BH02
ATE#2 test Specification

Edition 2006

Published by Neonseven s.r.l.,
Viale Stazione di Prosecco, 15

34010 Sgonico (Trieste) Italy

For questions on technology, delivery and prices please contact the Neonseven Offices in Italy Sgonico and Gorizia

Attention Please!

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact Neonseven.

Neonseven technologies may only be used in life-support devices or systems with the express written approval of Neonseven, if a failure of such technologies can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Table of Contents

1 Document Mission/Scope

1.1 Mission

This document describes adjustment in production of GSM/GPRS mobile phones based on BP30 platfrom.

1.2 Scope

This document in intended to be use by Infineon developers and customer involved in test/adjustement of modems/mobiles based on Infineon BP30 platform.

2 List of Acronyms

Abbreviation / Term	Explanation / Definition
IFX	Infineon
IFWD	Infineon Wireless Design
R&S	Rohde&Schwartz
RSSI	Receiver Signal Strength Indicator

3 Introduction

The aim of this document is to be a detailed description of procedures, functions and data structures used during adjustment of BP30 platform.
The function interface is DWDIO.DLL provided by IFWD.
The examples are related to R&S CMU200 RF test equipment and Agilent 66332A or 6632B power supply.

4 List of tests

In order to detect and handle hw problems a set of tests must be run before DUT RF adjustment (some HW defect may be hidden by further calibration).
At the end of adjustment other RF performance tests assure quality and covers possible defects.
Table below for list:

S/N measurement	Only before mass production	Optional
TX measurement	Non signaling mode, must be repeated for all bands	Recommended
RX measurement	Non signaling mode, must be repeated for all bands	Recommended
TX/RX performance test	Signalling mode, must be repeated for all bands	Recommended

Table 4-1

5 List of adjustments

Default compensation values are downloaded in phone EEPROM together with flashfile during SW-download. Some, but not all, of these compensation values must subsequently be adjusted, fine tuned, for each individual phone in production. The following Table specifies the full set of recommended/optional adjustments.

DCXO adjustment	Must be done in only one band	Recommended
TX adjustment	Must be repeated for all bands	Recommended
RX adjustment	Must be repeated for all bands	Recommended
ADC adjustment	Must be done for all ADC used for critical voltage measurements in the design. Eg. battery voltage and RF temperature.	Recommended
Audio adjustment		Optional (not covered)

Table 5-1

6 Test setup & Instruments

The figure shows a typical test setup for adjustment in production (Figure 6-1)

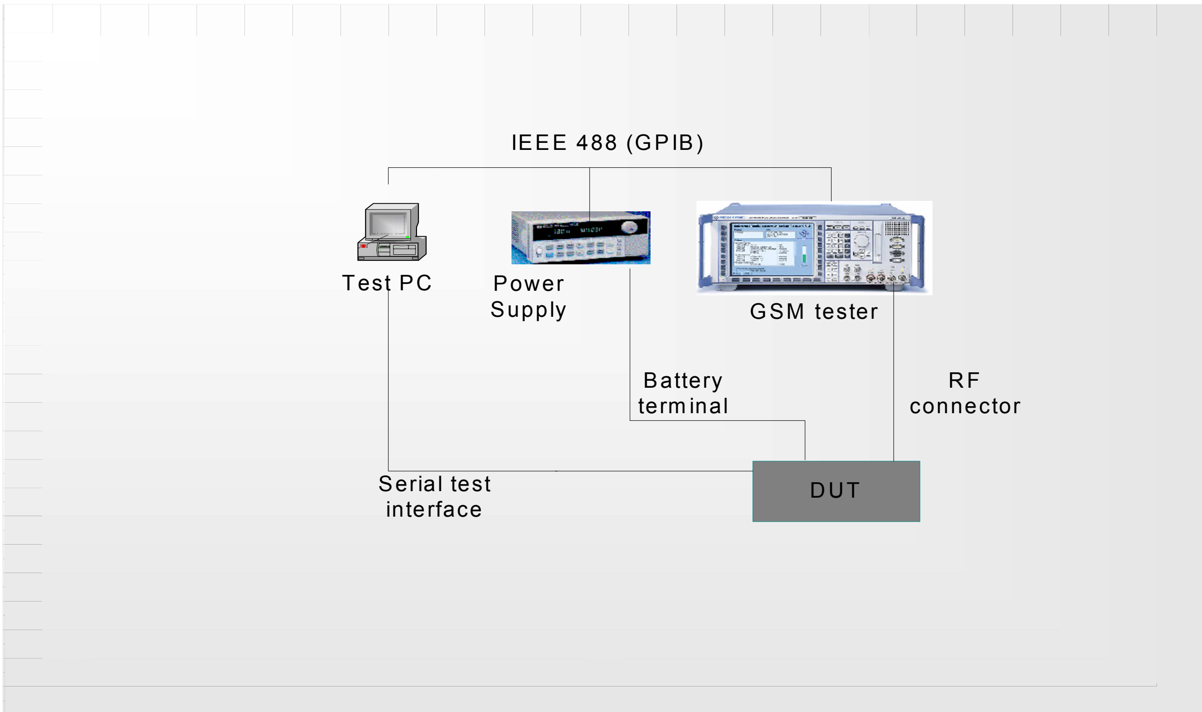


Figure 6-2

All RF tests and adjustments must be performed on the coaxial test connector. Adjustment should be done subsequently to plastic and shielding mount.

Recommended GSM tester is Rohde&Schwarz CMU200.
Levelconverter must be applied between test-PC and DUT (not shown on the figure).
If audio adjustments are applied, artificial mouth and ear must be added to the testsetup. (Not shown on the figure)

7 Serial test interface

Test and adjustment of Infineon phones in production is based on IFWD's production concept.
A dll (DWDIO.DLL) is provided, which constitutes the SW test interface between test-PC and the DUT.
The DWDIO.DLL offers a set of proprietary AT-commands providing all necessary functionality for adjustment of the phone. Moreover the DWDIO.DLL handles the serial communication protocol, security, mapping of EERPROM parameters into physical addresses, temporary storage of adjusted parameters and other aspects of the adjustment session. The DWDIO.DLL can be employed from most programming languages,.including Borland Builder C++, Visual Studio 5.0 and 6.0, NI LabView and NI LabWindowsCVI.

The DWDIO.dll commands relevant for the different adjustments are summarized in the following description of the adjustment procedures.
Refer to production_test_pc_dll_v8.2.pdf for complete documentation of DWDIO.dll.

8 DUT test startup and end sequences

The DUT will automatically turn on when battery- and charge voltage is applied in combination.
For products without battery charging support such as e.g. data modules, other means must be applied in order to turn on the DUT. The DUT itself can detect the power_on cause and behave accordingly.

During adjustments DUT must be supplied with nominal Vbat directly on battery terminals with a stable power supply capable of delivering min. 500 mA (particular measurements require different settings).

8.1 Power_up procedure in test_mode

The following procedure assumes the DUT is connected to comport_1 of TEST PC.
The power up sequence is as follows:

1. EEPROM startup parameter is already set to ptest mode. (eep_static.atc_startup.mode) during previous stage.
2. Apply battery- and charge-voltage. DUT turns on in ptest mode.
3. Wait 1 - 2 seconds to allow for the target SW to initialize.
4. Open the comport and set the serial speed to 115.200 (only required the first time)

```
int Handle = -1;
[…]
if (DWD_set_com_port(1,&Handle)
{
DWD_set_baud_rate(115200,Handle);
DWD_set_RTS(false,Handle);
}
```

5. DWD_check_comm_link command must be sent as the first command: returns TRUE if communication has been established.

```
if (DWD_check_comm_link( Handle ))
printf("linked");
```

6. Now DUT is linked and is ready to follow TEST PC commands.

8.1.1 Prototypes of functions involved

```
bool DWD_set_com_port(unsigned char com_port, int *handle);
bool DWD_check_comm_link(int handle);
bool DWD_set_baud_rate(int baud, int handle);
bool DWD_set_DTR(bool value, int handle);
bool DWD_set_RTS(bool value, int handle);
```

8.2 Entering Non_signalling_test_mode sequence

1. Send DWD_set_testmode(non-signalling mode) command. Entering inline (=non signaling) test mode the second parameter (arfcn) is not used.

```
DWD_set_test_mode(dwd_in_line_mode, 0, Handle )
```

2. The phone is now in non-signaling test mode ready to receive further commands.

8.2.1 Prototypes of functions involved

```
bool DWD_set_test_mode(unsigned int16 mode, unsigned int16 arfcn, int handle
```

8.3 Error handling

If an error condition is detected during DWDIO.DLL function call, then an error description can be retrieved through the following function:.

```
string Error;
[…]
Error = DWD_get_last_error();
```

8.3.1 Prototypes of functions involved

```
char *DWD_get_last_error(void);
```

9 Changing from 1800 to 1900 and back

When a channel must be specified in the 1800 or 1900 band, an auxiliary function allows setting of the different band (since the channel number is not enough)

1)Call the function below to use 1800 band

```
DWD_treat_1800_as_1900( dwd_treat_1800_as_1800 , Handle )
```

2) Call the function below to use 1900 band

```
DWD_treat_1800_as_1900( dwd_treat_1800_as_1900 , Handle )
```

9.1.1 Prototypes of functions involved

```
bool DWD_treat_1800_as_1900(unsigned int16 mode, unsigned int handle);
```

10 Receiver S/N measurements

S/N Measurement is done on 3 channels and on a per band basis
Input level is -102 dBm and RX receiver gain is set to high.
A CW signal with a offset of 10-30 kHz is applied (assuming proper DCXO calibration has occurred).
iqrms read is performed in following scenarios

RF signal is applied -> Signal + Noise, **S+N**
No RF signal is applied -> Noise only, **N**

To obtain a more stable measurement an average (internal to DUT) measurement is fetched.

