_version: reference fetched the security version
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the security version
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode
Atcptest:	atctst_get_sec_revision_req

8 Non-volatile memory (NV)

8.1 Production parameters

8.1.1 DWD_get_config

Function name:	Bool DWD_get_config(unsigned char config_id, **dwd_config_type** *config, int handle)
Parameters:	Config_id: *dwd_original_config* or *dwd_updated_config* Config: Reference to where the requested configuration must be placed. typedef struct { unsigned int16 id; unsigned int16 hw_version; unsigned char eep_version; unsigned char eep_revision; } **dwd_config_type**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the configuration parameters from the EEPROM. *dwd_original_config:* *eep_static.production_parms.orig_conf.id* *eep_static.production_parms.orig_conf.hw_version* *eep_static.production_parms.orig_conf.eep_version.ver* *eep_static.production_parms.orig_conf.eep_version.rev* *dwd_updated_config:* *eep_static.production_parms.upd_conf.id* *eep_static.production_parms.upd_conf.hw_version* *eep_static.production_parms.upd_conf.eep_version.ver* *eep_static.production_parms.upd_conf.eep_version.rev*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.2 DWD_store_config

Function name:	Bool DWD_store_config(unsigned char config_id, **dwd_config_type** *config, int handle)
Parameters:	config_id: *dwd_original_config* or *dwd_updated_config* config: Reference to where the requested configuration must be placed. typedef struct { unsigned int16 id; unsigned int16 hw_version;

	unsigned char eep_version; unsigned char eep_revision; } **dwd_config_type**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the specified "config" in the DWD DLL until the DWD_store_to_nv_memory is called. The *dwd_original_config* is supposed to be programmed in the production line while the *dwd_updated_config* is supposed to be programmed after repair or SW update. *dwd_original_config:* *eep_static.production_parms.orig_conf.id* *eep_static.production_parms.orig_conf.hw_version* *eep_static.production_parms.orig_conf.eep_version.ver* *eep_static.production_parms.orig_conf.eep_version.rev* *dwd_updated_config:* *eep_static.production_parms.upd_conf.id* *eep_static.production_parms.upd_conf.hw_version* *eep_static.production_parms.upd_conf.eep_version.ver* *eep_static.production_parms.upd_conf.eep_version.rev*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.3 DWD_get_customer_parms

Function name:	Bool DWD_get_customer_parms(**dwd_customer_parm_type** *customer, int handle)
Parameters:	Customer: This is the reference to the customer specific parameters typedef struct { unsigned int16 param_1; unsigned int16 param_2; unsigned int16 param_3; unsigned int16 param_4; unsigned int16 param_5; unsigned int16 param_6; unsigned int16 param_7; unsigned int16 param_8; unsigned int16 param_9; } **dwd_customer_parm_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the customer specific parameters. *eep_static.production_parms.cust_parms.param_1…9*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.4 DWD_store_customer_parms

Function name:	Bool DWD_store_customer_parms(**dwd_customer_parm_type** *customer, int handle)

Parameters:	Customer: Hold the new values. typedef struct { unsigned int16 param_1; unsigned int16 param_2; unsigned int16 param_3; unsigned int16 param_4; unsigned int16 param_5; unsigned int16 param_6; unsigned int16 param_7; unsigned int16 param_8; unsigned int16 param_9; } **dwd_customer_parm_type**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the customer specific parameters. *eep_static.production_parms.cust_parms.param_1…9*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.5 DWD_get_pn_number

Function name:	Bool DWD_get_pn_number(**dwd_pn_type** *number, int handle)
Parameters:	Number: Reference to a store of **dwd_pn_type** where the production number should be returned. typedef struct { unsigned char number[20]; } **dwd_pn_type**;
Returns:	Result of the operation: *ok or error*
Description	This function get the pn_number *eep_static.pn.number[0]…[19]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.6 DWD_store_pn_number

Function name:	Bool DWD_store_pn_number(**dwd_pn_type** *number, int handle)
Parameters:	Number: Reference to a store of **dwd_pn_type.** typedef struct { unsigned char number[20]; } **dwd_pn_type**;
Returns:	Result of the operation: *ok or error*
Description	This function store new pn_number parameter *eep_static.pn.number[0]…[19]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.7 DWD_get_production_date

Function name:	Bool DWD_get_production_date(**dwd_production_date_type** *date, int handle)
Parameters:	Date: Reference to where the requested *date* must be placed. typedef struct { unsigned char year; unsigned char month; unsigned char week; unsigned char day; } **dwd_production_date_type**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the production date parameter. *eep_static.production_parms.date.year* *eep_static.production_parms.date.month* *eep_static.production_parms.date.week* *eep_static.production_parms.date.day*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.8 DWD_store_production_date

Function name:	Bool DWD_store_production_date(**dwd_production_date_type** *date, int handle)
Parameters:	Date: Reference to where the requested *date* must be placed. typedef struct { unsigned char year; unsigned char month; unsigned char week; unsigned char day; } **dwd_production_date_type**;
Returns:	Result of the operation: *ok or error*
Description	This function store new value to production date parameter. *eep_static.production_parms.date.year* *eep_static.production_parms.date.month* *eep_static.production_parms.date.week* *eep_static.production_parms.date.day*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.9 DWD_get_serial_number

Function name:	Bool DWD_get_serial_number(unsigned int32 *serial_number, int handle)
Parameters:	Serial_number: Reference to where the serial number must be placed.
Returns:	Result of the operation: *ok or error*
Description	This function gets the serial number. *eep_static.production_parms.serial_no*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.10 DWD_store_serial_number

Function name:	Bool DWD_store_serial_number(unsigned int32 serial_number, int handle)
Parameters:	Serial_number: Variable holding the new value.
Returns:	Result of the operation: *ok or error*
Description	This function stores a new serial number. *eep_static.production_parms.serial_no*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.11 DWD_get_test_stations_parms

Function name:	Bool DWD_get_test_station_parms(unsigned char test_station_id, **dwd_test_station_type** *parms, int handle)

Parameters:	Test_station_id: The different id's of the testers are
	
	enum dwd_test_modes
	{
	dwd_in_line_mode,
	dwd_calib_mode,
	dwd_mmi_mode
	};
	
	Parms: Reference to where the requested test station parameters must be placed.
	
	typedef struct
	{
	unsigned int16 result;
	unsigned int16 nof_tests;
	unsigned int16 version;
	unsigned int16 sw_version;
	unsigned int16 spec;
	unsigned int16 testsite_id;
	} **dwd_test_station_type**;

Returns:	Result of the operation: *ok or error*
Description	This function gets the test station parameters for the specified tester. *dwd_in_line_mode:* *eep_static.production_parms.in_line.result* *eep_static.production_parms.in_line.nof_tests* *eep_static.production_parms.in_line.version* *eep_static.production_parms.in_line.sw_version* *eep_static.production_parms.in_line.spec* *eep_static.production_parms.in_line.testsite_id* *dwd_calib_mode:* *eep_static.production_parms.calib.result* *eep_static.production_parms.calib.nof_tests* *eep_static.production_parms.calib.version* *eep_static.production_parms.calib.sw_version* *eep_static.production_parms.calib.spec*

	eep_static.production_parms.calib.testsite_id *dwd_mmi_mode:* *eep_static.production_parms.mmi.result* *eep_static.production_parms.mmi.nof_tests* *eep_static.production_parms.mmi.version* *eep_static.production_parms.mmi.sw_version* *eep_static.production_parms.mmi.spec* *eep_static.production_parms.mmi.testsite_id*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.12 DWD_store_test_stations_parms

Function name:	Bool DWD_store_test_station_parms(unsigned char test_station_id, **dwd_test_station_type** *parms, int handle);

Parameters:	Test_station_id: The different id's of the testers are
	
	enum dwd_test_modes
	{
	dwd_in_line_mode,
	dwd_calib_mode,
	dwd_mmi_mode
	};
	
	parms: Reference to the test station parameters to be stored.
	
	typedef struct
	{
	unsigned int16 result;
	unsigned int16 nof_tests;
	unsigned int16 version;
	unsigned int16 sw_version;
	unsigned int16 spec;
	unsigned int16 testsite_id;
	} **dwd_test_station_type**;

Returns:	Result of the operation: *ok or error*
Description	This function gets the test station parameters for the specified tester. *dwd_in_line_mode:* *eep_static.production_parms.in_line.result* *eep_static.production_parms.in_line.nof_tests* *eep_static.production_parms.in_line.version* *eep_static.production_parms.in_line.sw_version* *eep_static.production_parms.in_line.spec* *eep_static.production_parms.in_line.testsite_id* *dwd_calib_mode:* *eep_static.production_parms.calib.result* *eep_static.production_parms.calib.nof_tests*

	eep_static.production_parms.calib.version *eep_static.production_parms.calib.sw_version* *eep_static.production_parms.calib.spec* *eep_static.production_parms.calib.testsite_id* *dwd_mmi_mode:* *eep_static.production_parms.mmi.result* *eep_static.production_parms.mmi.nof_tests* *eep_static.production_parms.mmi.version* *eep_static.production_parms.mmi.sw_version* *eep_static.production_parms.mmi.spec* *eep_static.production_parms.mmi.testsite_id*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.13 DWD_get_test_series_id

Function name:	Bool DWD_get_testseries_id(unsigned int16 *testseries_id, int handle);
Parameters:	Testseries_id: Reference to where the testseries_id must be placed.
Returns:	Result of the operation: *ok or error*
Description	This function gets the parameter *testseries_id*. *eep_static.production_parms.testseries_id*
Platform	P2002, P2002+, MP1, BP2, ULC

8.1.14 DWD_store_test_series_id

Function name:	Bool DWD_store_testseries_id(unsigned int16 testseries_id, int handle);
Parameters:	Testseries_id: hold the new value
Returns:	Result of the operation: *ok or error*
Description	This function stores a new value. Testseries_id: testseries_id to be assigned to the DUT
Platform	P2002, P2002+, MP1, BP2, ULC

8.2 RF Parameters

8.2.1 DWD_get_pa_offset

Function name:	Bool DWD_get_pa_offset(unsigned char band, **dwd_pa_offset_comp_table_type** *pa_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_offsets_ptr: Reference to fetched dac16 offsets for all power levels for the specified band. typedef signed int16 **dwd_pa_offset_comp_table_type[16]**
Returns:	Result of the operation: *ok or error*
Description	This function gets for the specified band the stored pa dac16 compensation offsets stored in the EEPROM. *eep_static.rf_adjcomp.pa_offset[0][0]…[3][19]*.
Platform	P2002, P2002+, BP2, ULC

8.2.2 DWD_store_pa_offset

Function name:	Bool DWD_store_pa_offset(unsigned char band, **dwd_pa_offset_comp_table_type** *pa_offsets_ptr, int handle);
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_offsets_ptr: Reference to fetched dac16 offsets for all power levels for the specified band. typedef signed int16 **dwd_pa_offset_comp_table_type[16]**
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of *dac16 offsets* (offsets to be added to the final value of the up ramp) for the specified band. *eep_static.rf_adjcomp.pa_offset[0][0]…[3][19]*.
Platform	P2002, P2002+, BP2, ULC

8.2.3 DWD_get_vhome_offset

Function name:	Bool DWD_get_vhome_offset(unsigned char band, **dwd_pa_vhome_offset_comp_table_type** *vhome_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Vhome_offsets_ptr: Reference to table with fetched *VHOME offsets* for all power levels for the specified band. typedef unsigned char **dwd_pa_vhome_offset_comp_table_type**[16];
Returns:	Result of the operation: *ok or error*
Description	This function fetches for the specified band the stored *VHOME* offsets stored in EEPORM. *eep_static.rf_adjcomp.pa_v_home_offset[0][0]*
Platform	P2002, P2002+, BP2

8.2.4 DWD_store_vhome_offset

Function name:	Bool DWD_store_vhome_offset(unsigned char band, **dwd_pa_vhome_offset_comp_table_type** *vhome_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Vhome_offsets_ptr: Reference to table with fetched *VHOME offsets* for all power levels for the specified band. typedef unsigned char **dwd_pa_vhome_offset_comp_table_type**[16];
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of *VHOME offsets* (offset to be added to the start value of the up ramp) for the specified band. *eep_static.rf_adjcomp.pa_v_home_offset[0][0]*
Platform	P2002, P2002+, BP2

