TRIMTRAC CALIBRATION PROCEDURES

1 INTRODUCTION 1

1.1 Overview 1

1.2 Measurable Parameters 1

1.3 Calibration and Development Signals 2

1.4 Equipment Required 2

1.5 Terminology 2

2 PRE-CALIBRATION 4

3 AUTOMATIC FREQUENCY CORRECTION 5

3.1 Purpose 5

3.2 Theory 5

3.3 Example 6

4 AUTOMATIC GAIN CONTROL 8

4.1 Purpose 8

4.2 Theory 8

4.3 Example 8

5 RSSI CORRECTION 10

5.1 Purpose 10

5.2 Theory 10

5.3 Example 11

6 TRANSMIT POWER SCALING FACTORS 12

6.1 Purpose 12

6.2 Theory 12

6.3 Example 13

7 TRANSMIT POWER FREQUENCY COMPENSATION 14

7.1 Purpose 14

7.2 Theory 14

8 POST CALIBRATION 17

8.1 Leaving Test Mode 17

8.2 Storing Calibration Data 17

1 INTRODUCTION

1.1 Overview

This document describes a suggested calibration procedure for all of the measurable parameters of the target platform in production.

Individual sections cover each procedure. Information on how the resultant calibration data may be incorporated into the target platform is included in the relevant configuration manual for the platform.

1.2 Measurable Parameters

The aim of the calibration procedure is to measure the following parameters:

Type	**Name**	**See Section**	**Size**
SignedInt32	`slopePPB`	3	
Int16	`nominalAfcValue`	3	
Int16	`gainControlCodeTable`	4	`RADIO_GAIN_TABLE_SIZE`
SignedInt16	`accurateRadioGainTable`	4	`RADIO_GAIN_TABLE_SIZE`
SignedInt8	`rssiCorrectionTable`	5	`BAND_SIZE` (number of channels in the band)
Int16	`rampFactorTable`	6	`NUM_POWER_CONTROL_LEVELS`
Int16	`txFreqCompensation`	7	`BAND_SIZE`/4

Note: the `gainControlCodeTable`, `accurateRadioGainTable`, `rssiCorrectionTable`, `rampFactorTable` and `txFreqCompensation` contain "size" values for each of the supported bands.

1.3 Calibration and Development Signals

The procedures describe use the following Calibration and Development (`CalDev`) signals:

- CalDevGsmReq & CalDevGsmCnf
- CalDevGsmFinishReq & CalDevGsmFinishCnf
- CalDevGsmRssiReq & CalDevGsmRssiCnf
- CalDevGsmFreqOffsetMeasReq & CalDevGsmFreqOffsetMeasCnf
- CalDevGsmSetPowerRampReq & CalDevGsmSetPowerRampCnf
- CalDevGsmDcOffsetReq & CalDevGsmDcOffsetCnf
- CalDevGsmGainProgramReq & CalDevGsmGainProgramCnf
- CalDevGsmBurstReq & CalDevGsmBurstCnf
- CalDevGsmSetAfcDacReq & CalDevGsmSetAfcDacCnf
- CalDevGsmSetBurstDataReq & CalDevGsmSetBurstDataCnf
- CalDevGsmRxControlReq & CalDevGsmRxControlCnf
- CalDevGsmRampScaleReq & CalDevGsmRampScaleCnf
- CalDevGsmSetBandModeReq

1.4 Equipment Required

The suggested equipment required is a PC with a GPIB card to control a Signal Generator and a Power Meter, or appropriate GSM tester. The target will be controlled via the serial interface.

