Technical Document

L1/DSP-DRP API DEFINITION

Document Number:	Document number to be assigned
Version:	1.10
Status:	Draft
Approval Authority:	Francois Mazard, Daniel Ezekiel, Francois Goeusse, Vincent Roussel
Creation Date:	2004-Nov-14
Last changed:	1 Feb 2006, Shrinivas Gadkari
File Name:	locosto_drp2_api_13_01_04_02719.doc

Important Notice

Change History

Date	Changed by	Approved by	Version	Status	Notes
15 Nov 2004	Francois Goeusse		0.0		Reference API document creation.

30 Mar 2005	Shrinivas Gadkari	Francois Mazard, Daniel Ezekiel, Francois Goeusse, Vincent Roussel	1.0	Draft	Restructuring, and rewriting reference API doc to reflect the state of Caplyso+-Uppcosto test platform scripts and calibration code. Updated the document corresponds to scripts 130.02 and Uppcosto version 2.02 calibration code adapted for Caplypso+.
10 May, 2005	Shrinivas Gadkari	Francois Mazard, Daniel Ezekiel, Francois Goeusse, Vincent Roussel	1.01		1. Added details on data structure storage in Flash, RAM assumed in the code. 2. RX_ON takes a input data in a consolidated format. 3. Added Test Mode Support for calibration section. 4. Editing of text, consolidation of figures tables. 5. Modified mixer, SCF pole calibration. 6. Added RX_ON, TX_ON details in appendix. 7. MACRO definitions, external dependencies listing in appendix.
26 May 2005	Shrinivas Gadkari	Francois Mazard, Daniel Ezekiel, Vincent Roussel	1.02		1. Changed RX_ON input parameter API structure element name. 2. Added API structure in appendix A.
3 June 2005	Shrinivas Gadkari	Francois Mazard, Daniel Ezekiel	1.03		Added BAND_INDEX input parameter for drp_afe_gain_calib1, and drp_afe_gain_calib2.
27 Jul 2005	Shrinivas Gadkari		1.04		Script timing update for 131.04
27 Jul 2005	Shrinivas Gadkari	Francois Mazard, Daniel Ezekiel	1.05		Update for Calibration-L1 integration
1 Aug 2005	Sumeer Bhatara		1.06		Script timing update for 131.05
19 Aug 2005	Sumeer Bhatara		1.07		Script timing update for 132.00
16 Sept 2005	Shrinivas Gadkari	Daniel Ezekiel	1.08		Update the drp calibration ETM commands
24 Oct 2005	Shrinivas Gadkari		1.09		Add DCO retiming and DCO ibias calibration.

1 Feb 2006	Shrinivas Gadkari		1.10		Update for scripts 134.01 Input signal for DRP calibration level changed. DCXO calibration supports GSM and EGSM band Multi-PCB configurations support RSSI gain estimation correction.

Table of Contents

1 Introduction

DRP reference SW stands for the complete SW delivered with the DRP HW module, which only together constitute a functional RF module.

This document provides a comprehensive description of the DRP Reference SW with respect to:

1. Interface specification with layer-1 code, including details of the data structures used.
2. Explanation of the Flash-RAM usage assumed within the code structure.
3. Detailed listing of API data structure, macros and build option definitions – along with the highlighting of the layer-1 dependencies contained in them.

Known Gaps in the Document:

1. Need for Gain compensation is not yet clear from characterization results.
2. DCXO calibration currently under study.
3. Temperature calibration currently under study.

2 DRP Reference SW Structure

The reference SW consists of the following entities:

1. The assembled DRP scripts and the register initialization data is contained in the data structure drp_ref_sw[ ]. This is a constant data, that is stored in the flash. The reference SW provides the routine drp_copy_ref_sw_to_drpsrm( ), to copy this data from the flash to the drp SRM memory. The Layer1 code is expected to call this function and pass a pointer to drp_ref_sw[ ].
2. The DRP SRM address information for downloading this data is contained within these data arrays itself as detailed in Appendix C.
3. Calibration data is managed via data structures of type T_DRP_SW_DATA.
    a. The data structure drp_sw_data_init stored in the flash contains the pre-calibration default values. During the calibration process itself, the layer-1 code creates a copy of this structure in RAM. Let us call this copy of the drp_sw_data_init in RAM as, drp_sw_data_calib. The layer1 code passes a pointer to drp_sw_data_calib to each of the calibration routines.
    b. The calibration routines modify the contents of the data structure drp_sw_data_calib. At the end of the calibration process, the modified drp_sw_data_calib structure is stored form this task saved by layer1 code into the FFS. Let us denote this saved copy of the calibrated data structure drp_sw_data_calib_saved.
    c. During the phone mode operation, the layer1 code passes a pointer to the structure drp_sw_data_calib_saved, to the reference SW routine drp_copy_sw_data_to_drpsrm( ), which copies the calibration data into the DRP SRM.
4. C routines are called by the layer-1 code during factory calibration of the phone. These are executed directly from the flash program memory. The minimal RAM required for the C-code variables is to be allocated from the heap RAM resource. The size of the stack needed is ???.
5. Note: There will be a single build that includes both the phone mode operation code and calibration routines.

Notes:
a. The data in the structure drp_sw_data_init is already included in drp_ref_sw[ ]. So copying drp_ref_sw into SRM includes download of the default values of the pre-calibration values of drp_sw_data.

2.1 Multiple PCB Configurations Support

It is anticipated that the Locosto device would be configured differently for RF connections by individual customers, see [1]. In this context there are seven configurations that need to be supported by the DRP SW. The DRP scripts and C-code will be reused as-is on each of the configurations. A global macro RF_BAND_SYSTEM_INDEX that will define the PCB configuration used. The layer-1 code will use this macro definition for DCXO calibration (section 4.3.1), and RX_ON input generation (section 3.1.2).

2.2 List of C Code Files

The C files in the reference SW are summarized here.

2.2.1 Source Files

1. drp_calib_main.c: This file contains all the calibration routines.
2. drp_main.c: There are two types of routines included within this file:
    a. Wrapper code routines to adapt the code in the drp_calib_main.c to the Locosto API.
    b. Routines to copy script and calibration data from Flash to DRP SRM.
3. drp_script_data.c: This file defines data structure that contains the assembled scripts and register initialization data. The format of these structures is summarized in Appendix C.

2.2.2 Header Files

1. drp_extern_dependencies.h: This file contains the macro and constant definitions needed by the DRP calibration software that are closely related to the integration of the calibration code with the layer1 code. Details of this file are included in the Appendix B.
2. drp_defines.h: This files defines constants that are internal to the DRP code.
3. drp_API.h: This files defines the API data structure, and the data structures used for calibration, and the drp registers structure. The details of the API data structure are given in Appendix A.
4. drp_ver.h: Contains the version of the C-code.
5. drp_calib_main.h: Header file for drp_calib_main.c.
6. drp_main.h: Header file for the drp_main.c file.