8.2.5 DWD_get_pa_timing_offset

Function name:	Bool DWD_get_pa_timing_offset(unsigned char band, **pa_timing_offset_table_type** *timing_offsets_ptr, int handle)

Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900.* Timing_offsets_ptr: References a table holding time offset for positioning the TX burst for all power levels for the specified band. typedef **dwd_pa_timing_elm_type pa_timing_offset_table_type[16];**
Returns:	Result of the operation: *ok or error*
Description	This function fetches for the specified band the stored *pa timing offsets* stored in EEPROM. *eep_static.rf_adjcomp.pa_timing_offset[0][0].rampup*
Platform	P2002, P2002+, MP1, BP2, ULC

8.2.6 DWD_store_pa_timing_offset

Function name:	bool DWD_store_pa_timing_offset(unsigned char band,pa_timing_offset_table_type *timing_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900.* Timing_offsets_ptr: References a table holding time offset for positioning the TX burst for all power levels for the specified band. typedef **dwd_pa_timing_elm_type pa_timing_offset_table_type[16];**
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of pa timing offsets for the specified band. *eep_static.rf_adjcomp.pa_timing_offset[0][0].rampup*
Platform	P2002, P2002+, MP1, BP2, ULC

8.2.7 DWD_get_max_pa_ch_comp

Function name:	bool DWD_get_max_pa_ch_comp(signed char *offset_ptr, int handle)
Parameters:	Offsets_ptr: Reference to the Max PA CH COMP value.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the max_pa_ch_comp value from the EEP *eep_static.rf_comp.pa_ch_comp[0][0]…[3][3]*
Platform	P2002, P2002+, BP2, ULC

8.2.8 DWD_store_max_pa_ch_comp

Function name:	Bool DWD_store_max_pa_ch_comp(unsigned char band, unsigned char ch_high_low, signed char offset, int handle);
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* ch_high_low: *dwd_low_ch* for the low range of channels of the specified band and *dwd_high_ch* for the high range of channels of the specified band. offsets: offset to be added to the DAC16 value for the max power for the given band.
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the channel compensation offset (offset to be added to the final value of the up ramp) for max power level of the specified band and channel range. *eep_static.rf_comp.pa_ch_comp[0][0]…[3][3]*
Platform	P2002, P2002+, BP2, ULC

8.2.9 DWD_get_gmsk_pa_ch_comp

Function name:	Bool DWD_get_gmsk_pa_ch_comp(unsigned char band, **pa_ch_comp_type** *pa_ch_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_ch_comp_ptr: References a table holding the channel compensation for all power levels for the specified band. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**; typedef **dwd_pa_ch_comp_elm_type pa_ch_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the table of channel compensation for all power levels for the specified band. *eep_static.gmsk_rf_comp.gmsk_pa_ch_comp[0][0][0]…[3][19][3]*
Platform	MP1

8.2.10 DWD_store_gmsk_pa_ch_comp

Function name:	Bool DWD_store_gmsk_pa_ch_comp(unsigned char band, **pa_ch_comp_type** *pa_ch_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_ch_comp_ptr: References a table holding the channel compensation for all power levels for the specified band. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**; typedef **dwd_pa_ch_comp_elm_type pa_ch_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of channel compensation for all power levels for the specified band *eep_static.gmsk_rf_comp.gmsk_pa_ch_comp[0][0][0]…[3][19][3]*
Platform	MP1

8.2.11 DWD_get_gmsk_pa_temp_comp

Function name:	Bool DWD_get_gmsk_pa_temp_comp(unsigned char band, **pa_temp_comp_type** *pa_temp_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_temp_comp_ptr: References a table holding the temperature compensation for all power levels for the specified band. typedef struct { signed char pa_temp_comp0; signed char pa_temp_comp1; signed char pa_temp_comp2; signed char pa_temp_comp3; signed char pa_temp_comp4; }**dwd_pa_temp_comp_elm_type**; typedef **dwd_pa_temp_comp_elm_type pa_temp_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the table of temperature compensation for all power levels for the specified band. *eep_static.gmsk_rf_comp.gmsk_pa_temp_comp[0][0][0]…[3][19][4]*
Platform	MP1

8.2.12 DWD_get_gmsk_pa_ch_comp_boundary

Function name:	Bool DWD_get_gmsk_pa_ch_comp_boundary(unsigned char band, **pa_ch_comp_limit_type** *limits_ptr, int handle);
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Limits_ptr: References a table holding the boundary value. typedef unsigned int16 **pa_ch_comp_limit_type[4]**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the boundary value for the specified band. *eep_static.gmsk_rf_comp2.gmsk_pa_ch_comp_limits[0][0] …[3][3]*
Platform	MP1

8.2.13 DWD_store_gmsk_pa_temp_comp

Function name:	Bool DWD_store_gmsk_pa_temp_comp(unsigned char band, **pa_temp_comp_type** *pa_temp_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_temp_comp_ptr: References a table holding the temperature compensation for all power levels for the specified band. typedef struct {

	signed char pa_temp_comp0; signed char pa_temp_comp1; signed char pa_temp_comp2; signed char pa_temp_comp3; signed char pa_temp_comp4; }**dwd_pa_temp_comp_elm_type**; typedef **dwd_pa_temp_comp_elm_type pa_temp_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of temperature compensation for all power levels for the specified band. *eep_static.gmsk_rf_comp.gmsk_pa_temp_comp[0][0][0]…[3][19][4]*
Platform	MP1

8.2.14DWD_get_edge_pa_ch_comp_boundary

Function name:	Bool DWD_get_edge_pa_ch_comp_boundary(unsigned char band, **pa_ch_comp_limit_type** *limits_ptr, int handle);
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Limits_ptr: References a table holding the boundary value. typedef unsigned int16 **pa_ch_comp_limit_type[4]**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the boundary value for the specified band. *eep_static.gmsk_rf_comp2.edge_pa_ch_comp_limits[0][0] …[3][3]*
Platform	MP1(E)

8.2.15 DWD_get_edge_pa_ch_comp

Function name:	Bool DWD_get_edge_pa_ch_comp(unsigned char band, **pa_ch_comp_type** *pa_ch_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_ch_comp_ptr: References a table holding the channel compensation for all power levels for the specified band. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**; typedef **dwd_pa_ch_comp_elm_type pa_ch_comp_type[16];**
Returns:	Result of the operation: *ok or error*
Description	This function gets the table of channel compensation for all power levels for the specified band.

	eep_static.edge_rf_comp.edge_pa_ch_comp[0][0][0]…[3][19][3]
Platform	MP1(E)

8.2.16 DWD_store_edge_pa_ch_comp

Function name:	bool DWD_store_edge_pa_ch_comp(unsigned char band, **pa_ch_comp_type** *pa_ch_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_ch_comp_ptr: References a table holding the channel compensation for all power levels for the specified band. typedef struct { signed int16 pa_ch_comp0; signed int16 pa_ch_comp1; signed int16 pa_ch_comp2; signed int16 pa_ch_comp3; }**dwd_pa_ch_comp_elm_type**; typedef **dwd_pa_ch_comp_elm_type pa_ch_comp_type[16];**
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of channel compensation for all power levels for the specified band. *eep_static.edge_rf_comp.edge_pa_ch_comp[0][0][0]…[3][19][3]*
Platform	MP1(E)

8.2.17 DWD_get_edge_pa_temp_comp

Function name:	Bool DWD_get_edge_pa_temp_comp(unsigned char band, **pa_temp_comp_type** *pa_temp_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_temp_comp_ptr: References a table holding the temperature compensation for all power levels for the specified band. typedef struct { signed char pa_temp_comp0; signed char pa_temp_comp1; signed char pa_temp_comp2; signed char pa_temp_comp3; signed char pa_temp_comp4; }**dwd_pa_temp_comp_elm_type**; typedef **dwd_pa_temp_comp_elm_type pa_temp_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function gets the table of temperature compensation for all power levels for the specified band. *eep_static.edge_rf_comp.edge_pa_temp_comp[0][0][0]…[3][19][4]*
Platform	MP1(E)

8.2.18 DWD_store_edge_pa_temp_comp

Function name:	Bool DWD_store_edge_pa_temp_comp(unsigned char band, **pa_temp_comp_type** *pa_temp_comp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Pa_temp_comp_ptr: References a table holding the temperature compensation for all power levels for the specified band. typedef struct { signed char pa_temp_comp0; signed char pa_temp_comp1; signed char pa_temp_comp2; signed char pa_temp_comp3; signed char pa_temp_comp4; }**dwd_pa_temp_comp_elm_type**; typedef **dwd_pa_temp_comp_elm_type pa_temp_comp_type[16]**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of temperature compensation for all power levels for the specified band *eep_static.edge_rf_comp.edge_pa_temp_comp[0][0][0]…[3][19][4]*
Platform	MP1(E)

8.2.19 DWD_get_edge_pa_timing_offset

Function name:	bool DWD_get_edge_pa_timing_offset(unsigned char band, **pa_timing_offset_table_type** *timing_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900.* Timing_offsets_ptr: References a table holding time offset for positioning the TX burst for all power levels for the specified band. typedef struct { signed char rampup; signed char rampdown; } **dwd_pa_timing_elm_type**; typedef **dwd_pa_timing_elm_type pa_timing_offset_table_type[16]**
Returns:	Result of the operation: *ok or error*
Description	This function gets for the specified band the stored *pa timing offsets* stored in EEPROM. *eep_static.rf_adjcomp.edge_pa_timing_offset[0][0].rampup…[3][19]* *eep_static.rf_adjcomp.edge_pa_timing_offset[0][0].rampdown…[3][19]*
Platform	MP1(E)

8.2.20 DWD_store_edge_pa_timing_offset

Function name:	Bool DWD_store_edge_pa_timing_offset(unsigned char band, **pa_timing_offset_table_type** *timing_offsets_ptr,int handle); Extern
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900.* Timing_offsets_ptr: References a table holding time offset for positioning the TX burst for all power levels for the specified band. typedef struct { signed char rampup; signed char rampdown; } **dwd_pa_timing_elm_type**; typedef **dwd_pa_timing_elm_type pa_timing_offset_table_type[16]**
Returns:	Result of the operation: *ok or error*
Description	This function stores the table of pa timing offsets for the specified band. *eep_static.rf_adjcomp.edge_pa_timing_offset[0][0].rampup…[3][19]* *eep_static.rf_adjcomp.edge_pa_timing_offset[0][0].rampdown…[3][19]*
Platform	MP1(E)

8.2.21 DWD_get_rxlev_ch_comp_offset

Function name:	Bool DWD_get_rxlev_ch_comp_offset(unsigned char band, **dwd_ch_comp_rxlev_offset_table_type** *rxlev_ch_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Rxlev_ch_offsets_ptr: Reference to fetched rxlev ch comp offsets for specified band. typedef signed char **dwd_ch_comp_rxlev_offset_table_type[8];**
Returns:	Result of the operation: *ok or error*
Description	This function fetches for the specified band the rxlev channel compensation offsets stored in EEPROM. *eep_static.rf_adjcomp.rxlev_ch_comp[0][0]..[3][7]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.2.22 DWD_store_rxlev_ch_comp_offset

Function name:	Bool DWD_store_rxlev_ch_comp_offset(unsigned char band, **dwd_ch_comp_rxlev_offset_table_type** *rxlev_ch_offsets_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Rxlev_ch_offsets_ptr: Reference to fetched rxlev ch comp offsets for specified band. typedef signed char **dwd_ch_comp_rxlev_offset_table_type[8];**
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived rxlev channel compensation offsets for the specified band. NOTE that the number of offsets is not the same for GSM900 and DCS1800. *Eep_static.rf_adjcomp.rxlev_ch_comp[0][0]..[3][7]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.2.23 DWD_get_rxlev_tmp_comp

Function name:	Bool DWD_get_rxlev_tmp_comp(signed char *tmp_ptr, int handle)
Parameters:	Tmp_ptr: reference to fetched rxlev_tmp_comp
Returns:	Result of the operation: *ok or error*
Description	This function fetches the rxlev_tmp_comp stored in EEPROM. *Eep_static.rf_comp.rxlev_temp_comp[0][0]…[3][4]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.2.24 DWD_store_rxlev_tmp_comp