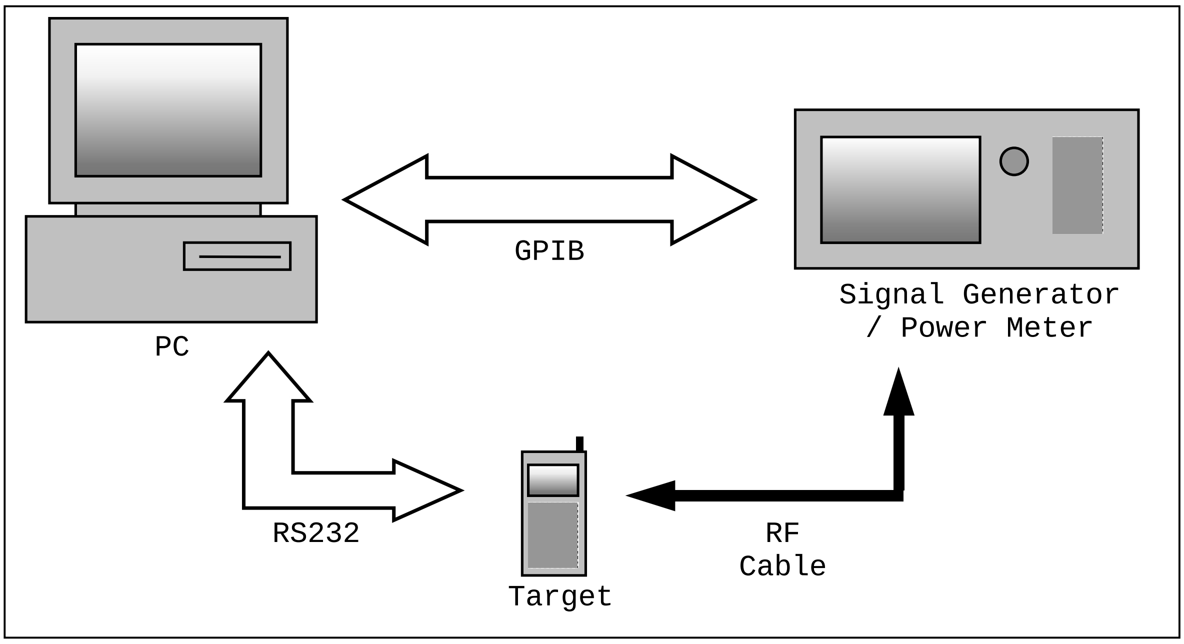

Figure 1 Typical calibration station

1.5 Terminology

The following abbreviations are used throughout this document:

AFC	Automatic Frequency Correction
AGC	Automatic Gain Control
ARFCN	Absolute Radio Frequency Channel Number
BBC	Base Band Codec
DAC	Digital to Analogue Converter
GKI	Generic Kernel Interface
GPIB	General Purpose Interface Bus
NVRAM	Non Volatile Random Access Memory
RSSI	Received Signal Strength Indication

2 PRE-CALIBRATION

Calibration may only be performed whilst the Layer 1 software running on the target platform is in the `TEST` state, which may only be reached from the `NULL` state.

`NULL` state is a state where the application layer, protocol stack and layer 1 have been shutdown. How this state is reached is dependent on the software build and hardware implementation, but it should be possible to enter `NULL` state by sending a signal to the target from the PC using Genie.

If a full Application Layer build is running, an `ApexEmEnterNullStateSig` needs to be sent to the Application Foreground Layer task. This will shutdown the Application Layer and Protocol Stack and causes an `MphDeactivateReq` signal to be sent, shutting down Layer 1.

If a PS application build is being used then an `MphDeactivateReq` signal may be sent directly to the Layer 1 Background task from Genie.

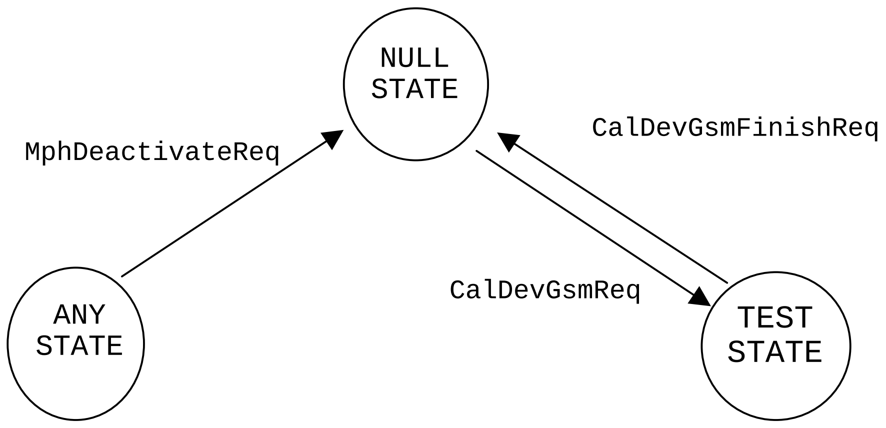

Figure 2 Excerpt of Layer 1 State Diagram

Prior to starting calibration, assuming that the platform is connected as illustrated in Figure 1 and in the `NULL` state, there are two steps to perform:

- notify the target to process received signals
- place the target into the `TEST` state

3 AUTOMATIC FREQUENCY CORRECTION

3.1 Purpose

Calibration of the AFC is carried out to ensure that the target's clock frequency can correctly synchronise to that of the network.

3.2 Theory

Because the relationship between the DAC and the clock frequency is an essentially linear one, the AFC calibration of an individual target may be characterised by two parameters:

- a nominal value
- the gradient of the VCXO

The nominal value is the DAC value required to give a zero frequency offset between the target and the signal generator tuned to the centre frequency of the mid-band channel. The gradient allows for the calculation of the correct DAC value for all other frequency offsets.

The calibration procedure outlined assumes that a theoretical nominal value has been calculated for the radio used.