2.2.3 Layer1-DRP Integration Dependency Checks

For the sake of Layer1-DRP integration convenience we list some important points to ensure they are not missed:

1. The drp_ref_sw[ ], and drp_sw_data_init[ ] are declared as constants and would be stored in the flash.
2. Following constants defined in drp_extern_dependencies.h file are platform dependent:
    a. DRP_REGS_BASE_ADD is the base address of the DRP registers/ internal memory as seen from MCU.
    b. DRP_SRM_API_ADD is the MCU view of the API structure address in the DRP SRM.
    c. DRP_SRM_DATA_ADD is the MCU view of the calibration structure address in the DRP SRM.
    d. DRP_EXTERNAL_MEMORY_ADD is the MCU view address of the DRP external memory.
    e. DRP_EXTERNAL_MEMORY_SIZE is the size in bytes of the DRP external memory.
3. The macros: READ_WORD, WRITE_WORD, WRITE_BYTE_HIGH, WRITE_BYTE_LOW, WRITE_WORD_AT_PTR will be platform dependent. The word to read/write can be in SRM memory or in external DRP memory. These macros are defined in Appendix D.
4. The macro to implement a wait, WAIT_US, will be platform dependent and needs to implement the prescribed wait for correct functioning of the DRP scripts. This macro are defined in Appendix D
5. The last 512 bytes of the DRP internal RAM and DRP external (wrapper) have fast access from the DSP. These memory locations should be used for locating the DRP API. Specifically, the API data structure will be located within the RAM segments: 0x2E00-0x2FFF, and 0x10E00-0x10FFF (DRP OCP addresses).
6. A global macro RF_BAND_SYSTEM_INDEX will be defined in the layer-1 code. It will be set to one of the following seven constants also defined in layer-1 code: RF_QUADBAND, RF_EU_TRIBAND, RF_EU_DUALBAND, RF_US_TRIBAND, RF_US_DUALBAND, RF_PCS1900_900_DUALBAND, RF_DCS1800_850_DUALBAND.

3 DRP Phone Mode Operation

To use the DRP reference SW in the phone mode the layer-1 code needs to perform the following to data tasks:

1. Layer1 code calls the reference SW routine, drp_copy_ref_sw_to_drpsrm( ), while passing pointer to the structure drp_ref_sw[ ]. This operation loads the following into the DRP memory (registers and SRM and external): (i) assembled scripts[1], (ii) assembled compensation routines running on the script processor, and (ii) register initializations.
2. The Layer1 code then calls the function drp_copy_sw_data_to_drpsrm( ) while passing the pointer to the structure drp_srm_data_calib_saved to load the calibration data into DRP SRM. (Note that the order of tasks 1 and 2 should be maintained for correct functioning of the DRP scripts.)
3. Invoke the DRP scripts using the DRP-layer1 API as described in the remainder of this section.
4. The Layer 1 can access to the DRP register through the DRP register structure drp_regs which is a structure of type - T_DRP_REGS_STR, specified in the file drp_API.h.
5. The location of the API data structure in the DRP SRM is specified by the constant DRP_SRM_API_ADD defined in the file drp_extern_dependencies.h file.

3.1 DRP Script Usage

To switch from one state to another, the layer1 code is required to execute DRP scripts as described next. In general, a script expects some input data in a particular format to be written into specified elements of the API data structure located within the DRP SRM. Also after the script execution, the script writes the output data into the specified elements of this API data structure.

Starting Scripts: Starting of a script is controlled by the SCRIPT_STARTL register as:

- Write the number of the script to be started in bits(3:0).
- Set the bit(7) to one.

SCRIPT_STARTL Address 0x0500

Bit	Name	Function	Default Value
7	script_start	Writing a '1' to this location starts the script number indicated by the field 'script_num'.	0x0
6:4	not used		
3:0	script_num	The script number of the script to be started, 0-15.	0x0

3.1.1 TX_ON

Script Number: 1

Description: This script is executed to put the DRP in a mode ready to transmit. See Appendix F for details.

Duration: 177 qbits (with the script processor running at 104 MHz).

Input Parameters

API Data Structure

- UWORD16 drp_srm_api->control.retiming
  - This parameter is used to enable/disable the clock retiming selectively. The specific mapping between the values of this parameter and the retiming enable/disable functionality is:
    - 0x0000 – retiming disable

[1] DRP Scripts are SW routines executing on the script processor of the DRP. These are described in section 3.1.

- 0x0001 – retiming enable in Tx-mode only
- 0x0002 – retiming enable in Rx-mode only
- 0x0003 – retiming enable in Tx and Rx mode

DRP register

RF_FREQL Address 0x0150

Bit	Name	Function	Default Value
15:0	Mem_rf_freq	DLO carrier frequency programmed in 100 kHz increments. The DLO internally adjusts for high / low band operation, so the programming for high band should be 17100 – 19900 while the programming for low band should be 8240 – 9600. Default from SRM for band select is low band (E-GSM), so low band frequency is programmed as default.	0x2032

Note: When the script is completed, DRP is ready for transmission, waiting for TX_START signal to be asserted. In case retiming feature is enabled during transmission, the script will take care of setting the proper DLO division ratio to re-generate DBB clocks matching the requirements.

Current restriction of use

TX_ON must be called from the DRP IDLE state.

Timing Details

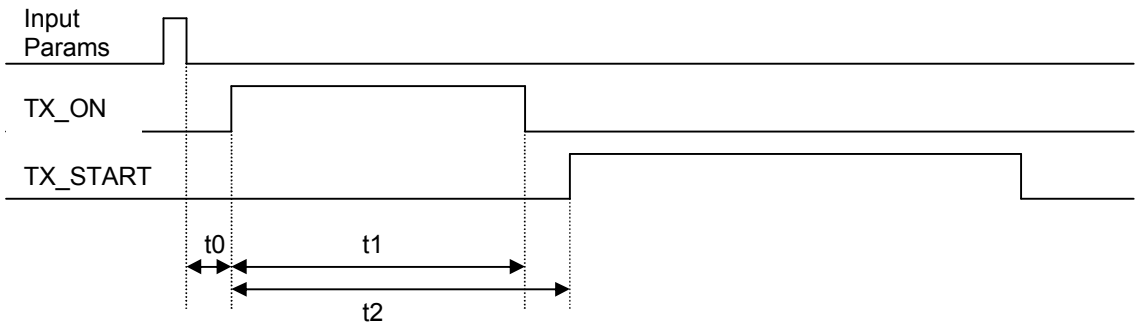

Figure 1 - TX_ON time line

Name	Description	Value		
		Min	Typical	Max
t0	Input parameters program→TX_ON start	> qbits	>0 qbits	
t1	TX_ON Duration	-	177 qbits	250 qbits
t2	Delay Tx_ON start → Tx START	-	TBD	250 qbits

3.1.2 RX_ON

Script Number: 2

Description: This script is executed to put the DRP in a mode ready to receive. See appendix F for details.