Function name:	bool DWD_store_rxlev_tmp_comp(unsigned char band, signed char *tmp_ptr, int handle)
Parameters:	Band: *dwd_gsm_900* or *dwd_dcs_1800 or dwd_gsm_850 or dwd_pcs_1900* Tmp_ptr: reference to fetched rxlev_tmp_comp
Returns:	Result of the operation: *ok or error*
Description	This function stored the fetched rxlev_tmp_comp values for a given band. *Eep_static.rf_comp.rxlev_temp_comp[0][0]…[3][4]*
Platform	P2002, P2002+, MP1, BP2, ULC

8.3 Temperature calibration

8.3.1 DWD_get_pmb_temp_offset

Function name:	Bool DWD_get_pmb_temp_offset(signed int16 *rf_temp_offset, int handle)
Parameters:	Rf_temp_offset: Reference to fetched rf_temp_offset.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the actual This function is used to get the actual *pmb temperature compensation offset* stored in EEPROM. *Eep_static.chr_pmb6253.temp_offset*
Platform	P2002, P2002+, MP1, BP2, ULC

8.3.2 DWD_store_pmb_temp_offset

Function name:	Bool DWD_store_pmb_temp_offset(signed int16 rf_temp_offset, int handle)
Parameters:	Rf_temp_offset: Reference to fetched rf_temp_offset.
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived *pmb temperature compensation offset.* *eep_static.chr_pmb6253.temp_offset*
Platform	P2002, P2002+, MP1, BP2, ULC

8.4 ADC calibration

8.4.1 DWD_get_adc

Function name:	Bool DWD_get_adc(**dwd_adc_adjusted_comp_parms_type** *comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to the ADC compensation values. typedef struct

	{ signed int16 vbat_gain; signed int16 vbat_offset; signed int16 tbat_gain; signed int16 tbat_offset; signed int16 tenv_gain; signed int16 tenv_offset; signed int16 btec_gain; signed int16 btec_offset; signed int16 tvco_gain; signed int16 tvco_offset; }**dwd_adc_adjusted_comp_parms_type**;
Returns:	Result of the operation: *ok or error*
Description	This function fetches the ADC parameters stored in EEPROM. *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_offset* *eep_static.chr_adjcomp.tbat_gain* *eep_static.chr_adjcomp.tbat_offset* *eep_static.chr_adjcomp.tenv_gain* *eep_static.chr_adjcomp.tenv_offset* *eep_static.chr_adjcomp.btec_gain* *eep_static.chr_adjcomp.btec_offset* *eep_static.chr_adjcomp.tvco_gain* *eep_static.chr_adjcomp.tvco_offset*
Platform	P2002, P2002+, BP2, ULC

8.4.2 DWD_get_adc_v2

Function name:	Bool DWD_get_adc_v2(**dwd_adc_adjusted_comp_parms_v2_type** *comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to the ADC compensation values. typedef struct { signed int16 vbat_gain; signed int16 vbat_offset; signed int16 tbat_gain; signed int16 tbat_offset; signed int16 tenv_gain; signed int16 tenv_offset; signed int16 btec_gain; signed int16 btec_offset; signed int16 tvco_gain; signed int16 tvco_offset; signed int16 current_gain; signed int16 current_offset; signed int16 accid_gain; signed int16 accid_offset;

	}**dwd_adc_adjusted_comp_parms_v2_type;**
Returns:	Result of the operation: *ok or error*
Description	This function fetches the ADC parameters stored in EEPROM. *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_offset* *eep_static.chr_adjcomp.tbat_gain* *eep_static.chr_adjcomp.tbat_offset* *eep_static.chr_adjcomp.tenv_gain* *eep_static.chr_adjcomp.tenv_offset* *eep_static.chr_adjcomp.btec_gain* *eep_static.chr_adjcomp.btec_offset* *eep_static.chr_adjcomp.tvco_gain* *eep_static.chr_adjcomp.tvco_offset* *eep_static.chr_adjcomp.current_gain* *eep_static.chr_adjcomp.current_offset* *eep_static.chr_adjcomp.accid_gain* *eep_static.chr_adjcomp.accid_offset*
Platform	MP1

8.4.3 DWD_store_adc

Function name:	Bool DWD_store_adc(**dwd_adc_adjusted_comp_parms_type** *comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to the ADC compensation values. typedef struct { signed int16 vbat_gain; signed int16 vbat_offset; signed int16 tbat_gain; signed int16 tbat_offset; signed int16 tenv_gain; signed int16 tenv_offset; signed int16 btec_gain; signed int16 btec_offset; signed int16 tvco_gain; signed int16 tvco_offset; }**dwd_adc_adjusted_comp_parms_type**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived ADC compensation values. *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_offset* *eep_static.chr_adjcomp.tbat_gain* *eep_static.chr_adjcomp.tbat_offset* *eep_static.chr_adjcomp.tenv_gain* *eep_static.chr_adjcomp.tenv_offset*

	eep_static.chr_adjcomp.btec_gain *eep_static.chr_adjcomp.btec_offset* *eep_static.chr_adjcomp.tvco_gain* *eep_static.chr_adjcomp.tvco_offset*
Platform	P2002, P2002+, BP2, ULC

8.4.4 DWD_store_adc_v2

Function name:	Bool DWD_store_adc(**dwd_adc_adjusted_comp_parms_v2_type ***comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to the ADC compensation values. typedef struct { signed int16 vbat_gain; signed int16 vbat_offset; signed int16 tbat_gain; signed int16 tbat_offset; signed int16 tenv_gain; signed int16 tenv_offset; signed int16 btec_gain; signed int16 btec_offset; signed int16 tvco_gain; signed int16 tvco_offset; signed int16 current_gain; signed int16 current_offset; signed int16 accid_gain; signed int16 accid_offset; }**dwd_adc_adjusted_comp_parms_v2_type**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived ADC compensation values. *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_gain* *eep_static.chr_adjcomp.vbat_offset* *eep_static.chr_adjcomp.tbat_gain* *eep_static.chr_adjcomp.tbat_offset* *eep_static.chr_adjcomp.tenv_gain* *eep_static.chr_adjcomp.tenv_offset* *eep_static.chr_adjcomp.btec_gain* *eep_static.chr_adjcomp.btec_offset* *eep_static.chr_adjcomp.tvco_gain* *eep_static.chr_adjcomp.tvco_offset* *eep_static.chr_adjcomp.current_gain* *eep_static.chr_adjcomp.current_offset* *eep_static.chr_adjcomp.accid_gain* *eep_static.chr_adjcomp.accid_offset*
Platform	MP1

8.4.5 DWD_store_adc_direct

Function name:	Bool DWD_store_adc_direct(signed int16 vbat_gain, signed int16 vbat_offset, signed int16 tbat_gain, signed int16 tbat_offset, signed int16 tenv_gain, signed int16 tenv_offset, signed int16 btec_gain, signed int16 btec_offset, signed int16 tvco_gain, signed int16 tvco_offset, int handle)
Parameters:	
Returns:	Result of the operation: *ok or error*
Description	This function is the same as DWD_store_adc but it easier to use in LabView program since the input parameter not use a pointer as input type.
Platform	P2002, P2002+, MP1, BP2, ULC

8.4.6 DWD_store_adc_direct_v2

Function name:	Bool DWD_store_adc_direct_v2(signed int16 vbat_gain, signed int16 vbat_offset, signed int16 tbat_gain, signed int16 tbat_offset, signed int16 tenv_gain, signed int16 tenv_offset, signed int16 btec_gain, signed int16 btec_offset, signed int16 tvco_gain, signed int16 tvco_offset, signed int16 current_gain, signed int16 current_offset, signed int16 accid_gain, signed int16 accid_offset, int handle)
Parameters:	
Returns:	Result of the operation: *ok or error*
Description	This function is the same as DWD_store_adc_v2 but it easier to use in Lab View program since the input parameter not use a pointer as input type.
Platform	MP1, BP2

8.5 AFC calibration

8.5.1 DWD_get_afc

Function name:	Bool DWD_get_afc(**dwd_afc_comp_type** *comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to the AFC compensation values. typedef struct { unsigned int16 default_dac_value; unsigned int16 dac_step_gsm_in_hz; } **dwd_afc_comp_type**;
Returns:	Result of the operation: *ok or error*
Description	This function fetches the AFC parameters stored in EEPROM. *eep_static.rf_adjcomp.afc.default_dac_value* *eep_static.rf_adjcomp.afc.dac_step_gsm_in_hz*
Platform	P2002, P2002+, MP1, BP2, ULC

8.5.2 DWD_store_afc

Function name:	Bool DWD_store_afc(**dwd_afc_comp_type** *comp_ptr, int handle)
Parameters:	Comp_ptr: Reference to derived AFC compensation values. typedef struct {

	unsigned int16 default_dac_value; unsigned int16 dac_step_gsm_in_hz; } **dwd_afc_comp_type**;
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived AFC compensation values. *eep_static.rf_adjcomp.afc.default_dac_value* *eep_static.rf_adjcomp.afc.dac_step_gsm_in_hz*
Platform	P2002, P2002+, MP1, BP2, ULC

8.6 GDD

8.6.1 DWD_get_lcd_contrast

Function name:	Bool DWD_get_lcd_contrast(unsigned char *contrast, int handle)
Parameters:	Contrast: Reference fetched the contrast value
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the contrast from the EEPROM *eep_static.lcd_parms.contrast_ctrl_reg*
Platform	P2002, P2002+, MP1, BP2, ULC

8.6.2 DWD_get_lcd_contrast_v2

Function name:	bool DWD_get_lcd_contrast_v2(unsigned int16 hw_id, unsigned int16 *lcd_id, unsigned char *contrast, int handle)
Parameters:	Hw_id: indicate the hardware you want to get the lcd_id and contrast value for. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. Contrast: Fetched the contrast value. Lcd_id: Fetched the lcd_id value. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the contrast and lcd_id from the EEPROM. GDD_MAIN_LCD parameter is located in: *eep_static.lcd_parms.contrast_ctrl_reg (Contrast)* *eep_static.lcd_parms.contrast_ctrl_reg_1 (LCD ID)*

	GDD_SUB_LCD parameter is located in: *eep_static.lcd_parms.contrast_ctrl_reg_2 (Contrast)* *eep_static.lcd_parms.contrast_ctrl_reg_3 (LCD ID)*
Platform	BP30

8.6.3 DWD_store_lcd_contrast

Function name:	Bool DWD_store_lcd_contrast(unsigned char contrast, int handle)
Parameters:	Contrast: Reference fetched the contrast value.
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived LCD contrast value. *eep_static.lcd_parms.contrast_ctrl_reg*
Platform	P2002, P2002+, MP1, BP2, ULC

8.6.4 DWD_store_lcd_contrast_v2

Function name:	bool DWD_store_lcd_contrast_v2(unsigned int16 hw_id, unsigned int16 lcd_id, unsigned char contrast, int handle)
Parameters:	Hw_id: indicate for which hardware you want to store lcd_id and contrast valuer. The value should be set to GDD_MAIN_LCD or GDD_SUB_LCD. Contrast: Fetched the contrast value. Lcd_id: Fetched the lcd_id value. typedef enum { /* Values not to be changed. */ GDD_MAIN_LCD = 250, GDD_SUB_LCD, GDD_GDC, GDD_CAM, GDD_MAIN_CAM = GDD_CAM, GDD_SUB_CAM, GDD_MAX_HW_ID } gdd_hw_id_type;
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived LCD ID and contrast value. GDD_MAIN_LCD parameter is located in: *eep_static.lcd_parms.contrast_ctrl_reg (Contrast)* *eep_static.lcd_parms.contrast_ctrl_reg_1 (LCD ID)* GDD_SUB_LCD parameter is located in: *eep_static.lcd_parms.contrast_ctrl_reg_2 (Contrast)* *eep_static.lcd_parms.contrast_ctrl_reg_3 (LCD ID)*
Platform	BP30

8.6.5 DWD_get_lcd_contrast_sub_display

Function name:	Bool DWD_get_lcd_contrast_sub_display(unsigned char *contrast, int handle)
Parameters:	Contrast: Reference fetched the contrast value
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the contrast value in EEPROM *eep_static.lcd_parms.contrast_ctrl_reg_2*
Platform	P2002+, MP1, BP2

8.6.6 DWD_store_sub_lcd_contrast

Function name:	Bool DWD_store_sub_lcd_contrast(unsigned char contrast, int handle)
Parameters:	Contrast: Reference fetched the contrast value.
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived SUB LCD contrast value. *eep_static.lcd_parms.contrast_ctrl_reg_2*
Platform	P2002+, MP1, BP2