10.1 Limits:

S/N	Lower Limit [dB]	Upper Limit [dB]
GSM850 128	11,5	16
GSM850 189	11,5	16
GSM850 251	11,5	16
GSM900 975	11,5	16
GSM900 18	11,5	16
GSM900 124	11,5	16
GSM1800 512	10,0	15
GSM1800 670	10,0	15
GSM1800 885	10,0	15
GSM1900 512	10,0	15
GSM1900 660	10,0	15
GSM1900 810	10,0	15

Table 10-1

10.2 Calculation

The output voltage could be calculated with

$$U = \left(const * 10^{\frac{RMSvalue}{16*20}} \right)$$

S/N measurements comes out of

$$S/_N = 10\log\left[\left(const * 10^{\frac{S+N}{16*20}} \right)^2 - \left(const * 10^{\frac{N}{16*20}} \right)^2\right] - 10\log\left[\left(const * 10^{\frac{N}{16*20}} \right)^2\right]$$

using const =0,077

Assuming N << (S+N) the formula above can be adequately approximated by

$$S/_N = 10\log\left[\left(const * 10^{\frac{S+N}{16*20}}\right)^2\right] - 10\log\left[\left(const * 10^{\frac{N}{16*20}}\right)^2\right] = \left(\frac{S+N}{16} - \frac{N}{16}\right)$$

10.3 Sequence

1) Set the input level to the phone to –102 dBm
2) Set the phone to RXTXMON measuring (both RX and MON slots are used for multi sampling)

```
DWD_set_rf_mode(dwd_rxtxmon , Handle );
```

3) Set phone receiver gain 63 (both RX and MON)

```
DWD_set_rf_gain( dwd_mon_gain_tp , 63 , Handle);
DWD_set_rf_gain(dwd_rx_gain_tp, 63 , Handle);
```

4) Set the phone to an ARFCN in 1st channel according to Table 10-1 above.

```
DWD_set_rf_channel( dwd_mon_arfcn_tp , 18 , Handle );
DWD_set_rf_channel(dwd_rx_arfcn_tp , 18 , Handle );
```

5) Set number of samples to be collected during iqrms measurements (average internal to DUT)

```
DWD_set_iqrms_sample_number( nof_samples , Handle );
```

6) Set RF generator frequency to arfcn+30 kHz offset
7) Read up measured RSSI value from the phone (S+N)

```
UINT16 iqrms_N, iqrms_SN;
[…]
DWD_get_iqrms( & iqrms_SN , Handle );
```

8) Set RF generator frequency to arfcn+3 MHz offset
9) Read up measured RSSI value from the phone (N)

```
DWD_get_iqrms( & iqrms_N , Handle );
```

10) Calculate S/N by following formula

```
SN = ( iqrms_SN - iqrms_N ) / 16
```

11) Check with limits
12) Set the phone to an ARFCN in 2nd channel …..and proceed with other channel

10.3.1 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_set_rf_gain(unsigned int16 gain_tp, signed char gain, int handle);
bool DWD_set_iqrms_sample_number( int16 nof_samples, int handle);
bool DWD_get_iqrms( unsigned int16 *iqrms, int handle);
```

11 TX hole-through test

TX hole-through tests allow detection of HW failures. Two concepts have been implemented, each leading to different informations.

11.1 Test of max power level

This test consists in setting TX power at maximum power level at middle channel (or most critical channel if statistical analisys available) and checking of RF power toward wide limits to assure radio and PA are generally working. Typical behaviour of RF TX system is tested. The maximum power level is PCL 5 for GSM850 and GSM900 bands, PCL 0 for DCS1800 and PCS 1900 bands.
The following description is given on a per band basis, and must be repeated for all bands.

11.1.1 Sequence

Test takes place in Not Signalling test mode. Please see example below for GSM900 band

1) Set normal GMSK modulation, 1 TX slot and training sequence (e.g. 5)

```
DWD_set_gmsk_mode( dwd_normal , 5 , Handle );
DWD_inl_set_nof_slot( 1 , Handle );
```

2) Setup ARFCN on centerchannel and enable transmission

```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 65 , Handle );
DWD_set_rf_mode( dwd_tx_burst , Handle );
```

3) Set Pout level and measure Pout

```
DWD_change_pa_level( MAX_PCL , 0 , Handle );
```

Pout read must be checked toward limits (limits must be found on statistical basis for every specific board).
After all bands are tested transmission must be switched off to avoid damages.

11.2 Max RF power (clipped power)

This test consists in setting TX power at maximum PA power output at middle channel (or most critical channel if statistical analisys available) and checking of RF power toward wide limits to assure radio and PA are generally working. Maximum TX performance of PA is tested. The rump hold value is set in order to obtain the maximum power output.
The following description is given on a per band basis, and must be repeated for all bands.

11.2.1 Sequence

Test takes place in Not Signalling test mode. Please see example below for GSM900 band

4) Set normal GMSK modulation, 1 TX slot and training sequence (e.g. 5)

```
DWD_set_gmsk_mode( dwd_normal , 5 , Handle );
DWD_inl_set_nof_slot( 1 , Handle );
```

5) Setup ARFCN on centerchannel and enable transmission

```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 65 , Handle );
DWD_set_rf_mode( dwd_tx_burst , Handle );
```

6) Set Pout level

```
DWD_change_pa_level( MAX_PCL , 0 , Handle );
```

7) Get MAX_PCL ramp values

```
unsigned int16 ramp_ptr1[32];
[…]
DWD_get_power_ramp( dwd_gsm_900 , MAX_PCL , ramp_ptr1 , Handle );
```

8) Set ramp to external ramp map (one ramp for each power control level)

```
DWD_inl_set_power_ramp_mode( dwd_external_pa_ramp , Handle );
```

9) Set MAX_PCL ramp values to obtain max output power

```
DWD_inl_set_power_ramps( ramp_ptr1, ramp_ptr1 , Handle );
```

10) Measure power and check toward limits

Pout read must be checked with limits (limits must be found on statistical basis for every specific board). After all bands are tested transmission must be switched off to avoid damages.

11.2.2 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
bool DWD_set_gmsk_mode( unsigned int16 mode, unsigned int16 tsc,int handle);
bool DWD_inl_set_nof_slot(unsigned char nof_tx_slot, int handle);
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_change_pa_level( unsigned char txpwr, unsigned char txpwr2, int handle);
bool DWD_get_power_ramp(unsigned char band, unsigned int16 pa_level, unsigned int16 *ramp_ptr, int handle);
bool DWD_inl_set_power_ramps(unsigned int16 *power_ramp1_ptr, unsigned int16 *power_ramp2_ptr, int handle);
bool DWD_inl_set_power_ramp_mode(unsigned char mode, int handle);
```

12 DCXO adjustment

On platforms that use Smarti SD2 (such as BP30), the AFC value is sent to the DCXO through a 3wire bus, and is 13 bits wide [0..8191].

12.1 Adjustment procedure for DCXO adjustment

DCXO adjustment procedure requires the computation of two parameters:

XoCal : [0..7] 3 bits
XoTune : [0..8191] 13 bits

The procedure described for AFC adjustment is an iterative approach, which relies on a good initial guess on XoCal and XoTune values: the better initial guess, the fewer iterations are required. Usually a one-step approach allows a good level of confidence. XoCal calibration can be skipped after statistical analisys (a default XoCal can be set in EEPROM and used for calibration).

12.1.1 Adjusted parameters for DCXO adjustment

eep_static.rf_adjcomp.afc.default_dac_value
eep_static.rf_adjcomp.afc.dac_step_gsm_in_hz

12.2 Switching RF TX transmission ON for DCXO calibration

The following sequence examples assume that calibration takes place in GSM900 band. (channel = 65, PCL = 18, TSC 5).

1) Turn off TX power (see 13.5)

```
DWD_set_rf_mode(dwd_stop , Handle );
```

2) Turn on TX on center ARFCN (typically 65) in 900 MHz band at PCL 18. Set 1 TX slot.