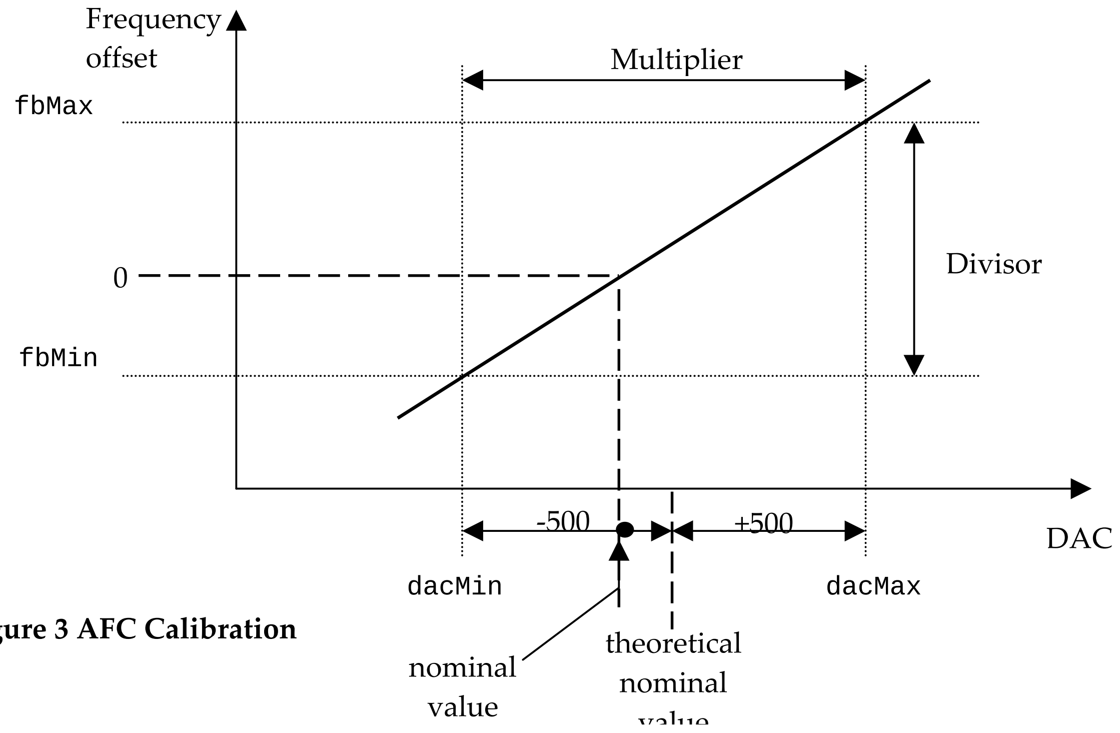

Figure 3 AFC Calibration

3.3 Example

Below is a description of how to calculate the AFC slope (`slopePPB`) and nominal value (`nominalAfcValue`) using `CalDevGsmFreqOffsetMeasReq` and `CalDevGsmFreqOffsetMeasCnf` signals. These signals measure frequency bursts to detect the frequency offset.

```
frequency = MID_CHANNEL_FREQUENCY + 0.0677083333 MHz /* see note 1 */
amplitude = -62dBm + CABLE_LOSS
SET Signal Generator TO frequency & amplitude WITH MODULATION_OFF
SEND CalDevGsmFreqOffsetMeasReq (MID_CHANNEL_ARFCN,
                                 dacMax)
WAIT FOR CalDevGsmFreqOffsetMeasCnf (freqBurstFound, frequencyOffset)

IF freqBurstFound = TRUE THEN
   fbMax = frequencyOffset
   SEND CalDevGsmFreqOffsetMeasReq (MID_CHANNEL_ARFCN,
                                    dacMin)
   WAIT FOR CalDevGsmFreqOffsetMeasCnf (freqBurstFound, frequencyOffset)

   IF freqBurstFound = TRUE THEN
      fbMin = frequencyOffset
      multiplier = dacMax - dacMin
      divisor = (-1) * (fbMax - fbMin) /* see note 2 */
      nominalAfcValue = dacMin + ((fbMin * multiplier) / divisor)
      slopePPB = (divisor / multiplier) * (10 / BAND)
                     * ((AFC_DAC_MAX_STEPS * 1000) / AFC_DAC_SPAN) /* see note 3 */
   END IF
END IF

/*
** Note 1 : this simulates a frequency burst of all 1's.
** Note 2 : the target reports how far away the network frequency is from where
**          it expects it to be, we want to know how far away the target's frequency
**          is from where the network is - we could multiply either:
**          each offset by -1 or the sum by -1.
** Note 3 : the AFC slope is calculated in units of PPB/Volt to remove the need for AFC
**          data to be stored for each band.
*/
```

Where:

`MID_CHANNEL_ARFCN` is the middle channel of the GSM/DCS/PCS band.

`MID_CHANNEL_FREQUENCY` matches the selected mid channel ARFCN.

`dacMax` is normally set to the theoretical nominal value +500.

`dacMin` is normally set to the theoretical nominal value -500.