8.7 Audio

8.7.1 DWD_get_audio_parms

Function name:	Bool DWD_get_audio_parms(signed int16 *scal_in, signed int16 *scal_out, int handle)
Parameters:	Scal_in & scal_out fetched the audio calibration parameter
Returns:	Result of the operation: *ok or error*
Description	This function fetches the stored *audio parameters* stored in EEPROM. *eep_static.aud_scal_parms[0]…[3].scal_in* *eep_static.aud_scal_parms[0]…[3].scal_out*
Platform	P2002, P2002+, ULC

8.7.2 DWD_store_audio

Function name:	Bool DWD_store_audio_parms(signed int16 scal_in, signed int16 scal_out, int handle)
Parameters:	*scal_in, *scal_out: References to the derived audio calibration parameters.
Returns:	Result of the operation: *ok or error*
Description	This function stores the derived audio calibration parameters. *eep_static.aud_scal_parms[0]…[3].scal_in* *eep_static.aud_scal_parms[0]…[3].scal_out*
Platform	P2002, P2002+, ULC

8.8 Bluetooth

8.8.1 DWD_btd_get_address (Blue moon single cellular)

Function name:	bool DWD_btd_get_address(**dwd_btd_address_type** *btd_address, int handle);
Parameters:	*btd_address: Fetched the 48bit Bluetooth device address

	typedef unsigned char dwd_btd_address_type[6];
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the 48bit Bluetooth device address. *eep_static.bd_data.bd_addr[0]…[5]*
Platform	MP1

8.8.2 DWD_btd_bmu_get_address (Bluemoon universal)

Function name:	Bool DWD_btd_bmu_get_address(**dwd_btd_address_type** *btd_address, int handle);
Parameters:	*btd_address: Fetched the 48bit Bluetooth device address typedef unsigned char dwd_btd_address_type[6];
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the 48bit Bluetooth device address. *eep_static.bd_data_bmu.BdAddr[0]…[5]*
Platform	MPE

8.8.3 DWD_btd_store_address (Blue moon single cellular)

Function name:	bool DWD_btd_store_address(**dwd_btd_address_type** *btd_address, int handle);
Parameters:	*btd_address: Reference to the the memory location where Bluetooth device address should be copied. typedef unsigned char dwd_btd_address_type[6];
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the 48bit Bluetooth device address. Please note that the setting of the Bluetooth device address is only activated in *normal mode*. Therefore the changed Bluetooth device address will not be used until the mobile is powered up in *normal mode*. *eep_static.bd_data.bd_addr[0]…[5]*
Platform	MP1

8.8.4 DWD_btd_bmu_store_address (Blue moon universal)

Function name:	bool CALL_CONVENTION DWD_btd_store_address(**dwd_btd_address_type** *btd_address, int handle)
Parameters:	*btd_address: Reference to the the memory location where Bluetooth device address should be copied. typedef unsigned char dwd_btd_address_type[6];
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the 48bit Bluetooth device address. Please note that the setting of the Bluetooth device address is only activated in *normal mode*. Therefore the changed Bluetooth device address will not be used until the mobile is powered up in *normal mode*.

	eep_static.bd_data_bmu.BdAddr[0]…[5]
Platform	MPE

8.8.5 DWD_btd_get_tx_power_offset (Blue moon single cellular)

Function name:	bool DWD_btd_get_tx_power_offset(**dwd_btd_tx_power_offset_type** *offset, int handle)
Parameters:	*Offset: The TX power offset value for the Bluetooth device. typedef enum { dwd_btd_tx_power_offset_0_db = 0, dwd_btd_tx_power_offset_minus_1_5_db = 1, dwd_btd_tx_power_offset_minus_3_db = 2, dwd_btd_tx_power_offset_1_5_db = 3, }dwd_btd_tx_power_offset_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the TX power level offset. *eep_static.bd_data.RF_Conf* **(bit:xxxx00xx)**
Platform	MP1

8.8.6 DWD_btd_bmu_get_tx_power_offset (Blue moon universal)

Function name:	bool DWD_btd_bmu_get_tx_power_offset(**dwd_btd_bmu_tx_power_offset_type** *offset, int handle)
Parameters:	*Offset: The TX power offset value for the Bluetooth device. typedef enum { dwd_btd_bmu_tx_power_offset_minus_4_db = 0, dwd_btd_bmu_tx_power_offset_minus_2_db = 1, dwd_btd_bmu_tx_power_offset_nominal = 2, dwd_btd_bmu_tx_power_offset_2_db = 3 }dwd_btd_bmu_tx_power_offset_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to fetch the TX power level offset. *eep_static.bd_data_bmu.RfConf* **(bit:xxxx00xx)**
Platform	MPE

8.8.7 DWD_btd_store_tx_power_offset (Blue moon single cellular)

Function name:	bool DWD_btd_store_tx_power_offset(**dwd_btd_tx_power_offset_type** *offset, int handle)
Parameters:	*Offset: The TX power offset value for the Bluetooth device. typedef enum {

	dwd_btd_tx_power_offset_0_db = 0, dwd_btd_tx_power_offset_minus_1_5_db = 1, dwd_btd_tx_power_offset_minus_3_db = 2, dwd_btd_tx_power_offset_1_5_db = 3, }dwd_btd_tx_power_offset_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the TX power level offset. The store tx power offset value is used in both *normal mode* and *test mode*. In test mode it is not necessary to power off/on the mobile to activate the new tx power offset value. To use the new value stored in EEPROM it is only necessary to press the START button in the phone tool when the Bluetooth mode is TESTMODE. *eep_static.bd_data.RF_Conf* **(bit:xxxx00xx)**
Platform	MP1

8.8.8 DWD_btd_bmu_store_tx_power_offset (Blue moon universal)

Function name:	bool DWD_btd_bmu_store_tx_power_offset(**dwd_btd_bmu_tx_power_offset_type** *offset, int handle)
Parameters:	*Offset: The TX power offset value for the Bluetooth device. typedef enum { dwd_btd_bmu_tx_power_offset_minus_4_db = 0, dwd_btd_bmu_tx_power_offset_minus_2_db = 1, dwd_btd_bmu_tx_power_offset_nominal = 2, dwd_btd_bmu_tx_power_offset_2_db = 3 }dwd_btd_bmu_tx_power_offset_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the TX power level offset. The store tx power offset value is used in both *normal mode* and *test mode*. In test mode it is not necessary to power off/on the mobile to activate the new tx power offset value. To use the new value stored in EEPROM it is only necessary to press the START button in the phone tool when the Bluetooth mode is TESTMODE. *eep_static.bd_data_bmu.RfConf* **(bit:xxxx00xx)**
Platform	MPE

8.8.9 DWD_btd_get_baud_rate (Blue moon single cellular)

Function name:	bool DWD_btd_get_baud_rate(**dwd_btd_baud_rate_type** *baudrate, int handle)
Parameters:	Baudrate: The baud rate between BTD and host typedef enum { dwd_btd_baud_115200 = 0x70,

	dwd_btd_baud_230400 = 0x37, dwd_btd_baud_460800 = 0x1B, dwd_btd_baud_921600 = 0x0D }dwd_btd_baud_rate_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to retrieve the selected baud rate in NV memory *eep_static.clk_data.Sabcal*
Platform	MP1

8.8.10 DWD_btd_bmu_get_baud_rate (Blue moon universal)

Function name:	bool DWD_btd_bmu_get_baud_rate(**dwd_btd_bmu_baud_rate_type** *baudrate, int handle);
Parameters:	Baudrate: The baud rate between BTD and host typedef enum { dwd_btd_bmu_baud_9600 = 0x00, dwd_btd_bmu_baud_19200 = 0x01, dwd_btd_bmu_baud_38400 = 0x02, dwd_btd_bmu_baud_57600 = 0x03, dwd_btd_bmu_baud_115200 = 0x04, dwd_btd_bmu_baud_230400 = 0x05, dwd_btd_bmu_baud_460800 = 0x06, dwd_btd_bmu_baud_921600 = 0x07, dwd_btd_bmu_baud_1843200 = 0x08, dwd_btd_bmu_baud_3250000 = 0x09 }dwd_btd_bmu_baud_rate_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to retrieve the selected baud rate in NV memory *eep_static.bd_data_bmu.UartBaudrate*
Platform	MPE

8.8.11 DWD_btd_store_baud_rate (Blue moon single cellular)

Function name:	bool DWD_btd_store_baud_rate(**dwd_btd_baud_rate_type** *baud_rate, int handle))
Parameters:	Baudrate: The baud rate between BTD and host typedef enum { dwd_btd_baud_115200 = 0x70, dwd_btd_baud_230400 = 0x37, dwd_btd_baud_460800 = 0x1B, dwd_btd_baud_921600 = 0x0D }dwd_btd_baud_rate_type;

Returns:	Result of the operation: *ok or error*
Description	This function is used to store selected baud rate in NV memory. Please note that the setting of the baud rate is only activated in *normal mode*. Therefore, a new stored baud rate will not be used until the mobile is powered up in *normal mode*. *eep_static.clk_data.Sabcal*
Platform	MP1

8.8.12 DWD_btd_bmu_store_baud_rate (Blue moon universal)

Function name:	bool DWD_btd_store_baud_rate(**dwd_btd_baud_rate_type** *baud_rate, int handle))
Parameters:	Baudrate: The baud rate between BTD and host typedef enum { dwd_btd_bmu_baud_9600 = 0x00, dwd_btd_bmu_baud_19200 = 0x01, dwd_btd_bmu_baud_38400 = 0x02, dwd_btd_bmu_baud_57600 = 0x03, dwd_btd_bmu_baud_115200 = 0x04, dwd_btd_bmu_baud_230400 = 0x05, dwd_btd_bmu_baud_460800 = 0x06, dwd_btd_bmu_baud_921600 = 0x07, dwd_btd_bmu_baud_1843200 = 0x08, dwd_btd_bmu_baud_3250000 = 0x09 }dwd_btd_bmu_baud_rate_type;
Returns:	Result of the operation: *ok or error*
Description	This function is used to store selected baud rate in NV memory. Please note that the setting of the baud rate is only activated in *normal mode*. Therefore, a new stored baud rate will not be used until the mobile is powered up in *normal mode*. *eep_static.bd_data_bmu.UartBaudrate*
Platform	MPE

8.8.13DWD_btd_get_device_name

Function name:	bool DWD_btd_get_device_name(unsigned char *btd_device_name, int handle)
Parameters:	Btd_device_name: Fetched the Bluetooth device name.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the Bluetooth device name save in the dynamic EEPROM *eep_dynamic.bt_parms.bt_local_name[0]*
Platform	MP1

8.8.14DWD_btd_store_device_name

Function name:	bool DWD_btd_store_device_name(unsigned char *btd_device_name, int handle)
Parameters:	*Btd_device_name: Fetched the BlueTooth device name.
Returns:	Result of the operation: *ok or error*
Description	This function is used to store the Bluetooth device name in dynamic EEPROM. Please note that the setting of the Bluetooth device name is only activated in *normal mode*. Therefore, a new stored Bluetooth device name will not be used until the mobile is powered up in *normal mode*. *eep_dynamic.bt_parms.bt_local_name[0]* **It is not necessary to call DWD_store_to_dynamic_nv_memory, in this case as you normally should do, after changing an EEPROM value. This dwdio.dll function will write the new value on the fly in order to save some time.**
Platform	MP1

8.8.15DWD_btd_get_mode

Function name:	bool DWD_btd_get_mode(bool *OnOff, int handle)
Parameters:	*OnOff: Fetched the Bluetooth mode
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the Bluetooth mode value name in dynamic EEPROM. It is bit5 in the parameter below: *eep_dynamic.bt_parms.bt_settings* **The value is stored local in the dwdio.dll. You need to call DWD_store_to_dynamic_nv_memory afterwards to store the value in the target.**
Platform	MP1