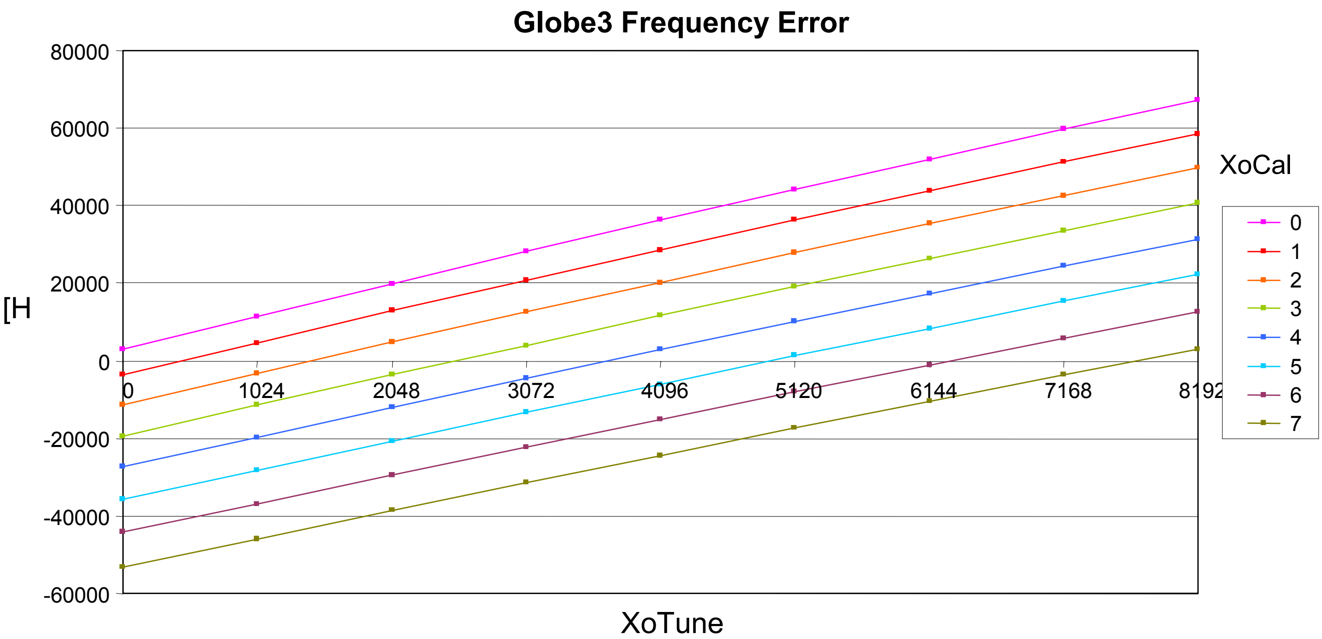
```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 65 , Handle );
DWD_set_gmsk_mode( dwd_normal , 5 , Handle );
DWD_inl_set_nof_slot( 1 , Handle );
DWD_change_pa_level( 18 , 18 , Handle );
DWD_set_rf_mode( dwd_tx_burst , Handle );
```

12.2.1 Prototypes of functions involved

```
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_set_gmsk_mode( unsigned int16 mode, unsigned int16 tsc ,int handle);
bool DWD_inl_set_nof_slot(unsigned char nof_tx_slot, int handle);
bool DWD_change_pa_level( unsigned char txpwr, unsigned char txpwr2, int handle);
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
```

12.3 XoCal adjustment procedure

For Smarti-SD2 based platform the value of XoCal should be calculated in order to obtain the most centered value for calibrated XoTune (next step). Here follows the typical frequency error of a Globe3 board measured at different (XoCal,XoTune) settings:



.

1) Set XoTune to 4095 and XoCal value to $XoCal_1$ (typically 0)

```
DWD_set_afc_v2(XoCal_1 , 4095 , Handle );
```

2) Measure corresponding frequency error $Ferr_1$ (average values to get stable measures)
3) Set XoTune to 4095 and XoCal value to $XoCal_2$ (typically 7)

```
DWD_set_afc_v2(XoCal_2 , 4095 , Handle );
```

4) Measure corresponding frequency error $Ferr_2$ (average values to get sTable measures).
5) Calculate subrange spacing (assuming they are equally spaced) :

$$F_{delta} = \frac{(Ferr_2 - Ferr_1)}{(XoCal_2 - XoCal_1)}$$

6) Calculate the subrange that gives the minimum frequency error @ XoTune = 4095 (center value) by the following formula:

$$XoCal = Round\left(-\frac{(XoCal_2 - XoCal_1)Ferr_1}{(Ferr_2 - Ferr_1)} \right)$$

7) Store locally (in application temporary variable) the calculated XoCal for following XoTune adjustment.

12.3.1 Prototypes of functions involved

```
bool DWD_set_afc_v2( unsigned int16 dac_xocal, unsigned int16 dac_xotune, int handle );
```

12.4 XoTune adjustment procedure

1) Set XoCal to previously found XoCal value.
2) Set XoTune value to XoTune $_i$ and XoTune $_{i+1}$ in order to obtain a boundary estimation of frequency error characteristic. (initial XoTune +/- 500, see Table 4)
3) Measure corresponding frequency errors $Ferr_i$ and $Ferr_{i+1}$ (average values to get sTable measures).
4) Assume a linear relationship between frequency and XoTune and calculate a 'new' nominal to achieve Ferr=0Hz @ 900 MHz.
5) Set XoTune to the new-calculated value and measure frequency error again
6) Repeat 3) -> 6) until ($Ferr_{i+1}$ * $Ferr_i$) is <= 0 or frequency error < = 100 Hz
7) Calculate the slope from $XoTune_{i+1}$, $XoTune_i$, $Ferr_{i+1}$ and $Ferr_i$ values found in the last iteration:

$$Slope = Round\left(\frac{Ferr_{i+1} - Ferr_i}{XoTune_{i+1} - XoTune_i}\right)$$

8) Store the values for nominal XoTune value and calculated Slope locally.

If the measurements exceed the limits of the Tables below then the DUT is defect and must be repaired.

MS with DCXO				
	Unit	Minimum	Typical	Maximum
Step size	Hz/step	tbd	7	tbd
XoTune	/	tbd	4095	tbd

Typically a single iteration produces accepTable values. Usually XoCal and XoTune calibration (including Slope calculation) take place in the same phase, in order to save setup and storage time.

12.4.1 Simplyfied sequence (assuming one step approach)

1) Set XoTune to (4095-500)

```
DWD_set_afc_v2( XO_cal , 4095 - 500 , Handle );
```

2) Measure Ferr_1
3) Set XoTune to (4095+500)

```
DWD_set_afc_v2( XO_cal , 4095 + 500 , Handle );
```

4) Measure Ferr_2
5) Calculate XoTune and Slope
6) Set XoTune and check toward limits.
7)

12.4.2 Prototypes of functions involved

```
bool DWD_set_afc_v2( unsigned int16 dac_xocal, unsigned int16 dac_xotune, int handle );
```

12.5 Switching TX transmission OFF

The transmission must be switched OFF after DCXO calibration using the following command:

```
DWD_set_rf_mode( dwd_stop , Handle )
```

12.5.1 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
```

12.6 Storage of data locally in DWDIO.DLL

XoCal, XoTune and Slope calculated must be stored in DWDIO.DLL, ready for following storage in DUT EEPROM.

1) Calculate default_dac_value

```
dwd_afc_comp_type afc_comp;
[…]
afc_comp.default_dac_value = (XO_cal << 13) | (XO_tune & 0x1FFF );
```

2) Calculate dac_step_gsm_in_hz

```
afc_comp.dac_step_gsm_in_hz = slope;
```

3) Store parameter in DWDIO.DLL

```
DWD_store_afc( &afc_comp , Handle );
```

12.6.1 Prototypes of functions involved

```
bool DWD_store_afc(dwd_afc_comp_type *comp_ptr, int handle);
```

13 TX adjustment

TX compensation during normal operation of the platform is implemented by adding or subtracting an offset, - the “Pa offset” (residing in EEPROM), to the default DAC value, that controls the top level of the burst template (“DAC16”). During non-signaling operation, this compensation is inactive in the SW and the output power is solely determined by the default DAC16 values residing in phones flash memory.

The recommended procedure for TX adjustment is based on a non-iterative approach where the individual Pout versus DAC characteristic for each phone (with default DAC values) are determined by output power measurements on a number of selected channels and PCL’s

When the Pout versus DAC characteristic for a given phone has been found through measurements, it’s possible to calculate a set of PA offset values for all PCL’s, based on the knowledge of the default DAC values and the target output powers.
Measurements on a few selected PCL’s in combination with linear interpolation turns out to give sufficient accuracy in the TX adjustment. Measurements can be done on e.g. max PCL, min PCL +5 and min PCL +2 leading to a two segment linear approximation of the Pout versus DAC curve for the phone.
No further recommendation as for the selection of a suitable subset of PCL´s will be given here. For a given platform the Pout versus DAC characteristic should be thoroughly measured on all PCL’s and evaluated in order to make this selection. More measurements gives better approximation of the Pout versus DAC characteristic and eventually better adjustment accuracy but will on the other hand increase testtime.

13.1 Center channel compensation

BP30 platfrom use what is called “centerchannel” compensation, meaning the calibration vs power parameters are computed on a ‘center’ channel, and calibration vs frequency parameters are computed on low and high channel range

	LOW		MID		HIGH	
			POW_1			
			…			
			POWi			
			…			
			POW_n			

Table 13-1

13.2 Adjusted parameters “Centerchannel”compensation

eep_static.rf_adjcomp.pa_offset[0..3][0..15]).
eep_static.rf_comp.pa_ch_comp[0..3][0..3].

13.3 Channel ranges for TX adjustements

	GSM850	**GSM900**	**GSM1800**	**GSM1900**
Channel low	128-138	975-985	512-522	512-522
Channel center	184-194	33-43	694-704	656-666
Channel high	241-251	114-124	876-885	800-810

Table 13-2

13.4 Standard sequence for “center channel” compensation

The following description is given on a pr. band basis, and must be repeated for all bands e.g 850, 900, 1800 and 1900 MHz bands.

In the following sequence the power adjustement takes place in GSM 900 band. Every supported band must be adjusted.

11) Read up default DAC values from the phone.

```
UINT16 DEF_DAC[20]
[…]
DWD_get_pa_dac16_range( dwd_gsm_900 , 5 , 19 , &result , DEF_DAC , Handle );
```

12) Set normal GMSK modulation, 1 TX slot and training sequence (e.g. 5)

```
DWD_set_gmsk_mode( dwd_normal , 5 , Handle );
DWD_inl_set_nof_slot( 1 , Handle );
```

13) Setup ARFCN on centerchannel and enable transmission

```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 37 , Handle );
DWD_set_rf_mode( dwd_tx_burst , Handle );
```

14) Set Pout level and measure Pout at a selected number of PCLs

```
DWD_change_pa_level( MAX_PCL , 0 , Handle );
DWD_change_pa_level( PCL_2 , 0 , Handle );
DWD_change_pa_level( PCL_3 , 0 , Handle );
```

15) Setup ARFCN in low range and MAX_PCL

```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 975 , Handle );
DWD_change_pa_level( MAX_PCL , 0 , Handle );
```

16) Measure Pout at max PCL only.
17) Setup ARFCN in high range and MAX_PCL

```
DWD_set_rf_channel( dwd_tx_arfcn_tp , 124 , Handle );
DWD_change_pa_level( MAX_PCL , 0 , Handle );
```

18) Measure Pout at max PCL only.
19) Calculate Pa offset values for all PCL´s (first element is PCL 5 for GSM 900 and 850, PCL 0 for DCS1800 and PCS 1900)
20) Calculate Max Pa offset for low and high range (see following paragraph **Error! Reference source not found.**).
21) Store Pa offsets for centerchannel together with max PA offset low and max PA offset high values for channel compensation locally in DWDIO.DLL

```
UINT8 offset_lowch, offset_highch;
dwd_pa_offset_comp_table_type PA_OFFSET;
[…]
DWD_store_max_pa_ch_comp(dwd_gsm_900, dwd_high_ch , offset_highch , Handle);
DWD_store_max_pa_ch_comp(dwd_gsm_900, dwd_low_ch , offset_lowch , Handle);
DWD_store_pa_offset(dwd_gsm_900, &PA_OFFSET) , Handle ) )
```

13.4.1 Prototypes of functions involved

```
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);
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_set_gmsk_mode( unsigned int16 mode, unsigned int16 tsc ,int handle);
bool DWD_inl_set_nof_slot(unsigned char nof_tx_slot, int handle);
bool DWD_change_pa_level( unsigned char txpwr, unsigned char txpwr2, int handle);
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
bool DWD_store_pa_offset(unsigned char band, dwd_pa_offset_comp_table_type *pa_offsets_ptr, int handle);
bool DWD_store_max_pa_ch_comp(unsigned char band, unsigned char ch_high_low, signed char offset, int handle);
```

13.5 Switching TX transmission OFF

The transmission must be switched OFF after TX calibration of all bands using the following command:

```
DWD_set_rf_mode( dwd_stop , Handle )
```

13.5.1 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
```

13.6 Calculations

For each line segment in the interpolated Pout versus DAC characteristic shown in Figure 13-1, the offset and slope can be calculated by:

slope = (y1 –y2)/(x1- x2) *and* *offset = y1 –((y1 –y2)/(x1 – x2))*x1*

- where x1 and x2 are the default DAC values read from phone, and y1 and y2 are the corresponding measured Pout values converted to linear domain by:

*y = sqrrt(exp(Pout*ln(10)/10)*

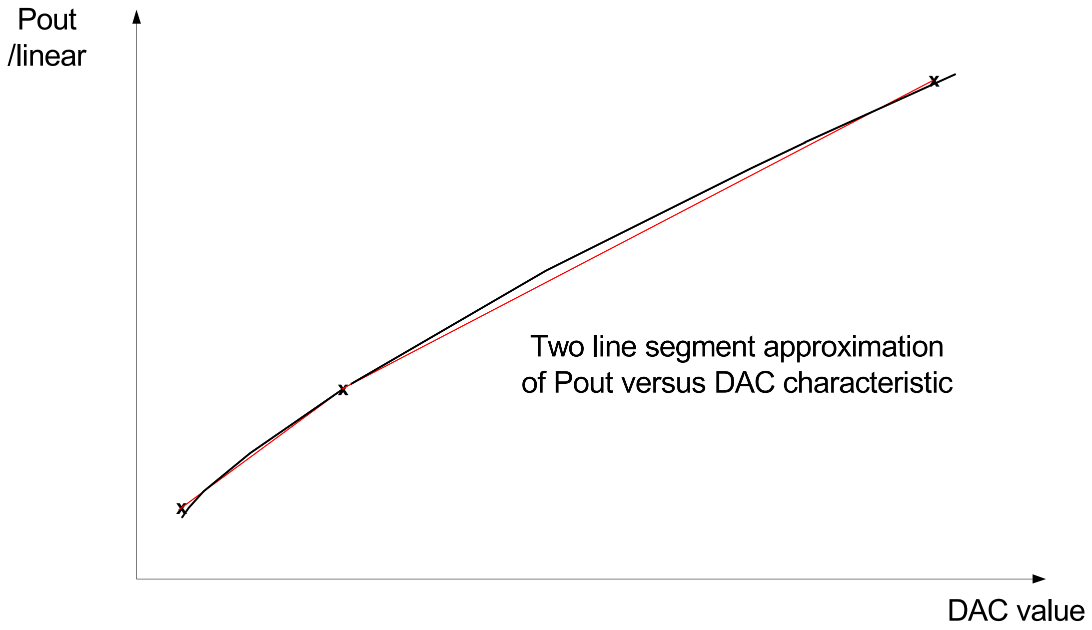


Figure 13-1

When slope and offset values for each line segment has been calculated, the assumed linear relationship between Pout and DAC value is used on every PCL to calculate the "correct" DAC value for any wanted output power level.

Recommended target output powers are listed in the Table 13-1 below. To calculate a correct DAC value for a given target output power, first convert target power to linear domain by the formula above.
Next use the formula:

Correct DAC value = (Y – offset)/slope

- where Y is the wanted output power converted to linear domain and slope and offset are the values for the relevant line segment.
Finally the PA offset value can be calculated as:

PA offset = correct DAC value – default DAC value.

For the channel compensation, note that following formula can be used (lowch as example)

temp_offset_lowch = (y_midch[max_level]; - y_lowch) / slope [max_level];

temp_offset_ch must be calculated as int16, and then checked towards range: the final values stored in target must be within -127 and +127, and stored as INT8 values.

13.7 Adjustment target values for GMSK.

Power level	GSM850	EGSM900	GSM1800	GSM1900
0	Na	na	29.2	29.2
1	Na	na	27.5	27.5
2	Na	na	26	26
3	Na	na	24	24
4	Na	na	22	22
5	32.2	32.2	20	20
6	30.5	30.5	18	18
7	29	29	16	16
8	27	27	14	14
9	25	25	12	12
10	23	23	10	10
11	21	21	8	8
12	19	19	6	6
13	17	17	4	4
14	15	15	2.5	2.5
15	13	13	1	1
16	11	11	na	na
17	9	9	na	na
18	7.5	7.5	na	na
19	6	6	na	na

Table 1. Adjustment target values for GMSK

13.8 Limits.

It´s recommended to check all initial power measurements before adjustment against limits in the order of:

Adjustment target values +/- 4 – 5 db

This should be done for the purpose of identifying hardware failures. Furthermore a continuous monitoring that the default values are not too far off the final values ensures a good quality of the calibration.

14 RX adjustment

RX compensation is implemented in BP30 platform in order to ascertain that reported RXL's during normal operation are within GSM specifications. Unlike TX, where just one compensation value accounts for unlinearity over both the dynamic and the frequency range, the RX compensation is implemented by adding of two values: a gain- and a channel -compensation value. There are 6 channel compensation values in 850 and 900 MHz band, and 8 values in 1800 and 1900 MHz bands. See Table 14-1 : ARFCNs for RX frequency compensation. For gain compensation, there are 70 entries in EEPROM, but those can be interpolated from just 3 readings at generatorlevels –55.5, –75.5 and –95.5 dbm .

The goal of the RX adjustment is to determine 6/8 channel compensation values and 70 gain compensation values. Like in the TX case, RX compensation is inactive in nonsignalling testmode, and hence the raw performance of the HW can be measured. The measurements are done as RSSI readings when applying known signal levels at different ARFCNs to the phone.

From the RSSI values read from the phone, RXL´s can be calculated the same way the phones SW calculate the RXL:

$$RXlev = 110 + floor((iqrms + 8)/16) - gain - HF\_CORR\_VALUE - HF\_CORR\_VALUE\_INL$$

Where

iqrms is the result read from phone with **DWD_get_iqrms** in dB/16.
floor denotes round to nearest integer.
HF_CORR_VALUE is a constant, which corrects for different RFIC and BBIC.
HF_CORR_VALUE_INL is a constant, which represents the difference between gain setting in nonsignalling test-mode and normal operational mode of the phone.
gain is the receiver gain, previously set with **DWD_set_rf_gain**

The testsignal from GSM tester must be a continuous signal, as the phone has no means of synchronizing to a bursted signal in non-signalling testmode. To excite the RSSI detector in the same way a modulated GSM signal does, the testssignal must either be gmsk modulated or alternatively offset 67 kHz from center frequency.

The following descriptions are given on a pr. band basis, and must be repeated for all bands e.g 850, 900, 1800 and 1900 MHz bands.

14.1 Adjusted parameters.

eep_static.rf_adjcomp.rxlev_ch_comp[0..3][0..7]
eep_static.rf_adjcomp.rxlev_gain_comp[0..3][0..69]

14.2 RX channel linearity adjustment.

14.2.1 Sequence

1) Set the input level to the phone to –75 dBm (see Table 14-1)
2) Set the phone to MON measuring

```
DWD_set_rf_mode( dwd_mon_burst , Handle );
```

3) Set phone receiver gain 57 (see **Table 14-2: Generator levels, gains, HF_CORR_VALUE and HF_CORR_VALUE_INL**): only specified values are admitted.

```
DWD_set_rf_gain( dwd_mon_gain_tp , 57 , Handle);
```

4) Set the phone to an ARFCN in 1st channel range according to Table 14-1 below.

```
DWD_set_rf_channel( dwd_mon_arfcn_tp , 12 , Handle );
```

5) Read up measured RSSI value from the phone

```
UINT16 *iqrms;
double *RX_level
double HF_CORR_VALUE=75.0;
 […]
DWD_get_iqrms( iqrms , Handle );
*RX_level = 110.0 + floor(( *iqrms + 8.0 )/16) -57 -HF_CORR_VALUE;
```

6) Set the phone to an ARFCN in 2nd channel range according to Table 14-1 below.

```
DWD_set_rf_channel( dwd_mon_arfcn_tp , channel , Handle );
```

7) ....
8) Set the phone to an ARFCN in last channel range according to Table 14-1 below.
9) Read up measured RSSI value from the phone
10) Calculate RX channel compensation values and store locally in DWDIO.DLL

```
INT8 ch_comp[8];
[…]
DWD_store_rxlev_ch_comp_offset(dwd_gsm_900 , &ch_comp , Handle );
```

14.2.2 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_set_rf_gain(unsigned int16 gain_tp, signed char gain, int handle);
bool DWD_get_iqrms( unsigned int16 *iqrms, int handle);
bool DWD_store_rxlev_ch_comp_offset(unsigned char band,dwd_ch_comp_rxlev_offset_table_type *rxlev_ch_offsets_ptr, int
        handle);
```

14.2.3 Calculations.

Based on the set of 6/8 RSSI readings over the frequency range, calculate the RXL's by the formula above. Next calculate the RX channel compensation values as:

$$rxlev\_ch\_comp[i] = rxlev\_ch\_comp[center] - rxlev\_ch\_comp[i]$$

where center ARFCN for each band emerges from table below

Reference : GSM850	**Offset**	**Ch. Measured**	**Channel Range**
dwd_ch_comp_rxlev_offset_Table_type[0]	RXLmeas[0]- RXLmeas[1]	138	128 to 147
dwd_ch_comp_rxlev_offset_Table_type[1]	**0**	**158**	**148 to 167**
dwd_ch_comp_rxlev_offset_Table_type[2]	RXLmeas[2]- RXLmeas[1]	178	168 to 187
dwd_ch_comp_rxlev_offset_Table_type[3]	RXLmeas[3]- RXLmeas[1]	198	188 to 207
dwd_ch_comp_rxlev_offset_Table_type[4]	RXLmeas[4]- RXLmeas[1]	218	208 to 227
dwd_ch_comp_rxlev_offset_Table_type[5]	RXLmeas[5]- RXLmeas[1]	238	228 to 251
Reference : GSM900	**Offset**	**Ch. measured**	**Channel Range**
dwd_ch_comp_rxlev_offset_Table_type[0]	RXLmeas[0]- RXLmeas[1]	12	0 to 25
dwd_ch_comp_rxlev_offset_Table_type[1]	**0**	**37**	**26 to 50**
dwd_ch_comp_rxlev_offset_Table_type[2]	RXLmeas[2]- RXLmeas[1]	63	51 to 75
dwd_ch_comp_rxlev_offset_Table_type[3]	RXLmeas[3]- RXLmeas[1]	88	76 to 100
dwd_ch_comp_rxlev_offset_Table_type[4]	RXLmeas[4]- RXLmeas[1]	113	101 to 124
dwd_ch_comp_rxlev_offset_Table_type[5]	RXLmeas[5]- RXLmeas[1]	1000	975 to 1023
Reference : GSM1800	**Offset**	**Ch. measured**	**Channel Range**
dwd_ch_comp_rxlev_offset_Table_type[0]	RXLmeas[0]- RXLmeas[3]	530	512 to 550
dwd_ch_comp_rxlev_offset_Table_type[1]	RXLmeas[1]- RXLmeas[3]	570	551 to 600
dwd_ch_comp_rxlev_offset_Table_type[2]	RXLmeas[2]- RXLmeas[3]	630	601 to 650
dwd_ch_comp_rxlev_offset_Table_type[3]	**0**	**670**	**651 to 700**
dwd_ch_comp_rxlev_offset_Table_type[4]	RXLmeas[4]- RXLmeas[3]	730	701 to 750
dwd_ch_comp_rxlev_offset_Table_type[5]	RXLmeas[5]- RXLmeas[3]	770	751 to 800
dwd_ch_comp_rxlev_offset_Table_type[6]	RXLmeas[6]- RXLmeas[3]	830	801 to 850
dwd_ch_comp_rxlev_offset_Table_type[7]	RXLmeas[7]- RXLmeas[3]	870	851 to 885
Reference : GSM1900	**Offset**	**Ch. measured**	**Channel Range**
dwd_ch_comp_rxlev_offset_Table_type[0]	RXLmeas[0]- RXLmeas[3]	530	512 to 550
dwd_ch_comp_rxlev_offset_Table_type[1]	RXLmeas[1]- RXLmeas[3]	570	551 to 590
dwd_ch_comp_rxlev_offset_Table_type[2]	RXLmeas[2]- RXLmeas[3]	610	591 to 630
dwd_ch_comp_rxlev_offset_Table_type[3]	**0**	**650**	**631 to 670**
dwd_ch_comp_rxlev_offset_Table_type[4]	RXLmeas[4]- RXLmeas[3]	690	671 to 710
dwd_ch_comp_rxlev_offset_Table_type[5]	RXLmeas[5]- RXLmeas[3]	730	711 to 750
dwd_ch_comp_rxlev_offset_Table_type[6]	RXLmeas[6]- RXLmeas[3]	770	751 to 790
dwd_ch_comp_rxlev_offset_Table_type[7]	RXLmeas[7]- RXLmeas[3]	800	791 to 810

Table 14-1 : ARFCNs for RX frequency compensation

14.2.4 Limits.

It’s recommended to check all initial RXL measurements before adjustment against limits in the order of:

Ideal RXL values +/- 4 – 5 db

14.3 RX gain linearity adjustment

14.3.1 Sequence

1) Set the phone to MON measuring

```
DWD_set_rf_mode( dwd_mon_burst , Handle );
```

2) Set the phone to the center ARFCN of the band

```
DWD_set_rf_channel( dwd_mon_arfcn_tp , channel , Handle );
```

3) Set the generator level to – 48 dBm
4) Set receiver gain to 23

```
DWD_set_rf_gain( dwd_mon_gain_tp , 23 , Handle);
```

5) Read up measured RSSI value from the phone

```
UINT16 *iqrms;
double *RX_level
double HF_CORR_VALUE=75.0;
[...]
DWD_get_iqrms( iqrms , Handle );
*RX_level = 110.0 + floor(( *iqrms + 8.0 )/16) -23 -HF_CORR_VALUE;
```

6) Set the generatorlevel to – 75 dBm
7) Set receiver gain to 57

```
DWD_set_rf_gain( dwd_mon_gain_tp , 57 , Handle);
```

8) Read up measured RSSI value from the phone

```
DWD_get_iqrms( iqrms , Handle );
*RX_level = 110.0 + floor(( *iqrms + 8.0 )/16) -57 -HF_CORR_VALUE;
```

9) Calculate RX gain compensation values

```
gain_comp[i] = Round(110+input_level[idx]-rx_level[idx]);
```

10) Store values locally in DWDIO.DLL

```
INT8 gain_comp[70];
[...]
DWD_store_rxlev_gain_offset(dwd_gsm_900 , &gain_comp , Handle );
```

14.3.2 Prototypes of functions involved

```
bool DWD_set_rf_mode(unsigned int16 mode, int handle);
bool DWD_set_rf_channel(unsigned int16 arfcn_tp, unsigned int16 arfcn, int handle);
bool DWD_set_rf_gain(unsigned int16 gain_tp, signed char gain, int handle);
bool DWD_get_iqrms( unsigned int16 *iqrms, int handle);
bool DWD_store_rxlev_gain_offset(unsigned char band,dwd_rxlev_offset_table_type *rxlev_offsets_ptr, int handle);
```

14.3.3 Calculations.

From the 2 RSSI readings on center ARFCN , the corresponding RXL's can be found by the formula above.

Due to the implementation of a constant gain receiver chain in the BP30 platform (mainly set by the integrated transceiver IC Smarti SD2) the RX level calibration is simplified. As the response of the overall gain versus input level is extremely linear over the whole input range it is sufficient to measure at only one point. The slope of the line is simply 1 [RXLEV] / 1 dB.

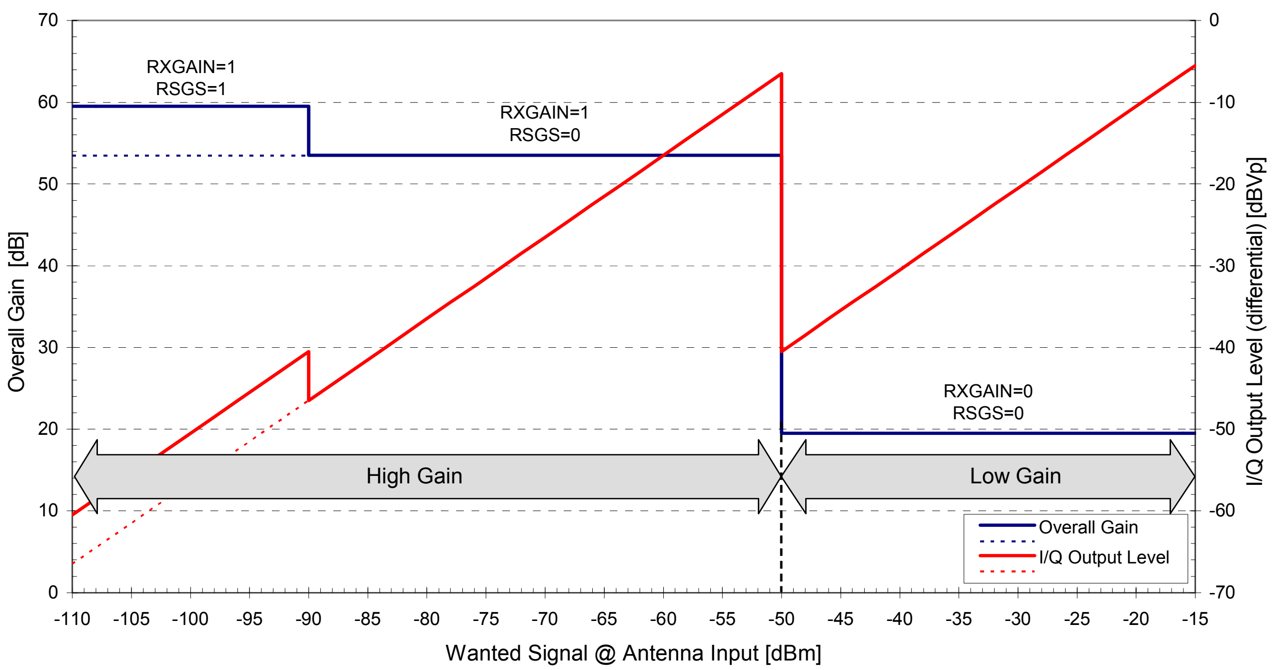
As can be seen from the pictures below there are 3 different states of the frontend gain between which the telephone is alternating in order to have the best dynamic performance at any moment. Thus it would theoretically be required to calibrate 3 different points that are spread among the 3 linear sections.
In fact the low gain and the first high gain mode are calibrated. There is no need to calibrate the highest gain section (RXGS = 1 for SD2) as the 3GPP specification allows a higher tolerance in the respective level range, anyway.

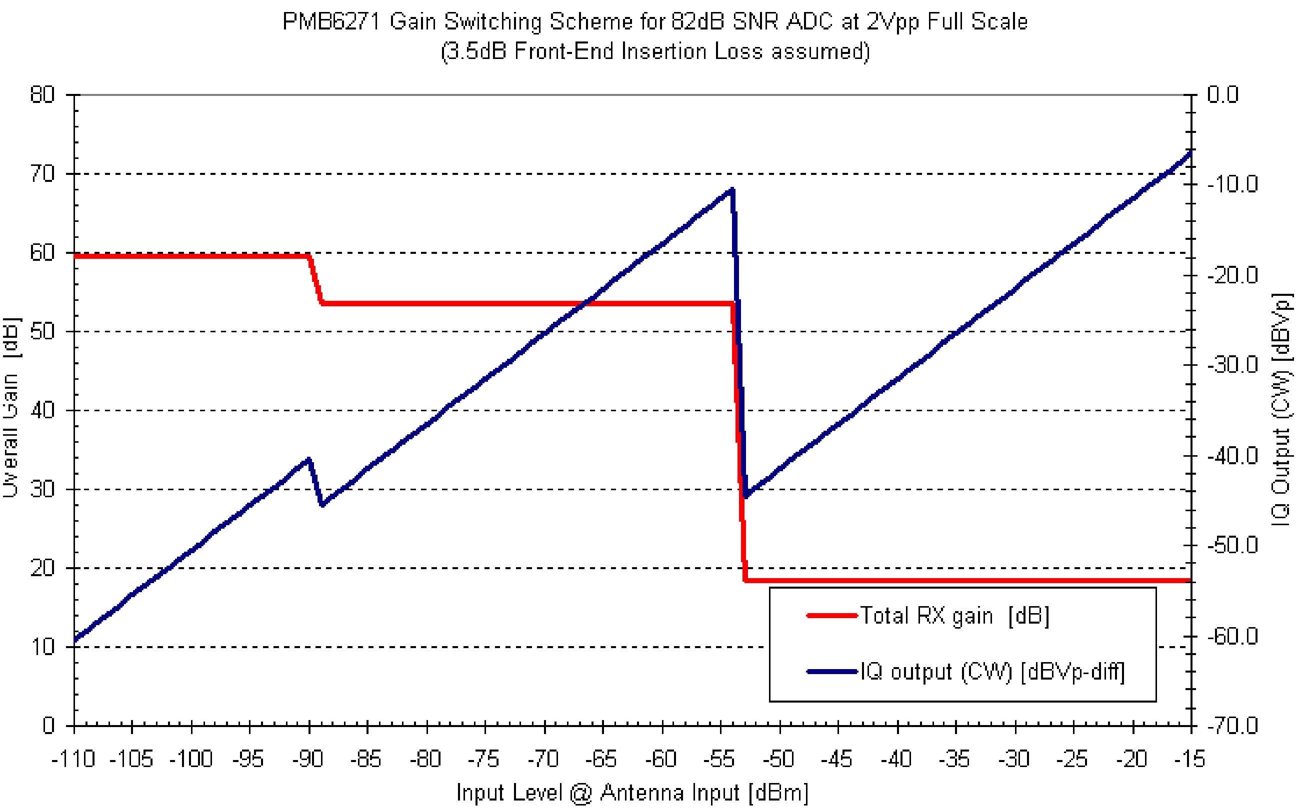
So the calibration yields only two correction values, one for high gain, evaluated at -75dBm, and one for low gain, evaluated at -48dBm. They are later used by the target software depending on what absolute gain is actually set for the respective radio frame. In order to use the same EEPROM parameters as the constant level receiver platforms those two values are stored under the same name:

eep_static.rf_adjcomp.rxlev_gain_comp[0..3][0..50] are filled with the value for high gain (57dB)
eep_static.rf_adjcomp.rxlev_gain_comp[0..3][51..69] are filled with the value for low gain (23dB)

Note that the entries at
eep_static.rf_adjcomp.rxlev_gain_comp[0..3][9] (→ highest gain),
eep_static.rf_adjcomp.rxlev_gain_comp[0..3][39] (→ high gain) and
eep_static.rf_adjcomp.rxlev_gain_comp[0..3][64] (→ low gain)
are used by the target software for correction.

PMB6270 Gain Switching Scheme Proposal for 82dB SNR ADC at 2Vpp Full Scale
(3.5dB Front-End Insertion Loss assumed; OGS=0)





The following setting are used for BP30 platform RX gain adjustement:

Receiver	**-48 dBm**	**-75 dBm**	*-75 dBm*	**HF_CORR_VALUE**	**HF_CORR_VALUE_INL**
Smarti SD2	23	57	*57*	75	0

Table 14-2: Generator levels, gains, HF_CORR_VALUE and HF_CORR_VALUE_INL

*Remark: the compensation values over frequency variation might be evaluated on a **statistical** basis during pre-production and stored as default values in EEPROM.*

14.3.4 Limits.

It´s recommended to check all initial RXL measurements before adjustment against limits in the order of:

Ideal values +/- 4 – 5 db

15 Sum of adjustement values

Sum of channel and gain compensation parameters should never exceed range [-6,+6].
Every value over this range is truncated during normal operation..

16 Store of calibration parameters to DUT EEPROM

During test and calibration all the values saved through the DWD_Store_XXXYYY functions are buffered in DWDIO.DLL and associated to handle.

16.1 Production params

Last step before storage is production data handling. It is possible to retrieve and modify a set of fields used for tracking and normal operation. In order to allow normal_mode regular behaviour the nof_test fileld in calibration related structure **must** be set to 1 or higher.

16.1.1 Modified parameters.

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

16.1.2 Sequence

1) Get calib tracking parameters

```
dwd_test_station_type calib;
[...]
DWD_get_test_station_parms( dwd_calib_mode , &calib , Handle );
```

2) Modify calib tracking parameters (e.g. increment nof_tests if nof_tests < 1)

```
calib.nof_tests++;
```

3) Store parameters locally in DWDIO.DLL

```
DWD_store_test_station_parms( dwd_calib_mode , &calib , Handle );
```

16.1.3 Prototypes of functions involved and data types

```
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;
```

```
bool DWD_store_test_station_parms(unsigned char test_station_id, dwd_test_station_type *parms, int handle);
bool DWD_get_test_station_parms(unsigned char test_station_id, dwd_test_station_type *parms, int handle);
```

16.2 EEPROM values storage

When the calibration has finished all the values modified can be flushed to DUT EEPROM.
This is obtained by the following function call

```
DWD_store_to_nv_memory( Handle );
```

16.2.1 Prototypes of functions involved

```
bool DWD_store_to_nv_memory(int handle);
```

17 TX/RX performance test

The mobile is in signaling test mode.
After communication has been established a call setup starts. When call is set up all RF tests are performed using CMU200 remote command interface. Voltage: 3.8V, T~25°C

17.1 Entering Signalling_test_mode sequence

1. Set DUT voltage to 3.8V
2. Power on DUT to ptest_mode
3. Set CMU200 to GSM900 overview mode, proper channel for BCCH and all BS param an needed
4. Send DWD_set_testmode(signalling mode,arfcn) command. The second parameter (arfcn) is used.to set channel for idle

```
DWD_set_test_mode(dwd_calib_mode, 38, Handle )
```

5. The phone is now in signaling test mode ready synchronize to CMU200 (assuming this is set to proper BCCH channel)
6. Poll DUT for status of synchronization (wait for Idle status)

```
Bool bIdle;
[...]
DWD_dut_in_idle( &bIdle , Handle )
```

7. Poll CMU200 for DUT status (MSYN)

17.1.1 Prototypes of functions involved

```
bool DWD_set_test_mode(unsigned int16 mode, unsigned int16 arfcn, int handle
bool DWD_dut_in_idle( bool *in_idle, int handle);
```

17.2 Call setup

1. Establish CALL

```
DWD_setup_emergency_call( Handle )
```

2. Poll CMU200 for DUT status (MCE)

17.2.1 Prototypes of functions involved

```
bool DWD_setup_emergency_call(int handle);
```

17.3 Sequence

After call setup every measurement is done through CMU200 test commands.

Explanations: BER_RBERRII:

4) GSM850 -106,5 dBm
5) GSM900 -106,5 dBm
6) GSM1800 -105,5 dBm
7) GSM1900 -105,5 dBm

Target Output Power for highest PCL is:

1) 32.00 dBm at **Ch189** for GSM850
2) 32.00 dBm at **Ch38** for EGSM
3) 29.25 dBm at **Ch698** for GSM1800
4) 29.25 dBm at **Ch661** for GSM1900

The following measurements are to be taken (tables are set on per band base):

Measurement GSM850	CH	PL	LL	UL	Remark
Power	128	5	31.25dBm	32.5dBm	
Burst	128	5	1(Pass)	1(Pass)	
Phase peak	128	5	0	12°	
Phase_rms	128	5	0	4.0°	
Freq_error	128	5	-75Hz	75Hz	
Switch Spectrum	128	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	128	5	-99 dBc -99 dBc	-60dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-50.5dBm	128	5	-54.5dBm	-46.5dBm	
BER_rberrII/-106.5 dBm	128	5	0	2.4%	
Power	189	5	31.25dBm	32.5dBm	
Burst	189	5	1(Pass)	1(Pass)	
Phase_peak	189	5	0	12°	
Phase_rms	189	5	0	4.0°	
Freq_error	189	5	-75Hz	75Hz	
Switch Spectrum	189	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	189	5	-99 dBc -99 dBc	-60dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-70.5dBm	189	5	-74.5dBm	-66.5dBm	
BER_rberrII/-106.5 dBm	189	5	0	2.4%	
Power	251	5	31.25dBm	32.5dBm	
Burst	251	5	1(Pass)	1(Pass)	
Phase_peak	251	5	0	12°	
Phase_rms	251	5	0	4.0°	
Freq_error	251	5	-75Hz	75Hz	
Switch Spectrum	251	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	251	5	-99 dBc -99 dBc	-60dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-106.5 dBm	251	5	-110.5dBm	-102.5dBm	
BER_rberrII/-106.5 dBm	251	5	0	2.4%	

Table 17-1

Measurement GSM900	CH	PL	LL	UL	Remark
Power	975	5	31.25dBm	32.5dBm	
Burst	975	5	1(Pass)	1(Pass)	
Phase peak	975	5	0	12°	
Phase_rms	975	5	0	4.0°	
Freq_error	975	5	-75Hz	75Hz	
Switch Spectrum	975	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	975	5	-99 dBc -99 dBc	-60dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-50.5dBm	975	5	-54.5dBm	-46.5dBm	
BER_rberrII/-106.5 dBm	975	5	0	2.4%	
Power	38	5	31.25dBm	32.5dBm	
Burst	38	5	1(Pass)	1(Pass)	
Phase_peak	38	5	0	12°	
Phase_rms	38	5	0	4.0°	
Freq_error	38	5	-75Hz	75Hz	
Switch Spectrum	38	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	38	5	-99 dBc -99 dBc	-60dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-70.5dBm	38	5	-74.5dBm	-66.5dBm	
BER_rberrII/-106.5 dBm	38	5	0	2.4%	
Power	124	5	31.25dBm	32.5dBm	
Burst	124	5	1(Pass)	1(Pass)	
Phase_peak	124	5	0	12°	
Phase_rms	124	5	0	4.0°	
Freq_error	124	5	-75Hz	75Hz	
Switch Spectrum	124	5	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	124	5	-99 dBc -99 dBc	-60dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-106.5 dBm	124	5	-110.5dBm	-102.5dBm	
BER_rberrII/-106.5 dBm	124	5	0	2.4%	

Table 17-2

Measurement GSM1800	CH	PL	LL	UL	Remark
Power	512	0	28.5dBm	30.0dBm	
Burst	512	0	1(Pass)	1(Pass)	
Phase_peak	512	0	0	12°	
Phase_rms	512	0	0	4.0°	
Freq_error	512	0	-150Hz	150Hz	
Switch Spectrum	512	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	512	0	-99 dBc -99 dBc	-58 dBc -30 dBc	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-50.5dBm	512	0	-54.5dBm	-46.5dBm	
BER_rberrII/-105.5 dBm	512	0	0	2.4%	
Power	698	0	28. 5dBm	30.0dBm	
Burst	698	0	1(Pass)	1(Pass)	
Phase_peak	698	0	0	12°	
Phase_rms	698	0	0	4.0°	
Freq_error	698	0	-150Hz	150Hz	
Switch Spectrum	698	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	698	0	-99 dBc -99 dBc	-58 dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-70.5dBm	698	0	-74.5dBm	-66.5dBm	
BER_rberrII/-105.5 dBm	698	0	0	2.4%	
Power	880	0	28. 5dBm	30.0dBm	
Burst	880	0	1(Pass)	1(Pass)	
Phase_peak	880	0	0	12°	
Phase_rms	880	0	0	4.0°	
Freq_error	880	0	-150Hz	150Hz	
Switch Spectrum	880	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
Modulation Spectrum	880	0	-99 dBc -99 dBc	-58 dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-105.5 dBm	880	0	-109.5dBm	-101.5dBm	
BER_rberrII/-105.5 dBm	880	0	0	2.4%	

Table 17-3

Measurement GSM1900	CH	PL	LL	UL	Remark
Power	512	0	28. 5dBm	30.0dBm	
Burst	512	0	1(Pass)	1(Pass)	
Phase_peak	512	0	0	12°	
Phase_rms	512	0	0	4.0°	
Freq_error	512	0	-150Hz	150Hz	
Switch Spectrum	512	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
ModulationSpectrum	512	0	-99 dBc -99 dBc	-58 dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-50.5dBm	512	0	-54.5dBm	-46.5dBm	
BER_rberrII/-105.5 dBm	512	0	0	2.4%	
Power	661	0	28. 5dBm	30.0dBm	
Burst	661	0	1(Pass)	1(Pass)	
Phase_peak	661	0	0	12°	
Phase_rms	661	0	0	4.0°	
Freq_error	661	0	-150Hz	150Hz	
Switch Spectrum	661	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
ModulationSpectrum	661	0	-99 dBc -99 dBc	-58 dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-70.5dBm	661	0	-74.5dBm	-66.5dBm	
BER_rberrII/-105.5 dBm	661	0	0	2.4%	
Power	810	0	28.5dBm	30.0dBm	
Burst	810	0	1(Pass)	1(Pass)	
Phase_peak	810	0	0	12°	
Phase_rms	810	0	0	4.0°	
Freq_error	810	0	-150Hz	150Hz	
SwitchSpectrum	810	0	-50 dBm -50 dBm	-25 dBm -28 dBm	Offset +/- 400 Offset +/- 600 kHz
ModulationSpectrum	810	0	-99 dBc -99 dBc	-58 dB -30 dB	Offset +/-400 and +/- 600 kHz Offset +/-200 kHz
Read_RxLevel/-105.5dBm	810	0	-109.5dBm	-101.5dBm	
BER_rberrII/-105.5 dBm	810	0	0	2.4%	

Table 17-4

18 BT baseband test

Bluetooth can be tested only if the DUT is set to pTestMode.
The tests include chip type detection, communication test, reset of chip and firmware reading.

18.1 Check HOST interface sequence

1. Set DUT voltage to 3.8V

2. Power on DUT to ptest_mode

3. Check HOST by following command

```
Bool bRet=false;
[…]
DWD_btd_check_host_interface(&bRet,Handle);
```

bRet is set to true when BT chip is detected.

4. Check HW interface.: it must be BlueMoon Universal (3) and serial interface must be set to UART (1)

```
unsigned int16 bluemoon_type = 0;
unsigned int16 serial_interface = 0;
[…]
DWD_btd_get_hw_info(&bluemoon_type,&serial_interface,Handle);
if (bluemoon_type != 3) {
   printf ("Wrong CHIP type: only BlueMoon Universal is supported");
   return false;
}
if (serial_interface != 1){
   printf("Wrong serial interface: only UART is supported");
   return false;
}
```

18.1.1 Limits

Bluemoon type must be checked to be BlueMoon Universal, so the type should be 3.
The supported serial is UART.

18.2 Check firmware sequence

1. DUT is set in ptest_mode

2. Read firmware version

```
dwd_btd_bmu_firmware_version_type bmu_data;
[…]
DWD_btd_get_bluemoon_firmware_version(&bmu_data,handle);
```

18.3 Read BT address

1. Read BT address (can be used to speed up further connection skipping enquiry).

```
dwd_btd_address_type data;
[…]
DWD_btd_bmu_get_address(&data,Handle);
```

18.4 Structure types involved

```
typedef unsigned char dwd_btd_address_type[6];

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;
```

18.5 Prototypes of functions involved

```
bool DWD_btd_check_host_interface(bool *result, int handle);
bool DWD_btd_get_hw_info(unsigned int16 *bluemoon_type, unsigned int16 *serial_interface_type, int handle);
bool DWD_btd_get_bluemoon_firmware_version(void *version,int handle);
bool DWD_btd_bmu_get_address(dwd_btd_address_type *btd_address, int handle);
```

19 BT RF power test (non signaling)

Bluetooth can be tested only if the DUT is set to pTestMode.
This tests TX power verification.
Non signaling test can be performed by powermeter, spectrum analyzer or RF tester (CMU200)

19.1 Check HOST interface sequence

1. Set DUT voltage to 3.8V
2. Power on DUT to ptest_mode
3. Check HOST by following command

```
Bool bRet=false;
[…]
DWD_btd_check_host_interface(&bRet,Handle);
```

bRet is set to true when BT chip is detected.

19.2 TX non signaling sequence

1. Set TX parameters :

```
unsigned char packet;
unsigned char tx_channel;
unsigned char tx_interval;
unsigned char pattern;
unsigned char tx_coarse;
unsigned char tx_fine;
[…]
packet = 0x01;          // DH1
tx_channel = 0;         // 0..78
tx_interval = 2;
tx_fine = dwd_btd_bmu_tx_power_offset_nominal;
```

2. Set parameters on analyzer

3. Start TX transmission in non signaling mode

```
[…]
DWD_btd_set_tx_burst_mode(packet, tx_interval, tx_channel, 0, tx_fine, 0, Handle);
```

4. Measure TX power and check towards limits

5. To stop transmission the BT chip can be reset by the following function call (check result bRet)

```
bRet=false;
 […]
DWD_btd_sw_reset(&bRet, Handle);
```

19.2.1 Limits

The tests should be done on 3 TX-Channels (Table 19-1).

	min	mid	max
TX channel	20	tbd	60

Table 19-1

The average output power should be measured at the TX channels and must be within following limits

	Upper Limit	Lower Limit
output power	+ 4.5 dBm	- 6.0 dBm

Table 19-2

In case the test is used to check BT antenna connection it is not needed to perform test on all three channels (in this case the whole test must be performed in earlier stage).

19.3 Prototypes of functions involved

```
bool DWD_btd_check_host_interface(bool *result, int handle);
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);
bool DWD_btd_sw_reset(bool *result, int handle);
```

20 BT signaling test (BT test mode)

The following test is performed using R&S CMU200 with BT option K53.
The mobile is powered up in ptestmode, BT testmode is set and the tests are performed through CUM200.
Two options are available (test time can change according to the choosen one): in the sequence below they are marked as option A and option B.

20.1 Check HOST interface sequence

1. Set DUT voltage to 3.8V
2. Power on DUT to ptest_mode
3. Check HOST by following command

```
Bool bRet=false;
[…]
DWD_btd_check_host_interface(&bRet,Handle);
```

bRet is set to true when BT chip is detected.

20.2 Testmode sequence

1. Read BT address (option A).

```
dwd_btd_address_type data;
[…]
DWD_btd_bmu_get_address(&data,Handle);
```

2. Set tester in BT mode

3. Start BT testmode on target

```
Bool bRet;
[…]
DWD_btd_enter_rf_test_mode(&bRet,Handle);
```

4. Set tester according to BT address read (option A).

5. Set TX parameters on tester (Table 20-1).

Testmode Type	Transmitter
Hopping Scheme	RX/TX single Frequency
Pattern Type	10101010
Packet Type	DH1
Length of Test Sequence	27 byte
Whitening	off
Statistic Count	10 Bursts

Table 20-1

6. Start enquiry sequence on tester to detect DUT and collect BT address (option B).

7. Connect to DUT.

8. Perform all TX tests (including frequency drift) driving CMU as master (DUT is the slave)

9. Check measures towards limits (Table 20-4).

10. Set RX parameters on tester ().

Testmode Type	Loopback
Hopping Scheme	RX/TX single Frequency
Pattern Type	Static PRBS
Packet Type	DH3
Length of Test Sequence	183 byte
Whitening	off
Delay	off
Packets	69
Master Signal TX-Level	-73 dBm
Dirty Transmitter Mod Index	off
Dirty Transmitter Freq Offset	off

Table 20-2

11. Perform all RX tests driving CMU as master (DUT is the slave)

12. Check measures towards limits (Table 20-5)

13. Detach DUT from tester

14. Reset BT chip

```
bRet=false;
 [...]
DWD_btd_sw_reset(&bRet, Handle);
```

20.2.1 Limits

Tests must be performed on 3 TX channels (Table 19-1) and on 3 RX channels (Table 20-3)

	min	mid	max
RX channel	20	tbd	60

Table 20-3

All measured TX values should be in between limits (Table 20-4)

	Upper Limit	Lower Limit
Average output power	+ 4.5 dBm	- 6.0 dBm
frequency accuracy	+ 75 kHz	- 75 kHz
frequency drift	+ 25 kHz	- 25 kHz

Table 20-4

Measured RX values should be in between limits (Table 20-5)

	Upper Limit
BER	0.1 %

Table 20-5

20.3 Prototypes of functions involved

```
bool DWD_btd_check_host_interface(bool *result, int handle);
bool DWD_btd_enter_rf_test_mode(bool *result, int handle);
bool DWD_btd_sw_reset(bool *result, int handle);
```

21 BT address storage

A valid and unique BT address can be stored at the end of BT verification by the following test sequence.

21.1 BT address storage sequence

1. Write BT address in DUT (the example below uses a dummy value)

```
dwd_btd_address_type btd_address;
 […]
btd_address[0]=0x00;
btd_address[1]=0x01;
btd_address[2]=0x02;
btd_address[3]=0x03;
btd_address[4]=0x04;
btd_address[5]=0x05;
DWD_btd_bmu_store_address(, Handle);
```

21.2 Prototypes of functions involved

```
bool DWD_btd_bmu_store_address(dwd_btd_address_type *btd_address, int handle);
```

22 Final step

The last step of communication toward DUT must always be the closure of the com handle.
This step must be executed even if an error has been detected during tests.

```
DWD_close_com_port(Handle);
```

22.1 Prototypes of functions involved

```
bool DWD_close_com_port(int handle);
```

23 References

23.1 External

23.2 Internal

Title	Doc ID

24 Document change report

	Change Reference		Record of changes made to previous released version	
Rev	**Date**	**CR**	**Section**	**Comment**
1.0	14/07/2005		Document stored on server	
1.1	25/07/2005	Diego Barbana	Hole through and performance tests added	NA
1.2	27/12/2005	Diego Barbana	Blue Tooth tests added	NA
1.3	28/12/2005	Diego Barbana	Sections 12.2,13.4 and 13.6 changed	NA
1.4	28/12/2005	Diego Barbana	Sections 10.3,20.2 changed	S/N fixed. BT frequency drift settings added

25 Approval

Revision	Approver(s)	Date	Source/signature
1.4	Valter Jelicic	16/02/2006	Document stored on server