Duration: 146 qbits (with the script processor running at 104 MHz).

Input Parameters

API Data Structure

- UWORD16 - dr_srm_api->control.retiming
  - This parameter is used to enable/disable the clock retiming selectively. The specific mapping between the values of this parameter and the retiming enable/disable functionality is:
    - 0x0000 – retiming disable
    - 0x0001 – retiming enable in Tx-mode only
    - 0x0002 – retiming enable in Rx-mode only
    - 0x0003 – retiming enable in Tx and Rx mode
- UWORD16 - drp_srm_api->inout.rx.rxon_input
  - Bit(0) used to select the desired IF frequency as:
    - Bit(0) = 0, for IF frequency of 0 KHz.
    - Bit(0) = 1, for IF frequency of 100 KHz.
      - Depending on the RF Validation results this frequency value may change.
- UWORD16 - drp_srm_api->inout.rx.rxon_input
  - Bit(1) is used to enable/ disable the gain compensation algorithm as
    - Bit(1) = 0, to disable the gain compensation.
    - Bit(1) = 1, to enable the gain compensation.
- UWORD16 - drp_srm_api->inout.rx.rxon_input
  - Bits(5:2) specify the ABE gain and Bit(6) specifies the AFE gain as:
    - Bits(5:2) = 0, corresponds to ABE gain = 0 dB
    - Bits(5:20) = 1, corresponds to ABE gain = 2 dB
    - Bits(5:2) = 2, corresponds to ABE gain = 5 dB
    - Bits(5:2) = 3, corresponds to ABE gain = 8 dB
    - Bits(5:2) = 4, corresponds to ABE gain = 11 dB
    - Bits(5:2) = 5, corresponds to ABE gain = 14 dB
    - Bits(5:2) = 6, corresponds to ABE gain = 17 dB
    - Bits(5:2) = 7, corresponds to ABE gain = 20 dB
    - Bits(5:2) = 8, correspond to a ABE gain = 23 dB
    - Bit(6) = 0, corresponds to a AFE gain = 11 dB
    - Bit(6) = 1, corresponds to a AFE gain = 38 dB
- UWORD16 - drp_srm_api->inout.rx.rxon_input
  - Bit(7) used to select the switched capacitor filter (SCF) corner freq setting as
    - corner_freq = 0, to set the SCF corner frequencies to 270 KHz, 400 KHz, the mixer corner freq is set to 400 KHz.
    - corner_freq = 1, to set the SCF corner frequencies to 400 KHz, 400 KHz, the mixer corner freq is set to 400 KHz.
- UWORD16 - drp_srm_api->inout.rx.rxon_input
  - Bit(9:8) used to set the parameter DBBIF setting, needed for selecting the LNA input for DRP. This value is obtained by calling the drp function drp_generate_dbbif_setting_arfcn ( ) (see section 3.2.1).

DRP register

RF_FREQL Address 0x0150

Bit	Name	Function	Default Value

15:0	mem_rf_freq	DLO carrier frequency programmed in 100 kHz increments. The DLO internally adjusts for high / low band operation, so the programming for high band should be 17100 – 19900 while the programming for low band should be 8240 – 9600. Default from SRM for band select is low band (E-GSM), so low band frequency is programmed as default.	0x2032

Output Parameters

- UWORD16 – drp_srm_api->inout.rx.agc.output.compensated_gain
  - Specifies the total gain (AFE+ABE) applied by the gain compensation algorithm, the value is specified in 1024*log2(gain) format.

Note: When the script is completed, DRP is ready for receiving, waiting for RX_START signal to be asserted. In case retiming feature is enabled during transmission, the script will take care of setting the proper DLO division ratio to re-generate DBB clocks matching the requirements.

Current restriction of use

RX_ON must be called from the DRP IDLE state.

Timing Details

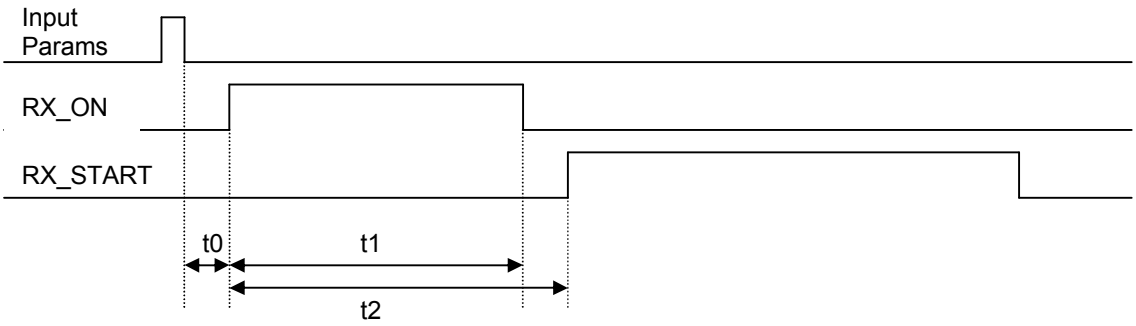

Figure 2 - RX_ON time line

Name	Description	Value		
		Min	Typical	Max
t0	Input parameters programming → RX_ON	-	> 0	-
t1	RX_ON script duration	-	146 qbits[1]	158 qbits
t2	RX_ON → RX_START HIGH	-	TBD	158 qbits

[1] The maximum time period allowed from the end of last DRP burst (Rx or Tx) to end of the RX_ON script is 145 us. This includes the IDLE script execution, AFC script execution (if needed).

3.1.3 ROC

Script Number: 4

Description: This script is automatically called on RX_START rising edge to perform residual offset measurements in single or multislot mode.

Duration: This script runs in the back ground when the RX_START signal is high and the burst samples are being received.

Input Parameters

SRM data

- UWORD16 - drp_srm_api->inout.rx.roc.input.window_start
  - This is the sample number within the burst to start accumulation for residual DC estimation.
- UWORD16 - drp_srm_api->inout.rx.roc.input.window_stop
  - This is the sample number within the burst to stop accumulation for residual DC estimation.
- UWORD16 - drp_srm_api->inout.rx.roc.input.lut_start_phase
  - This is the lookup table phase index is reset to the value indicated by this parameter after the RX_START is raised high.