8.8.16 DWD_btd_store_mode

Function name:	bool DWD_btd_store_mode(bool OnOff, int handle)
Parameters:	*OnOff: Fetched the Bluetooth mode value you want to store.
Returns:	Result of the operation: *ok or error*
Description	This function is used to store a new Bluetooth mode value name in dynamic EEPROM. It's bit5 in the parameter below. Please note that the setting of the Bluetooth mode is only activated in *normal mode*. Therefore, a new stored Bluetooth mode will not be used until the mobile is powered up in *normal mode*. *eep_dynamic.bt_parms.bt_settings* **It is not necessary to call DWD_store_to_dynamic_nv_memory, in this case as you normally should do, after changing a EEPROM value. This dwdio.dll function will write the new value on the fly in order to save some time.**
Platform	MP1

8.9 EEP generic function

8.9.1 DWD_dump_dynamic

Function name:	Bool DWD_dump_dynamic(unsigned char *data, int nof_bytes, int handle)

8.9.2 DWD_dump_parameter

Function name:	Bool DWD_dump_parameter(unsigned char *data_start, unsigned short int size, unsigned short int start_addr,unsigned short int dynamic_start_addr, int handle)

8.9.3 DWD_dump_static

Function name:	Bool DWD_dump_static(unsigned char *data, int remaining_to_process, int handle)

8.9.4 DWD_dump_static_wchksum

Function name:	Bool DWD_dump_static_wchksum(unsigned char *data, int remaining_to_process, unsigned int16 chksum, bool *chksum_ok, int handle)

8.9.5 DWD_get_eep_info

Function name:	Bool DWD_get_eep_info(**dwd_get_eep_info_type** *eep_ptr, int handle)
Parameters:	Eep_ptr: Reference fetched the EEP info. typedef struct { unsigned int16 static_size; unsigned int16 dynamic_size; unsigned int16 org_version; unsigned long eep_start_addr; unsigned int16 upd_revision; } **dwd_get_eep_info_type**;
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the EEP info from the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC
Atcptest:	atctst_get_eep_info_req

8.9.6 DWD_get_eep_revision

Function name:	Bool DWD_get_eep_revision(unsigned int16 *eep_revision, int handle)
Parameters:	Eep_revision: Reference fetched the EEP revision.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the EEP revision from the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC

Atcptest:	atctst_get_eep_revision_req

8.9.7 DWD_get_eep_version

Function name:	Bool DWD_get_eep_version(int *eep_version, int handle)
Parameters:	Eep_version: Reference fetched the EEP version.
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the EEP version from the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.8 DWD_get_generic_tag_parms

Function name:	Bool DWD_get_generic_tag_parms(char *parm_name, void *data, unsigned int nof_bytes, char *tag_name, unsigned char *tag, int handle)
Parameters:	*parm_name: Input string with the name of the parameter to be fetched. The name MUST be the complete name from EEP.CFG file *data: Reference indicating where the requested data should be placed nof_bytes: Number of bytes to be fetched *tag_name: Input string with the name of the *tag* associated with the tag parameters. The name MUST be the complete name from eep.cfg. *tag: Reference indicating where the value of the associated tag should be placed.
Returns:	Result of the operation: *ok or error*
Description	Used to get the specified *tag parms* (customized parms) and *tag* value from the NV memory
Platform	P2002, P2002+, MP1

8.9.9 DWD_store_generic_tag_parms

Function name:	Bool DWD_store_generic_tag_parms(char *parm_name, void *data, unsigned int nof_bytes, char *tag_name, int handle)
Parameters:	*parm_name: Input string with the name of the parameter to be stored. The name MUST be the complete name from EEP.CFG. *data: Reference to the data to be stored in the given parameter nof_bytes: Number of bytes to be stored *tag_name: Input string with the name of the *tag* associated with the tag parameters. The name MUST be the complete name from EEP.CFG.
Returns:	Result of the operation: *ok or error*
Description	Used to store *tag parms* (customized parms). Furthermore, it increments the associated *tag*, which forces an update of The Dynamic NV block at next power up.
Platform	P2002, P2002+, MP1, BP2

8.9.10 DWD_get_parameters

Function name:	Bool DWD_get_parameters(unsigned char *dest, int nof_bytes, unsigned short int start_addr, unsigned short int dynamic_start_addr, int handle)

8.9.11 DWD_get_parm_xx

Function name:	Bool DWD_get_parm_xx(char *parameter_name, unsigned int nof_bytes, void *data_ptr, int handle)
Parameters:	parameter_name: fetched NULL-terminated ASCII string with the complete name of the wanted EEPROM static parameter as specified in the *.CFG file. Nof_bytes: fetched the number of bytes to read. Data_ptr: reference fetched the data store.
Returns:	Result of the operation: *ok or error*
Description	This function gets the value of the wanted static EEPROM parameter.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.12 DWD_get_dynamic_parm_xx

Function name:	bool DWD_get_dynamic_parm_xx(char *parameter_name, unsigned int nof_bytes, void *data_ptr, int handle)
Parameters:	parameter_name: fetched NULL-terminated ASCII string with the complete name of the wanted EEPROM dynamic parameter as specified in the *.CFG file. Nof_bytes: fetched the number of bytes to read. Data_ptr: reference fetched the data store.
Returns:	Result of the operation: *ok or error*
Description	This function gets the value of the wanted dynamic EEPROM parameter.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.13 DWD_store_parm_xx

Function name:	Bool DWD_store_parm_xx(char *parameter_name, unsigned int nof_bytes, void *data_ptr, int handle)
Parameters:	Parameter_name: Fetched NULL-terminated ASCII string with the complete name of the wanted EEPROM static parameter as specified in the *.CFG file. Nof_bytes: Fetched the number of bytes to read. Data_ptr: Reference fetched the data store.
Returns:	Result of the operation: *ok or error*
Description	This function can store new value for selected EEPROM static parameter as specified in the *.CFG.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.14 DWD_store_dynamic_parm_xx

Function name:	bool DWD_store_dynamic_parm_xx(char *parameter_name, unsigned int nof_bytes, void *data_ptr, int handle)
Parameters:	Parameter_name: Fetched NULL-terminated ASCII string with the complete name of the wanted EEPROM dynamic parameter as specified in the *.CFG file. Nof_bytes: Fetched the number of bytes to read. Data_ptr: Reference fetched the data store.
Returns:	Result of the operation: *ok or error*
Description	This function can store new value for selected EEPROM dynamic parameter as specified in the *.CFG.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.15 DWD_read_dynamic_nv

Function name:	Bool DWD_read_dynamic_nv(unsigned short int start_addr, unsigned short int nof_bytes, void *pdest, int handle)
Parameters:	Start_addr: Fetched the start address to start the reading. Zero (0) always specifies the first address of *dynamic* NV part. Nof_bytes: Fetched the number of bytes to read. pdest: Reference fetched the parameters reads from the *dynamic* NV.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read parameter from the *dynamic* NV block.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.16 DWD_read_static_nv

Function name:	Bool DWD_read_static_nv(unsigned short int start_addr, unsigned short int nof_bytes, void *pdest, int handle)
Parameters:	Start_addr: Fetched the start address to start the reading: Zero (0) always specifies the first address of *static* NV part. Nof_bytes: Fetched the number of bytes to be read. pdest: Reference fetched the parameters read from *static* NV.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read parameter from the *static* NV block.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.17 DWD_reset_dyn_nv_to_default

Function name:	Bool DWD_reset_dyn_nv_to_default(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	Used to reset the dynamic NV parameter to default.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.18 DWD_store_item_in_nv_men

Function name:	Bool DWD_store_item_in_nv_mem(unsigned short int item_index, int handle)

8.9.19 DWD_store_to_nv_memory

Function name:	Bool DWD_store_to_nv_memory(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to write the calibration (static) parameters stored in the DWDIO DLL to the *non-volatile memory* of the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.20 DWD_store_to_nv_wchksum

Function name:	Bool DWD_store_to_nv_wchksum(bool *checksum_ok, int handle)
Parameters:	Checksum_ok: Reference fetched the result of the operation. T*rue or false*.
Returns:	Result of the operation: *ok or error*
Description	This function is used to write the calibration (static) parameters stored in the DWDIO DLL to the *non-volatile memory* of the DUT. There is a checksum calculation.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.21 DWD_store_to_dynamic_nv_memory

Function name:	Bool DWD_store_to_dynamic_nv_memory(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to write the dynamic parameters stored in the DWDIO DLL to the *non-volatile memory* of the DUT.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.22 DWD_write_dynamic_nv

Function name:	Bool DWD_write_dynamic_nv(unsigned int16 start_addr, unsigned int16 nof_bytes, void *psrc, int handle)
Parameters:	Start_addr: Fetched the start address to start the writing. Zero (0) always specifies the first address of *dynamic* NV part. Nof_bytes: Fetched the number of bytes to write. pdest: Reference fetched the parameters there will be write to the *dynamic* NV.
Returns:	Result of the operation: *ok or error*
Description	This function is used to write parameters to the *dynamic* NV block.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.23 DWD_write_static_nv

Function name:	Bool DWD_write_static_nv(unsigned int16 start_addr, unsigned int16 nof_bytes, void *psrc, int handle)
Parameters:	Start_addr: Fetched the start address to start the writing. Zero (0) always specifies the first address of *static* NV part. Nof_bytes: Fetched the number of bytes to write. pdest: Reference fetched the parameters there will be write to *static* NV.
Returns:	Result of the operation: *ok or error*
Description	This function is used to write parameter to the static NV block.
Platform	P2002, P2002+, MP1, BP2, ULC

8.9.24 DWD_set_eep_cfg_filename

Function name:	Bool DWD_set_eep_cfg_filename(char *eep_input_name,int handle)
Parameters:	Eep_input_name: Reference to the EEP prefix ex.
Returns:	Result of the operation: *ok or error*
Description	This function is used to change the default EEP prefix.

Platform	All platforms.

8.9.25 DWD_use_eep_archive

Function name:	Void DWD_use_eep_archive(bool use)
Parameters:	Use: true or false
Returns:	No return value
Description	This function is used to the dwdio.dll is should get the EEP.CFG from a zip file.
Platform	All platforms.

9 FFS

9.1 DWD_get_ffs_revision

Function name:	bool DWD_get_ffs_revision(unsigned int16 *ffs_revision, int handle);
Parameters:	Ffs_revision: Reference fetched the ffs_revision
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the FFS revision from the DUT
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_ffs_generic_func_req

9.2 DWD_get_ffs_version

Function name:	Bool DWD_get_ffs_version(unsigned int16 *ffs_version, int handle)
Parameters:	Ffs_version: Reference fetched the ffs_version
Returns:	Result of the operation: *ok or error*
Description	This function is used to get the FFS version from the DUT
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode
Atcptest:	atctst_get_ffs_version_req

10 Workbench (Reduced signaling test)

This is a description of these function there are used in the Reduced signaling test mode. This test mode is only available on the MP1 platform.