A GSM tester may be used instead of a signal generator. In both these cases `freqBurstFound` is true when a frequency burst has been detected, but `calDevGsmFreqOffsetMeasCnf.status` will hold a value of `FO_CW_DETECTED` if a signal generator has been used or `FO_GSM_FB_DETECTED` if a tester was used and a frequency burst was detected. If a frequency burst is not detected the reported value will be `FO_NOT_MEASURED`.

`frequencyOffset` is the measured offset relative to the target.

`BAND` is a value related to the frequency of the band in which the measurement has been performed: 9 for GSM, 18 for DCS and 19 for PCS.

`AFC_DAC_MAX_STEPS` is the maximum AFC DAC input value.

`AFC_DAC_SPAN` is the range of DAC output voltages in mV

4 AUTOMATIC GAIN CONTROL

4.1 Purpose

The range of levels seen at the antenna is assumed to be between -10dBm and -110dBm (this slightly exceeds the range defined in GSM 05.05). The range acceptable at the input to the BBC is considerably less.

Calibration of the AGC is required to ensure that received signals remain within the useable range of the BBC.

4.2 Theory

To keep the software controller (and the calibration scheme) manageable the input dynamic range is covered by *n* discrete gain steps, nominally *m* dB per step, where *n* and *m* are the values `RADIO_GAIN_TABLE_SIZE` and `STEP_SIZE` respectively, as described in Section **Error! Reference source not found.**. Associated with each step are two parameters:

- a `gainControlCodeTable` value
- an `accurateRadioGainTable` value

The `gainControlCodeTable` value is the gain word programmed to the radio to apply the correct amount of gain to the signal to present the optimum level to the BBC. This optimum level is the `SET_POINT` as described in Section **Error! Reference source not found.**.

The `accurateRadioGainTable` value (measured in $^{1}/_{16}$ths of a dB) is used to calculate the actual received level based upon the level seen at the input to the BBC. The theoretical value for each `accurateRadioGainTable` entry is given by the formula:

```
accurateRadioGainTable[step] = ((m x step) + (SET_POINT - -10)) x 16
```

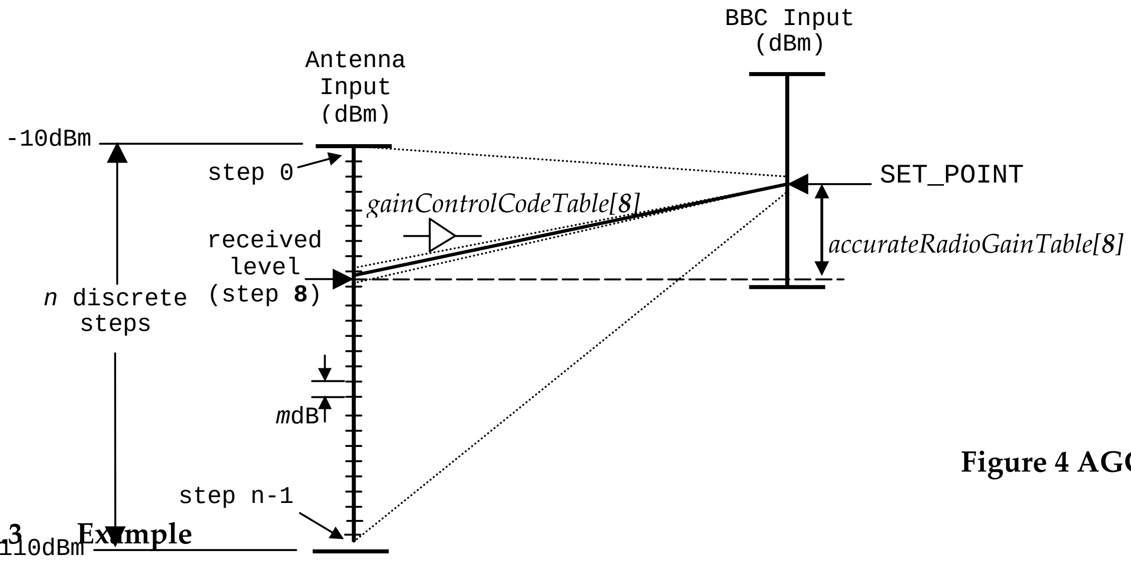

Figure 4 AGC Calibration

4.3 Example

Calibration of the AGC should be carried out at the mid-band frequency for each supported band.

Sending the signal `CalDevGsmGainProgramReq`, set to the `GAIN_CODE_WRITE` mode, with the required step (gain number) and gain word will set the gain to be applied to the incoming signal.

Sending the `CalDevGsmRssiReq`, set to `DM_RSSI_NORMAL` mode, with the required ARFCN and gain number, will measure the RSSI level seen at the BBC. The results are returned in the associated `CalDevGsmRssiCnf`.