DRP register

None

Output Parameters

- WORD32 - drp_srm_api->inout.rx.roc.output.accum_I[4]
  - The accumulated value of the DC Offset on the I-path for upto 4 consecutive Rx bursts, with accum_I[0] corresponding to the accumulated offset value for the slot number 0.
- WORD32 - drp_srm_api->inout.rx.roc.output.accum_Q[4]
  - The accumulated value of the DC Offset on the I-path for upto 4 consecutive Rx bursts.

Notes:

1. Measurements given are accumulation values, meaning that the ROC accumulator values have to be divided in the DSP by the number of samples accumulated.
2. The sampling frequency for ROC computation is 4 times the GSM symbol rate. Thus the sample numbers for starting and stopping the accumulation are to be specified at this sample rate.
3. The arrays are filled with the accumulator values per time slot in the sequence 0, 1, 2, 3.
4. Irrespective of the number of time slots the script is active, the next time the script is called the index for filling the array is reset to zero.

Event Return

None

Timing Details

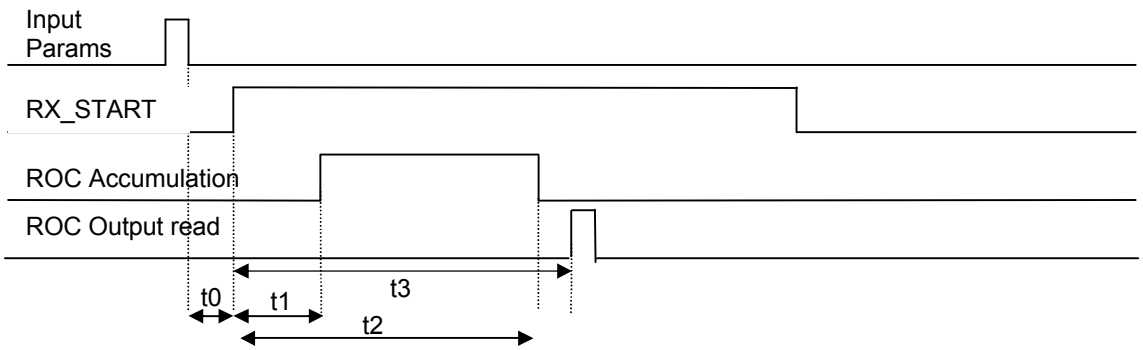

Figure 3 - ROC time line Single slot

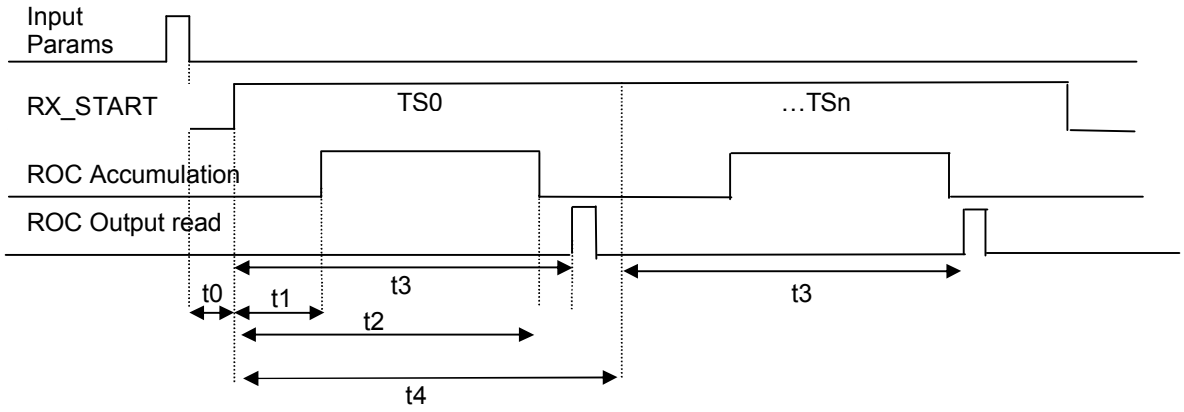

Figure 4 - ROC time line Multi Slot

Name	Description	Value		
		Min	Typical	Max
t0	Input parameters programming → RX_START	> 0	NA	NA
t1	Time slot start → First accumulated sample in time slot	64 qbits[1]	NA	500 qbits
t2	Time slot start → Last accumulated sample in time slot	64 qbits[1]	NA	500 qbits
t3	Time slot start → ROC output ready for read	520 qbits	520 qbits	520 qbits
t4	Time slot duration	625 qbits	625 qbits	625 qbits

3.1.4 IDLE

Script Number: 15

Description: This script puts the DRP in an idle mode where the Tx, Rx and the digitally controlled oscillator (DCO) are shut off. The LDOs however are on.

Duration: 10 qbits

Input Parameters

None

[1] Here assumed 8 Symbols IIR Group Delay

3.1.5 AFC

Script Number: 13

Description: This script is run to update the control word of DCXO frequency (Automatic Frequency control - AFC). It compares the value of the mem_xtal requested by the L1/DSP with the current value of DCXO_XTALL register and changes it in steps of +/- 2 ppm with a wait period of 10 us to allow for DXCO amplitude settling after each change.

Duration:
Approximately given by CEILING(Step/2ppm)*(10μs)

Step is the difference between current AFC control word and the control word set in mem_xtal.

Input Parameters

SRM data

- WORD16 – drp_srm_api->inout.afc.input.mem_xtal
  - mem_xtal value is 14-bit signed value. Its is sign extended to a 16-bit integer number.
  - This computed by AFC algorithm in the DSP

DRP register

DCXO_XTALL Address0x0440

Bit	Name	Function	Default Value
13:0	mem_xtal_ctl	14 bit control of DCXO frequency Top 10 bits are direct DAC control. Bottom 4 bits are Sigma-Delta. (This is signed format – 0x0000 is midrange value) 0x1FFF – max frequency, 0x2000 – min frequency Cannot access this register every clock! Must wait for the synchro-nizing circuit to step through its modes.	0x2000

Note:
AFC script must be called in DRP IDLE mode only.

3.1.6 REG_ON

Script Number: 0

Description: This script turns on ths LDOs in the DRP, and loads the ARX registers from the SRM shadow of these registers.

Duration: 337 qbits.

Note:
Used at power on or wake up phase only

3.1.7 REG_OFF

Script Number: 7

Description: This script turns off ths LDOs in the DRP.

Duration: 1 qbit.

Note:
Used to put DRP into Deep Sleep mode.

3.1.8 TEMP_MEAS

Script Number: 3

Description: This script reads the ADC value in DCXO and coverts it to a temperature reading using a calibration lookup table.

Duration:
16 us

Output Parameters

- WORD16 – drp_srm_api->inout.temperature.output
  - Temperature measured given in Celcius

Notes:
1) The TEMP_MEAS script can not be called during Rx window if ROC script is on going.

3.2 DRP C Functions for Phone mode

3.2.1 DBBIF Setting Generation

```
UINT16 drp_generate_dbbif_setting_arfcn(UINT16 pcb_config, UINT16 arfcn)
```

This routine generates the DBBIF setting needed to be passed to the RX_ON script via the API parameter drp_srm_api->inout.rx.rxon_input(9:8)

Input

- UWORD16 pcb_config
  - This is the value of the global macro RF_BAND_SYSTEM_INDEX .
- UINT16 arfcn

- This is the layer-1 definition of the arfcn value for the current channel.

Return

- DBBIF setting value.

3.2.2 DRP Gain Correction

```
SINT16 drp_gain_correction(UINT16 arfcn, UCHAR lna_off, UINT16 agc)
```

This routine generates the gain correction based on the actual DRP AFE and ABF gains set during the RX_ON script execution versus the requested gain values.

Input

- UINT16 arfcn
  - This is the layer-1 definition of the arfcn value for the current channel.
- UCHAR lna_off
  - This parameter = 1, for low LNA gain, and =0 for high LNA gain selection.
- UINT16 agc
  - This parameter is the ABE gain index set via drp_srm_api->inout.rx.rxon_input(5:2)

Return

- Correction based on actual DRP gains set versus the requested DRP gains.

3.3 DRP Power Management

3.3.1 Wakeup / Power ON

This is achieved by running the REG_ON script, this turns on the LDOs within the DRP and puts in a mode where we are ready to run TX_ON, RX_ON or any of the other scripts described in section 3.1.

3.3.2 Deep Sleep

This is achieved by running the REG_OFF script, which turns off the LDO's within the DRP. Note that to come out of deep-sleep it is necessary to run the REG_ON.

4 DRP Calibration

We first describe the calibration procedure step-by-step, followed by the support needed for automation of this procedure using the extended test mode (ETM) shell. We then list and briefly describe the routines provided within the reference SW C-code along with the parameters that each routine expects.

4.1 Calibration Procedure

The calibration procedure can be summarized as:

1. Layer 1 calls the Ref SW function drp_copy_ref_sw_ to_drpsrm( ), while passing the address of the data structure drp_ref_sw[ ]. This will load the calibration script data into the DRP RAM.
2. Layer1 makes a copy the structure drp_sw_data_init from flash into RAM. This RAM copy of the data structure will be called drp_sw_data_calib.
3. Calibration engineer supplies the RF signal input expected by the DCXO calibration routine. Use ETM command to call the drp_dcxo_calib( ) routine.
4. Use ETM command to call the drp_tx_rx_common_calib( ) routine.
5. Set BAND_INDEX = 0.
6. Supply the RF signal expected by LNA center frequency calibration routine.
7. Use ETM command to call drp_lna_cfreq_calib( ).

8. Use ETM command to call drp_iqmc_calib().
9. Use ETM command to call drp_mixer_pole_calib( ).
10. Use ETM command to call drp_scf_poles_calib( ).
11. Use ETM command to call drp_afe_gain_calib1( ).
12. If BAND_INDEX = 3, go to 14, else BAND_INDEX++, and go to 6.
13. Use ETM command to save the calibrated data_structure drp_sw_data_calib in the Flash.

4.2 Extended Test Mode Shell Support for Calibration

The above listed calibration procedure would be executed using the ETM shell. Here we suggest a possible ETM shell command format with input parameters and the actions the layer1 code is expected to perform for each of these ETM commands.

IMPORTANT: ETM implementation will delay the sending back of the ETM acknowledgement until the completion of the corresponding calibration routine.

4.2.1 Calibration ETM Shell Command Format

The following format is suggested for a TM calibration command: drpcal param1 param2
The layer1 actions for different values of param1 are listed next.

- param1 = 3
  - There is no param2 for this case.
  - Layer1 code will call the drp_dcxo_calib routine, and pass the BAND_INDEX (depending on the PCB configuration), and the pointer to the structure drp_sw_data_calib as argument to this function.
- param1 = 4
  - There is no param2 for this case.
  - Layer1 code will call the drp_tx_rx_common_calib routine, and pass the pointer to the structure drp_sw_data_calib as argument to this function.
- param1 = 5
  - param2 needs to be specified. Possible values 0,1,2,3
  - Layer1 and calls the function drp_lna_cfreq_calib( ). The two arguments passed to this function are
    - Pointer to the structure drp_sw_data_calib, and
    - BAND_INDEX = param2
- param1 = 6
  - param2 needs to be specified. Possible values 0,1,2,3
  - Layer1 calls the function drp_iqmc_calib( ). The two arguments passed to this function are
    - Pointer to the structure drp_sw_data_calib, and
    - BAND_INDEX = param2
- param1 = 7
  - param2 needs to be specified. Possible values 0,1,2,3
  - Layer1 calls the function drp_mixer_pole_calib( ). The two arguments passed to this function are
    - Pointer to the structure drp_sw_data_calib, and
    - BAND_INDEX = param2
- param1 = 8
  - param2 needs to be specified. Possible values 0,1,2,3
  - Layer1 calls the function drp_scf_pole_calib( ). The two arguments passed to this function are
    - Pointer to the structure drp_sw_data_calib, and
    - BAND_INDEX = param2
- param1 = 9
  - param2 needs to be specified. Possible values 0,1,2,3
  - Layer1 calls the function drp_afe_gain_calib_1( ). The two arguments passed to this function are
    - Pointer to the structure drp_sw_data_calib, and
    - BAND_INDEX = param2

- param1 = 11
  - There is no param2 for this case.
  - Layer1 copies the structure drp_sw_data_calib that has been modified by the calibration routines, back into flash.

4.3 Calibration and Supporting C-Routines

4.3.1 DCXO Calibration

```
void drp_dcxo_calib(UWORD16 BAND_INDEX, T_DRP_SW_DATA *sw_data_ptr)
```

This routine calibrates the DCXO, coarse frequency array and also the DCXO, DAC current.

Input

- UWORD16 BAND_INDEX
  - This parameter is used to select between the GSM band, and, EGSM band, depending on the PCB configuration. Specifically:
    - EGSM – 0
    - GSM – 1
  - For GSM band, the routine expects an input signal of frequency 869.2 MHz at -70 dBm, and for EGSM band the routine expects 925.2 MHz signal at – 70 dBm.
- T_DRP_SW_DATA *drp_sw_data_calib_ptr is a pointer to the RAM copy of the drp_sw_data_init data structure.