10.1 DWD_wrk_get_ptm

Function name:	Bool DWD_wrk_get_ptm(unsigned int32 *fn, **dwd_equ_data** *equ_data, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_get_ptm_req

10.2 DWD_wrk_get_si

Function name:	Bool DWD_wrk_get_si(unsigned int16 *status, unsigned int16 *metric, unsigned char *channel_type, unsigned int16 *arfcn, unsigned long *fn, unsigned int16 *rms, signed int16 *toffs,signed int32 *qual, signed int16 *foffs, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_get_si_req

10.3 DWD_wrk_start_fcb_sb

Function name:	Bool DWD_wrk_start_fcb_sb(unsigned int16 arfcn, unsigned char fcb_only, unsigned char *fcb_result, signed int16 *freq_offs, unsigned int16 *qual, unsigned int16 *rms, unsigned char *bsic, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_fcb_req

10.4 DWD_wrk_start_idle

Function name:	Bool DWD_wrk_start_idle(unsigned int16 arfcn, unsigned char txpwr, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_idle_req

10.5 DWD_wrk_start_ptm

Function name:	bool DWD_wrk_start_ptm(unsigned int16 trx_mode, **dwd_timeslot_allocation** uplnk, **dwd_timeslot_allocation** downlnk, **dwd_radio_freq_no** arfcn, **dwd_timeslot_source** timeslot_source, **dwd_loopback_mode_type** loopback_mode, unsigned int16 tx_mix_control_cs, **dwd_cs_type** tx_mix_any_other_cs, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_ptm_req

10.6 DWD_wrk_start_rach

Function name:	Bool DWD_wrk_start_rach(unsigned int16 ch_req_msg, **dwd_tsc_type** tsc, unsigned int16 timing_advance, **dwd_block_format_type** access_burst_type, **dwd_channel_type** channel_type, **dwd_radio_freq_no** arfcn, **dwd_txpwr** cs_txpwr, **dwd_txpwr** ps_txpwr, unsigned char iq_swap, unsigned int16 bsic, **dwd_rach_type** rachtype, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_rach_req

10.7 DWD_wrk_start_rxlev_band

Function name:	Bool DWD_wrk_start_rxlev_band(unsigned char band, unsigned int16 start_arfcn, unsigned char nof_arfcn_in,unsigned char *nof_arfcn_out, unsigned char *rxlev, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_rxlev_band_req

10.8 DWD_wrk_start_si

Function name:	Bool DWD_wrk_start_si(unsigned int16 arfcn, unsigned int16 si_read_mode, unsigned int16 si_indicator, unsigned char pbcch_lr_start, unsigned char bcch_descr_psi1_repeat_period, unsigned char bcch_descr_bs_pbcch_blks, unsigned char bcch_descr_pb,unsigned char bcch_descr_tsc, unsigned char bcch_descr_tn, unsigned char bcch_descr_chan_sel_present, unsigned int16 bcch_descr_chan_sel_union_rf, unsigned int16 bcch_descr_mobile_alloc_rf_chan_cnt, unsigned int16 *bcch_descr_mobile_alloc_rf_chan_no, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest_	atctst_wrk_start_si_req

10.9 DWD_wrk_start_tch

Function name:	Bool DWD_wrk_start_tch(unsigned int32 ta, **dwd_l1_channel_elem** chan, **dwd_amr_config_type** amr_config, unsigned char audio_enable, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_start_tch_req

10.10 DWD_wrk_start_tch_loop

Function name:	Bool DWD_wrk_start_tch_loop(unsigned char tch_loop_mode, int handle)
Platform:	MP1
Mode:	Default test mode

Atcptest:	atctst_wrk_start_tch_loop_req

10.11 DWD_wrk_stop_fcb_sb

Function name:	DWD_wrk_stop_fcb_sb(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_fcb_req

10.12 DWD_wrk_stop_ptm

Function name:	Bool DWD_wrk_stop_ptm(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_ptm_req

10.13 DWD_stop_rach

Function name:	Bool DWD_wrk_stop_rach(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_rach_req

10.14 DWD_wrk_stop_rxlev_band

Function name:	Bool DWD_wrk_stop_rxlev_band(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_rxlev_band_req

10.15 DWD_wrk_stop_si

Function name:	Bool DWD_wrk_stop_si(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_si_req

10.16 DWD_wrk_stop_tch

Function name:	Bool DWD_wrk_stop_tch(int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_wrk_stop_tch_req

10.17 DWD_fwval_setup_adjust_mode

Function name:	Bool DWD_fwval_setup_adjust_mode(unsigned char flags, int handle)
Platform:	MP1
Mode:	Default test mode
Atcptest:	atctst_fwval_setup_adjust_mode_req

11 GTB (Generic test bench)

11.1 DWD_gtb_generic

Function name:	Bool DWD_gtb_generic(unsigned int16 opcode, unsigned char fileID, int handle)

Parameters:	Opode: Reference fetched the opcode.
	FileId: Reference fetched the file ID.
	
	enum dwd_gtb_opcode
	{
	dwd_gtb_transfer_ffs_file_to_ram_1,
	dwd_gtb_store_ram_1_data_to_ffs_file,
	dwd_gtb_transfer_ffs_file_to_ram_2,
	dwd_gtb_store_ram_2_data_to_ffs_file,
	dwd_reprog_dsp_patch
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to send some generic test bench data to the DUT.
Platform	MP1
Mode:	Default test mode

11.1.1 DWD_gtb_generic_timeout_delay

Function name:	Bool DWD_gtb_generic(unsigned int16 opcode, unsigned char fileID, unsigned int timeout_delay, int handle)

Parameters:	Opode: Reference fetched the opcode.
	FileId: Reference fetched the file ID.
	Timeout_delay: Reference fetched the timeout value for dwdio.dll
	
	enum dwd_gtb_opcode
	{
	dwd_gtb_transfer_ffs_file_to_ram_1,
	dwd_gtb_store_ram_1_data_to_ffs_file,
	dwd_gtb_transfer_ffs_file_to_ram_2,
	dwd_gtb_store_ram_2_data_to_ffs_file,
	dwd_reprog_dsp_patch
	};

Returns:	Result of the operation: *ok or error*
Description	This function is used to send some generic test bench data to the DUT.
Platform	MP1
Mode:	Default test mode

12 Debugging utilities

12.1 DWD_memory_read

Function name:	Bool DWD_memory_read(unsigned char *dest, unsigned int16 nof_bytes, unsigned int32 address, int handle)
Parameters:	Dest: Reference fetched the destination data. Nof_bytes: Reference fetched the number of bytes. Address: Reference fetched the address to read.
Returns:	Result of the operation: *ok or error*
Description	This function is used to read the memory at a given address.
Platform	All platforms.
Mode:	Default test mode
Atcptest:	atctst_memory_read_req

12.2 DWD_ram_modify

Function name:	Bool DWD_ram_modify(unsigned int32 address, unsigned int16 nof_bytes, unsigned char *src, int handle)
Parameters:	Address: Reference fetched the address to modify. Nof_bytes: Reference fetched the number of bytes to modify. Src: Reference fetched the data.
Returns:	Result of the operation: *ok or error*
Description	This function is used to modify the RAM at a given address.
Platform	All platforms.
Mode:	Default test mode
Atcptest:	atctst_ram_modify_req

12.3 DLL trace

In order to ease the debugging of the production test implementation trace of the input and output of the serial line communication can be setup. Depend on how many COM ports you have set up will some files be generated after calling DWD_stop_trace.

DWD1.LOG
DWD2.LOG

The .LOG files are time stamped log of what has been received and sent over the serial line. If you want to make the log file more readable can it be done by calling the function:
14.9***DWD_convert_function_trace2txt***

12.3.1 DWD_start_trace

Function name:	Bool DWD_start_trace(void)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*

Description	This function is used to start the trace.
Platform	All platforms.
Mode:	Default test mode

12.3.2 DWD_stop_trace

Function name:	Bool DWD_stop_trace(void)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to stop the trace.
Platform	All platforms.
Mode:	Default test mode

12.4 Exception

12.4.1 DWD_clear_excp_log

Function name:	Bool DWD_clear_excp_log(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used clear the exception log in the DUT.
Platform	All platforms.
Mode:	Default test mode
Atcptest:	atctst_clear_excp_log_req

12.4.2 DWD_provoke_excp

Function name:	Bool DWD_provoke_excp(int handle)
Parameters:	N/A
Returns:	Result of the operation: *ok or error*
Description	This function is used to provoke an exception in the DUT.
Platform	All platforms.
Mode:	Default test mode
Atcptest:	atctst_provoke_excp_req

13 Misc. function

13.1 Generic test function:

All generic function has the same input parameter as follow:

- opcode (Operation code, the ID's can be found in dwdptest.h or in the individual xxx_opcode.h files from the module.
- Nof_input_bytes(Indicates the bytes send to target, the maximum is 200).
- Pinput (input buffer).
- Nof_output_bytes (The number of bytes in the output buffer).
- Poutput (output buffer).

For some function is there one more parameter called *timeout_delay*. This parameter is used to increase the default timeout delay in the DLL. The default value is 2500 (2.5 second).

13.1.1 DWD_monitor_pin_generic_func

Function name:	Bool DWD_monitor_pin_generic_func(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the monitor signaling test
Platform	MP1
Mode:	Default test mode

13.1.2 DWD_cam_generic

Function name:	bool DWD_cam_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the camera test.
Platform	P2002+
Mode:	Default test mode

13.1.3 DWD_afr_generic

Function name:	Bool DWD_afr_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the AFR (AM / FM Radio) driver
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode

13.1.3.1 DWD_afr_generic_timeout_delay

Function name:	bool DWD_afr_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function the to AFR (AM / FM Radio) driver
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode

13.1.4 DWD_mmci_generic

Function name:	Bool DWD_mmci_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to MMCI driver
Platform	P2002+, MP1, BP2
Mode:	Default test mode

13.1.4.1 DWD_mmci_generic_timeout_delay

Function name:	Bool DWD_mmci_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to MMCI driver
Platform	P2002+, MP1, BP2
Mode:	Default test mode

13.1.5 DWD_lcd_generic_func

Function name:	Bool DWD_lcd_generic_func(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *input, unsigned int16 *nof_output_bytes, unsigned char *output, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to LCD driver
Platform	P2002, P2002+, BP2
Mode:	Default test mode

13.1.6 DWD_pow_management_generic_func

Function name:	Bool DWD_pow_management_generic_func(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to SM power driver
Platform	MP1, ULC
Mode:	Default test mode

13.1.7 DWD_rtt_generic

Function name:	bool DWD_rtt_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the RTT driver
Platform	MP1
Mode:	Default test mode

13.1.8 DWD_sec_generic

Function name:	Bool DWD_sec_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the SEC driver
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

13.1.8.1 DWD_sec_generic_timeout_delay

Function name:	Bool DWD_sec_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay,int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the SEC driver
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

13.1.9 DWD_sim_generic

Function name:	Bool DWD_sim_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the SIM driver
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

13.1.10 DWD_vib_generic

Function name:	Bool DWD_vib_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to vibrator

	Please refer to VIB interface document for detailed description of the different operation code.
Platform	MP1
Mode:	Default test mode

13.1.11 DWD_aud_generic

Function name:	Bool DWD_aud_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to AUD driver. **Please refer to audio interface documents for details on supported functionality.** **Warning!!! If you use the opcode "set_dai_mode" be aware that this function may change the default configuration of Input\Output ports and thereby destroy other functions. Please check the specific Printed Circuit Board schematics for further information.**
Platform	MP1, BP2, ULC
Mode:	Default test mode

13.1.11.1 DWD_aud_generic_timeout_delay

Function name:	Bool DWD_aud_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to AUD driver. **Please refer to audio interface documents for details on supported functionality.** **Warning!!! If you use the opcode "set_dai_mode" be aware that this function may change the default configuration of Input\Output ports and thereby destroy other functions. Please check the specific Printed Circuit Board schematics for further information.**
Platform	MP1, BP2, ULC
Mode:	Default test mode

13.1.12 DWD_aud_generic_longint

Function name:	Bool DWD_aud_generic_longint(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to AUD driver and can be used where there are problem to use DWD_aud_generic with 32 byte alignment. **Please refer to audio interface documents for details on supported functionality.**

	Warning!!! If you use the opcode "set_dai_mode" be aware that this function may change the default configuration of Input\Output ports and thereby destroy other functions. Please check the specific Printed Circuit Board schematics for further information.
Platform	MP1
Mode:	Default test mode

13.1.12.1 DWD_aud_generic_longint_timeout_delay

Function name:	DWD_aud_generic_longint_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to AUD driver and can be used where there are problem to use DWD_aud_generic_timeout_delay with 32 byte alignment. **Please refer to audio interface documents for details on supported functionality.** **Warning!!! If you use the opcode "set_dai_mode" be aware that this function may change the default configuration of Input\Output ports and thereby destroy other functions. Please check the specific Printed Circuit Board schematics for further information.**
Platform	MP1
Mode:	Default test mode

13.1.13 DWD_gdd_generic

Function name:	Bool DWD_gdd_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to GDD driver
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

13.1.14 DWD_bluetooth_generic

Function name:	bool DWD_bluetooth_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to Bluetooth driver
Platform	MP1
Mode:	Default test mode

13.1.14.1 DWD_bluetooth_generic_timeout_delay

Function name:	bool DWD_bluetooth_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to Bluetooth driver
Platform	MP1
Mode:	Default test mode

13.1.15 DWD_acc_generic

Function name:	bool DWD_acc_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes,unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to Accessory driver
Platform	MP1
Mode:	Default test mode

13.1.16 DWD_chr_generic

Function name:	Bool DWD_chr_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to charger driver
Platform	P2002, P2002+, MP1, BP2, ULC
Mode:	Default test mode

13.1.17 DWD_ffs_generic

Function name:	Bool DWD_ffs_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FFS driver. **Reference document is: "FFS PC-interface_v_x.x".**
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode

13.1.17.1 DWD_ffs_generic_timeout_delay

Function name:	Bool DWD_ffs_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FFS driver, but this function here has one more input parameter, where you can set the dwdio.dll timeout time.