Below is a description of how the `gainControlCodeTable` and `accurateRadioGainTable` are derived, using the `CalDevGsmGainProgramReq`, `CalDevGsmGainProgramCnf`, `CalDevGsmRssiReq` and `CalDevGsmRssiCnf` signals. It is assumed that the `STEP_SIZE` is 4dBm.

```
bbcSetPoint  = SET_POINT dBm
stepSize = 4dBm
initialLevel = -10dBm
frequency = MID_CHANNEL_FREQUENCY
FOR each supported band
   FOR step = 0 TO (RADIO_GAIN_TABLE_SIZE - 1)
      amplitude = (initialLevel - (step * stepSize))
      sgAmplitude = amplitude + CABLE_LOSS
      SET Signal Generator to frequency and sgAmplitude WITH GSM_MODULATION
      gainWord = validGainControlCodeValue[step]
      DO SEND CalDevGsmGainProgramReq ( step, gainWord )
         WAIT FOR CalDevGsmGainProgramCnf
         SEND CalDevGsmRssiReq ( MID_CHANNEL_ARFCN, step )
         WAIT FOR CalDevGsmRssiCnf ( rssiLevel )

         IF rssiLevel != (bbcSetPoint +/- acceptableRssiError) THEN
            IF rssiLevel > bbcSetPoint + acceptableRssiError THEN
               Increase gainWord
            ELSE
               Decrease gainWord
            END IF
         END IF
      WHILE rssiLevel != (bbcSetPoint +/- acceptableRssiError)
      gainControlCodeTable [step] = gainWord
      accurateRadioGainTable [step] = 16*((rssiLevel/16) - amplitude)
   NEXT step
NEXT band
```

Where:

`SET_POINT` is a BBC specific constant (see section **Error! Reference source not found.**).

`CalDevGsmGainProgramReq(step,gainWord)` should be sent with `mode` set to `GAIN_CODE_WRITE`.

In order for the RSSI value returned in the `CalDevGsmRssiCnf` to be the value measured at the BBC input, the value in the `accurateRadioGainTable`[] in the target used during the RSSI measurement must be zero. This can be done using the `CalDevGsmGainProgramReq` signal with mode set to `ACCURATE_GAIN_WRITE` and `gainWord` set to zero to temporarily overwrite the `accurateRadioGainTable`[].

`validGainControlCodeValue[RADIO_GAIN_TABLE_SIZE]` contains the best guess gain word for a given gain step in order to shorten the time to achieve the `bbcSetPoint`.

`acceptableRssiError` is used to determine how accurate a `bbcSetPoint` is to be achieved. To improve the resolution, both of these signed values are in sixteenths of a dB.

5 RSSI CORRECTION

5.1 Purpose

RSSI correction is calibrated to ensure that received levels are reported accurately across the entire frequency band.

5.2 Theory

RSSI correction calibration is performed across the supported band at one mid-range level (i.e. -62dBm) and is measured in $^{1}/_{16}$ths of a dB relative to that level.

The RSSI correction value is applied to the RSSI level calculated using the `accurateRadioGainTable`[] to allow for accurate level reporting at all levels and at all frequencies.

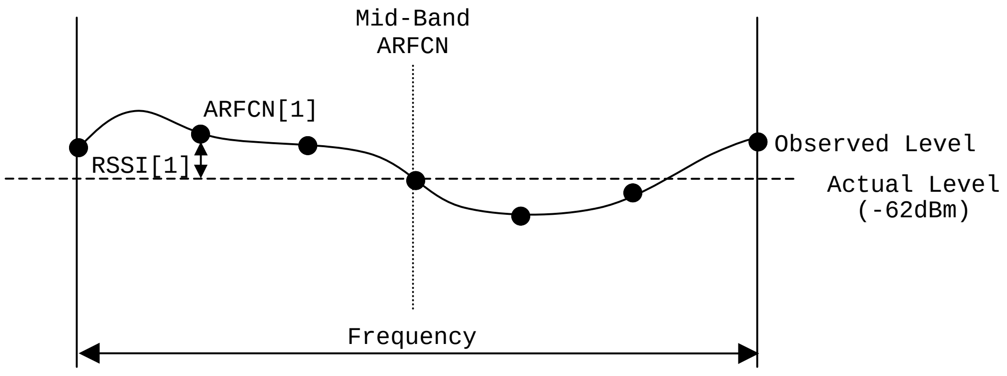

Figure 5 RSSI Correction

5.3 Example

```
amplitude = -62dBm
sgAmplitude = amplitude + CABLE_LOSS
FOR each supported band
   FOR loopCount = BOTTOM_CHANNEL TO TOP_CHANNEL
      frequency  =  frequencyMeasPoint( loopCount )
      SET Signal Generator TO frequency and sgAmplitude WITH GSM_MODULATION
      SEND CalDevGsmRssiReq ( loopCount, MID_GAIN_NUMBER )
      WAIT FOR CalDevGsmRssiCnf ( rssiLevel )
      rssiCorrectionTable[ loopCount ] = -1 * ((16*amplitude - rssiLevel)
   NEXT loopCount
NEXT band
```

Where:

`MID_GAIN_NUMBER` is `RADIO_GAIN_TABLE_SIZE` / 2 (i.e. 13 for Bright radios)

`BOTTOM_CHANNEL` is the lowest ARFCN in the band (i.e. for GSM 900 ARFCN 1 is used; for DCS 1800 ARFCN 512 is used; for PCS 1900 ARFCN 512 is used).

`TOP_CHANNEL` is the highest ARFCN in the band (i.e. for GSM 900 ARFCN 124 is used; for DCS 1800 ARFCN 885 is used; for PCS 1900 ARFCN 810 is used).

`frequencyMeasPoint(loopcount)` is the frequency of the selected ARFCN.

`rssiLevel` is the measured level.

Note: Using one entry per ARFCN allows the correction value to be determined quickly at the expense of memory. If storage of calibration data is an issue then it may be preferable to measure the correction values at a number of points across the band and interpolate to find the correction for a specific ARFCN.

6 TRANSMIT POWER SCALING FACTORS

6.1 Purpose

Scaling factors are calibrated to ensure that the transmitted output power of the target meets the levels specified in GSM 05.05 for each supported power control level.

Note: This assumes that the ramp profiles for each power level have been defined.

6.2 Theory

The target's transmitted output power may be modified in two ways:

- modifying the ramp profile
- modifying the scaling factor

The amount of memory required by the ramp profiles precludes their inclusion in NVRAM (they are stored in ROM) so modification of the ramp shapes would require an individual software build for each target. Therefore, modification of the scaling factor associated with each ramp shape is used.

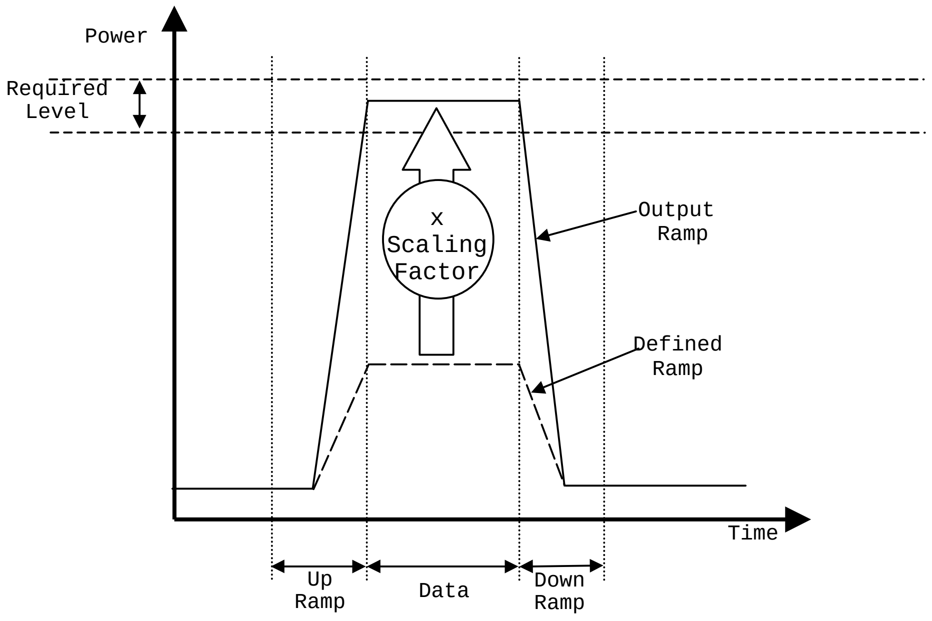

Figure 6 Scaling Factors

6.3 Example

```
FOR each supported band
   FOR powerLevel = FIRST_POWER_LEVEL TO LAST_POWER_LEVEL
      scalingFactor = DEFAULT_SCALING_FACTOR
      powerLevelAchieved = FALSE
      SEND CalDevGsmBurstReq ( MID_CHANNEL_ARFCN,
                               transmit NORMAL_BURST,
                               powerControlLevel(powerLevel) | 0x0100)
      WAIT FOR  CalDevGsmBurstCnf
      REPEAT
         SEND CalDevGsmRampScaleReq ( scalingFactor )
         WAIT FOR CalDevGsmRampScaleCnf
         MEASURE measuredPowerLevel USING Power Meter
         measuredPowerLevel = measuredPowerLevel + CABLE_LOSS
         IF measuredPowerLevel IS WITHIN powerLevel +/- (tolerance) dBm THEN
            powerLevelAchieved = TRUE
         ELSE
            IF  measuredPowerLevel < powerLevel THEN
               scalingFactor = scalingFactor + POWER_INCREMENT
            ELSE
               scalingFactor = scalingFactor - POWER_INCREMENT
            ENDIF
         ENDIF
      UNTIL powerLevelAchieved = TRUE
      rampFactorTable[powerControlLevel(powerLevel)] = scalingFactor
   NEXT powerLevel
NEXT band
```

Where:

`FIRST_POWER_LEVEL` is 5dBm for GSM900 or 0dBm for DCS1800/PCS1900

`LAST_POWER_LEVEL` is 33dBm for GSM 900 or 30dBm for DCS 1800/PCS1900

`POWER_INCREMENT` is an integer value, which would give a 1dBm change

`MID_CHANNEL_ARFCN` is the middle channel of the GSM/DCS/PCS band.

`scalingFactor` is the scaling factor to be applied to the stored power ramp

`powerLevel` is the output power level in dBm that is to be achieved. It is incremented by 2dBm every time round the loop.

`powerControlLevel(powerLevel)` gives the associated power control level (0-31) for each `powerLevel` in dBm, as given in 05.05. When the power control level is passed in `CalDevGsmBurstReq.txPowerLevel` then bit 8 of this field should be set to 1 (e.g. for power control level 5 send 133). This is so the dynamic scale factor sent in the `CalDevGsmRampScaleReq` is used when calculating the power ramp, otherwise the default scale factor compiled into the target code will be used.

7 TRANSMIT POWER FREQUENCY COMPENSATION

7.1 Purpose

Frequency compensation is calibrated to ensure that the transmitted output power of the target meets the levels specified in GSM 05.05 across the supported frequency band.

7.2 Theory

Calibration of frequency compensation is typically performed at several points across all supported bands at the highest supported power level where the specified tolerances are tightest.

The frequency compensation factor modifies the scaling factor in the following manner:

```
scalingFactor = (scalingFactor x frequencyCompensation) >> 15
```

Therefore: A frequency compensation of 0x8000 is unity.
A frequency compensation < 0x8000 lowers the output power.
A frequency compensation > 0x8000 raises the output power.

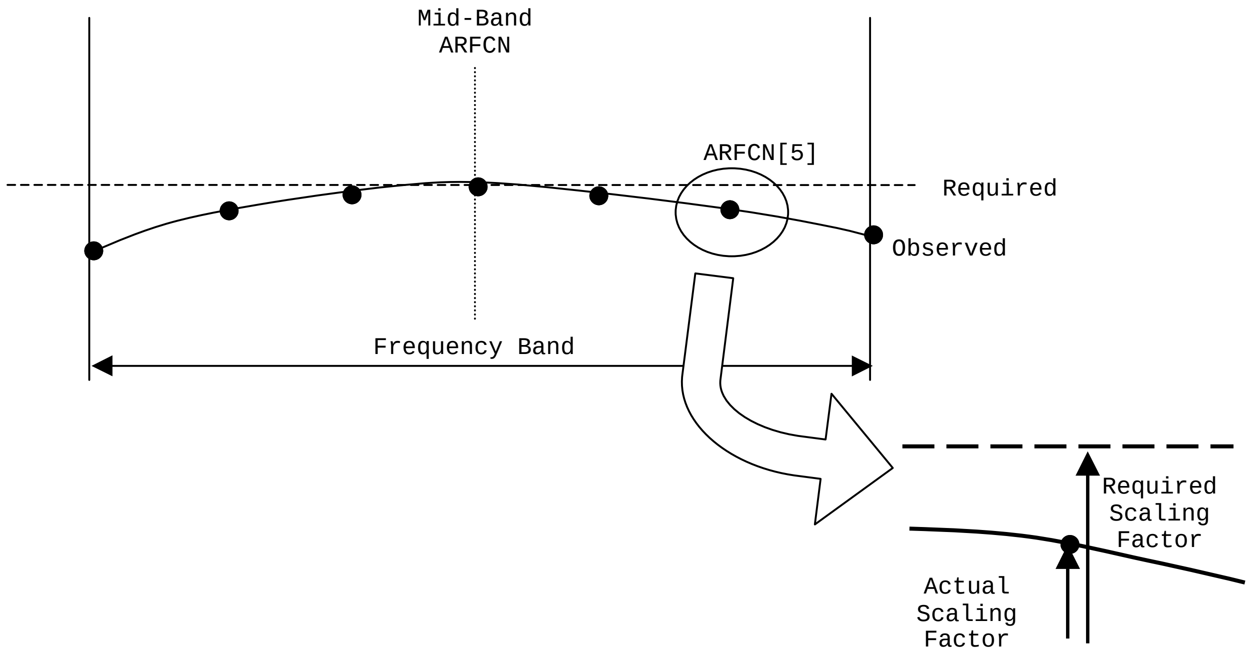

Figure 7 Frequency Compensation

The frequency compensation is calculated from the actual (mid-band) scaling factor and the scaling factor required to reach the required output power level at that particular frequency. With these two factors the necessary frequency compensation may then be calculated thus:

```
txFreqCompensation[point] = (requiredScalingFactor[point] * 0x8000)
                                     actualScalingFactor
```

Frequency compensation is typically performed every 4 ARFCNs across the band to decrease the amount of calibration data stored in NVRAM. Each band has `NUM_CHANNELS`/4 locations reserved in the `txFreqCompensation`[] table, with each location giving the compensation value for four ARFCNs.

It may be that measurements can be performed for a much reduced set of channels, using interpolation to calculate the values between measured points. This will speed up calibration times at the expense of accuracy.

The location in the array for a particular ARFCN is given by the following algorithm:

```
IF ( BAND(arfcn) == EGSM )
   index = int( ((1024 – MIN_EGSM_ARFCN + arfcn) MOD 1024) / 4 )
ELSE IF ( BAND(arfcn) == GSM )
   index = int( ((NUM_EGSM_CHANNELS + 3) / 4) + (arfcn / 4) )
ELSE IF ( BAND(arfcn) == DCS )
   index = int( ((NUM_EGSM_CHANNELS + NUM_GSM_CHANNELS + 3) / 4) + ((arfcn – 512) / 4) )
ELSE
   invalid ARFCN
ENDIF
```

Example

The following is an example of how scaling factors maybe calculated for the top power level across a supported frequency band.

```
FOR  each supported band
   FOR  loopCount = 0 TO CHANNELS_TO_CAL
      SEND CalDevGsmBurstReq ( txArfcnList[loopCount],
                              transmit NORMAL_BURST,
                              HIGH_POWER_CONTROL_LEVEL | 0x0100 )
      WAIT FOR  CalDevGsmBurstCnf
      scalingFactor = rampFactorTable[HIGH_POWER_CONTROL_LEVEL]
      powerLevelAchieved = FALSE
      REPEAT
         SEND CalDevGsmRampScaleReq ( scalingFactor )
         WAIT FOR CalDevGsmRampScaleCnf
         MEASURE measuredPowerLevel USING Power Meter
         measuredPowerLevel = measuredPowerLevel + CABLE_LOSS
         IF measuredPowerLevel IS WITHIN powerLevel +/- (tolerance)dBm THEN
            powerLevelAchieved = TRUE
         ELSE
            IF  measuredPowerLevel < powerLevel THEN
               scalingFactor = scalingFactor + POWER_INCREMENT
            ELSE
               scalingFactor = scalingFactor - POWER_INCREMENT
            ENDIF
         END IF
      UNTIL powerLevelAchieved = TRUE
      CALCULATE txFreqCompensation[loopCount] USING rampFactorTable[] & scalingFactor
   NEXT loopCount
NEXT band
```

Where:

`HIGH_POWER_CONTROL_LEVEL` is 5 for GSM 900 and 0 for DCS1800.

`rampfactorTable[]` is the calibrated scaling factor.

`txArfcnList[CHANNELS_TO_CAL]` contains the list of ARFCNs that are to be calibrated (i.e. for GSM 900 ARFCN 1, 5, 9, 13, … 117, 121 are used; for DCS1800 ARFCN 512, 516, 520… 880, 884 are used; for PCS1900 ARFCN 512, 516, 520… 804, 808 are used).

8 POST CALIBRATION

8.1 Leaving Test Mode

Once the calibration procedure has completed the target must be taken out of `TEST` mode. This may be achieved in one of two ways:

- send a `CalDevGsmFinishReq`.
- power the target off

The target should respond to a `CalDevGsmFinishReq` with a `CalDevGsmFinishCnf`.

The target should now be in NULL state.

8.2 Storing Calibration Data

Once calibration is completed each data table can be stored in NVRAM on the target using signal `L1AlNvramWriteCalReq` sent from the PC. This signal contains a Calibration ID field, `CalDataId`, which is a reference giving the table that is being stored. It also includes a data array in which to place the table data. NOTE: in the default code this array is 255 bytes long, which is not large enough to contain the `rssiCorrectionTable[]` for the DCS or PCS bands. To store these items the target code must be recompiled using a larger data array size.

A `L1AlNvramWriteCalCnf` will be returned indicating whether the data was written correctly.

To protect the calibration data area, data for each Calibration ID may only be stored once. If the Calibration ID has already been used it is not possible to overwrite the data stored without erasing the calibration NVRAM entirely. This may be performed using signal `L1AlNvramEraseCalReq`. A `L1AlNvramEraseCalCnf` will be returned indicating whether the memory was erased correctly.

To read back data stored in NVRAM signal `L1AlNvramReadCalReq` should be used, specifying the Calibration ID. A `L1AlNvramReadCalCnf` will be returned containing the data.

Once calibration data has been stored in NVRAM it will be used immediately.