Return

- 0 if no error, -1 if error.

4.3.2 Tx-Rx Common Calibration

```
SINT16 drp_tx_rx_common_calib(T_DRP_SW_DATA *drp_sw_data_calib_ptr)
```

This routine performs the following calibrations:

1. DLO Period Inverse Calibration.
2. DLO acquisition to tracking ratio calibration.
3. PVT Bank calibration.
4. KDCO Inverse calibration.
5. ABE gain calibration using an internally generated signal.

Input

- T_DRP_SW_DATA *drp_sw_data_calib_ptr
  - is a pointer to the RAM copy of the drp_sw_data_init data structure.

Return

- 0 if no error, -1 if error.

4.3.3 LNA Center Frequency Calibration

```
SINT16 drp_lna_cfreq_calib(UWORD16 BAND_INDEX, T_DRP_SW_DATA
            *drp_sw_data_calib_ptr)
```

This routine adjusts the LNA register settings to provide a maximum gain at the center of the particular band of operation.

Inputs

- UWORD16 BAND_INDEX
  - Specifies the cellular band to be calibrated, with the following BAND_INDEX – band mapping:
    - EGSM – 0
    - GSM – 1
    - DCS – 2
    - PCS –3

- T_DRP_SW_DATA *drp_sw_data_calib_ptr
  - is a pointer to the RAM copy of the drp_sw_data_init data structure.
- This routine expects a sinusoidal RF input with a frequency corresponding to the center of cellular band specified by the BAND_INDEX, at a level –45 dBm. We list below the band center frequencies:
  - EGSM – 942.4 MHz
  - GSM – 881.4 MHz
  - DCS – 1842.4 MHz
  - PCS – 1960 MHz

Ouput

The calibration results are stored in the elements of the drp_sw_data structure copy located in the RAM.

Return

- 0 if no error, -1 if error.

4.3.4 IQMC Coefficient Calibration

```
SINT16 drp_iqmc_calib(UWORD16 BAND_INDEX, T_DRP_SW_DATA
            *drp_sw_data_calib_ptr)
```

This routine calibrates the coefficients used by the IQMC block to correct for IQ-mismatch.

Inputs

- UWORD16 BAND_INDEX
  - Specifies the cellular band to be calibrated, with the following BAND_INDEX – band mapping:
    - EGSM – 0
    - GSM – 1
    - DCS – 2
    - PCS –3

- T_DRP_SW_DATA *drp_sw_data_calib_ptr
  - is a pointer to the RAM copy of the drp_sw_data_init data structure.
- This routine expects a sinusoidal RF input with a frequency corresponding to the center of cellular band specified by the BAND_INDEX, at a level – 45 dBm. We list below the band center frequencies:
  - EGSM – 942.4 MHz

- GSM – 881.4 MHz
- DCS – 1842.4 MHz
- PCS – 1960 MHz

Return

- 0 if no error, -1 if error.

4.3.5 Mixer Pole Calibration

```
SINT16 drp_mixer_pole_calib(UWORD16 BAND_INDEX, T_DRP_SW_DATA
            *drp_sw_data_calib_ptr)
```

This routine calibrates the 400 KHz pole of the switched capacitor filter (SCF).

Inputs

- UWORD16 BAND_INDEX
  - Specifies the cellular band to be calibrated, with the following BAND_INDEX – band mapping:
    - EGSM – 0
    - GSM – 1
    - DCS – 2
    - PCS –3
- T_DRP_SW_DATA *drp_sw_data_calib_ptr
  - is a pointer to the RAM copy of the drp_sw_data_init data structure.
- This routine expects a sinusoidal RF input with a frequency corresponding to the center of cellular band specified by the BAND_INDEX, at a level – 45 dBm. We list below the band center frequencies:
  - EGSM – 942.4 MHz
  - GSM – 881.4 MHz
  - DCS – 1842.4 MHz
  - PCS – 1960 MHz

Return

- 0 if no error, -1 if error.

4.3.6 SCF Pole Calibration

```
SINT16 drp_scf_pole_calib(UWORD16 BAND_INDEX, T_DRP_SW_DATA
            *drp_sw_data_calib_ptr)
```

This routine calibrates the 270 KHz, and 400 KHz poles of the switched capacitor filter (SCF).

Inputs

- UWORD16 BAND_INDEX
  - Specifies the cellular band to be calibrated, with the following BAND_INDEX – band mapping:
    - EGSM – 0
    - GSM – 1
    - DCS – 2
    - PCS –3
- T_DRP_SW_DATA *drp_sw_data_calib_ptr

  - is a pointer to the RAM copy of the drp_sw_data_init data structure.
- This routine expects a sinusoidal RF input with a frequency corresponding to the center of cellular band specified by the BAND_INDEX, at a level – 45 dBm. We list below the band center frequencies:
  - EGSM – 942.4 MHz
  - GSM – 881.4 MHz
  - DCS – 1842.4 MHz
  - PCS – 1960 MHz

Return

- 0 if no error, -1 if error.

4.3.7 AFE Gain Calibration - part1

```
SINT16 drp_afe_gain_calib1(UWORD16 BAND_INDEX, T_DRP_SW_DATA
            *drp_sw_data_calib_ptr)
```

This routine calibrates a part of the AFE gain steps. (Higher gain steps)

Inputs

- T_DRP_SW_DATA *drp_sw_data_calib_ptr
  - is a pointer to the RAM copy of the drp_sw_data_init data structure.
- This routine expects a sinusoidal RF input with a frequency corresponding to the center of cellular band specified by the BAND_INDEX, at a level – 45 dBm. We list below the band center frequencies:
  - EGSM – 942.4 MHz
  - GSM – 881.4 MHz
  - DCS – 1842.4 MHz
  - PCS – 1960 MHz

Return

- 0 if no error, -1 if error.

4.3.8 Copy Script Data from Flash to DRP

```
SINT16 drp_copy_ref_sw_to_drpsrm(UWORD8 *ref_sw_ptr)
```

This routine copies the script data contained in the structures drp_ref_sw[ ] and drp_ref_sw_calib[ ], from Flash into the DRP RAM. The DRP RAM address information is contained within the data structures.

- This function should be executed every time the phone is powered up with flash address of the drp_sw_data[ ] as the input argument.
- At calibration time, the input argument should be the flash address of the drp_ref_sw_calib[ ] structure.
- The constant DRP_REG_BASE_ADD needs to be correctly specified in the drp_extern_dependencies.h file.

Inputs

UWORD8 *ref_sw_ptr - pointer to the script data structure in flash.

Return

- 0 if no error, -1 if error.