	Reference document is: "FFS PC-interface_v_x.x".
Platform	P2002, P2002+, MP1, BP2
Mode:	Default test mode

13.1.18 DWD_util_generic

Function name:	Bool DWD_util_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to UTIL driver
Platform	MP1
Mode:	Default test mode

13.1.18.1 DWD_util_generic_timeout_delay

Function name:	Bool DWD_util_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to UTIL driver
Platform	MP1
Mode:	Default test mode

13.1.19 DWD_ftl_generic

Function name:	Bool DWD_ftl_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FTL driver
Platform	MP1
Mode:	Default test mode

13.1.19.1 DWD_ftl_generic_timeout_delay

Function name:	Bool DWD_ftl_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FTL driver
Platform	MP1
Mode:	Default test mode

13.1.20 DWD_nand_generic

Function name:	Bool DWD_nand_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int

	handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to NAND driver
Platform	MP1
Mode:	Default test mode

13.1.20.1 DWD_nand_generic_timeout_delay

Function name:	Bool DWD_nand_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to NAND driver
Platform	MP1
Mode:	Default test mode

13.1.21 DWD_fs_generic

Function name:	Bool DWD_fs_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FS driver **Reference: document: ”FS PC-interface”**
Platform	P2002+, MP1
Mode:	Default test mode

13.1.21.1 DWD_fs_generic_timeout_delay

Function name:	Bool DWD_fs_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to FS driver **Reference: document: ”FS PC-interface”**
Platform	P2002+, MP1
Mode:	Default test mode

13.1.22 DWD_led_generic

Function name:	Bool DWD_led_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to LED driver
Platform	MP1
Mode:	Default test mode

13.1.22.1 DWD_led_generic_timeout_delay

Function name:	Bool DWD_led_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to LED driver
Platform	MP1
Mode:	Default test mode

13.1.23 DWD_gtl_generic

Function name:	Bool DWD_gtl_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to GTL driver
Platform	MP1
Mode:	Default test mode

13.1.23.1 DWD_gtl_generic_timeout_delay

Function name:	Bool DWD_gtl_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to GTL driver
Platform	MP1
Mode:	Default test mode

13.1.24 DWD_ee_generic

Function name:	Bool DWD_ee_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to EE driver
Platform	MP1, ULC
Mode:	Default test mode

13.1.24.1 DWD_ee_generic_timeout_delay

Function name:	Bool DWD_ee_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to EE driver
Platform	MP1, ULC
Mode:	Default test mode

13.1.25 DWD_key_generic

Function name:	Bool DWD_key_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to keyboard driver
Platform	MP1, ULC
Mode:	Default test mode

13.1.25.1 DWD_key_generic_timeout_delay

Function name:	Bool DWD_key_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput,unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to keyboard driver
Platform	MP1, ULC
Mode:	Default test mode

13.1.26 DWD_fs_ms_generic

Function name:	Bool DWD_fs_ms_generic(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle);
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to fs_ms driver
Platform	MP1
Mode:	Default test mode

13.1.26.1 DWD_fs_ms_generic_timeout_delay

Function name:	Bool DWD_fs_ms_generic_timeout_delay(unsigned int16 opcode, unsigned int16 nof_input_bytes, unsigned char *pinput,unsigned int16 *nof_output_bytes, unsigned char *poutput, unsigned int timeout_delay, int handle);
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to fs_ms driver
Platform	MP1
Mode:	Default test mode

13.1.27 DWD_ver_generic

Function name:	bool DWD_ver_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char *pinput, unsigned int16 *nof_output_bytes, unsigned char *poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the VER driver

Platform	ULC
Mode:	Default test mode

13.1.27.1 DWD_ver_generic_timeout_delay

Function name:	bool DWD_ver_generic_timeout_delay(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the VER driver
Platform	ULC
Mode:	Default test mode

13.1.28 DWD_trap_generic

Function name:	bool DWD_trap_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the TRAP driver
Platform	ULC
Mode:	Default test mode

13.1.28.1 DWD_trap_generic_timeout_delay

Function name:	bool DWD_trap_generic_timeout_delay(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to TRAP driver
Platform	ULC
Mode:	Default test mode

13.1.29 DWD_emd_generic

Function name:	bool DWD_emd_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the EMD driver
Platform	ULC
Mode:	Default test mode

13.1.29.1 DWD_emd_generic_timeout_delay

Function name:	bool DWD_emd_generic_timeout_delay(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, unsigned int timeout_delay, int handle)

Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the EMD driver
Platform	ULC
Mode:	Default test mode

13.1.30 DWD_rtc_generic

Function name:	bool DWD_rtc_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the RTC driver
Platform	ULC
Mode:	Default test mode

13.1.30.1 DWD_rtc_generic_timeout_delay

Function name:	bool DWD_rtc_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the RTC driver
Platform	ULC
Mode:	Default test mode

13.1.31 DWD_iic_generic

Function name:	bool DWD_iic_generic(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the IIC driver
Platform	ULC
Mode:	Default test mode

13.1.31.1 DWD_iic_generic_timeout_delay

Function name:	bool CALL_CONVENTION DWD_iic_generic_timeout_delay(unsigned int16 opcode, unsigned int32 nof_input_bytes, unsigned char * pinput, unsigned int16 * nof_output_bytes, unsigned char * poutput, unsigned int timeout_delay, int handle)
Returns:	Result of the operation: *ok or error*
Description	This is a generic function to the IIC driver
Platform	ULC
Mode:	Default test mode

14 Others

14.1 DWD_Get_DLL_handle

Function name:	HINSTANCE DWD_Get_DLL_Handle()
Description:	This function can be useful if another *.dll should call dwdio.dll

14.2 DWD_baseband_present

Function name:	Bool DWD_baseband_present(unsigned int16 *baseband, int handle)

Description:	Get information about which base band the target software is compiled to. Phone tool are using the command to recognize the different base band platform.
	
	enum dwd_baseband_present_codes
	{
	dwd_bb_none = 0,
	dwd_egold_plus_v1_2 = 2,
	dwd_egold_v3 = 4,
	dwd_sgold = 8,
	dwd_sgold_lite = 16,
	dwd_sglite_v1_1 = 32,
	dwd_egold_lite = 64,
	dwd_egold_radio = 65,
	dwd_sgold2 = 128
	};

14.3 DWD_get_hasp_id

Function name:	Bool DWD_get_hasp_id(unsigned int32 *id, int handle)
Description:	Used to read a the unique ID number from the HASP dongle.

14.4 DWD_feature_present

Function name:	Bool DWD_features_present(unsigned int32 *features, int handle)
Description:	This function is used to get which feature there are represent by the target software.

14.5 DWD_get_AT_wrapped_len

Function name:	Int DWD_get_AT_wrapped_len(unsigned char *data_src, int src_len)

14.6 DWD_get_buffer

Function name:	Void DWD_get_buffers(unsigned char **rx,unsigned char **tx, unsigned char **txAT)

14.7 DWD_get_com_port_for_handle

Function name:	Int DWD_get_com_port_for_handle(int handle)
Description	This function can be used to get the comport for a given handle.

14.8 DWD_dll_version

Function name:	Bool DWD_get_dll_version(char *dll_version)
Description:	This function is used to get the version of the dwdio.dll

14.9DWD_convert_function_trace2txt

Function name:	Bool DWD_convert_function_trace2txt(char *func_file, char *output_file)

14.10 DWD_external_debug_message_displayer

Function name:	Void DWD_external_debug_message_displayer(void (*show_func)(char *msg))
Description:	

14.11 DWD_external_msg_handler

Function name:	Void DWD_external_msg_handler(bool (*process)(void))

14.12 DWD_get_mail

Function name:	Int DWD_get_mail(int handle)

14.13 DWD_rf_present

Function name:	Bool DWD_rf_present(unsigned int16 *rf, int handle)

14.14 DWD_rx_monitor_start

Function name:	Bool DWD_rx_monitor_start(int handle, void (*process_line)(unsigned char *, int))
Description:	// This is NOT like trace - this only picks up what is received nothing can be done while its listening, // except calling "DWD_rx_monitor_stop()"

14.15 DWD_rx_monitor_stop

Function name:	Bool DWD_rx_monitor_stop(int handle)
Description:	Stop RX monitor.

14.16 DWD_send_at_command

Function name:	bool DWD_send_at_command(unsigned char *at_cmd_str, unsigned char len,

	unsigned int wait_for_response_timer, unsigned int wait_for_char, unsigned char *at_cmd_resp_str, bool *timeout, int handle)

14.17 DWD_send_buffer

Function name:	Bool DWD_send_buffer(int handle, int length);

14.18 DWD_send_buffer_special

Function name:	Bool DWD_send_buffer_special(int handle, int length)

14.19 DWD_send_dsp_command

Function name:	Bool DWD_send_dsp_cmd(unsigned int16 nof_dsp_cmd, dsp_cmd_type *data,int handle)
Description:	This command is used by the DSP command in Phone Tool. This menu is used in chip validation.

14.20 DWD_user_command

Function name:	Bool DWD_user_command(unsigned int16 id,unsigned char *data,unsigned char length, void (*rxdone)(unsigned char *rxdata,int rx_length),int handle)

14.21 DWD_user_command_timeout_delay

Function name:	Bool DWD_user_command_timeout(unsigned int16 id, unsigned char *data,unsigned char length, void (*rxdone)(unsigned char *rxdata,int rx_length), unsigned int timeout, int handle)

14.22 DWD_wrap_at_data

Function name:	DWD_wrap_AT_data(unsigned char *dest, unsigned char *data_src, int src_len)
Description:	This is a function to wrap the data string you want to send to target.

14.23 DWD_get_instantaneous_load

Function name:	Bool DWD_get_instantaneous_load(unsigned char *load, int handle)
Description:	The function is used by the load meter in Phone Tool, to measure the load of the base band chip.

14.24 DWD_get_trap_version

Function name:	Bool DWD_get_trap_version(unsigned int16 *trap_version, int handle)
Description:	Get the trap version from target. This is used by Phone tool.

14.25 DWD_get_last_error

Function name:	Char *DWD_get_last_error(void)
Parameters:	Return: Reference to a string holding the last detected error in the DLL.
Returns:	Result of operation: *ok* or *error*
Description:	This function is used to fetch the latest detected error in the DLL.
Status:	Implemented

14.25.1 Error causes

Following possible error strings exist:

"No errors",
"eepX.cfg file not found",
"Error in the eepX.cfg file",
"Invalid handle",
"COMPORT not open or invalid handle",
"A DWD DLL upgrade is needed",
"Error reading parameters from phone (wrong length returned)",
"Error reading parameters from phone",
"Please check cable and ensure that DUT is powered on (E1)",
"Please check cable and ensure that DUT is powered on (E2)",
"Connection ok, but DUT isn't replying correctly (E3)",
"Error while storing parameters in DUT",
"Out of memory!",
"Operation rejected due to currently running operation - please retry",
"Shutting down",
"Output buffer full - please retry later",
"Timeout while receiving - please check connection and that DUT is ON",
"Error in confirm mail from DUT",
"IMEI already programmed",
"IMEI contents error",
"Error while programming IMEI",
"Please ensure that the DWD dongle is present",
"Hardware problem (E42)! Please contact DWDIO.DLL supplier for upgrade",
"Semaphore not present - is the COM port set?",
"Timeout while waiting for thread to leave critical region",
"DWDIO DLL and dongle didn't match",
"Parameter is not defined in eepX.cfg file",
"Unknown RF band support",
"eep.cfg filename not set, default eep.cfg filename used.",
"Tag name is not defined!",
"cust_key.sap file not found",
"An error in cust_key.sap file has occurred",
"Error in sending TX buffer",
"Error in receiving RX buffer",
"Error!- Parameter didn't exists in eep.cfg file",
"Error in opening file",
"JPEG encode error",
"JPEG decode error",
"EEP checksum error",
" "Too many CRs in TX buffer - unable to make AT wrap",
"Error! - Number of receiving bytes set to high",
"Error! - DWDIO DLL corrupted.",
"Error! - eep archive file not found.",
"Error! - eep cfg file not found in archive file.",
"Error! - eep prefix has not been set.",
"Operation interrupted by user",
"Error occure in dwdio.dll",

"New request not allowed",
"Error - Closing comport failed",
"Error - Received eep_static checksum",
"Error - Set Bluetooth TX burst mode failed",
"Error - Set Bluetooth TX continues mode failed",
"Error - Set Bluetooth RX continues mode failed",
"Error in last error - memory may be corrupted",

14.26 DWD_get_dll_id_no

Function name:	Bool DWD_get_dll_id_no(unsigned int16 *id)
Description:	This function is used to get the dwdio.dll ID no. Each dwdio.dll variant has it own number.