4.3.9 Copy Calibration Data from Flash to DRP SRM

```
SINT16 drp_copy_sw_data_to_drpsrm(T_DRP_SW_DATA *ptr_flash, T_DRP_SRM_DATA
            *ptr_srm)
```

This routine copies the calibration data from the structure drp_sw_data_calib_saved into the DRP SRM. The data is written in the DRP SRM into a structure of the type T_DRP_SRM_DATA.

- This function is called at power on just after the function drp_copy_ref_sw_to_drpsrm( ) completes execution.
- The pointer argument *ptr_flash, should point to the flash address of the drp_sw_data_calib_saved structure.
- The pointer argument *ptr_srm should be set to DRP_SRM_DATA_ADDR defined in the drp_extern_dependencies.h file.

Inputs

T_DRP_SW_DATA *ptr_flash - pointer to the saved calibration data structure in flash
T_DRP_SRM_DATA *ptr_srm - pointer to the calibration data structure in DRP SRM

Return

- 0 if no error, -1 if error.

A. DRP_SRM_API Structure

```
/*****************************************************/
/*              API STRUCTURE                */
/*****************************************************/

/*-----------------------------**
** ROC Script Input Parameters **
**-----------------------------*/
typedef struct
{
  UWORD16 window_start;
  UWORD16 window_stop;
  WORD16  lut_start_phase;
}
T_DRP_ROC_IN;

/*-----------------------------**
** ROC Script Output Parameters **
**-----------------------------*/
typedef struct
{
  UWORD32 accum_I[4];
  UWORD32 accum_Q[4];
}
T_DRP_ROC_OUT;

/*-----------------------------**
** ROC Script InOut Parameters **
**-----------------------------*/
typedef struct
{
  T_DRP_ROC_IN input;
  T_DRP_ROC_OUT output;
```

```
}
T_DRP_ROC_INOUT;

/*------------------------------**
** AGC       Output Parameters **
**------------------------------*/
typedef struct
{
  UWORD16 compensated_gain;       // ABF + AFE gain in DRP after compensation
                                  // 1024*log2(Pout/Pin)
  UWORD16 meas_abe_vpp_i;         // V-pp for I path
  UWORD16 meas_abe_vpp_q;         // V-pp for Q path
}
T_DRP_AGC_OUT;

/*------------------------------**
** Temperature InOut Parameters **
**------------------------------*/
typedef struct
{
  WORD16 output;
}
T_DRP_TEMP_OUT;

/*----------------------**
** RX InOut Parameters **
**----------------------*/
typedef struct
{
  UWORD16 rxon_input;          // b0 - IF_FREQ index
                               // b1 - Gain compensation enable
                               // b5:b2 - ABE gain index
                               // b6 - AFE gain index
                               // b7 - SCF Pole index
                               // b9:8 - dbb if setting
  T_DRP_AGC_OUT agc;           //agc control structure
#if LOCOSTO_VALID
  T_DRP_ROC_INOUT roc;
#endif
}
T_DRP_RX_INOUT;


/*------------------------------**
** AFC script Input Parameters **
**------------------------------*/
typedef struct
{
  UWORD16 mem_xtal;
}
T_DRP_AFC_IN;

/*------------------------------**
** AFC script Output Parameters **
**------------------------------*/
typedef struct
{
  UWORD16 dcxo_amp;
}
```

```
T_DRP_AFC_OUT;

/*----------------------------**
** AFC script InOut Parameters **
**----------------------------*/
typedef struct
{
  T_DRP_AFC_IN input;
  T_DRP_AFC_OUT output;
}
T_DRP_AFC_INOUT;

/*---------------------**
** DRP InOut Parameters **
**---------------------*/
typedef struct
{
  T_DRP_RX_INOUT rx;
  T_DRP_AFC_INOUT afc;
  T_DRP_TEMP_OUT temperature;
}
T_DRP_INOUT;

/*--------------------------**
** DRP General control words **
**--------------------------*/

typedef struct
{
  UWORD16 retiming;     // enable-disable control of retiming
  UWORD16 calib_ctl;    // b0 - FCW bypass for Tx
                        // b1 - FCW bypass for RX
                        // b2 - Bypass ABE gain comp call
                        // b3 - Bypass AFE gain comp call
                        // b4 - mixer disable during ABE gain measure
                        // b5 - ibias setting bypass
                        // b6 - flyback setting bypass
                        // b7 - use CVKD2 for TX_ON DCOIF
}
T_DRP_CTL;

/*-----------------------------------**
** DRP SRM API Data Structure **
**-----------------------------------*/
typedef struct
{
  T_DRP_INOUT inout;       // input-output DRP parameters - phone mode
  T_DRP_CTL control;       // contains the DRP operating mode control words
} T_DRP_SRM_API;
```

B. Script and Initialization Data Array Format

The script data and the DRP register initialization data is stored into data array drp_ref_sw[ ], in the following format:

```
const unsigned char drp_ref_sw[] = {

/* Tag  */ 0x00, 0x00, 0x00, 0x00,
/* Vers */ 0x01, 0x00, 0x00, 0x00,
```

```
/* size */ 0x64, 0x00, 0x00, 0x00,
/* addr */ 0x80, 0x20, 0x00, 0x00,

0x78,0x56,0x00,0xa1,0x34,0x12,0x00,0xa2,0x01,0x00,0x08,0x9f,0x02,0x00,0x10,0x9f,
0x03,0x00,0x18,0x9f,0x04,0x00,0x20,0x9f,0x05,0x00,0x28,0x9f,0x06,0x00,0x30,0x9f,
0x07,0x00,0x38,0x9f,0xf8,0xff,0x40,0x9f,0xf7,0xff,0x48,0x9f,0xf6,0xff,0x50,0x9f,
0xf5,0xff,0x58,0x9f,0xf4,0xff,0x60,0x9f,0xf3,0xff,0x68,0x9f,0xf2,0xff,0x70,0x9f,
0xf1,0xff,0x78,0x9f,0x00,0x00,0x00,0x00,0x00,0x03,0x08,0x81,0x00,0x49,0x10,0x81,
0x00,0x51,0x18,0x82,0x00,0x00,0x20,0xa2,0xff,0xff,0x20,0xa1,0x00,0x08,0x28,0x83,
0x00,0x08,0x30,0x84,0x00,0x4b,0x40,0xad,0x00,0x4f,0x48,0x8e,0x00,0x09,0x60,0x85,
0xbc,0x8a,0x68,0xa2,0xf9,0xde,0x68,0xa1,0x00,0x51,0x73,0x86,0x00,0x51,0x7b,0x87,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x9f,0x64,0x00,0x58,0x9f,0x00,0xcb,0x39,0x81,
0x00,0xd6,0x51,0x89,0xfd,0xbf,0x02,0xaa,0x00,0x00,0x00,0x00,0xf0,0x2f,0x70,0xa1,
0x00,0x00,0x70,0xa2,0x00,0x80,0x03,0x9b,0x00,0x80,0x43,0x93,0x02,0x80,0x4b,0x91,
0x03,0x80,0x53,0x90,0x04,0x80,0x03,0x9a,0x04,0x80,0x5b,0x93,0x08,0x80,0x03,0x99,
0x08,0x80,0x23,0x93,0x00,0x00,0x00,0x00,

/* size */ 0x08, 0x00, 0x00, 0x00,
/* addr */ 0x0C, 0x05, 0x00, 0x00,

0x80,0x20,0x00,0x00,0xc8,0x20,0x00,0x00,0x04,0x21,0x00,0x00,0x1c,0x21,0x00,0x00,

/* Final record */
/* size */ 0x00, 0x00, 0x00, 0x00,
};
```

Following the tag, first information shall indicate the version of the reference software. Although, 32-bits have been reserved for SW version number, the current format needs only 16bits. The number represented by the bits b(15:8) denotes the major release number. This number ranges from 128-255. The bits b(7:0) denote the intermediate release number, this number ranges from 0-255.

Next, several data blocks are defined. A block defines a memory range initialization with the data contained in this block. To specify a block, 3 parameters shall be indicated:

- **The block size (32 bits) measured in WORD16**
- **The OCP start address of the block (32 bits)**
- **The data contained in the block**

The number of blocks to define is not fixed. The final record defines the end of the drp memory initialization. In this final record, the block size shall be set to 0x00000000. The value of the address field is not used and therefore not mentioned.

C. Macro Definitions

1. WRITE_BYTE_LOW(a,b)
    a) a is the 16 bit unsigned integer to be written into.
    b) b is in general a 16-bit value or variable.
    c) The macro should copy the least significant byte from b into a.
2. WRITE_BYTE_HIGH(a,b)
    a) a is the 16 bit unsigned integer to be written into.
    b) b is in general a 16-bit value or variable.
    c) The macro should copy the most significant byte from b into a.
3. WRITE_WORD(a,b)
    a) a is the 16 bit unsigned integer to be written into.
    b) b is in general 16-bit value or variable.
    c) The macro should copy the b into a.
4. WRITE_WORD_AT_PTR(a,b)
    a) a is a pointer to a 16 bit unsigned integer to be written into.
    b) b is in general 16-bit value or variable.
    c) The macro should copy the b into location pointed to by a.
5. READ_WORD(a,b)

a) a is the 16 bit unsigned integer to be read.
b) b is a 16-bit unsigned variable to be read into.
c) The macro should copy the contents of a into b.

6. WAIT_US(a)
   a) a is the wait in micro seconds.

D. RX_ON Script Details

Here we describe step-by-step functionality of the RX_ON script:

1. Read the temperature,
2. Estimate the ABE and AFE gains using the temperature coefficients
3. Find the AFE and ABE gains closet to the gains requested by the AGC gain step.
4. Apply the AFE gain, disable the AFE.
5. ***Apply an ABE gain of 11 dB.***
6. Set the RF_FREQ register.
7. Disable the RX Front End Module (FEM).
8. Enable DCO (***mixer clocks enabled***), set DBB_IF
9. Load calibrated values of KDCO, PVT, IQMC, DLO_AT, SCF poles
10. Enable mixer speedup.
11. Enable FBDAC, ADC
12. Compute fcw, resampler_fcw
13. DLO_common_1 call: Reset DLO, Step through PVT
14. Set the MCU, DSP divider, enable TDC_SIGMA_DELTA on integer channels.
15. Wait 10 us for PVT step through
16. DLO_common_2 call: step through acquisition (10 us wait), step into tracking, start period inv accumulation.
17. Disable mixer speedup
18. rx_gain_measure_part1: Enable DCU, PCU, apply positive input to FCU_DAC
19. dlo_common_3: wait 20 us, stop period inverse accumulation.
20. Compute the period inverse.
21. Reset DC estimation registers.
22. Gain_measure_part2: read the DC offset (MAX+MIN)/2, apply negative input to FCU_DAC.
23. Dlo_common_4: Start IIR filters, gear shifting, enable type-II loop (wait of 15 us)
24. Wait for 5 us, reset the DC estimation registers.
25. Gain_measure_part3: read the DC offset, compute the Vpp swing resulting from positive and negative FCU_DAC input.
26. Enable AFE.
27. Wait for ***10-15*** us.
28. Reset DC estimation registers.
29. Wait for ***30-35*** us.
30. Rx_gain_measure_part4: DC offset estimation and FCU compensation (***no SCF loss factor, mixer clocks were enabled all the while***)
31. ***Set the correct ABE gain.***
32. Prefilter in operational mode, disable ARX bus and shadow.
33. Script Done (enable the FEM).
34. ***Remaining gain compensation***

Notes:

a. We estimate and compensate DC offset with a modest ABE gain of 11 dB.
b. It has been experimentally verified that with 11 dB ABE gain the DC offset introduced with AFE in max gain setting is comfortably below the ADC saturation level.
c. It is now expected that the DC offset compensation will ensure that the ADC will always remain well outside the saturation limits and any remaining DC offset will be handled by the residual DC offset compensation.

The remaining gain compensation mentioned in step 34 is supposed to correct for the difference in measured ABE gain versus the estimated ABE gain. It is not very clear at present if the ABE gain measured via FB-DAC is a more accurate estimate of ABE gain than the ABE gain estimated from calibrated gains and temperature coefficient. Need comprehensive characterization results here.

E. TX_ON Script Details

Here we describe step-by-step functionality of the TX_ON script:

1. Load calibrated values of DLO_PER_INV, DLO_TUNE_A_T, KDCO, PVT
2. Enable DCO, set DBB_IF
3. Enable SAM block clock.
4. Compute fcw
5. DLO_common_1 call: Reset DLO, Step through PVT
6. Set the MCU, DSP divider, enable TDC_SIGMA_DELTA on integer channels.
7. Wait 10 us for PVT step through
8. DLO_common_2 call: step through acquisition (10 us wait), step into tracking, start period inv accumulation.
9. dlo_common_3: wait 20 us, stop period inverse accumulation.
10. Compute the period inverse.
11. Dlo_common_4: Start IIR filters, gear shifting, enable type-II loop (wait of 15 us)
12. RUN the KDCO_INV calibration algorithm, includes running modulation with 1100 bit-pattern, and includes over 85 us of wait.
13. Allow TX modulation.
14. Enable the SAM block.

References

[1] LNA Swap SW change request.

LNA_Swap_SW_implementation_proposal