14.27 DWD_get_ComWindowsHandle

Function name:	Unsigned int DWD_Get_ComWindowsHandle(int handle)
Description:	Return the comport windows handle.

14.28 DWDIO_SetTimeOutDelay

Function name:	Void DWDIO_SetTimeOutDelay(unsigned int delay)
Description:	Ex. Delay value 25 means 2.5 second, which is the default value.

14.29 DWDIO_GetTimeOutDelay

Function name:	Unsigned long DWDIO_GetTimeoutDelay(void)
Description:	Get the actual time out value.

14.30 DWDIO_ResetTimeOutDelay

Function name:	Void DWDIO_ResetTimeoutDelay(void)
Description:	Reset the time out value to default. (approx. 2.5 sec)

14.31 DWD_interrupt_requested_mail

Function name:	void DWD_interrupt_requested_mail(int handle)
Description:	Interrupt requested mail, so you don’t want to wait for a time out.

14.32 DWD_send_atcpwroff

Function name:	Bool DWD_send_atcpwroff(int handle)
Description:	This command send a normal AT command (at+cpwroff) to target. **The function will only work if target is started up in normal mode and ready to receive normal AT command. You must also make sure that the target software has support for AT command.**

14.33 DWD_send_atcfun_startup_cmd

Function name:	Bool DWD_send_atcfun_startup_cmd(int handle)
Description:	This command send a normal AT command (at+cfun=1) to target. **The function will only work if target is started up in normal mode and ready to receive normal AT command. You must also make sure that the target software has support for AT command.**

14.34 Current_dirrectory

Function name:	Void current_directory(char *path)
Description:	This function is used to get path where the dwdio.dll has been loaded from. This could be important to know. It could be important information if you want to know where the dwdio.dll try to find the correct EEP configuration file.

14.35 Set_current_directory

Function name:	Void set_current_directory(char *path)
Description:	This function is used to set a new path, if you want the dwdio.dll to load the EEP configuration files from another directory than default. **Remember to use two backslashes in the pathname e.g. "C:\\test\\test1"**

14.36 DWD_check_for_ptest

Function name:	Bool DWD_check_for_ptest(bool *result ,int handle)
Description:	This function can be used to check target is started up in production test mode or not.

14.37 DWD_get_handle_for_com_port

Function name:	int DWD_get_handle_for_com_port(unsigned char comport)
Description:	This function return the handle for the specified comport.

15 Appendix 1: Reference DLL functions used by “Quick Adjust”

Here is a table to show which function we have used in our RF Quick adjust program to GSM /GPRS.

Category	Function Name	Applicable NonSign / Sign	
Comport setting	DWD_set_comport	x	x
	DWD_set_dtr	x	x
	DWD_set_rts	x	x
DLL commands	DWD_set_eep_cfg_filename	x	x
	DWD_set_baud_rate	x	x
	DWD_check_comlink	x	x
	DWD_get_last_error	x	x
Test mode command	DWD_set_testmode	x	x
Dedicated HW ctrl	DWD_check_rtc	x	
	DWD_get_rf_temp	x	
	DWD_set_rf_channel	x	
	DWD_set_gmsk_mode	x	
	DWD_set_rf_mode	x	
	DWD_change_pa_level	x	
	DWD_set_afc	x	
	DWD_meas_all_adc_direct	x	
	DWD_inl_set_power_ramp	x	
	DWD_inl_set_power_ramp_mode	x	
	DWD_set_rf_gain	x	
	DWD_get_iqrms	x	
Get from EEP/flash	DWD_get_pmb_temp_offset	x	x
	DWD_rf_bands_supported	x	x
	DWD_get_pa_dac16_range	x	x
	DWD_get_edge_pa_dac16_range	x	x
	DWD_get_power_ramp	x	x
Store to EEP	DWD_store_adc_direct	x	x
	DWD_store_pmb_temp_offset	x	x
	DWD_store_afc	x	x
	DWD_store_max_pa_ch_comp	x	x
	DWD_store_pa_offset	x	x
	DWD_store_edge_pa_ch_comp	x	x
	DWD_store_gmsk_pa_ch_comp	x	x
	DWD_store_rxlev_gain_offset	x	x
	DWD_store_rxlev_ch_comp_offset	x	x
Store to NV	DWD_store_to_nv_memory	x	x

16 Appendix 2: Reference DLL functions used by “Quick Adjust” for MPE platform.

Category	Function Name	Applicable NonSign / Sign	
Setup comport	DWD_set_comport	x	x
	DWD_set_dtr	x	x
	DWD_set_rts	x	x
	DWD_get_last_error	x	x
	DWD_set_baudrate	x	x
Setup Inline mode (Non-signaling)	Set_current_directory	x	x
	DWD_set_eep_cfg_filename	x	x
	DWD_set_v24_mode	x	x
	DWD_check_comlink	x	x
	DWD_set_serial_interface_mode	x	x
	DWD_sw_reset	x	x
	DWD_set_test_mode	x	x
	DWD_rf_bands_supported	x	x
Versions	DWD_get_sw_version	x	x
	DWD_get_eep_version	x	x
	DWD_get_dll_version	x	x
RTC-Clock	DWD_check_rtc	x	
Setup TX	DWD_set_rf_mode	x	
	DWD_set_gmsk_mode	x	
	DWD_set_rf_channel	x	
	DWD_change_pa_level	x	
		x	
BAT- Calc	DWD_bl_set_backlight	x	
	DWD_get_adc	x	
	DWD_meas_all_adc_direct	x	
	DWD_store_adc_direct	x	
	DWD_set_rf_gain	x	
		x	
DCXO	DWD_set_afc_v2	x	
	DWD_store_afc	x	
PA_adjust	DWD_get_pa_dac16_range	x	x
	DWD_store_gmsk_pa_ch_comp	x	x
EDGE PA calib	DWD_get_edge_pa_dac16_range	x	x

	DWD_store_edge_pa_ch_comp	x	x
	DWD_store_edge_txcorr	x	x
RX- Cal	DWD_set_rf_mode	x	
	DWD_set_rf_gain	x	
	DWD_get_iqrms	x	
	DWD_store_rxlev_ch_comp_offset	x	x
	DWD_store_rxlev_gain_offset	x	x
DUT-Parms	DWD_store_serial_number	x	x
	DWD_write_static_nv	x	x
	DWD_get_testseries_id	x	x
	DWD_get_production_date	x	x
	DWD_get_config	x	x
	DWD_store_testseries_id	x	x
	DWD_store_production_date	x	x
	DWD_store_config	x	x
	DWD_get_test_station_parms	x	x
	DWD_store_test_station_parms	x	x
Store calib data	DWD_store_to_nv_memory	x	x
Read eep file	DWD_get_parameters	x	x

17 Appendix 3: Interface between dwdio.dll and DUT

17.1 Protocol

The physical interface between TEST STATION and MS is RS232 or USB. IN CASE OF USB, it is assumed that Common Device Class is used meaning that the DWDIO.DLL accesses the USB as an ordinary COM port. The protocol for exchanging data over the interface is AT/V24 based meaning that all commands are implemented in a proprietary AT command named 'AT#'. The actual test commands are embedded in this specific AT command.
The AT# command is only supported in test mode.

Requirements for communication between TESTSTATION and MS:

- Data format: 8N1
- Max. Data rate: Depending on platform
- Each AT command send from DTE must be confirmed by a result code send from DCE before next AT commands can be initiated.
- No need for flow control handling (hardware not possible) software not needed.
- Each AT command starts with "AT" or "at" and ends with <CR>. **Note:** If the character <CR> occurs in the content of the AT commands it must be substituted by the character <LF>. The info field indicating the substituted <CR's> follows right after 'AT#' as {*nof_substitutes, offset1+14, .. offsetN+14*}. The content in between "AT#" and <CR> are given in type definition defined in **atctstif.h**:

  BUILDING OF THE AT# COMMAND:

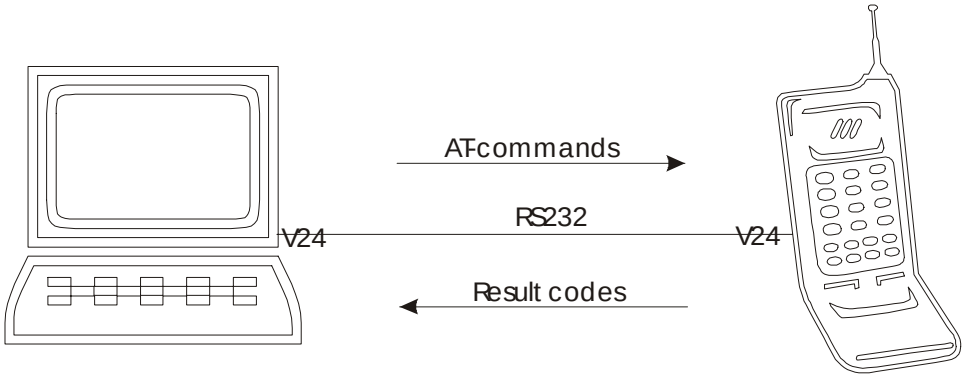
  AT# format: <AT#><CR substitution description><data><CR>

  EXAMPLE OF AT# COMMAND:

  Data to be transmitted: 0x09, 0x0a, 0x0b, 0x0c, 0x0d

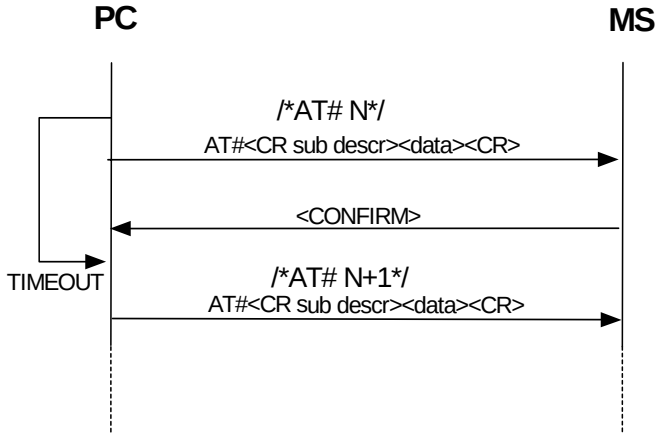
  Resulting AT# command: <AT#><0x01/*Nof CR substituted*/, 0x12/*Position of substituted CR*/>, <0x09, 0x0a, 0x0b, 0x0c, 0x0c><CR>

- Maximum size of data between the "AT#" and <CR> is 230 bytes.
- The Result codes serves as *confirm* corresponding to the *request* issued in the AT# command. The data of the result code is given by the type definition defined in **atctstif.h**.



- When an AT# command has been issued the Test PC must wait for a “CONFIRM” response (or an internal timeout surveying the “CONFIRM” response) before issuing the next AT# command:

When the PC has issued an AT# command it must wait for the "CONFIRM" before issuing the next AT# command.



17.2 Test commands

The data part of the AT# command is the actual *request* test command and is formatted as shown below:

```
typedef struct
{
    unsigned int16 funccode;
    /*parameter part may vary from one command to another*/
} atctst_test_command_req_type;
```

The *funccode* is used to identify the test command on the target side and all the different *funccodes* is defined in the **enum** *atctst_funccodes*

The result code data is the actual *confirm* test command and is formatted as shown below:

```
typedef struct
{
  unsigned int16 funccode;
  unsigned int16 length;
} atctst_header_type;


typedef struct
{
```

```
  atctst_header_type hd;
  /*parameter part may vary from one command to another*/
} atctst_test_command_con_type;
```

The header *hd* consist of a funccode indicating the actual confirm and a length indicating the number of data bytes following the length. The different *funccodes* is defined in the **enum** *atctst_funccodes.*

All the test commands are defined in *atctstif.h* and the handling on the target side can be found in *atcptest.c*.

17.3 Command flow

Command flow between DLL and target is illustrated in the figure below, where “Driver API“ represents the different drivers in the system: