SC1905 Serial Peripheral Interface Programming Guide

UG6831; Rev 0; 11/18

Abstract

This document provides the information necessary to develop the host software to communicate with the SC1905 by way of the Serial Peripheral Interface (SPI).

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Scope

This document provides the information necessary to develop the host software to communicate with the SC1905 through the Serial Peripheral Interface (SPI).

1.2. Acronyms

Acronyms	Description
AGC	Automatic Gain Control
CCDF	Complementary Cumulative Distribution Function
EEPROM	Electrically Erasable, Programmable, Read-Only Memory
OTP	One Time Programmable memory
EVB	Evaluation Board
EVK	Evaluation Board Kit
PAR	Peak-to-Average Ratio
PVT	Process, Voltage and Temperature.
RF	Radio Frequency
RFFB	RF Feedback
RFIN	RF Input
RFOUT	RF Output
RFPAL	RF PA Linearization
SPI	Serial Peripheral Interface
SSN	SPI Slave Select Enable

1.3. Additional References

1. GUI Installation Guide
2. SC1905 Hardware Design Guide
3. SC1905 Release Notes
4. SC1905 Data Sheet
5. Microchip 25A512 Data Sheet

2. Hardware Interface

2.1. SPI Bus Hardware

The SPI bus comprises four signals: SCLK, SSN, SDI and SDO. The SC1905 can only be used as a slave and the SPI clock can operate from 50KHz to 4MHz. The SPI bus can be shared with multiple slave devices (including several SC1905's). In this case, each slave must have a distinct Slave Select signal (SSN) from the host controller (see Figure 1). Refer to the SC1905 Hardware Design Guide for additional information.

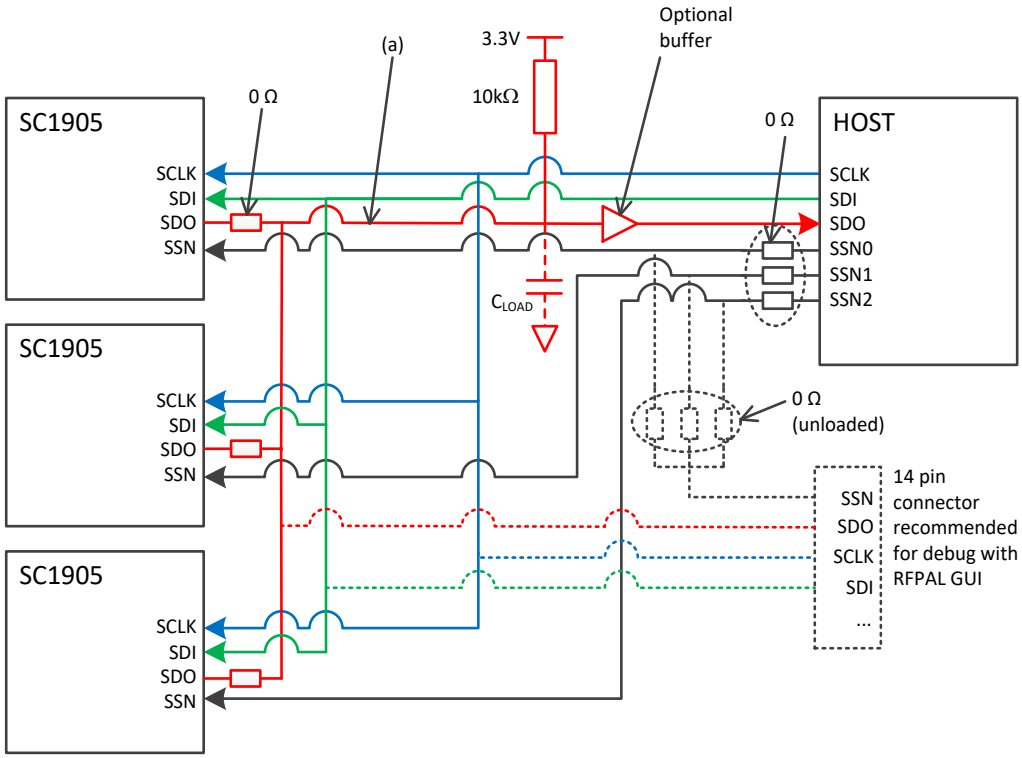


Figure 1: Host SPI Connection for Multiple SC1905 Applications

IMPORTANT: *It is highly recommended to use a 14-pin connector for debug with the GUI (see Figure 2) GUI only supports one SSN, so it is recommended to add a 0Ω resistor as shown in Figure 1 to be able to debug with the GUI.*

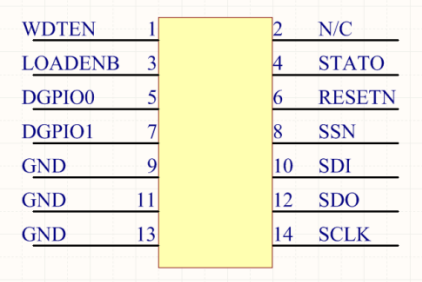


Figure 2: Interface Connector for Development

The SPI bus operates in Mode 0 (CPOL = 0 and CPHA = 0), which means that the data is sampled on the rising edge, and is generated on the falling edge, of SCLK. The signals use 3.3V digital CMOS levels. A detailed signal description is provided below:

- SCLK input: should receive a clock signal from the host during SPI transactions. The clock must have a 50% duty cycle. Internal to the SC1905, this pin is pulled down to ground through a 50KΩ resistor.
- SSN (Slave Select input) functions as an active-low slave selector. Internal to the SC1905, this pin is pulled up to DVDD33 by a 50KΩ resistor.
- SDI input: functions to receive addresses, messages/commands, and data values from the host. This signal should be wired to the MOSI (master out/slave in) signal from the Bus Master. Internal to the SC1905, this pin is pulled down to ground through a 50KΩ resistor.
- SDO three-state output: this signal should be wired to the host MISO signal (master in/slave out). This pin does not have an internal pullup/pulldown and must be externally pulled-up by a 10KΩ resistor to DVDD33. This pin is capable of driving 12mA. Below is the equation for determining the maximum load capacitance for the SDO pin:
  - C_{MAX} (shunt to ground) = 3.75e-4/f_{SPI} (in Farad), where f_{SPI} is the frequency of the SPI communication in Hz.
  - For example: for f_{SPI} = 3MHz, the maximum load capacitance (C_{MAX}) to ground is 125pF. The SC1905 SDO pin capacitance is 2.8pF and must be taken into account when calculating C_{MAX}.
  - For values greater than C_{MAX}, a buffer such as the NC7WZ16P6X would be required.

IMPORTANT: *SSN should be disabled after each transaction as described below.*

In addition to these pins, pin 49 (RESETN) and pin 60 (LOADENB) are required for:

- operating the GUI
- upgrading FW
- configuring the SC1905 from the host.

See sections 2.2 and 2.3 for additional details.

Figure 3 to Figure 7 illustrate examples of transactions used for SPI host message communication (section 3) or EEPROM read and write instructions (section 4).
See Microchip 25A512 data sheet for additional details on transactions with the internal EEPROM.

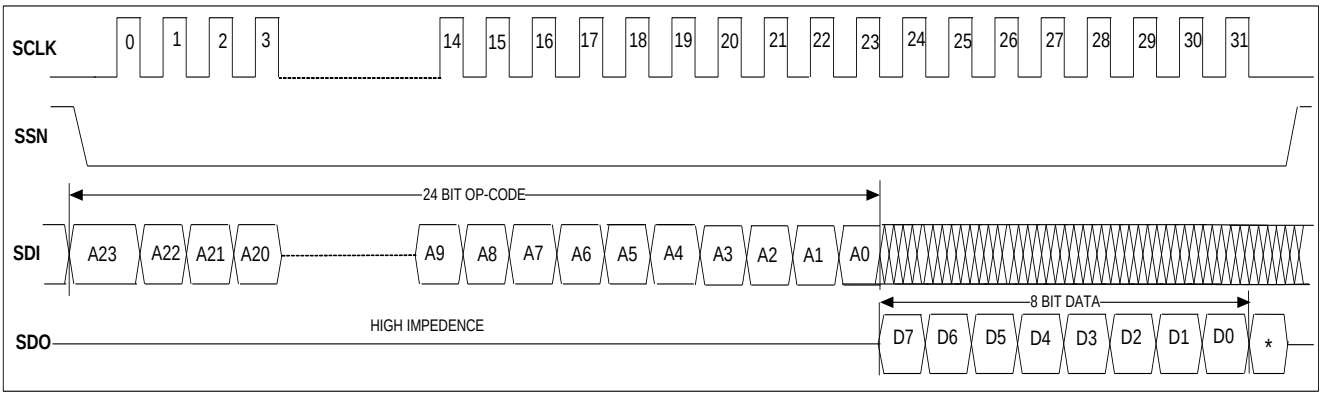


Figure 3: Single-Byte Read

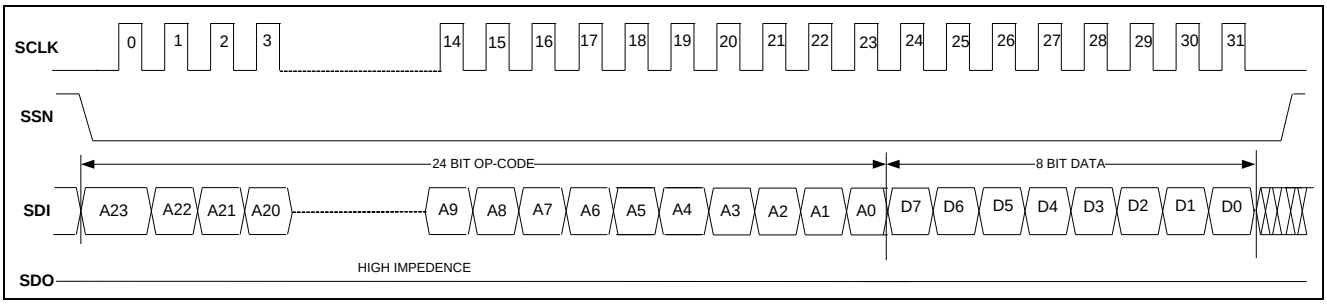


Figure 4: Single-Byte Write

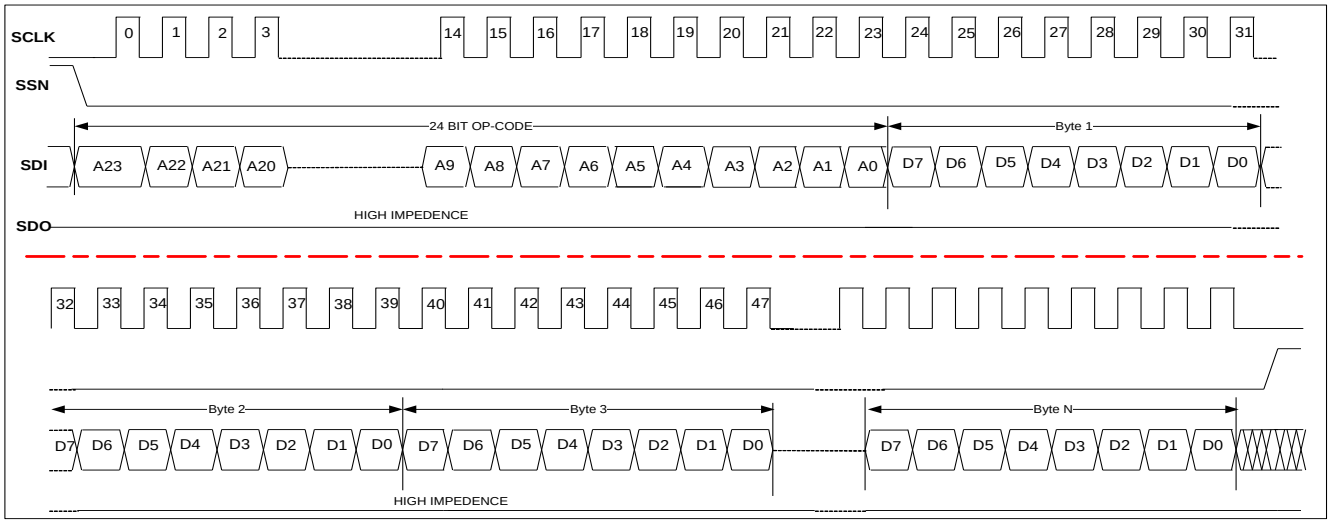


Figure 5: Four-Byte Write

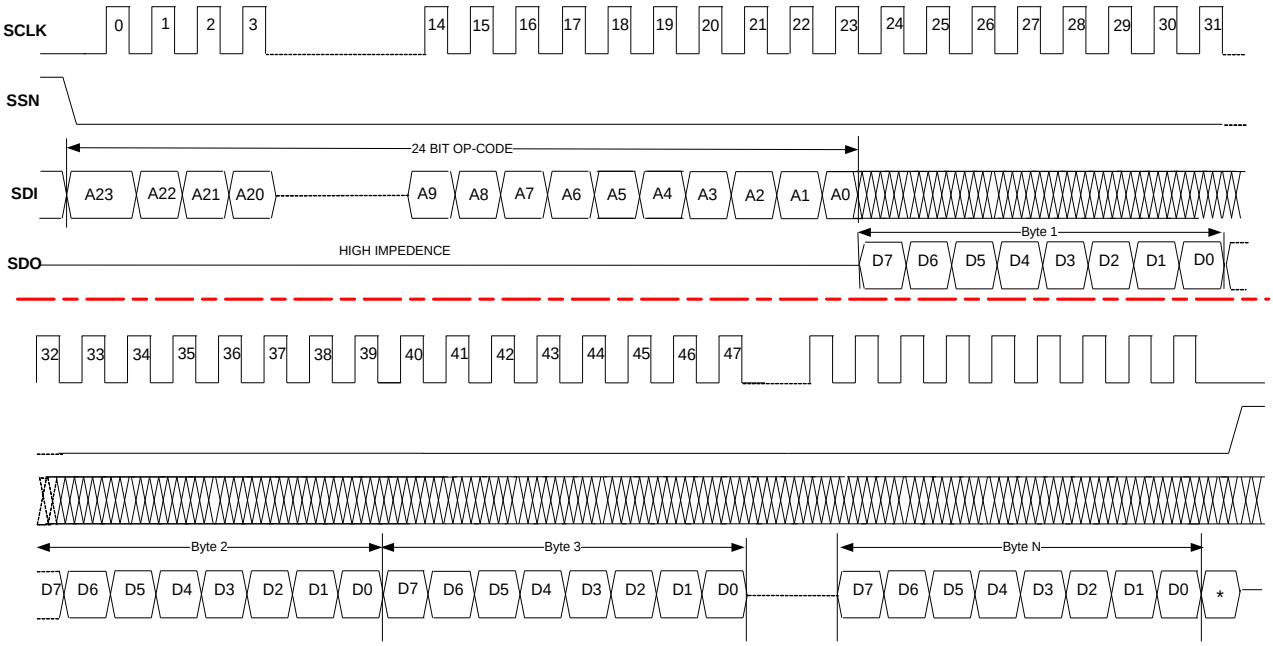


Figure 6: Four-Byte Read

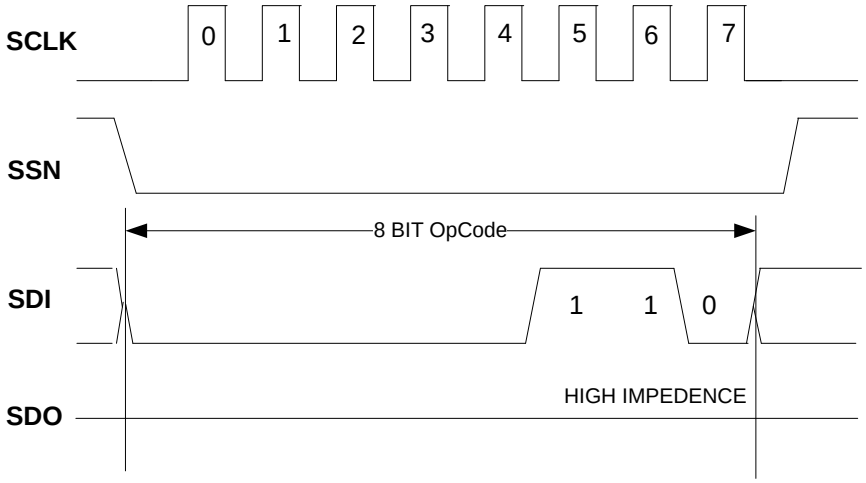


Figure 7: SPI Special Command 06

2.2. LOADENB (Pin 60) For Firmware Upgrade

LOADENB (pin 60) should be utilized when updating firmware. Input to the pin should be 3.3V CMOS level. Internally, this pin is pulled down to ground through a 50KΩ resistor. If the system has a host controller, it is recommended to connect this pin to one of its GPIO's. While this signal is set LOW, the SC1905 will be in normal operational mode. When the LOADENB signal is high, the SC1905 will be placed in a special mode where the SPI Bus is directly connected to the internally embedded EEPROM. In this mode, the SC1905 must be placed in a continuous reset mode by setting pin 49 (RESETN) to low. Throughout firmware updates, LOADENB must be at logic level high and. at the completion of the process, the signal must transition to logic level low. After the programming has been completed, a hard reset should be initiated by commanding the RESETN input low for at least 1µs, then toggled high.

2.3. RESETN (Pin 49) to Reset SC1905

It is required that RESETN, pin 49, be connected to a host processor through a GPIO connection or use a 1uF capacitor connected between pin 49 and ground. The RESETN pin is internally pulled-up to DVDD33 through a 50KΩ resistor. The RESETN (active-low) signal must be kept low for at least 100µs after the last supply is ramped to at least 90% of its final level; or it can be pulsed (from high to low, kept there for at least 1µs, and then back to high). When this signal is low, the SC1905 will be in a reset mode. When the signal goes high, the SC1905 will begin to boot-up and will complete this process in approximately 1 to 3 seconds (depending on the firmware version). After the boot-up process, SC1905 will start adapting towards optimal linearization.

Implementing a host GPIO connection to pin 49, RESETN allows the Host Processor to remotely reset the SC1905 if a re-initialization is required.

3. SPI Host Message Communication

The SC1905 requires a remote interface connection to configure critical parameters, like frequency range and min/max frequency scan bounds. In addition, this allows the user to provide the operational status and error/warning information that are critical for board start-up and debugging. The SC1905 should either be connected to an external host or SPI connector to be able to use the GUI. This provides the following benefits:

1. Ability to download updated versions of SC1905 firmware with incremental feature sets.
2. Ability to obtain continuous operational status.
3. Ability to obtain error/warning alarms.
4. Ability to configure the SC1905 to the appropriate frequency range with the corresponding Min and Max Frequency scanning range limits.

To exchange information with the SC1905 internal memory over the SPI bus, the Host must follow the communication protocol described in the following sections.

3.1. Host Procedure for 4-Byte Message Communication

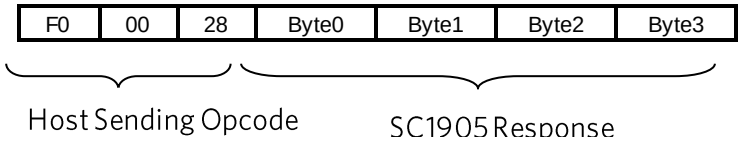
This section summarizes the Host procedure for 4-byte message communication. Refer to Figure 8 for the host flow diagram. Three types of registers are used in this message communication protocol.

1. **MRB**: 4-byte Message Reply Buffer
2. **RSR**: 1-byte Read Status Register
3. **CHK**: 1-byte Checksum Register

Five transmissions are required to read/write these registers.

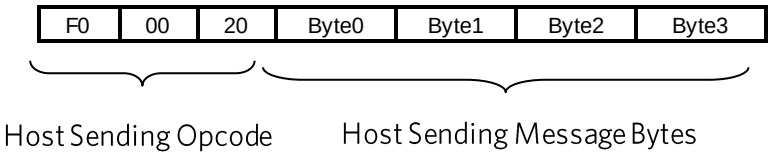
3.1.1. MRB Read Transaction

This transaction consists of two parts. In the first part of the transaction, the Host sends an Opcode which indicates an MRB register read transaction. In the second part of the transaction, the SC1905 sends the 4 bytes of the MRB register. The total transaction length is 56 SCLK cycles. This transaction is described in Figure 8.



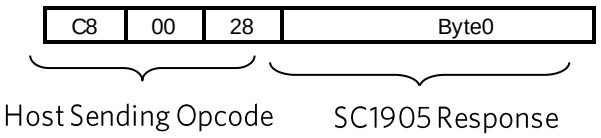
3.1.2. MRB Write Transaction

This transaction consists of two parts. In the first part, the Host sends the Opcode indicating a write transaction to MRB registers. In the second part of transaction, the Host sends 4 message bytes to be written to MRB registers. The total transaction length is 56 SCLK cycles. This transaction is described in Figure 8.



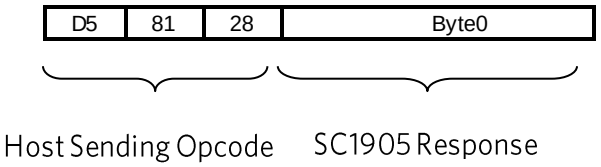
3.1.3. **RSR Read Command**

This transaction consists of two parts. In the first part of transaction, the Host sends the Opcode to read the RSR register. In the second part of the transaction, the SC1905 sends 1-byte which is the contents of the RSR register. The total transaction length is 32 SCLK cycles. Note that RSR is a read only register. This transaction is described in Figure 8.



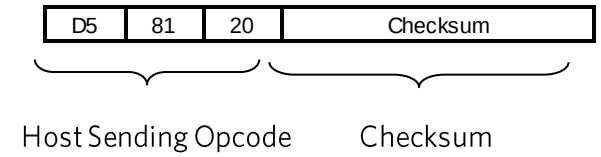
3.1.4. **CHK Read Command**

This transaction consists of two parts. In the first part of the transaction, the Host sends the Opcode to read the CHK register. In the second part of the transaction, the SC1905 sends the 1-byte content of the CHK register. The total transaction length is 32 SCLK cycles. This transaction is described in Figure 8.



3.1.5. **CHK Write Command**

In this transaction, the Host sends the Opcode to write the CHK register along with the checksum value. The total transaction length is 32 SCLK cycles. This transaction is described in Figure 8.



3.1.6. **Read/Write Message Protocol**

The following steps summarize the read/write message protocol:

1. Wait at least one second following reset before proceeding with step 2.
2. Host reads the RSR to determine current value.
3. Compose the four-byte message to write to the MRB, then compute the modulo-256 sum over these four bytes. Then compute the one's complement of the resulting value (i.e., invert each bit). The resulting value is the checksum for the message.
4. Write the checksum found in step 3 to the CHK register.
5. Write the four-byte message to the MRB. Start a timer with expiration value of one second upon completion of the write.
6. Host reads the RSR by using the RSR read command, C8 00 28. Read byte can assume the following four values:

RSR Value	SC1905 Status
0x0F or 0xF0	RSR values 0x0F and 0xF0 alternate indicating that the SC1905 response to prior command is ready. These values are also referred to as ACK0/1, respectively
0xFF	RSR of 0xFF indicates the most recently issued command was not received correctly. This value is also referred to as a NAK.
0x00	Indicates that the SC1905 has not yet completed processing of any commands since being reset

Host should keep polling the RSR every 5ms until either the timer expires or the RSR changes value. Any value other than the four in this table, is treated the same as a NAK.

1. If the expected ACK is returned before the timer expires, host reads the MRB registers by issuing an MRB read command, then read the CHK register by issuing a CHK read command. If timer expires before the RSR changes, or new RSR value is anything other than expected ACK, return to step 4.
2. Host computes the ones complement modulo-256 checksum over the five bytes consisting of the four bytes read from the MBR register plus the one byte from the RSR. Compare against the value read from the CHK register. If the values match, then transaction is complete. Otherwise, return to step 4.

IMPORTANT: *After Reset, the first read byte of MRB will remain 0x00 until the first command response is available.*
The SSN should be disabled after each transaction as described above.
Before sending a command, it is required to read the RSR.

1. *If 0x0F is read, then 0xF0 will indicate that the response to the command is ready.*
2. *Similarly, if 0xF0 was read before sending the command, then 0x0F will indicate that the response of the command is ready.*
3. *If the chip was reset, then "0x00" will be read and either 0xF0 or 0x0F will indicate that the response to the command is ready.*

Figure 8 provides a flow diagram that illustrates the sequence of actions from the start of a transaction to the completion.
See section 3.6 for examples of SPI Message Communication Commands.

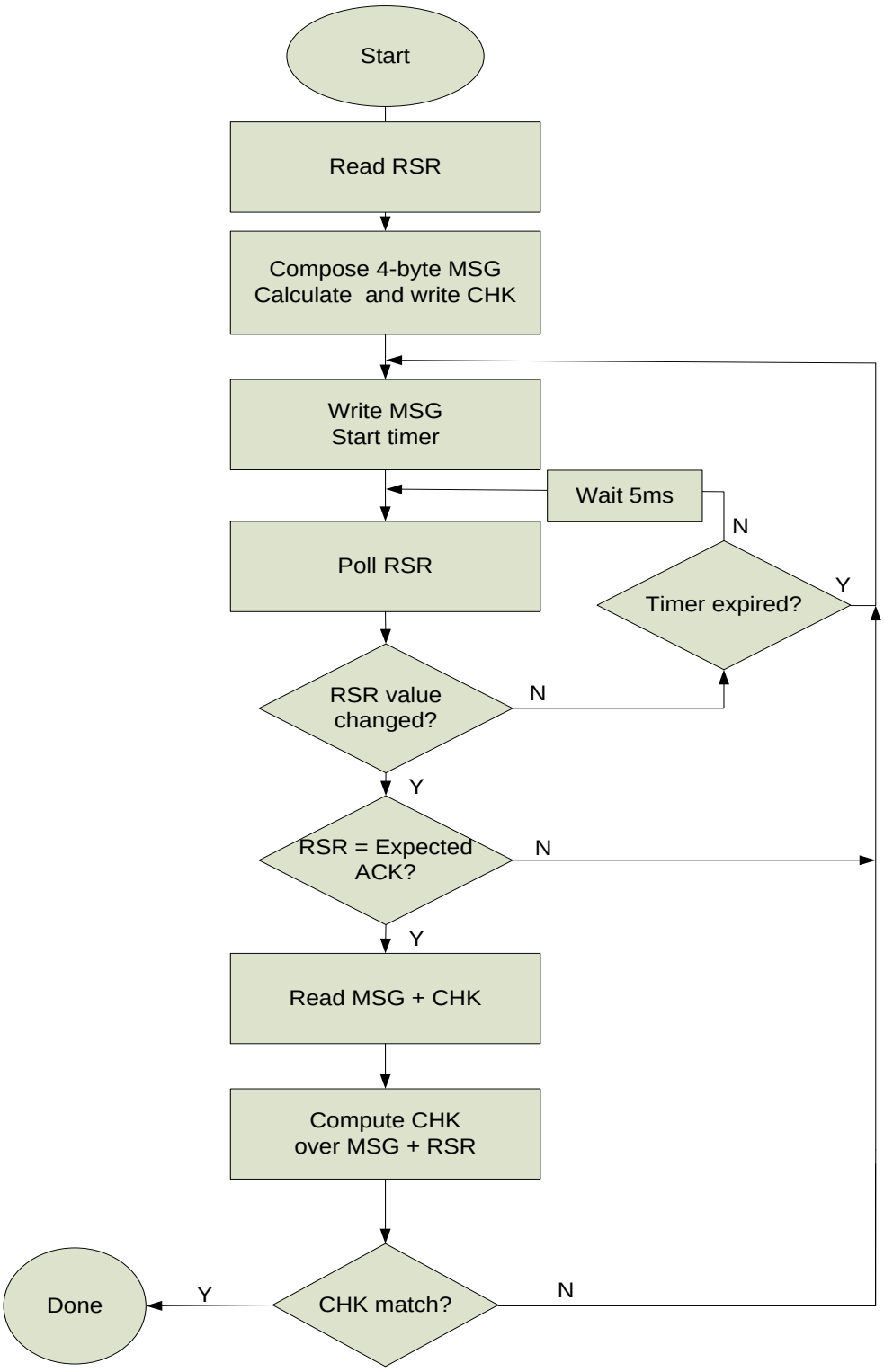


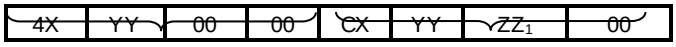
Figure 8: Host Flow Diagram

3.2. SPI Message Read/Write Command Format

SPI messages to read/write 1 or 2-byte are described in the following sections. These commands only allow accessing the first 4K bytes of the scratch memory. For some of the parameters, it is required to access address beyond the 4K limit. Then it is required to send a special command to extend the readable range.

3.2.1. Host Message to Read 1-byte from the Scratch Memory

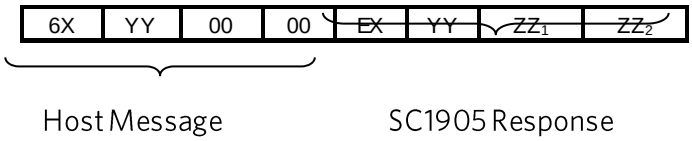
Host Message to read 1-byte from the scratch is



Where XYY is the hexadecimal address in the scratch and ZZ_1 the 1-byte value read.

3.2.2. Host Message to Read 2-bytes From the Scratch Memory

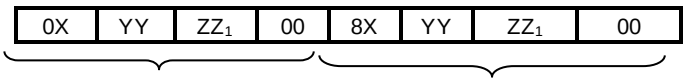
Host Message to read 2-bytes from the scratch is



Where XYY is the hexadecimal address in the scratch and ZZ_1 ZZ_2 the 2-byte value read.

3.2.3. Host Message to Write 1-byte to the Scratch Memory

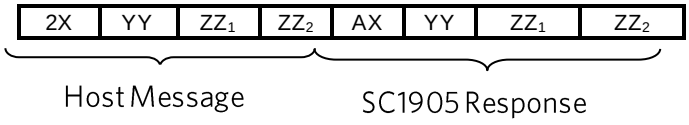
Host Message to write 1-byte to the scratch is



Where XYY is the hexadecimal address in the scratch and ZZ_1 the 1-byte value written.

3.2.4. Host Message to Write 2-bytes to the Scratch Memory

Host Message to write 2-bytes to the scratch is



Where XYY is the hexadecimal address in the scratch and ZZ_1 ZZ_2 the 2-byte value written.

3.2.5. Supported SPI Message Communication Commands

These SPI messages use the host message protocol described in section 1. Please refer to Figure 8 for Host Flow Diagram. See section 8.2 for example code for reading SPI message parameters.

Table 1: Scratch Parameters Available Through SPI Messages

Scratch Address (Hex)	Size/Access	Variable Name	Description
003	8-bit R	FW Version	Get Firmware version. Represents W.X.YY.ZZ where each hexadecimal digit for W and X is separately displayed as a decimal where a value 0x60 would be displayed as 6.0.YY.ZZ
004	8-bit R	Get FW Build MSB	Get Firmware build MSB. Represents W.X.YY.ZZ where YY is the value converted to ASCII decimal where a value WX = 0x60 and YY = 01 is displayed as 6.0.01.ZZ
005	8-bit R	Status	The Host should Get Status at least every 2s and not faster than every 100ms. Bit#7 Error occurred if bit contains "1" Bit#6 Warning occurred if bit contains "1" Bit#5-0 contains value describing overall status: 000000 = INIT 000001 = FSA (Full Speed Adaptation) 000011 = TRACK (Tracking) 000110 = CAL (Calibrating) 001001 = PDET (Calibrating) Other values not valid modes If bit#7 is set, host should "Read Error" within 6s. SC1905 will reset 6s after an error occurs. Error information will be lost after reset. If bit#6 is set, host should "Read Warning" and "Clear Warning".
006	8-bit R	Error	Host to read error code from SC1905. XX is the decimal error number. 00 means no error. Any other values mean that the chip has an internal failure and should not typically happen. Please refer to the release note for Error code.
007	8-bit R	Warning	Host to read warning code from SC1905. YY is the decimal warning number. Please refer to the release note for Warning code.
008	8-bit RW	Output Mode	Output Mode XX = 00 = RFOUT Disabled (Adaptation is frozen and SC1905 is not linearizing the PA) XX = 01 = "FW Control". This means RFOUT is enabled, by default, but can be disabled by the firmware. For example, in CAL, RFOUT Status is OFF, even if the mode is set to "FW Control". **Required**: After changing Output Mode, send the "Activate Outputs" messages to be effective. See Table 2
00A	8-bit R	FW Build LSB	Firmware Build LSB. Represents W.X.YY.ZZ where ZZ is the value converted to ASCII decimal where a value WX = 0x60, YY = 01 and ZZ = 00 is displayed as 6.0.01.00
010	8-bit R	Frequency Range	XX hexadecimal value of Frequency Range. XX = 04: 698MHz -1040MHz XX = 05: 1040MHz - 2080MHz XX = 06: 698MHz - 2700MHz XX = 07: 1800MHz - 2700MHz XX = 08: 2700MHz - 3500MHz XX = 09: 3300MHz - 3800MHz

Scratch Address (Hex)	Size/Access	Variable Name	Description
011	16-bit R	MinFrequencyScan	XX YY hexadecimal value of 2xMinFrequency Scan (MHz). For example, XX YY = 0E 10 corresponds to MinFrequency= 1800MHz. The MinFrequencyScan is the lowest frequency that the SC1905 examines when searching for the signal center frequency.
013	16-bit R	MaxFrequencyScan	XX YY hexadecimal value of 2 x MaxFrequency Scan (MHz). For example, XX YY = 15 E0 corresponds to MaxFrequencyScan = 2800MHz. The MaxFrequencyScan is the highest frequency that the SC1905 examines when searching for the signal center frequency.
017	8-bit R	Adaptation Mode	XX hexadecimal value of Adaptation Mode XX = 00 = Duty Cycled Feedback OFF (Default State) XX = 01 = Duty Cycled Feedback ON (Not recommended for TDD applications)
018	16-bit R	Signal Bandwidth	XX YY hexadecimal value of 2 x SignalBandwidth (MHz). For example, XX YY = 00 1E corresponds to signal Bandwidth = 15MHz.
01A	16-bit R	Unscaled Center Frequency	XX YY hexadecimal value of 2 x Unscaled Center Frequency (MHz). For example, XX YY = 0F A3 corresponds to signal center Frequency = 2001.5MHz. See section 4.1.2 for details
023	8-bit RW	Adaptation State	XX hexadecimal value of Adaptation State XX = 00 = Frozen (Freeze Adaptation) XX = 01 = Running (Default state: Adaptation Running)
032	8-bit R	Get Output Status	Get Output Status. = 00 = RFOUT OFF (RFOUT disable: Adaptation is frozen and SC1905 is not linearizing the PA) = 01 = RFOUT ON
033	8-bit R	Normalization_Factor	8-bit unsigned value of Normalization Factor. 0x2A = 42. See section 0 for details.
034	16-bit R	Unnormalized_Coeff	16-bit unsigned value of the un-normalized coefficient value. See section for details.
23C	8-bit R	RFIN AGC	8-bit unsigned value of RFIN AGC (PDET) value between 0 and 15
959	8-bit R	Product ID	8-bit unsigned value of Product ID. 1894 for SC1894 and 1905 for SC1905
9C4	8-bit R	RFFB AGC	8-bit unsigned value of RFFB AGC value between 0 and 29
BA8	16-bit	Scaled Center Frequency	The center frequency is scaled to support various external XO reference clock frequencies. XX YY hexadecimal value of 2 x ScaledFrequency (MHz). For example, XX YY = 0F A3 corresponds to scaled signal center Frequency = 2001.5MHz. See section 4.1.2 for details

IMPORTANT: *If the Message is less than 4-bytes long, it is required that 4-bytes be sent before disabling the SSN (although the content of the additional bytes may not have any particular value). So, for 2-byte messages, it is required to add two dummy bytes. Similarly, if the Reply is fewer than 4-bytes long, it is required that all 4-bytes are received before disabling the SSN. So, for a 2-byte response, 2 extra bytes will be received and discarded.*
After changing Output Mode, it required to send the "Activate Outputs" messages to be effective. See Table 2.

3.3. Special Commands

Table 2: Special SPI Commands

Message (Hex)	Reply (Hex)	Command Name	Description
10 03 00 00	90 03 00 00	Clear Warning	Clears the Warning Code
10 04 00 00	90 04 00 00	Activate Output	Activate Output Mode. This command must be issued following a Set Output Mode command, described in above message
10 05 00 00	90 05 00 00	Request Rescan	Requests rescan for signal frequency. Not disruptive to PA linearization.
10 CD 00 00	90 CD 00 00	Extend Scratch Readable Access Enable	Enables extended readable range of the scratch by adding address offset of 0x800. The instructions in section 3.2 will then access address XYZ + 0x800.
10 CE 00 00	90 CE 00 00	Extend Scratch Readable Access Disable	Disables extended readable range of the scratch by removing address offset of 0x800.
10 FA 00 00	90 FA 00 00	Wake-up	When the Duty Cycled Feedback is ON, the SC1905 will be mostly powered down during the OFF time (1s). This command will force SC1905 to power back up by going back to ON state. The wake process takes about 1ms to complete.

3.4. Special SPI Commands for Smooth Mode Calibration

The SC1905 can operate in one of two modes: Smooth Mode and Optimized Mode. In Optimized Mode, the firmware will execute the full AGC routines to find the optimal settings for the various attenuators and amplifiers in the SC1905 hardware whenever the conditions change. For example, if the PA output power is stepped down several dB, the AGC routines will be rerun. This causes significant spectral distortion during the time it occurs, but the final correction performance will usually be the best achievable by the hardware. Optimized mode is therefore not suitable for dynamic operation. It is mainly intended for tuning PAs and debugging performance issues. To switch to Smooth Mode, a calibration procedure is run. The procedure essentially involves applying a waveform, then issuing some SPI commands to the firmware. The full AGC routines are run and the values stored in EEPROM by the firmware. Once these calibration parameters have been written, the device is operating in Smooth Mode. In Smooth Mode, if some condition (such as PA output power or temperature) changes, the firmware estimates what the optimal values are for the various AGC indices for the new condition, rather than rerunning the full AGC routines. The final settings may not be optimal, but they will be close enough that performance is only slightly degraded relative to what would be achieved in Optimized mode. The main benefit is that there is little spectral disruption in dynamic conditions. If the calibration values in EEPROM are zeroed out, and the device reset, then it reverts back to Optimized Mode. The SPI commands used for Smooth Mode calibration are described in Table 3. Use of these commands is described in Section 7, with example code provided in sections 8.3 and 0.

Table 3: SPI Messages Communication Commands for Smooth Mode Calibration

Message (Hex)	Reply (Hex)	Command Name	Description
10 F3 00 00	90 F3 00 00	Clear MaxPWRCalParameters A	Firmware to clear maximum power amplifier output power calibration parameter and adaptation coefficients in EEPROM for frequency A. Then read MaxPWRClearOnGoing for command status. See Table 4
10 F4 00 00	90 F4 00 00	Clear MaxPWRCalParameters B	Firmware to clear maximum power amplifier output power calibration parameter and adaptation coefficients in EEPROM for frequency B. Then read MaxPWRClearOnGoing for command status. See Table 4
10 F5 00 00	90 F5 00 00	Write MaxPWRCalParameters A	Firmware to write maximum power amplifier output power calibration parameter and adaptation coefficients in EEPROM for frequency A. Then read MaxPwrCalAOngoing for command status. See Table 4
10 F6 00 00	90 F6 00 00	Write MaxPWRCalParameters B	Firmware to write maximum power amplifier output power calibration parameter and adaptation coefficients in EEPROM for frequency B. Then read MaxPwrCalBOngoing for command status. See Table 4

Table 4: Scratch Parameters for Smooth Calibration Command Status

Scratch Address (Hex)	Size/ Access	Variable Name	Description
DC3	UNIT8 R	MaxPWRClearOnGoing	Flag used to indicate the execution status of the command "Clear MaxPWRCalParameters" Keep executing reads of this flag until a value of 0x00 is returned by the SC1905.
DC4	UNIT8 R	MaxPwrCalAOngoing	Flag used to indicate the execution status of the command "Write MaxPWRCalParameters A" Keep executing reads of this flag until a value of 0x00 is returned by the SC1905.
DC6	UNIT8 R	MaxPwrCalBOngoing	Flag used to indicate the execution status of the command "Write MaxPWRCalParameters B" Keep executing reads of this flag until a value of 0x00 is returned by the SC1905.

3.5. Average of the Magnitudes of the Coefficients

In "FSA" and "TRACK" states, the average of the magnitudes of the coefficients is computed as follows:

Average_Coeff = Unnormalized_Coeff/Normalization_Factor;

Refer to Section 3.2 for instructions on how to get these parameters.

Example:

Unnormalized_Coeff = 0x04 59 = 1113

Normalization_Factor = 42

Then Average_Coeff = 1113/42 = 26.5

3.6. Examples of SPI Message Communication Commands

3.6.1. To Read the Product ID from Scratch

productID = rfpal_msgCmdRead(h, hex2dec('959'),1)
-> D5 81 20 3D %CHK Write Command CHK = 3D
-> C8 00 28 00 %Read RSR
<- FF FF FF 0F % RSR Value is 0F
-> F0 00 20 69 59 00 00 % Send command With MRB Write Transaction
% CHK Computation. Mod256(0x69+0x59) =Mod256(0xC2) = 0xFF-0xC2 = 0x3D = CHK
-> C8 00 28 00 %Read RSR
<- FF FF FF F0 % Value is F0 indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF E9 59 07 71 %Response = 0x0771 = 1995
-> D5 81 28 00 %Read CHK to make sure it matches the computed CHK
<- FF FF FF 55 %Read CHK = 0x55
CHK computation. Mod256(0xF0 + 0xE9 + 0x59 +0x07 + 0x71) = 0xFF – 0xAA = 0x55
productID = 1905 % For the SC1905

3.6.2. To Read the FW Build LSB from Scratch

The following shows the different SPI transactions to read the FW Build LSB from scratch:
FWBuildLSB = rfpal_msgCmdRead(h, hex2dec('00A'), 0)
-> D5 81 20 **B5** %CHK Write Command CHK = B5
-> C8 00 28 00 %Read RSR
<- FF FF FF **0F** % Value is 0F
-> F0 00 20 40 0A 00 00 % Send command With MRB Write Transaction
% CHK Computation. Mod256(0x40+0x0A) = Mod256(0x4A) = CHK = 0xFF-0x4A = **0xB5**
-> C8 00 28 00 %Read RSR
<- FF FF FF **F0** % Value is F0 indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF C0 0A **08** 00 %Response to Read FW Build LSB is 08
-> D5 81 28 00 %Read CHK to make sure it matches the computed CHK
<- FF FF FF **3D** %Read CHK = 3D
% CHK computation. Mod256(0xF0+0xC0+0x0A+0x08+0) = 0xFF - 0xC2 = **0x3D = CHK**
FWBuildLSB = 8

3.6.3. To Enable/Disable RFOUT

To disable RFOUT, the following shows the different SPI commands:
err = SPImsgRfOutEnable(h, 0)
-> D5 81 20 **F7** % Write CHK = 0xF7
-> C8 00 28 00 %Read RSR
<- FF FF FF **F0** %Value is F0
-> F0 00 20 00 08 00 00 % Send MRB Write Transaction to write '0' to Output Mode scratch variable
% CHK Computation. Mod256(0x0+0X8) = 0x8. CHK = 0xFF-0x8 = **0xF7**
-> C8 00 28 00 %Read RSR
<- FF FF FF **0F** %Value is 0F indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF 80 08 00 00 %SC1905 response
-> D5 81 28 00 %CHK Read Command to verify response CHK
<- FF FF FF 68 % CHK = 0x68 is read
% CHK computation. Mod256(0x0F+ 0x80+0x08+0x00+0x00+0) = 0x97. CHK = 0xFF-0x97 = **0x68**
-> D5 81 20 EB %Write Command CHK = 0xEB
-> C8 00 28 00 %Read RSR
<- FF FF FF 0F % Value is 0F
-> F0 00 20 10 04 00 00 %Activate Output command is sent with MRB Write Transaction
% CHK Computation. Mod256(0x10+0X04+0x00+0x00) = 0x14. CHK = 0xFF-0x14 = **0xEB**
-> C8 00 28 00 %Read RSR
<- FF FF FF F0 %Value is F0 indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF 90 04 00 00 %SC1905 response
-> D5 81 28 00 %CHK Read Command to verify response CHK
<- FF FF FF 7B %CHK = 0x7B is read
%CHK computation. Mod256(0xF0+ 0x90+0x04+0x00+0x00+0) = 0x84. CHK = 0xFF-0x84 = **0x7B**
err = 0
To Set RFOUT to FW Control, the following shows the different SPI commands:
err = SPImsgRfOutEnable(h, 1)
-> D5 81 20 F6 %Write CHK = 0xF6
-> C8 00 28 00 %Read RSR
<- FF FF FF **F0** %Value is F0
-> F0 00 20 00 08 01 00 % Send MRB Write Transaction to write '1' to Output Mode scratch variable
-> C8 00 28 00 %Read RSR
<- FF FF FF **0F** %Value is 0F indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF 80 08 01 00 %SC1905 response to Output Mode write command
-> D5 81 28 00 %CHK Read Command to verify response CHK
<- FF FF FF 67 %CHK = 0x67 is read
% CHK computation. Mod256(0x0F+ 0x80+0x08+0x01+0x00+0) = 0x98. CHK = 0xFF-0x97 = **0x67**
-> D5 81 20 EB %Write Command CHK = 0xEB
-> C8 00 28 00 %Read RSR
<- FF FF FF **0F** %Value is 0F
-> F0 00 20 10 04 00 00 %Activate Output command is sent with MRB Write Transaction

-> C8 00 28 00 %Read RSR
<- FF FF FF **F0** %Value is F0 indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF 90 04 00 00 %4-byte SC1905 response
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 7B %CHK = 0x7B is read
%CHK computation. Mod256(0xF0+0x90+0x04+0x00+0x00)=0x84. CHK = 0xFF-0x84 = **0x7B**
err = 0

3.6.4. To Clear Warnings

err = rfpal_msgSa(h,03)
-> D5 81 20 EC %CHK Write Command
-> C8 00 28 00 %Read RSR
<- FF FF FF **F0** %Value is F0
-> F0 00 20 10 03 00 00 %Clear warning command is sent with MRB Write Transaction
-> C8 00 28 00 %Read RSR
<- FF FF FF **0F** %Value is 0F indicates that the response to the command is ready to be read
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF 90 03 00 00 %4-byte SC1905 response
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 5D %CHK = 0x5D is read
%CHK computation. Mod256(0x0F+ 0x90+0x03+0x00+0x00)=0xA2. CHK = 0xFF-0xA2 = **0x5D**
err = 0
To Clear MaxPWRCalParameters A
SC1905clearMaxPWRCalParameters(h, 0)
-> D5 81 20 **FC** %CHK Write Command
-> C8 00 28 00 %RSR Read Command
<- FF FF FF **0F** %0F is read from RSR before sending Command. Waiting for F0 for response to be ready
-> F0 00 20 10 F3 00 00 %Clear Max PWR Cal Parameters command
% Mod256(0x10+0xF3+0+0)=0x03. CHK = FF-0x03 = FC
-> C8 00 28 00 %Send RSR Read Command until F0 is read
<- FF FF FF **0F** %0F is read from RSR. Response not ready. Waiting for F0 for response to be ready
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready. Waiting for F0 for response to be ready
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **0F** %0F is read from RSR. Response not ready.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF **F0** % Value is F0, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read

<- FF FF FF 90 F3 00 00 %SC1905 Command response 90 F3 00 00
-> D5 81 28 00 %CHK Read Command
<- FF FF FF 8C %CHK = 0x8C
% CHK computation. Mod256(0xF0+0x90+0xF3+0+0)=0x73. CHK = 0xFF-0x73 = 0x8C
-> D5 81 20 EF %Write CHK=0xEF for Command
-> C8 00 28 00 %Send RSR Read
<- FF FF FF F0 %Value is 0xF0
-> F0 00 20 4D C3 00 00 %Read MaxPWRClearOnGoing value with MRB Write Transaction
% Mod256(0x4D+0xC3+0+0)=0x10. CHK = FF-0x10 = EF is correct.
-> C8 00 28 00 %Send RSR Read
<- FF FF FF 0F %Value is 0F, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF CD C3 00 00 %SC1905 4-byte response to Command to Read MaxPWRClearOnGoing
% 0 means that “Clear MaxPWRCalParameters” Command is completed.
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 60 % CHK = 0x60 is read
% CHK computation. Mod256(0x0F+ 0xCD+0xC3+0x00+0x00)=0x9F. CHK = 0xFF-0x9F = 0x60

3.6.5. **Write MaxPWRCalParameters A**

The following shows the different SPI transactions for the Write MaxPWRCalParameters A:
rfpal_msgSa(h,hex2dec('F5'))
-> D5 81 20 FA %Write CHK = FA
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value is F0
-> F0 00 20 10 F5 00 00 %Send "Write MaxPWRCalParameters A" command with MRB Write Transaction
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the SC1905 response
<- FF FF FF 90 F5 00 00 %SC1905 4-byte response to Command
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 6B %CHK = 0x6B is read
% CHK computation. Mod256(0x0F+ 0x90+0xF5+0x00+0x00)=0x94. CHK = 0xFF-0x94 = **0x6B**
ans = 0
Then it is required to read the MaxPwrCalAOngoing flag to make sure the "Write MaxPWRCalParameters A" command has been completed.
calAOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC4'),0);
-> D5 81 20 EE %Write CHK = EE
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value is F0
-> F0 00 20 4D C4 00 00 %Read MaxPwrCalAOngoing value with MRB Write Transaction
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the SC1905 response
<- FF FF FF CD C4 01 00 %SC1905 4-byte response:
%1 means "Write MaxPWRCalParameters A" not complete yet.
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 5E %CHK = 0x5E is read
%CHK computation. Mod256(0x0F+ 0xCD+0xC4+0x01+0x00)=0xA1. CHK = 0xFF-0xA1 = **0x5E**
-> D5 81 20 EE %Write CHK = EE
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F
-> F0 00 20 4D C4 00 00 %Read MaxPwrCalAOngoing value with MRB Write Transaction
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value is F0, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the SC1905 response
<- FF FF FF CD C4 00 00 %SC1905 4-byte response:
% 0 means "Write MaxPWRCalParameters A" is complete.
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 7E %CHK = 0x7E is read
% CHK computation. Mod256(0xF0 + 0xCD + 0xC4 + 0x00 + 0x00) = 0x81. CHK = 0xFF-0x81 = **0x7E**
>> calAOngoingFlg = 0

IMPORTANT: *It might be required to Read MaxPwrCalAOngoing several times as the Write MaxPWRCalParameters A could take 1-2s*

3.6.6. To Clear MaxPWRCalParameters B

SC1905clearMaxPWRCalParameters(h, 1)
-> D5 81 20 FB %CHK Write Command
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F
-> F0 00 20 10 F4 00 00 %Send Clear Cal Param B with MRB Write Transaction
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F. Response is not ready yet
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value is F0. Response is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the SC1905 response
<- FF FF FF 90 F4 00 00 %SC1905 4-byte response
-> D5 81 28 00 %CHK Read Command
<- FF FF FF 8B %CHK = 0x8B
% CHK computation. Mod256(0xF0 + 0x90 + 0xF4 + 0 + 0) = 0x74. CHK = 0xFF-0x74 = 0x**8B**
-> D5 81 20 EF %Write CHK = 0xEF for Command
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value is F0
-> F0 00 20 4D C3 00 00 %Read MaxPWRClearOnGoing value with MRB Write Transaction
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value is 0F, which means that the response to Command is ready
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF CD C3 00 00 %SC1905 4-byte response to Command to Read MaxPWRClearOnGoing
%0 means that "Clear MaxPWRCalParameters" Command is completed.
-> D5 81 28 00 %CHK Read Command to verify response CHK with SC1905 4-byte response+RSR value
<- FF FF FF 60 %CHK = 0x60 is read
% CHK computation. Mod256(0x0F+ 0xCD+0xC3+0x00+0x00)=0x9F. CHK = 0xFF-0x9F = **0x60**

3.6.7. To Read Cost Function Value

Cost_function_bytes = double(rfpal_msgCmdRead(h, hex2dec('20D'), 1))
-> D5 81 20 90 %CHK Write Command
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value of 0F
-> F0 00 20 62 0D 00 00 %MRB Write to read 2-byte from @ 0x20D = 525
%CHK computation. 0x62 + 0xD + 0 + 0 = 0x6F = 111. CHK = dec2hex(255 - 111) = 0x90
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value of F0. SC1905 response to command is ready
-> F0 00 28 00 00 00 00 %MRB Read
<- FF FF FF E2 0D EA 68 %SC1905 Command response
-> D5 81 28 00 %CHK Read Command
<- FF FF FF CE %CHK from SC1905 Command response
% CHK computation. 0xF0 + 0xE2 + 0x0D + 0xEA + 0x68 = 0x331. Mod256 = 0x31 = 49.
% CHK = dec2hex(255-49) = 0xCE
Cost_function_bytes = 60008 = 0xEA68

Since 0xEA68>7FFF Then Cost = 60008-65536 = -5528

3.6.8. To Read Temperature IC

ic_temp_bit = uint16(rfpal_msgCmdRead(h, hex2dec('23D'), 1));
IC_temp = Read16B_signed_Scratch(ic_temp_bit);
-> D5 81 20 60 %Write CHK = 60
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %0F is read from RSR before sending Command. Waiting for F0 for response to be ready
-> F0 00 20 62 3D 00 00 %Read IC Temp parameter with MRB Write command
%CHK computation. 0x62+0x3D+0=0x9F. CHK=0xFF-0x9F=0x60
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %F0 is read from RSR. Response is ready.
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF E2 3D 00 28 %SC1905 response is 0x28 = 40C. See section 8.12 for negative temperature conversion.
-> D5 81 28 00 %CHK Read Command
<- FF FF FF C8 %CHK = 0xC8 is read
% CHK computation. Mod256(0xF0 + 0xE2 + 0x3D + 0x00 + 0x28) = 0x37. CHK = 0xFF-0x37 = **0xC8**

3.6.9. To Read RFIN and RFFB PMU Values

RFIN_PMU_bytes = double(rfpal_msgCmdRead(h, hex2dec('247'), 1)) %Address 0x247 = 583
-> D5 81 20 56 %Write CHK = 56
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 00 %0 is read from RSR before sending Command as the chip was just Reset.
-> F0 00 20 62 47 00 00 %Read RFIN PMU parameter with MRB Write command
%CHK computation. 0x62+0x47+0+0=0xA9. CHK=0xFF-0xA9=0x56
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %0F is read from RSR. Response is ready.
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF E2 47 09 99 %SC1905 4-byte response 0x0999 = 2457
-> D5 81 28 00 %CHK Read Command
<- FF FF FF 25 %CHK = 0x25 is read
% CHK computation. Mod256(0x0F+0xE2+ 0x47+0x09+0x99)=0xDA. CHK = 0xFF-0xDA = **0x25**
RFIN_PMU_bytes = 2457 = 256*0x09+0x99=256*9+153
% See section 6.1.3 for conversion and sections 8.6 and 8.12 for MATLAB® example code
RFIN.RMS = 3.01*Read16B_signed_Scratch(RFIN_PMU_bytes)/1024= 7.2222 dBm

RFFB_PMU_bytes = double(rfpal_msgCmdRead(h, hex2dec('245'), 1)) %Address 0x245 = 581
-> D5 81 20 58 %Write CHK = 58
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %0F is read from RSR before sending Command. Waiting for F0 for response to be ready
-> F0 00 20 62 45 00 00 %Read RFFB PMU parameter with MRB Write command
%CHK computation. 0x62+0x45+0+0=0xA7. CHK=0xFF-0xA7=0x58
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %F0 is read from RSR. Response is ready.
-> F0 00 28 00 00 00 00 %MRB Read Transaction to read the response
<- FF FF FF E2 45 F2 C5 %SC1905 4-byte response 0xF2C5 = 62149
-> D5 81 28 00 %CHK Read Command
<- FF FF FF 31 %CHK = 0x31 is read
% CHK computation. Mod256(0xF0+0xE2+ 0x45+0xF2+0xC5)=0xCE. CHK = 0xFF-0xCE = **0x31**
RFFB_PMU_bytes = 62149
% RFFB RMS Power (dBm/40ms) over a 40ms measurement window. Updated every 300ms
% See section 6.1.3 for conversion and sections 8.6 and 8.12 for MATLAB example code
RFFB.RMS= 3.01*Read16B_signed_Scratch(RFFB_PMU_bytes)/1024 = -9.9559dBm

3.6.10. To Read RFIN and RFFB AGC Values

RFIN_AGC_PDET=rfpal_msgCmdRead(h, hex2dec('23C'), 0) %Address 0x23C = 572
-> D5 81 20 81 %Write CHK = 81
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %0F is read from RSR before sending Command. Waiting for F0 for response to be ready
-> F0 00 20 42 3C 00 00
-> C8 00 28 00
<- FF FF FF F0
-> F0 00 28 00 00 00 00
<- FF FF FF C2 3C 08 00
-> D5 81 28 00
<- FF FF FF 09
RFIN_AGC_PDET = 8

RFFB_AGC =double(rfpal_msgCmdRead(h, hex2dec('9C4'), 0)) %Address 0x9C4 = 2500
-> D5 81 20 F2 %Write CHK = F2
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %0F is read from RSR before sending Command. Waiting for F0 for response to be ready
-> F0 00 20 49 C4 00 00
-> C8 00 28 00
<- FF FF FF F0
-> F0 00 28 00 00 00 00
<- FF FF FF C9 C4 1D 00
-> D5 81 28 00
<- FF FF FF 65
RFFB_AGC = 29

4. Reprogramming the EEPROM

IMPORTANT: *To reprogram the EEPROM with updated firmware and new customer configuration parameters, it is important to know the EEPROM mapping, as described in Table 5, as the firmware download must start at address 0x0000 and not go over 0XDFFF. Additionally, the EEPROM addresses for customer configuration parameters are listed in Table 6.*

The same EEPROM read and write instructions described in sections 4.1.10 and 4.3 are used to upload new firmware or update the customer configuration parameters.
See Microchip 25A512 data sheet for additional details on the EEPROM inside the SC1905.

IMPORTANT: *The number of EEPROM erase/write cycles is limited to one million.*

4.1. EEPROM Mapping and Customer Configuration Parameters

Table 5: EEPROM Mapping

EEPROM Addressed (Hex)	Description
0000-DFFF	Download firmware, starting at address 0x0000 **Note: Firmware size may be smaller.** ***IMPORTANT:*** *Do not write in the range: (end of firmware):0xDFFF*
E000-F77F	Reserved. **IMPORTANT: Do not write in this range**
F780-F7FF	ATE Calibration Parameters. See 4.1.9 for details **IMPORTANT: Do not write in this range**
F800-FBFF	Reserved. **IMPORTANT: Do not write in this range**
FC00-FFFF	Customer configuration parameters. See Table 6 for details.

Table 6: EEPROM Addresses for Customer Configuration Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
FC00	UINT16	MinFrequencyScan	Freq Range Minimum Bound: 16-bit value of 2*MHz value of 2 x MinFrequencyScan (MHz)
FC02	UINT16	MaxFrequencyScan	Freq Range Maximum Bound: 16-bit value of 2 x MaxFrequencyScan (MHz)
FC04	UINT8	Frequency Range	Frequency Range Option: 09: 3300MHz-3800MHz 08: 2700MHz-3500MHz 07: 1800MHz-2700MHz 06: 698MHz-2700MHz 05: 1040MHz-2080MHz 04: 698MHz-1040MHz
FC05 FC0F		Reserved	Reserved (DO NOT CHANGE VALUES)
FC10	UINT8	SemMeasBW_MHz	See Table 11 and Table 25 for details.
FC11	INT8	LowerSemFreqA_MHz	See Table 11 and Table 25 for details.
FC12 FC14		Reserved	Reserved (DO NOT CHANGE VALUES)
FC15	UINT8	Duty Cycle Feedback Mode	Adapt Mode 00 = Duty Cycle Feedback OFF (Default State) 01 = Duty Cycle Feedback ON (Not recommended for TDD applications)
FC16		Reserved	Reserved (DO NOT CHANGE VALUE)
FC17	INT16	RFFB Reference Offset	RFFB PMU Reference offset in dBN. dBm = 3.01*dBN/1024. See Table 21 for details.
FC19	INT16	RFIN Reference Offset	RFIN PMU Reference offset in dBN. dBm = 3.01*dBN/1024. See Table 21 for details.
FC1B	INT16	MaxPWRCalParameter1A (RFFB Max PWR A)	16-bit signed value of maximum Power Amplifier output power calibration parameter 1 at frequency A. Maximum RFFB RMS power level for the smooth mode calibration at frequency A. Used by the GUI to determine the operation Mode: 00 = Optimized Correction Mode. Other values = Smooth Adaptation Mode.
FC1D	UINT8	MaxPWRCalParameter2A (RFIN AGC Index A = PDET)	8-bit value of maximum Power Amplifier output power calibration parameter 2 at frequency A. Coarse PDET Attenuation Index between 0 and 15. See section 4.1.10 for PDET temperature compensation.
FC1E	INT16	MaxPWRCalParameter3A (IC Temp A)	16-bit value of maximum Power Amplifier output power calibration parameter 3 (IC Temp) at frequency A
FC20	UINT8	MaxPWRCalParameter4A (Corr VGA Index A)	8-bit value of maximum Power Amplifier output power calibration parameter 4 at frequency A CORR VGA Index. Valid values are {0, 1, 2, 3}.
FC21	UINT8	MaxPWRCalParameter5A (PDET DC offset DAC Index A)	8-bit value of maximum Power Amplifier output power calibration parameter 5 at frequency A. PDET DC offset DAC. Integer between 0 and 15.
FC22		Reserved	Reserved (DO NOT CHANGE VALUES)
FC23	UINT8	TDD Duty Cycle Factor %	Duty cycle of TDD waveform for PMU measurements. See Table 21 for details.

EEPROM @ (Hex)	Size	Variable Name	Description
FC24	UINT8	PDET Temperature Compensation Flag	8-bit value of PDET Temperature Compensation Flag 0 = Enabled (Default). PDET is adjusted based on internal gain fluctuation over temperature 1 = Disabled. See section 4.1.10 for PDET temperature compensation details.
FC25 FC2E		Reserved	Reserved (DO NOT CHANGE VALUES)
FC2F	UINT8	Upper Freeze Threshold	Adaptation Upper Freeze Threshold. Default 2 for 6dB. See section 3.2.1 for details.
FC30	UINT8	Lower Freeze Threshold	Adaptation Lower Freeze Threshold. Default 5 for 15dB. See section 3.2.1 for details.
FC31 FC36		Reserved	Reserved (DO NOT CHANGE VALUES)
FC37	UINT8	MaxPWRCalParameter6A	8-bit value of maximum Power Amplifier output power calibration parameter 6 at frequency A Fine PDET Attenuation Index between 0 and 15.
FC38	UINT8	MaxPWRCalParameter7A	8-bit value of maximum Power Amplifier output power calibration parameter 7 at frequency A
FC39 FC50	24 UINT8	MaxPWRCalParameter8A	24 8-bit values of maximum Power Amplifier output power calibration parameter 8 at frequency A
FC51	INT16	MaxPWRCalParameter9A (RFIN Max PWR A)	16-bit signed value of maximum Power Amplifier output power calibration parameter 9 at frequency A
FC53	UINT16	MaxPWRCalParameter10A (Center Freq A)	16-bit signed value of maximum Power Amplifier output power calibration parameter 10 at frequency A; LO Freq(MHz) = MaxPWRCalParameter10A*0.5MHz
FC55	INT16	MaxPWRCalParameter1B (RFFB Max PWR B)	16-bit signed value of maximum Power Amplifier output power calibration parameter 1 at frequency B
FC57	UINT8	MaxPWRCalParameter2B (RFIN AGC Index B)	8-bit value of maximum Power Amplifier output power calibration parameter 2 at frequency B Coarse PDET Attenuation Index between 0 and 15.
FC58	INT16	MaxPWRCalParameter3B (IC Temp B)	16-bit value of maximum Power Amplifier output power calibration parameter 3 at frequency B
FC5A	UINT8	MaxPWRCalParameter4B (Corr VGA Index B)	8-bit value of maximum Power Amplifier output power calibration parameter 4 at frequency B CORR VGA Index. Valid values are {0, 1, 2, 3}.
FC5B	UINT8	MaxPWRCalParameter5B (PDET DC offset DAC Index B)	8-bit value of maximum Power Amplifier output power calibration parameter 5 at frequency B. PDET DC offset DAC. Integer between 0 and 15.
FC5C	INT16	MaxPWRCalParameter9B (RFIN Max PWR B)	16-bit signed value of maximum Power Amplifier output power calibration parameter 9 at frequency B
FC5E	UINT16	MaxPWRCalParameter10B (Center Freq B)	16-bit signed value of maximum Power Amplifier output power calibration parameter 10 at center frequency B. 16-bit value of 2*MHz value of 2xCenter Frequency B (MHz)
FC60	UINT8	PDET PA Gain Compensation Flag	8-bit value of PDET PA Gain Compensation Flag 0 = Enabled (Default), 1 = Disabled. PDET is adjusted based on PA gain fluctuation over temperature. It is assumed that the PA output AGC is done before RFIN. See section 4.1.10 for PDET PA Gain compensation details.
FC61		Reserved	Reserved (DO NOT CHANGE VALUES)

EEPROM @ (Hex)	Size	Variable Name	Description
FC62	UINT8	Guard Band	Configurable guard band to make sure in-band signal doesn't get used for coefficient adaptation as this will negatively impact the performance. See Table 11 for details
FC63	UINT8	MaxPWRCalParameter6B	8-bit value of maximum Power Amplifier output power calibration parameter 6 at frequency B Fine PDET Attenuation Index between 0 and 15.
FC64	UINT8	MaxPWRCalParameter7B	8-bit value of maximum Power Amplifier output power calibration parameter 7 at frequency B
FC65 FC7C	24 UINT8	MaxPWRCalParameter8B	24 8-bit values of maximum Power Amplifier output power calibration parameter 8 at frequency B
FC7D FCAE	50 INT8	MaxPWRCalCoeffA	50 8-bit values of maximum Power Amplifier output power calibration Coefficients at frequency A
FCAF FCE0	50 INT8	MaxPWRCalCoeffB	50 8-bit values of maximum Power Amplifier output power calibration Coefficients at frequency B
FCE1 FCEC		Reserved	Reserved (DO NOT CHANGE VALUES)
FCED	UINT8	PLL Ref Divider	See Table 8 for details
FCEE	UINT8	PLL Output Divider	See Table 8 for details
FCEF	UINT8	PLL Feedback Divider	See Table 8 for details
FCF0	INT8	LowerSemFreqB_MHz	See Table 11 and Table 25 for details.
FCF1	INT8	UpperSemFreqA_MHz	See Table 11 and Table 25 for details.
FCF2	INT8	UpperSemFreqB_MHz	See Table 11 and Table 25 for details.
FCF3	INT16	SemB_HighThrsld	See Table 11 and Table 25 for details.
FCF5	UINT16	Power Change Detection Integration Time	Power change detection integration time in 50 microsecond units. Default = 1000 (50ms). See section 1 for details.
FCF7 FD3A		Reserved	Reserved (DO NOT CHANGE VALUES)
FD3B	UINT8	CCDF Mode	See Table 22 for details.
FD3C FD5D		Reserved	Reserved (DO NOT CHANGE VALUES)
FD5E	UINT8	Linearizer Operation Mode	Linearizer Operation Mode. Unsigned 8-bit value. See Table 11 for details. = 0: Normal Cost Function. = 1: Use SEM Range A and range B defined above = 2: Use SEM ranges A and B with weighting factors
FD5F FD93		Reserved	Reserved (DO NOT CHANGE VALUES)
FD94	UINT8	Lower NOOB Weight Factor	NOOB/FOOB ratio for lower side of carrier. See Table 11 for details.
FD95	UINT8	Upper NOOB Weight Factor	NOOB/FOOB ratio for upper side of carrier. See Table 11 for details.
FD96 FD9E		Reserved	Reserved (DO NOT CHANGE VALUES)
FD9F	UINT16	Power Change Detection Delta	Power change detection delta in 0.25 dB units. Default = 3 (0.75 dB). See section 1 for details.
FDA1	UINT8	Duty Cycle FSA Enable Flag	0: Duty Cycle is enabled at the end of CAL. > 0: duty cycle is enabled 10 seconds after entering TRACK.

EEPROM @ (Hex)	Size	Variable Name	Description
FDA2 FFA3		Reserved	Reserved (DO NOT CHANGE VALUES)
FDA4	UINT16	Power Step Down Iteration Count	Aggressiveness of reaction on power steps down. 0 = default behavior is 400 iterations of FSA2. >0: Number of adaptation iterations. See section 4.1.5 for details.
FDA6 FDA8		Reserved	Reserved (DO NOT CHANGE VALUES)
FDA9	UINT16	Power Step Up Iteration Count	Aggressiveness of reaction on power steps up. 0 = default behavior is 400 iterations of FSA2. >0: Number of adaptation iterations. See section 4.1.5 for details.
FDAB		Reserved	Reserved (DO NOT CHANGE VALUES)
FDAC	UINT8	GaN PA Mode enable	Enables special behavior to optimize performance with GaN PAs 0 = LDMOS PAs (Default) >0: GaN Pas. Firmware parameters optimized for GaN PAs. See section 3.2.1 for details.
FDAD FDB2		Reserved	Reserved (DO NOT CHANGE VALUES)
FDB3	UINT8	ATE Calibration Offset Zone Written	Parameter used to determine if ATE calibration Offsets were written in EEPROM = 0xA5: Cal offset data written to EEPROM Otherwise: data is not in EEPROM
FDB4 FFFE		Reserved	Reserved (DO NOT CHANGE VALUES)
FFFF	UINT8	Checksum	Checksum = Modulo256(SUM(FC00:FFFE))

IMPORTANT

- *16-bit values are little-endian.*
- *Address 0xFFFF checksum = Modulo256(SUM(FC00:FFFE))*
  If the checksum does not match, the firmware will issue an error 3

4.1.1. Frequency Range Configuration

Table 7: SC1905 Frequency Ranges

Frequency Range	Guaranteed Frequency Ranges[1] GUI Frequency Band Select Options		Frequency Ranges Available for Experimental Testing	
Range Index	Min Freq	Max Freq	Min Freq	Max Freq
04	698	1040	520	1040
05	1040	2080	1040	2080
06	698	2700	520	3049
07	1800	2700	1616	3049
08	2700	3500	2666	4200
09	3300	3800	3191	4200

1. *Operation outside these frequency ranges is not guaranteed.*

4.1.2. External Clock Configuration

Table 8 defines the different EEPROM parameters that need to be configured to support the following clock standard system clock rates: 10MHz, 13MHz, 15.36MHz, 19.2MHz, 20MHz, 26MHz, and 30.72MHz.

Table 8: External Clock Configuration EEPROM Parameters

EEPROM at (Hex)	Size	Variable Name	Description
FCED	UINT8	PLL Ref Divider	PLL Reference Divider to be configured for external clock with system clock rates = 10MHz, 13MHz, 15.36MHz, 19.2MHz, 20MHz, 26MHz, and 30.72MHz. See Table 9 for detailed configuration.
FCEE	UINT8	PLL Output Divider	PLL Output Divider to be configured for external clock with system clock rates = 10MHz, 13MHz, 15.36MHz, 19.2MHz, 20MHz, 26MHz, and 30.72MHz. See Table 9 for detailed configuration.
FCEF	UINT8	PLL Feedback Divider	PLL Feedback Divider to be configured for external clock with system clock rates = 10MHz, 13MHz, 15.36MHz, 19.2MHz, 20MHz, 26MHz, and 30.72MHz. See Table 9 for detailed configuration.

Table 9 defines the different configuration values for these parameters.

Table 9: External Clock Configuration values

Clock Rate	PLL Ref Divider	PLL Output Divider	PLL Feedback Divider
10	20	2	14
13	26	4	16
15.36	31	2	32
19.2	38	2	38
20	0	0	0
26	52	4	40
30.72	62	2	48

For certain reference clock frequencies, the carrier center frequency is scaled. There are two scratch variables that report the center frequency: one is unscaled, and the other is scaled. For reference clock frequencies which do not require scaling, these variables will have the same value. The GUI reads the unscaled center frequency and displays it. However, this may be misleading or confusing to customers in cases where the scaled frequency is different.

Table 10: Scratch center Frequency Parameters

Scratch Address (Hex)	Size/Access	Variable Name	Description
01A	16-bit R	Unscaled Center Frequency	XX YY hexadecimal value of 2 x Unscaled Center Frequency (MHz). For example, XX YY = 0F A3 corresponds to the signal center Frequency = 2001.5MHz.
BA8	16-bit	Scaled Center Frequency	The center frequency is scaled to support various external XO reference clock. XX YY hexadecimal value of 2 x ScaledFrequency (MHz). For example, XX YY = 0F A3 corresponds to the scaled signal center Frequency = 2001.5MHz

4.1.3. Wideband Optimization Customer Configuration Parameters

The following parameters can be used to optimize wideband performances.

Table 11: Wideband Performance EEPROM Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
FC10	UINT8	SemMeasBW_MHz	2*BandWidth over which spectral emission is measured for adaptation. Unsigned 8-bit value.
FC11	INT8	LowerSemFreqA_MHz	2*Lower Offset A in MHz from the lower edge of the signal for adaptation. Signed 8-bit value.
FC62	UINT8	Guard Band	Frequency band between the carrier edge (defined by the -24dBc signal bandwidth) and the IMD measurement region used for adaptation. 0 = 20% of the signal bandwidth is used (Default) Other value X =~ X*0.5MHz is used (1 = 0.5MHz, 2 = 1MHz, 3 = 1.5MHz, 4 = 2MHz, etc…) Signal bandwidth is defined by the -24dBc points. The actual in-band signal bandwidth might be wider and it is critical to careful configure the guard band to avoid using in-band signal for the adaptation as this will negatively impact the performance.
FD94	UINT8	Lower NOOB Weight Factor	NOOB/FOOB ratio for lower side of carrier
FD95	UINT8	Upper NOOB Weight Factor	NOOB/FOOB ratio for upper side of carrier
FCF0	INT8	LowerSemFreqB_MHz	2*Lower Offset B in MHz from the Lower edge of the signal for adaptation. Signed 8-bit value.
FCF1	INT8	UpperSemFreqA_MHz	2*Lower Offset A in MHz from the Upper edge of the signal for adaptation. Signed 8-bit value.
FCF2	INT8	UpperSemFreqB_MHz	2*Lower Offset B in MHz from the Upper edge of the signal for adaptation. Signed 8-bit value.
FD5E	UINT8	Linearizer Operation Mode	Linearizer Operation Mode. Unsigned 8-bit value. = 0: Normal Cost Function. = 1: Use SEM Range A and range B defined above. = 2: Use SEM ranges A and B with weighting factors

By default (value of “0”), the Guard Band is set to 20% of the signal bandwidth. This default configuration is optimal for IMD5 performance optimization of contiguous carriers with signal bandwidth greater than 40MHz. For non-contiguous carriers or for close-in IMD optimization, it is recommended to try different options.

With Linearizer Operation Mode = 1, Table 11 and Figure 9 illustrate a setting example for the SEM parameters with two LTE 10 MHz waveform separated by 60MHz.

Linearizer Operation Mode = 2 uses the SEM ranges A and B as with linearizer operation mode = 1, but adds the use of weighting factors. The range of distortion defined by the A SEM parameters (e.g., UpperSemFreqA_MHz) is referred to as Near Out of Band distortion (NOOB). The range of distortion defined by the B SEM parameters (e.g., UpperSemFreqB_MHz) is referred to as Far Out Of Band distortion (FOOB). It is possible to weight the ratio of NOOB/FOOB so as to cause the SC1905 to favor correction of NOOB over FOOB. In Linearizer Operation Mode = 0 or 1, all distortion is weighted equally and the linearizer does not try harder to correct distortion in some regions than others. There may be cases where one wants the linearizer to put more effort into correcting distortion very close to the carrier to meet some mask specification, for example. Or, one may want the linearizer to focus on just one side of the carrier and ignore the other side; for example, if a filter present in the system means that linearization is only required on one side of the carrier. The NOOB weight factor parameters provide this flexibility. The default value of the

NOOB weight factors is 40. Hence the NOOB is weighted 40X more heavily than the FOOB. This effectively causes the linearization algorithm to ignore the FOOB.
Note that even if Linearizer Operation Mode = 1 or 2, the linearizer acts as if the mode is 0 if adaptation is in the FSA0 or FSA1 stages. The mode setting of 1 or 2 will take effect during FSA2 and TRACK.

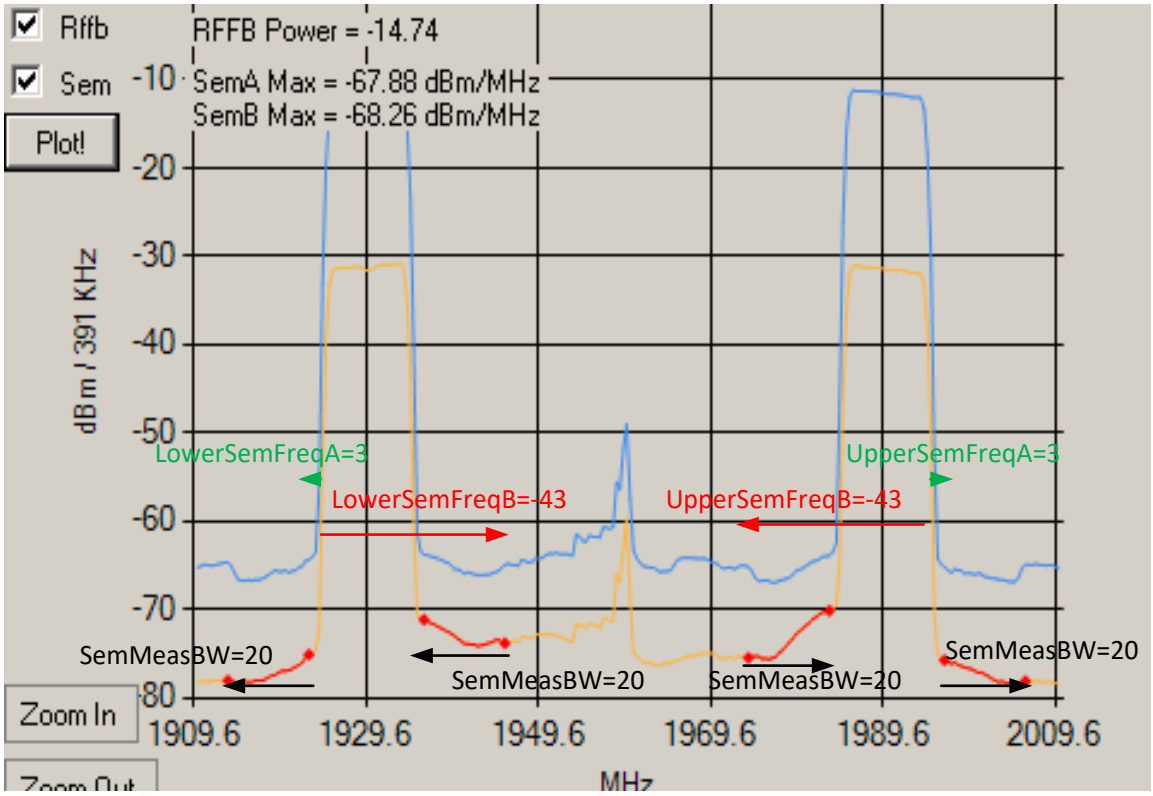


Figure 9: Setting for Wideband Mode for Two Separated LTE 10MHz Carriers

Table 12: SEM Parameters or Wideband Mode or Two Separated LTE 10 MHz Carriers

Variable Name	Value
SemMeasBW_MHz	20 = 10MHz
LowerSemFreqA_MHz	3 = 1.5MHz
LowerSemFreqB_MHz	-43 = -21.5MHz
UpperSemFreqA_MHz	3 = 1.5MHz
UpperSemFreqB_MHz	-43 = -21.5MHz

4.1.4. Meeting Spectral Emission Limits Very Close to Carrier

If a particular SEM specification requires that distortion very close to the carrier be reduced more than it is with the default settings, then use of the Wideband Optimization customer configuration parameters may help to achieve the required specification. One example is the so-called FCC Band 41 Block Edge specification which requires spectral emissions be below -13dBm/MHz at a point 1MHz from the edge of the carrier.

With a relatively wideband carrier such as 20MHz, changing the Guard Band parameter from the default setting may help. The default setting is for the guard band region to be 20% of the carrier bandwidth. For 20MHz carriers, this means that the distortion in the 4MHz region on either side of the carrier is ignored. There will be some reduction of the distortion at 1MHz offset due to the linearizer acting on the IM3 distortion that it is considering, but potentially more reduction can be achieved at the 1MHz offset point by reducing the guard band region. It is not a good idea to use a value of 1 since carrier power may be inadvertently included in the distortion, but a value of 2 should be safe. Each unit represents approximately 0.5MHz. The guard band is illustrated in Figure 10.

Guard Band
1. Used in CAL and FSA1 for linearize mode 1 and 2
2. Used in CAL, FSA and Track in linearize mode 0

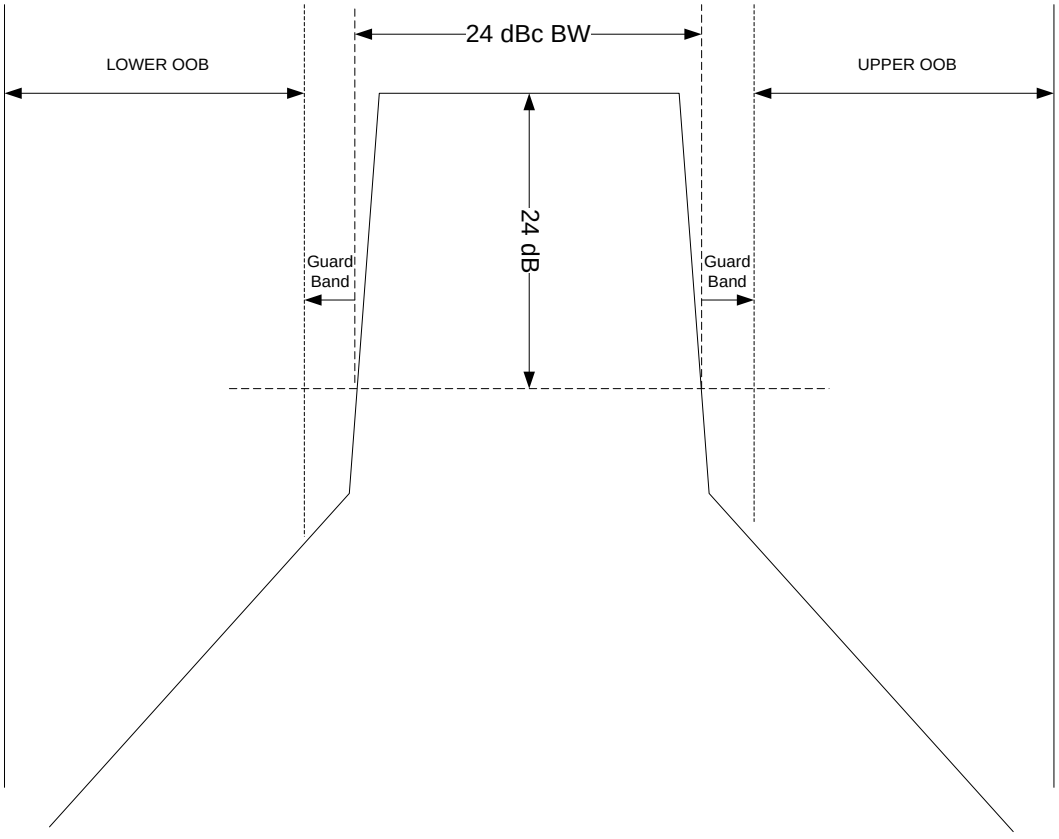


Figure 10: Guard Band

For a narrowband carrier, such as 5MHz, the carrier power is now contained within ¼ the bandwidth meaning the ACLR1 has to be 6dB better than for the 20MHz carrier to achieve the same absolute power density at a 1MHz offset. For a narrowband carrier, changing the Guard Band parameter will likely not help, but the Linearizer Operation Mode setting of 2 may help. Setup the NOOB and FOOB regions using the SEM parameters and experiment with the NOOB weighting factors to heavily weight the NOOB. Figure 11 illustrates the regions of spectrum that are defined as NOOB and FOOB. NOOB and FOOB both have a width of 2*SemMeasBW_MHz (MHz).
The Upper NOOB starting frequency is defined by the 2*UpperSemFreqA_MHz parameter and the Upper FOOB starting frequency is defined by the 2*UpperSemFreqB_MHz parameter.
The Lower NOOB starting frequency is defined by the 2*LowerSemFreqA_MHz parameter, and the lower FOOB starting frequency defined by the 2*LowerSemFreqB_MHz parameter.

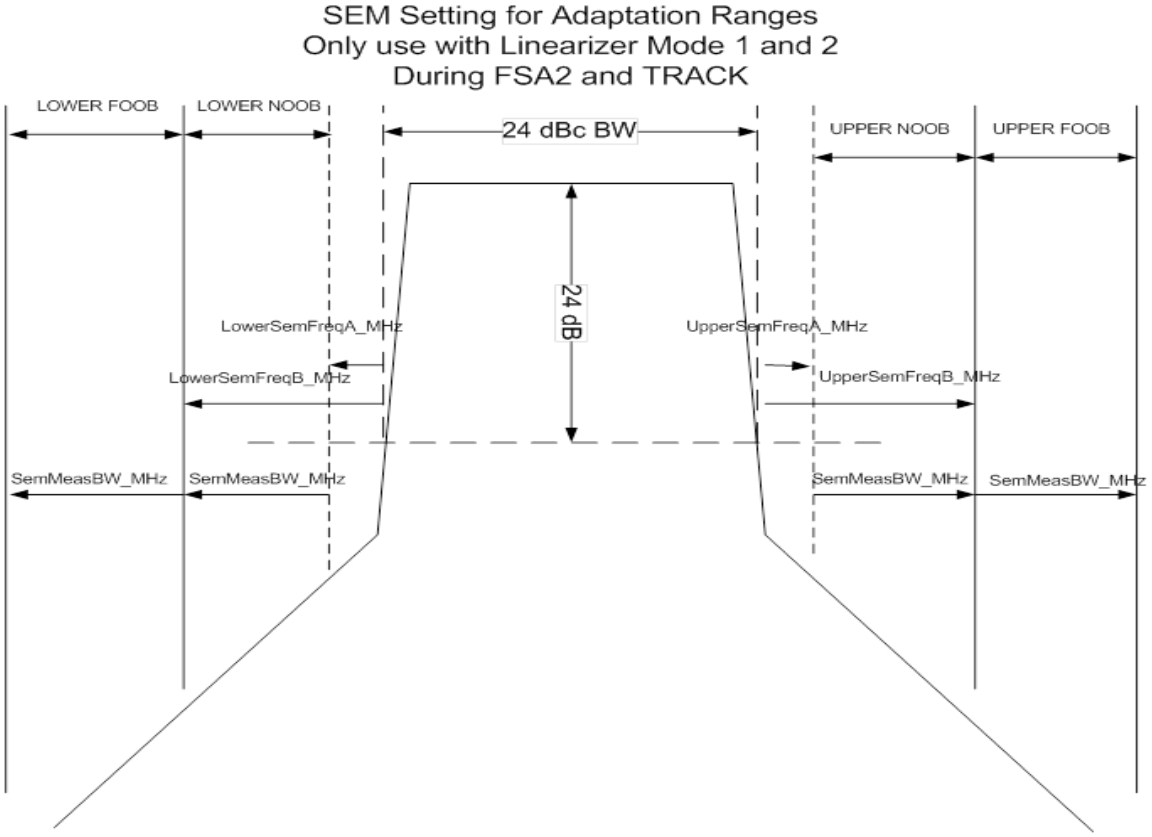


Figure 11: NOOB and FOOB Definitions

The starting frequency of NOOB defines the guard band during the period when linearize mode 0 is not being used. So during the period that linearize mode 1 or 2 is used, the Guard Band parameter is ignored. It is of course necessary to set the NOOB starting frequency close to the carrier if one is trying to reduce close-in distortion. Ultimately, the customer will need to determine empirically what settings give the best results in their particular setup, but a suggested starting point that gives good performance with both 5MHz and 20MHz LTE carriers is as follows:

SemMeasBW_MHz = 10 (5MHz)
LowerSemFreqA_MHz = 2 (1MHz)
LowerSemFreqB_MHz = 12 (6MHz)

UpperSemFreqA_MHz = 2 (1MHz)
UpperSemFreqB_MHz = 12 (6MHz)
Linearizer Operation Mode = 2
Guard Band = 2
Upper NOOB Weight Factor = 0 (use default of 40)
Lower NOOB Weight Factor = 0 (use default of 40)

When measuring compliance against an SEM spec, make sure the spectrum analyzer noise floor is not affecting the measurement. Set the attenuation in the front end of the SA to the minimum value that can be used given the dynamic range of the signal. Make use any preamp in the front end of the SA if available to further help with the instrument noise floor.

4.1.5. **Aggressiveness of Re-adaptation on Power Steps**

It was determined that for some PA's the re-adaptation on power steps (up or down) needs to be more aggressive in order to avoid being trapped in local minima of the cost surface.
As shown in Figure 12, the adaptation iteration counter represents the adaptation state:

- 0 < Counter < 300: CAL state
- 300 ≤ Counter < 1100: FSA1 state
- 1100 ≤ Counter < 2100: FSA2 state
- 2100 ≤ Counter: TRACK state

From CAL to TRACK, the adaptation goes from the most aggressive to the least aggressive.
The two EEPROM parameters defined in Table 13 allows setting the adaptation iteration counter to an earlier state with more aggressive re-adaptation.

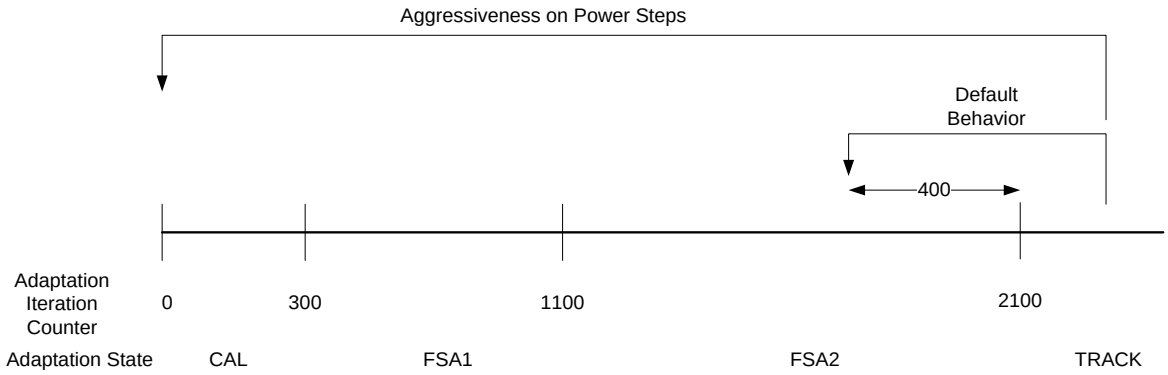


Figure 12: Aggressiveness on Power Steps

Table 13: EEPROM Parameters or Aggressiveness of Re-adaptation n Power Steps

EEPROM @ (Hex)	Size	Variable Name	Description
FDA4	UINT16	Power Step-Down Iteration Count	Adaptation Iteration counter on power steps down. 0 = default behavior is iteration counter of 1700 (so 400 iterations of FSA2) >0: Number of adaptation iteration counter
FDA9	UINT16	Power Step-Up Iteration Count	Aggressiveness of reaction on power steps up. 0 = default behavior is iteration counter of 1700 (so 400 iterations of FSA2) >0: Number of adaptation iteration counter

4.1.6. Power Change Detection Trigger Parameters

The firmware is continuously monitoring the RMS power of the RFFB signal in order to detect a change in the power level. The coefficients for a given power level may be completely invalid for a different power level so if power changes, some action must be taken. This includes running some FSA adaptation iterations, and for power steps up, loading max power coefficients. The trigger for detection of a power change is determined by two factors: the measurement integration time, and the amount of change in power level. The default values are 50ms for integration time, and 0.75dB for the change or delta. In some cases, particularly with LTE waveforms that have very dynamic traffic (resource block utilization), the power change detection mechanism can be triggered so frequently that overall correction performance suffers. It may be beneficial, in such cases, to make the trigger less sensitive. The two power change detection parameters can be used to configure the trigger sensitivity. For example, integrating over 100ms rather than 50ms will make the trigger less sensitive to short-term variations in RB utilization.

Table 14: Power Change Detection Trigger Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
FCF5	UINT16	Power Change Detection Integration Time	Power change detection integration time in 50 microsecond units. Default = 1000 (50ms).
FD9F	UINT16	Power Change Detection Delta	Power change detection delta in 0.25dB units. Default = 3 (0.75dB).

4.1.7. Lower Freeze Threshold

There is an EEPROM parameter provided for controlling over what range of PA output power the SC1905 actively adapts its linearization coefficients. This is the Lower Freeze Threshold. This parameter is defined relative to the max calibrated power level and is in units of 3dB. For example, a value of 5 corresponds to a 15dB backoff with respect to the max calibrated power. If the PA output power is below the Lower Freeze Threshold, then adaptation is immediately frozen. Frozen adaptation means that the linearization coefficients are fixed at their last values. This means, for example, that if the SC1905 is reset when the PA output power is below the lower freeze threshold, the adaptation will not start, the coefficients will therefore be stuck at their initial zero values and no correction of the PA distortion will occur.

The reason for including the Lower Freeze Threshold is that with LDMOS transistors, the non-linear distortion usually decreases significantly the more the PA output power is reduced. At more than 15dB backoff, the distortion is usually low enough that the linearization coefficients become very small. There is no real need to continue adapting them with so little distortion present. Freezing the adaptation also eliminates any fluctuation in the corrected distortion. With GaN

transistors, the distortion at backoff can be still high and there is usually a need to actively adapt at lower power levels than is necessary with LDMOS transistors. For this reason, when GaN PA Mode is enabled, the default setting for the Lower Freeze Threshold is changed to 7 from 5 lowering the level to -21dB from -15dB.

4.1.8. GaN PA Mode Optimization

A lot of work was done with GaN PA and the two most significant differences between GaN and LDMOS PAs as far as firmware operation is concerned are

1. Increasing non-linearity at high back-off
2. Gain expansion in back-off.

Whereas LDMOS PAs typically have very little non-linearity at high back- off (> 15dB from P_{MAX}), to the extent that SC1905 output could be disabled, GaN PA can have significant distortion at such high back-offs, possibly more than at higher power levels. Furthermore, for LDMOS PAs, the gain of the PA is independent of output power level, at least to a first approximation, but it is common to see the gain of a GaN PA increase > 3dB at 10dB back-off for example.

The Volterra series correction signal is generated by the Correction Block (CORR) of the SC1905. A simplified block diagram is shown in Figure 13. The elements affected by GaN PA behavior are shown in red.

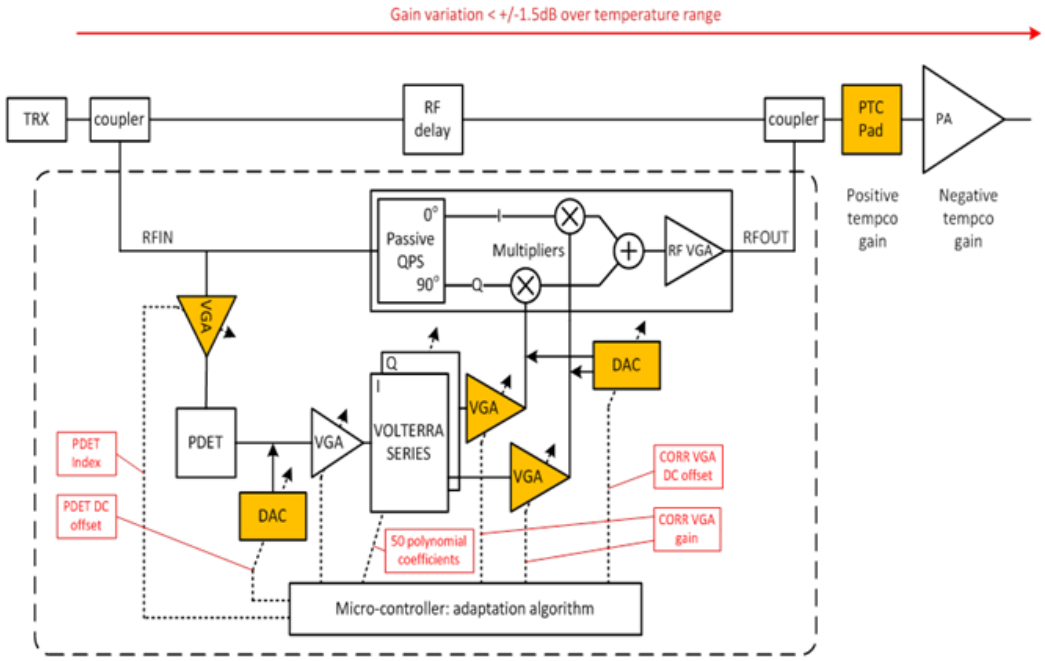


Figure 13: SC1905 Correction Path Block Diagram

To help with GaN PA performance, it is recommended to first enable GaN PA Mode. See Table 15 for details.

Table 15: GaN PA Mode EEPROM Parameter

EEPROM Address (Hex)	Size/Access	Variable Name	Description
FDAC	UINT8	GaN PA Mode Enable	Enables special behavior to optimize performance with GaN PA 0 = LDMOS PAs (Default) >0 : GaN PA. Firmware parameters optimized for GaN PA.

IMPORTANT: *Whenever the state of this parameter is changed between zero and non-zero, it is required to redo the Smooth Mode calibration.*

The following list summarizes the effects of GaN PA Mode Enable = 1 versus LDMOS PA mode (GaN PA Mode Enable = 0)

1. Power Step-Down Iteration Count and Power Step-Up Iteration Count default settings are changed to 301 instead of 1700. See Section 5 for details.
2. Default Lower Freeze Threshold is changed to 7, instead of 5, to enable adaptation at high back-off
3. Default EDET AGC Index Offset is changed to -2 instead of 0.
4. Default dynamic PDET and EDET DC offsets are changed to -30 and -30, respectively, instead of -40 and -70.

4.1.9. **ATE Calibration Parameters**

These ATE calibration parameters will affect the smooth mode calibration parameters. The EEPROM parameters defined in Table 16 and Table 17 are calibrated at ATE.

Table 16: ATE Calibration Offset Zone Written EEPROM Parameter

EEPROM Address (Hex)	Size/Access	Variable Name	Description
FDB3	UINT8	ATE Calibration Offset Zone Written	Parameter used to determine if ATE calibration Offsets were written in EEPROM = 0xA5: Cal offset data written to EEPROM Otherwise: data is not in EEPROM

Table 17: EEPROM Addresses for ATE Calibration Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
F780-F7B3		Reserved	Reserved (DO NOT CHANGE VALUES)
F7B4	UINT16	Temperature Offset	Temperature Offset 0= default = 283 >0: Offset value If withTemp Offset=0, the IC temperature is Temp_IC while the expected Temp IC is Exp_Temp_IC, then set Temp_Offset = 283 + Temp_IC - Exp_Temp_IC
F7B6-F7FF		Reserved	Reserved (DO NOT CHANGE VALUES)

4.1.10. Smooth Mode Temperature and Gain Compensation Discussion

This section describes the Custom configuration parameters in the EEPROM that can be adjusted to compensate for PA gain and RFPAL internal gain variation over temperature. The EEPROM addresses of these different parameters are in found in Table 6.

IMPORTANT: *These temperature compensation methods only apply to Smooth Mode operation (smooth mode system calibration is performed in the factory at maximum power). SC1905 internal temperature compensation algorithms assume that the PA output power is held constant by changing the RFIN input signal using the transmit system modulator ALC or transceiver ALC.*

The PDET (power detector) circuit generates a signal which is proportional to the instantaneous power of the envelope. The peak voltage out of the PDET has a big impact on correction performance of the RFPAL. An attenuator within the PDET is controlled by an AGC loop to ensure that the peak voltage stays within the desired range as the RFIN level or temperature changes. The setting of this attenuator is referred to as the PDET Index. The initial PDET Index is determined during the initial calibration step by running the PDET AGC. Over temperature, PA gain and RFPAL internal gain will vary. Depending on these variations, the PA system AGC loop will adjust the RFIN level to maintain constant PA output power. As the RFIN level increases, the optimal PDET index needs to increase.
Using optimized correction mode, changes in either temperature or RFFB power level can trigger a re-calibration of the PDET Index. When the SC1905 is actively predistorting the PA, this PDET recalibration process will result in some transient, sometimes called a pop-up. Using smooth adaptation mode, the fixed PDET index value (stored during maximum power calibration) can be gradually adjusted with any temperature and PA gain variation. There are three flag bytes described in Table 18 that control the PDET compensation in smooth adaptation mode.

Table 18: PDET Compensation Flags

PDET Temperature Compensation (Enable = 0)	PDET PA Gain Compensation (Enable = 0)	PDET Compensation Mode Description
1	1	Disabled. PDET is fixed regardless of temperature and system gain variation. No PA Gain Compensation.
0	1	PDET is compensated for RFPAL internal temperature variation. The gain variation of the PDET circuit over temperature is stored in a lookup table which is used to keep the output level of the PDET constant over temperature.
0	**0**	**Default, Recommended Setting.** PDET is compensated both for RFPAL temperature variation and system gain variation.

IMPORTANT: *Only use the settings included in this table.*

4.1.11. PDET Temperature Compensation Disabled

- PDET Temp Compensation Flag
    - 1 = Disabled. Smooth Mode PDET constant over all conditions
- PDET PA Gain Compensation Flag
    - 1 = Disabled. Smooth Mode PDET constant over PA gain variation

Why disable the PDET temperature compensation? When the PDET Index is adjusted for temperature variations, short degradations in ACLR correction may occur. For some applications, these short degradations in ACLR correction are not acceptable. Therefore, set the "PDET Temperature Compensation Flag" to '1' to disable the PDET index temperature compensation. Setting the PDET PA Gain Compensation Flag to '1' will disable any adjustment of the PDET index based on system gain.
Keeping both flags set to '1' will hold the smooth mode calibrated PDET value constant over all conditions. The tradeoff is that the correction performance may degrade at extreme temperatures. This amount of degradation is a function of the PA's gain and P1dB over temperature within the temperature range required. If ACLR degrades unacceptably, an alternative method for externally compensating PDET over temperature using NTC attenuators is described in the Hardware Design Guide.

4.1.12. Automatic PDET Temperature Compensation

- PDET Temp Compensation Flag.
    - 0 = Enabled (Default). PDET adjusted based on internal chip temperature variations.
- PDET PA Gain Compensation Flag.
    - 1 = Disabled. No PDET adjustments for PA gain fluctuations.

With PDET Temperature Compensation Flag enabled, the PDET attenuation level is automatically adjusted for internal RFPAL variation over temperature.

4.1.13. Automatic PDET Temperature Compensation with PA Gain Compensation

Recommended configuration

- PDET Temp Compensation Flag.
    - 0 = Enabled (Default). PDET adjusted based on internal chip temperature variations.
- PDET PA Gain Compensation Flag.
    - 0 = Enabled (Default). PDET adjusted based on PA gain fluctuations.

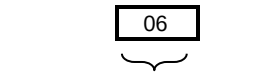
This automatic PDET temperature compensation with PA gain compensation mode is the recommended mode, especially for PA with a large gain variation over temperature. In addition to the automatic internal gain compensation, the SC1905 also monitors the changes in PA gain using RFIN and RFFB power measurements. PDET is then additionally varied to compensate for the change in PA gain. (PA gain variation is assumed to affect the RFIN level almost exclusively as the PA output power is held constant by the system ALC.)

4.2. EEPROM Write Instruction

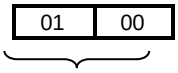
The same procedure is used to write either to the firmware zone or the customer Configuration Parameters zone or the Advance Customer Configuration Parameters. It is recommended to write 64-bytes page at a time.

The following steps must be used for SPI write to EEPROM:

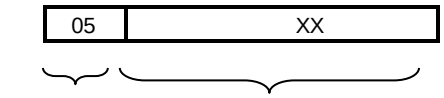
1. Operate the SPI Bus at up to 4MHz.
2. LOADENB (pin 60) needs to be set HIGH ("1") Host is now directly communicating with the embedded EEPROM. See the Microchip 25A512 data sheet for detailed instructions.
3. UNLOCK EEPROM.
   a) Issue a WREN (**0x06**) command to enable write operations to EEPROM



   b) Write zero to STATUS register to unlock: **01 00**



4. Make sure EEPROM is UNLOCKED by Reading STATUS register

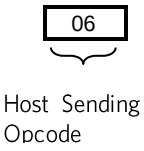


STATUS register (XX) is 8-bit status register (bits 7 MSB to bit 0 LSB).

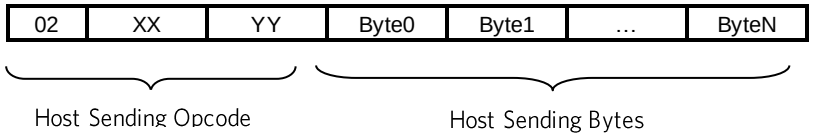
7	6	5	4	3	2	1	0
W/R	—	—	—	W/R	W/R	R	R
WPEN				BP1	BP0	WEL	WIP
W/R = Writable/Readable. R = Read-Only							

- The **Write-In-Process (WIP)** bit indicates whether the EEPROM is busy with a write operation. When set to a '1', a write is in progress, when set to a '0', no write is in progress. This bit is read-only.
- The Block **Protection (BP0 and BP1)** bits indicate if the EEPROM is locked or unlocked.
  - BP1 BP0 = 11 then EEPROM is locked
  - BP1 BP0 = 00 then EEPROM is unlocked

5. Issue a WREN (**0x06**) command to enable write operations to EEPROM



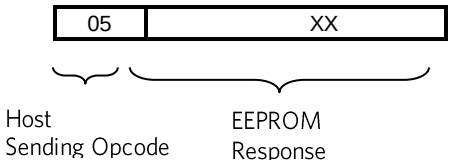
6. Issue a WRITE (0x02) command followed by the 16-bit address to be written followed by the contents to be written into that 64-byte page. If this is the last page, it is acceptable to write less than 64-bytes.
   a) Assert the SPI chip select, SSN which is active low, and begin toggling the SCLK while driving the 24-bit Op-code (02 command + 16 bit EEPROM address) on the SDI pin.
   b) The following N*8 clock edges clock in the N bytes of write data as shown below.
   c) Following bit 0 of the last byte to be written, de-assert SSN



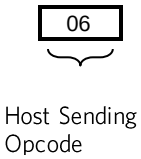
With XX YY 16-bit EEPROM address as described in Table 5 and Table 6.

***IMPORTANT:** Up to 64-bytes can be written with one write instruction. Burst accesses should not cross 128-bytes page boundaries.*

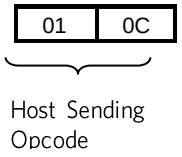
7. Poll the STATUS register until the Write-In-Progress (WIP Bit 0) status changes from '1' (write in progress) to '0' (write completed).



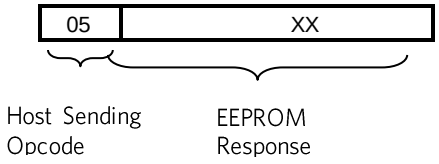
8. Then repeat steps 5 to 7 until all bytes are written.
9. LOCK EEPROM to disable writes to the EEPROM.
   a) Issue a WREN (**0x06**) command to enable write operations to EEPROM



   b) Write "0C" to STATUS register to lock: **01 0C**



10. Make sure EEPROM is LOCKED by Reading STATUS register



XX is 8-bit status register (bits 7 MSB to bit 0 LSB).

- If bit 2 – 3 = 11, then EEPROM is locked
- If bit 2 – 3 = 00, then EEPROM is unlocked

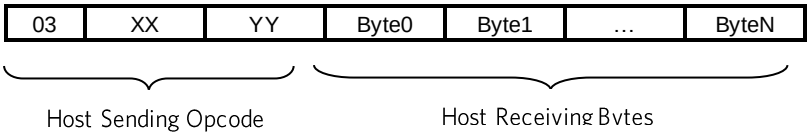
1. LOADENB (pin 60) needs to be set back to low ("0") when done writing to EEPROM.
2. Reset using pin 49 (RESETN)

4.3. EEPROM Read Instruction

The same procedure is used to read either to the firmware zone or the customer Configuration Parameters zone or the Advance Customer Configuration Parameters.

The following steps must be used for SPI read to EEPROM:

1. Operate the SPI Bus at up to 4MHz.
2. LOADENB needs to be set high ("1").
3. Issue a READ (0x03) command followed by the 16-bit address to be read.
   a) Assert the SPI chip-select, SSN, which is active-low, and begin toggling the SCLK while driving the 24-bit Op-code (03 read command + 16-bit EEPROM address) on the SDI pin.
   b) The following N*8 clock edges clock in the N bytes of write data
   c) Following bit 0 of the last byte to be written, de-assert SSN



With XX YY 16-bit EEPROM address as described in Table 5 and Table 6.

4. LOADENB (pin 60) needs to be set back to LOW ("0") when done reading the EEPROM.

IMPORTANT: *No restrictions on Read instructions for Number of bytes to be read with one Read instruction*
Must follow the setup and hold times as well as other timing requirements as described in the data sheet.

4.4. EEPROM Endurance

Table 19 shows the guaranteed number of EEPROM write/erase cycles across worst case supply voltage and temperature range unless otherwise specified.

Table 19: SC1905 EEPROM Endurance

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
EEPROM Write/Erase Cycles			1M			

5. Reading the Cost Function Variable

The cost function is measured at the RFFB input and is a scalar value proportional to ACLR measurement. The magnitude of this scalar will depend on the modulation type. Monitoring the relative change of this scalar will provide a measure of a given PA ACLR.

IMPORTANT: *Averaging of the cost function value is required for more accurate measurements.*

It is recommended to take at least 30 measurements to get more reliable results as follows:

1. Wait for Track. Section 3.2 for details.
2. Wait at least 5 more seconds (Strongly recommended)
3. Freeze the SC1905 adaptation. Section 3.2 for details.
4. Read 16-bit Cost Function variable at address 0x20D:
   a) Execute a 2-byte read at address 525 = 0x20D
      i. MSB at address: 525 = 0x20D
      ii. LSB at address: 526 = 0x20E
5. Unfreeze the SC1905 adaptation. Section 3.2 for details.
6. Wait 0.2s
7. Average the result for improved accuracy by repeating steps 4 to 6 (≥30 iterations).

IMPORTANT: *16-bit values read from SC1905 are in big-endian format.*

To Read 16-bit Cost Variable at 0x20D, the following commands are exchanged over the SPI bus.

Cost_function_bytes = double(rfpal_msgCmdRead(h, hex2dec('20D'), 1))
-> D5 81 20 90 %CHK Write Command
-> C8 00 28 00 %RSR Read Command
<- FF FF FF 0F %Value of 0F
-> F0 00 20 62 0D 00 00 %MRB Write to read 2-byte from @0x20D=525
%CHK computation. 0x62 + 0xD + 0 + 0 = 0x6F = 111. CHK = dec2hex (255 - 111) = 0x90
-> C8 00 28 00 %RSR Read Command
<- FF FF FF F0 %Value of F0. SC1905 response to command is ready
-> F0 00 28 00 00 00 00 %MRB Read
<- FF FF FF E2 0D EA 68 %SC1905 Command response
-> D5 81 28 00 %CHK Read Command
<- FF FF FF CE %CHK from SC1905 Command response
% CHK computation. 0xF0 + 0xE2 + 0x0D + 0xEA + 0x68 = 0x331. Mod256 = 0x31 = 49.
% CHK = dec2hex(255-49) = 0xCE

Cost_function_bytes = 60008 = 0xEA68

Since 0xEA68>7FFF Then Cost = 60008-65536 = -5528

See section 8.5 for MATLAB example code.

6. Debug Features

6.1. Power Measurement Unit (PMU)

6.1.1. PMU Calibration Flow

For absolute accuracy, a one-time, single point calibration of the converted RFIN and RFFB values is required due to dependence on end system characteristics and also on the part-to-part variation of the SC1905 boards. Different reference points are possible for both RFIN and RFFB. It is possible to calibrate the RFIN power level (dBm) into the RFIN coupler or into power amplifier input power level by reading from an external power meter or by applying a known power level. Similarly, the RFFB_PMU value can be calibrated into RFFB Balun or at the power amplifier output level.

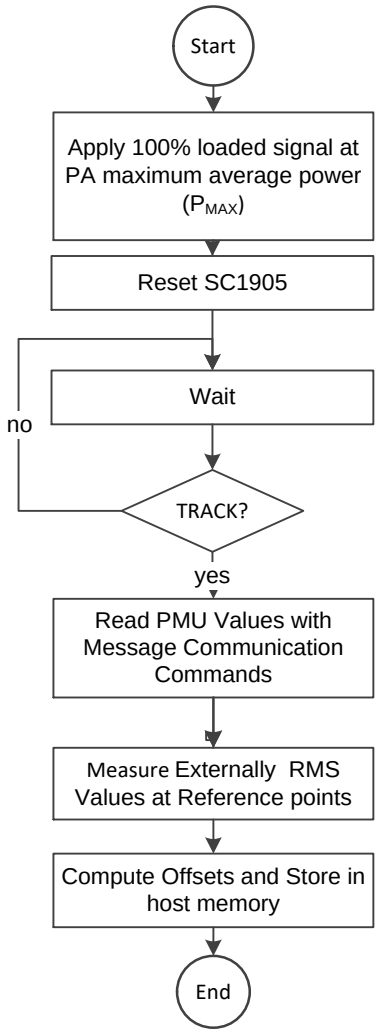


Figure 14: PMU Calibration Flow

6.1.2. PMU Scratch Parameters

See section 8.6 for MATLAB example code.

Table 20: Power Measurement Unit Scratch Parameters

Scratch at (Hex)	Size/Access	Variable Name	Description
037	16-bit R	RFIN_PeakPower_10ns	RFIN_PeakPower_10ns = RFIN Highest 10ns average values over the 40ms average window. See section 6.1.3 and 6.1.4 for conversion to dBm value.
03D	16-bit R	RFFB_PeakPower_10ns	RFFB_PeakPower_10ns = RFFB Highest 10ns average values over the 40ms average window. See section 6.1.3 and 6.1.4 for conversion to dBm value.
047	16-bit R	RFFB_MAX_40us	RFFB_MAX_40us Maximum value over 40µs measurement windows over 40ms. See section 6.1.3 and 6.1.4 for conversion to dBm value.
049	16-bit R	RFFB_MIN_40us	RFFB_MIN_40us Minimum value over 40µs measurement windows over 40ms. See section 6.1.3 and 6.1.4 for conversion to dBm value.
04B	16-bit R	RFIN_MAX_40us	RFFB_MAX_40us Maximum value over 40µs measurement windows over 40ms. See section 6.1.3 and 6.1.4 for conversion to dBm value.
04D	16-bit R	RFIN_MIN_40us	RFFB_MIN_40us Minimum value over 40µs measurement windows over 40ms. See section 6.1.3 and 6.1.4 for conversion to dBm value.
245	16-bit R	RFFB_ RMS	RFFB RMS Power (dBm/40ms) over a 40ms measurement window. Signed 6.10 signed Value. See section 6.1.3 and 6.1.4 for conversion to dBm value.
247	16-bit R	RFIN_ RMS	RFIN RMS Power (dBm/40ms) over a 40ms measurement window. Signed 6.10 signed Value. See section 6.1.3 and 6.1.4 for conversion to dBm value.

IMPORTANT: *Power measurements are updated every 340ms. The measurement time is 40ms and the coefficients are not adapted during that measurement period.*

From the parameters defined in Table 21, it is possible to compute the RFIN and RFFB Peak PAR as follows:
RFIN_Peak_PAR = RFIN_PeakPower_10ns – RFIN_RMS
RFFB_Peak_PAR = RFFB_PeakPower_10ns – RFFB_RMS
See section 8.6 for MATLAB example code.

6.1.3. Conversion of Read PMU values to dBm values

Customers can choose to calibrate the RFIN power level (dBm) into any reference point with the following formula:

$$P_{RFIN}[Reference] = \frac{RFIN\_PMU * 3.01}{1024} + \text{RFIN\_Reference\_Offset}$$

The RFFB power level into any reference point (RFFB (dBm) Balun or the power amplifier output power) can be computed as follow:

$$P_{RFFB}[Reference] = \frac{RFFB\_PMU * 3.01}{1024} + \text{RFFB\_Reference\_Offset}$$

The $OFFSET_{RFIN\_Reference}$ and $OFFSET_{RFFB\_Reference}$ are dependent on end system characteristics and also on the board to board variation with SC1905. So, a one-time calibration is required to determine the offsets.

6.1.4. TDD Considerations—Operation with < 100% Duty Cycle

The SC1889 PMU operates continuously over the measurement window—it does not discard samples which may have been taken when the PA is off. This will affect the reading for waveforms with less than 100% duty cycle as would be seen in TDD applications. For example, the PMU value read for a 50% PA on time (duty cycle) will be 3dB lower than the value with 100% duty cycle. It is straightforward to calculate the PA on time power from the PMU value as described in the following sections.

6.1.4.1. For Systems with a Fixed Rx/Tx Duty Cycle

For systems with a fixed Rx/Tx duty cycle, it is recommended to calibrate the PMU with the procedure above using a waveform with the same Rx/Tx duty cycle as would be seen in the field. This is the preferred method. In this case, duty cycle factor will be included in the computed offsets as described in section 6.1.3.

6.1.4.2. For Systems with a Variable Rx/Tx Duty Cycle

For systems with variable Rx/Tx duty cycle, the host controller can be used to scale the measurement value by the duty cycle (D_{CYCLE}) and calibrate the PMU values with a 100% duty cycle. Then the conversion of read PMU values into dBm values will be as follow:

$$P_{RFIN}[Reference] = \frac{RFIN\_PMU * 3.01}{1024} + \text{RFIN\_Reference\_Offset} - 10 * log10(D_{cycle})$$

$$P_{RFFB}[Reference] = \frac{RFFB\_PMU * 3.01}{1024} + \text{RFFB Reference Offset} - 10 * log10(D_{cycle})$$

NOTE: *For systems that are designed such that they can't operate at 100% duty cycle for thermal reasons, it is possible to calibrate with one duty cycle and then D_{CYCLE} is defined as the scaling duty cycle between the calibrate duty cycle and the current duty cycle.*

RFIN_Reference_Offset and RFFB_Reference_Offset are EEPROM parameters. See Table 21 for details.

6.1.5. PMU EEPROM Parameters

The duty factor can be stored in the SC1905 Customer Configuration Parameters zone in the EEPROM for the SC1905 to perform the scaling. This is not the recommended method. See section 6.1.4 for preferred methods.

Table 21: PMU EEPROM Parameter

EEPROM @ (Hex)	Size	Variable Name	Description
FC17	INT16	RFFB Reference Offset	RFFB Reference offset in dBN. dBm = 3.01*dBN/1024. See section 6.1.3 and 6.1.4 for details.
FC19	INT16	RFIN Reference Offset	RFIN Reference offset in dBN. dBm = 3.01*dBN/1024. See section 6.1.3 and 6.1.4 for details.
FC23	UINT8	TDD Duty Cycle Factor %	8-bit value of PMU Duty Cycle Factor for TDD system: integer value between 0 and 100 that represents the TDD duty cycle in percent. E.g. a value of 60 or 0x3C corresponds to 60%. A value of zero corresponds to 100% duty cycle and is the default value. Hence not programming this byte results in no duty cycle correction being applied (same as if 100d = 0x64 is programmed)

IMPORTANT: *Make sure to update checksum at address 0xFFFF when changing the PMUDutyCycleFactor value.*

6.2. CCDF Parameters

Table 22: CCDF EEPROM Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
FD3B	UINT8	CCDF Mode	CCDF Mode: 0 = Automatic or 1 = Manual. In Automatic mode, firmware will set CCDF1_dB = ~Peak_PAR(dB)-0.25dB, CCDF2_dB = ~Peak_PAR(dB)-1dB CCDF3_dB = ~Peak_PAR(dB)-2dB

IMPORTANT: *Make sure to update checksum at address 0xFFFF when changing the CCDF Mode value.*
In Manual mode, host will need to configure these RFIN_CCDFX_dB and RFFB_CCDFX_dB parameters after each reset.

The CCDF_dB parameters use the format defined in section 6.1.3 with no offset.

Table 23: CCDF Scratch Parameters

Scratch @ (Hex)	Size/Access	Variable Name	Description
51	UINT16 RW	RFIN_CCDF1_dB	RFIN_CCDF1_dB is the threshold 1(dBN) for RFPAL to find the percentage of samples with power level above RMS Power (dBm/40ms) + RFIN_CCDF1_dB. To convert dBN to dB
53	UINT16 RW	RFIN_CCDF2_dB	RFIN_CCDF2_dB is the threshold 2(dB) for RFPAL to find the percentage of samples with power level above RMS Power(dBm/40ms) + RFIN_CCDF2_dB
55	UINT16 RW	RFIN_CCDF3_dB	RFIN_CCDF3_dB is the threshold 3(dB) for RFPAL to find the percentage of samples with power level above RMS Power(dBm/40ms) + RFIN_CCDF3_dB
45	UINT16 R	RFIN_CCDF1_Per	RFIN_CCDF1_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFIN_CCDF1_dB = Percentage value *CCDF_Per_format (2^13)
61	UINT16 R	RFIN_CCDF2_Per	RFIN_CCDF2_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFIN_CCDF2_dB = Percentage value *CCDF_Per_format (2^13)
57	UINT16 R	RFIN_CCDF3_Per	RFIN_CCDF3_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFIN_CCDF3_dB = Percentage value *CCDF_Per_format (2^13)
2E	UINT16 RW	RFFB_CCDF1_dB	RFFB_CCDF1_dB is the threshold 1(dB) for RFPAL to find the percentage of samples with power level above RMS Power(dBm/40ms) + RFFB_CCDF1_dB.
4F	UINT16 RW	RFFB_CCDF2_dB	RFFB_CCDF2_dB is the threshold 2(dB) for RFPAL to find the percentage of samples with power level above RMS Power (dBm/40ms) + RFFB_CCDF2_dB
5F	UINT16 RW	RFFB_CCDF3_dB	RFFB_CCDF3_dB is the threshold1(dB) for RFPAL to find the percentage of samples with power level above RMS Power (dBm/40ms) + RFFB_CCDF3_dB
59	UINT16 R	RFFB_CCDF1_Per	RFFB_CCDF1_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFFB_CCDF1_dB = Percentage value *CCDF_Per_format (2^13)
5B	UINT16 R	RFFB_CCDF2_Per	RFFB_CCDF2_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFFB_CCDF2_dB = Percentage value *CCDF_Per_format (2^13)
5D	UINT16 R	RFFB_CCDF3_Per	RFFB_CCDF3_Per represents the percentage of samples with power level above RMS Power(dBm/40ms) + RFFB_CCDF3_dB = Percentage value *CCDF_Per_format (2^13)

*RFIN_CCDFX_dB and RFFB_CCDFX_dB are in dBN. dB=3.01*dBN/1024.*

6.3. Internal Temperature Sensor

Table 24: Internal Temperature Sensor Scratch Parameters

Scratch at (Hex)	Size/Access	Variable Name	Description
23D	INT16 R	IcTemp	16-signed bit Internal Temperature.

IcTemp is in big-endian format. It is in units of °C.

6.4. Spectrum Reporting (SEM and PSD)

6.4.1. Spectrum Emission Mask (SEM) Parameters

Table 25: SEM EEPROM Parameters

EEPROM @ (Hex)	Size	Variable Name	Description
FC10	UINT8	SemMeasBW_MHz	2*BandWidth in MHz over which spectral emission is measured.
FC11	INT8	LowerSemFreqA_MHz	2*Lower Offset A in MHz from the lower-edge of the signal
FCF0	INT8	LowerSemFreqB_MHz	2*Lower Offset B in MHz from the lower-edge of the signal
FCF1	INT8	UpperSemFreqA_MHz	2*Upper Offset A in MHz from the upper-edge of the signal
FCF2	INT8	UpperSemFreqB_MHz	2*Upper Offset B in MHz from the upper-edge of the signal

Table 26: SEM Scratch Parameters

Scratch @ (Hex)	Size/Access	Variable Name	Description
F1A	INT16 R	SEM_MaxPsdMeas	Highest 40µs PSD value in dBN/MHz over 40ms window. Updated every 340ms dBm = 3.01*dBN/1024
F1C	INT16 R	SEM_MaxRangeA	Maximum Spectrum Emission Mask of Lower and Upper SEM bands defined by LowerSemFreqA and UpperSemFreqA over SemMeasBw. Updated every 340ms.
F1E	INT16 R	SEM_MaxRangeB	Maximum Spectrum Emission Mask of Lower and Upper SEM bands defined by LowerSemFreqB and UpperSemFreqB over SemMeasBw. Updated every 340ms.

6.4.2. **Power Spectrum Density (PSD) Parameters**

See section 0 for MATLAB example code.

Table 27: PSD Scratch Parameters

Scratch @ (Hex)	Size/Access	Variable Name	Description
02C	UINT8 RW	EnablePsdMeas	PSD enable measurement. 1 for RFFB PSD capture 2 for RFIN PSD capture After setting this parameter to 1 or 2, the host should read EnablePsdMeas until it cleared to "0". Then the 256 PSD points in dBN format will be available in MeasEmplog2Psd.
BC8	UINT8 RW	Frequency Span	Frequency Span in MHz for PSD measurement. 0 or 1 = 100MHz (Default) 2 = 50MHz 3 = 25MHz 4 = 12.5MHz 5 ≥ 6.25MHz 100MHz will be used if not set.
CEC	UINT16 RW	PSD_LO_Frequency	2*PSD LO Frequency in MHz at which the PSD is taken. If not set, the signal center frequency will be used by default.
1340	256 INT16 R	MeasEmplog2Psd	256 PSD points in dBN format. Need to use special command to extend scratch readable range as described in section 3.3 The resulting PSD will need to be spectrally inverted

Before setting EnablePsdMeas to 1 or 2, it is possible to first change the configuration of the Frequency Span or the PSD_LO_Frequency.

7. Factory Calibration

7.1. Smooth Mode Calibration

The smooth adaptation calibration is done at factory alignment of the power amplifier system. It is possible to calibrate at either one or two center frequencies. With the system at maximum average output power and maximum signal bandwidth and a constant average output power, the SC1905 adapts and stores certain parameters in the EEPROM. For LTE signals, a signal with 100% resource block loading should be used. For TDD systems; it is recommended to use 100% resource block loading during the TX ON period.
The smooth adaptation calibration procedure for frequency A is described in Figure 15

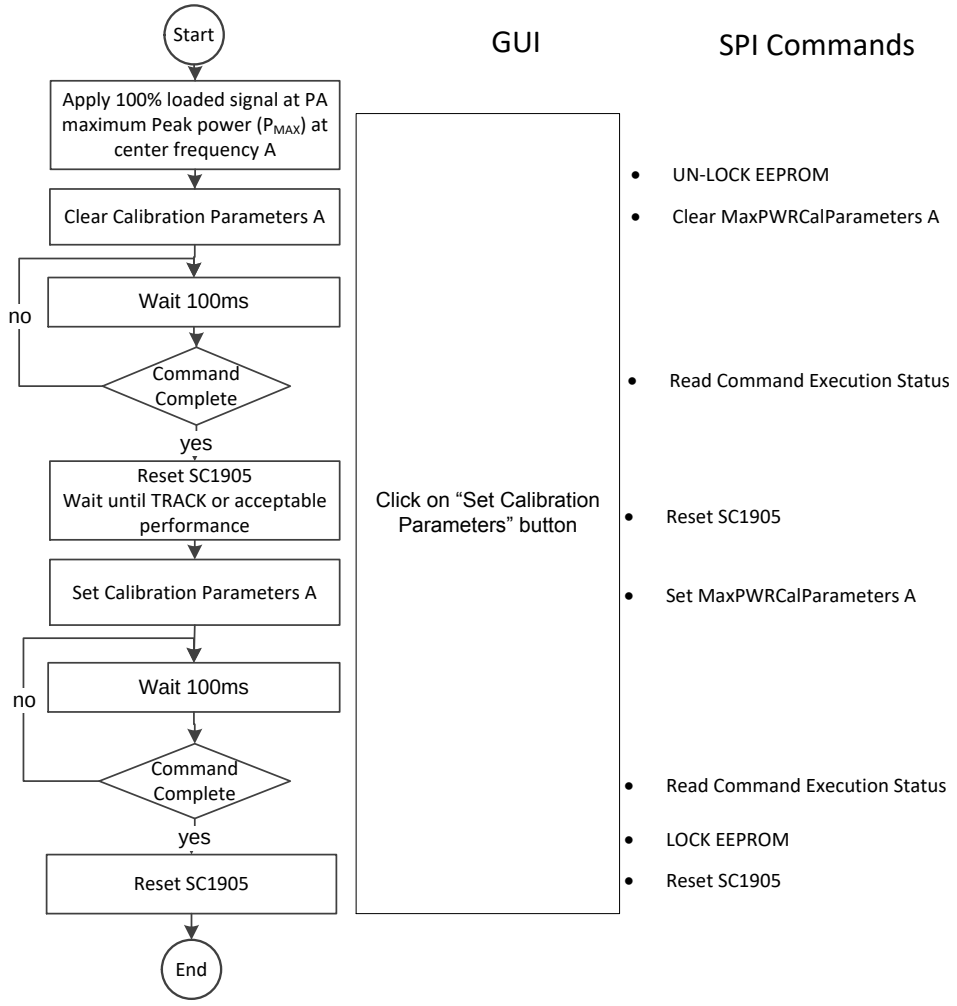


Figure 15: Smooth adaptation Calibration Procedure at Center Frequency A

IMPORTANT: *Smooth Adaptation Calibration has to be done with the system at Maximum Peak PA output power with minimum PAR (for Maximum RMS power) and it is recommended to wait until achieving the expected performance before issuing the Set Calibration command.*

7.1.1. Single Point of Calibration at Frequency A

The SPI Messages Communication Commands for smooth adaptation calibration are described in section 3.4 with their corresponding status flags. **See example code in sections 8.3 and 0.**
For a single point of calibration at frequency A, the sequence below should be followed:

1. Set the center frequency to frequency A and adjust signal levels to get the target PA output power.
2. Operate the SPI Bus at up to 4MHz.
3. LOADENB (pin 60) needs to be set HIGH ("1"). Host is now directly communicating with the embedded EEPROM. See the Microchip 25A512 data sheet for detailed instructions.
4. UNLOCK EEPROM. See section 4.1.10 for detailed instructions.
5. Make sure EEPROM is UNLOCKED by Reading STATUS register
6. LOADENB (pin 60) needs to be set LOW ("0").
7. Send Clear MaxPWRCalParameters A SPI command 10 F3 00 00. See section 3.4
8. Wait for at least 100 ms.
9. Read MaxPwrClearOnGoing flag until a value of 0x00 is returned by the SC1905.
10. Reset the SC1905 by toggling the RESETN line.
11. Wait 1 second.
12. Send Write MaxPWRCalParameters A SPI command 10 F5 00 00. See section 3.4
13. Wait for at least 100ms.
14. Read MaxPwrCalAOngoingflag until a value of 0x00 is returned by the SC1905.
15. If two points of calibration are required, then proceed to step 21 without executing the following commands. If only one point of calibration is required, then proceed to step 16
16. LOADENB (pin 60) needs to be set HIGH ("1").
17. LOCK EEPROM to disable writes to the EEPROM. See section 4.1.10 for detailed instructions.
18. LOADENB (pin 60) needs to be set LOW ("0").
19. Reset using pin 49 (RESETN)
20. One frequency calibration is complete.

7.1.2. Second Point of Calibration at Frequency B

If a second frequency calibration is desired, then continue with the sequence below:

21. Set the center frequency to frequency B and adjust signal levels to get the target PA output power.
22. Send Clear MaxPWRCalParameters B SPI command 10 F4 00 00. See section 3.4
23. Wait for at least 100ms.
24. Read MaxPwrClearOnGoing flag until a value of 0x00 is returned by the SC1905.
25. Reset the SC1905 by toggling the RESETN line.
26. Wait 1 second.
27. Send Write MaxPWRCalParameters B SPI command 10 F6 00 00. See section 3.4
28. Wait for at least 100ms.
29. Read MaxPwrCalAOngoingflag until a value of 0x00 is returned by the SC1905.
30. LOADENB (pin 60) needs to be set high ("1").
31. LOCK EEPROM to disable writes to the EEPROM. See section 4.1.10 for detailed instructions.
32. LOADENB (pin 60) needs to be set low ("0").
33. Reset using pin 49 (RESETN)
34. Two frequency points calibration is complete.

8. MATLAB Example Code

8.1. Set Frequency Range Example Code

```
function [Err] =SetFrequencyRange_Example_Code(FreqRange)
% Frequency Range
%               04: 698-1040 MHz, 05: 1040-2080 MHz, 06: 698-2700MHz
%                           07: 1800-2700 MHz, 08: 2700-3500 MHz, 09: 3300-3800MHz
% Error (out): =1 if an error occurs; =0 if OK
Err=0;
rfpal_eepromWriteStatus (0); % Set LOADENB High and Unlock the EEPROM

%Read all 1024 bytes of the customer Configuration Parameters with one read
%instruction
customerConfigParameters = rfpal_eepromRead(hex2dec('FC00'),1024);

switch FreqRange
   case (4)
      customerConfigParameters(5)= 04; %Frequency Range 04: 698MHz-1040MHz
      Freq_Scan_min= 698;             % Default Min Frequency is 698MHz
      Freq_Scan_max= 1000;             % Default Max Frequency is 1040MHz
   case (5)
      customerConfigParameters(5)= 05; % Frequency Range 05: 1040MHz-2080MHz
      Freq_Scan_min= 1040;             % Default Min Frequency is 1040MHz
      Freq_Scan_max= 2000;             % Default Max Frequency is 2080MHz
   case (6)
      customerConfigParameters(5)= 06; % Stiched Frequency Range 06: 698MHz-2700MHz
      Freq_Scan_min= 698;             % Default Min Frequency is 700MHz
      Freq_Scan_max= 2700;             % Default Max Frequency is 2700MHz
   case (7)
      customerConfigParameters(5)= 07; %Frequency Range 07: 1800MHz-2700MHz
      Freq_Scan_min= 1800;             % Default Min Frequency is 1800MHz
      Freq_Scan_max= 2700;             % Default Max Frequency is 270 MHz
   case (8)
      customerConfigParameters(5)= 08; %Frequency Range 08: 2700MHz-3500MHz
      Freq_Scan_min= 2700;             % Default Min Frequency is 2700MHz
      Freq_Scan_max= 3500;             % Default Max Frequency is 3500MHz
   case (9)
      customerConfigParameters(5)= 09; %Frequency Range 09: 3300MHz-3800MHz
      Freq_Scan_min= 3300;             % Default Min Frequency is 3300MHz
      Freq_Scan_max= 3800;             % Default Max Frequency is 3800MHz
   otherwise
      Err=1;
end

if (Err==0)
   customerConfigParameters(2)= floor (2*Freq_Scan_min/256);  %2xMin Freq Scan MSB
   %2xMin  Freq Scan LSB
   customerConfigParameters(1)= 2*Freq_Scan_min-256*floor (2*Freq_Scan_min/256);
   customerConfigParameters(4)= floor  (2*Freq_Scan_max/256);%2xMax Freq Scan MSB
   %2xMax  Freq Scan LSB
```

```
    customerConfigParameters(3)= 2*Freq_Scan_max-256*floor (2*Freq_Scan_max/256);

    %Computing New Checksum
    checksum = double(0);
    for i=1:1023
       checksum = double(checksum + double(customerConfigParameters(i)));
    end
   %Compute the New Checksum: Modulo256 of all bytes added from FC00 to FFFE
   customerConfigParameters(1024) = uint8(mod(checksum,256));

   fprintf(1, 'Storing Customer Configuration Parameters to 64K EEPROM\n');
   % rfpal_eepromWrite will divide customerConfigParameters into 64-bytes pages
   % and write 64-byte with one write instruction
   rfpal_eepromWrite(h,hex2dec('FC00'),customerConfigParameters(1:64));
   rfpal_eepromWrite(h,hex2dec('FC40'),customerConfigParameters(65:128));
   rfpal_eepromWrite(h,hex2dec('FC80'),customerConfigParameters(129:192));
   rfpal_eepromWrite(h,hex2dec('FCC0'),customerConfigParameters(193:256));
   rfpal_eepromWrite(h,hex2dec('FD00'),customerConfigParameters(257:320));
   rfpal_eepromWrite(h,hex2dec('FD40'),customerConfigParameters(321:384));
   rfpal_eepromWrite(h,hex2dec('FD80'),customerConfigParameters(385:448));
   rfpal_eepromWrite(h,hex2dec('FDC0'),customerConfigParameters(449:512));
   rfpal_eepromWrite(h,hex2dec('FE00'),customerConfigParameters(513:576));
   rfpal_eepromWrite(h,hex2dec('FE40'),customerConfigParameters(577:640));
   rfpal_eepromWrite(h,hex2dec('FE80'),customerConfigParameters(641:704));
   rfpal_eepromWrite(h,hex2dec('FEC0'),customerConfigParameters(705:768));
   rfpal_eepromWrite(h,hex2dec('FF00'),customerConfigParameters(769:832));
   rfpal_eepromWrite(h,hex2dec('FF40'),customerConfigParameters(833:896));
   rfpal_eepromWrite(h,hex2dec('FF80'),customerConfigParameters(897:960));
   rfpal_eepromWrite(h,hex2dec('FFC0'),customerConfigParameters(961:1024));
   rfpal_eepromWriteStatus (3); % Lock the EEPROM
   rfpal_hardReset; % % Reset and Set LOADENB LOW
end
end %End of the function
```

8.2. Get SPI Message Parameters Example Code

```
function Get_SPI_Message_Parameters_Example_code()
%Check Product ID: SC1894 or SC1905
productID = rfpal_msgCmdRead(h, hex2dec('959'),1);
fprintf( 'SC%4d Chip\n',productID)
%Read Firmware Version
FWVer = rfpal_msgCmdRead(hex2dec('03'), 0); %8-bit FW Version in Hexadecimal format
FWBuildMSB= rfpal_msgCmdRead(hex2dec('04'), 0); %8-bit FW Build MSB in Decimal format
FWBuildLSB= rfpal_msgCmdRead(hex2dec('0A'), 0); %8-bit FW Build LSB in Decimal format
fprintf( 'Firmware %1X %2d %2d \n',FWVer,FWBuildMSB,FWBuildLSB);
% Get 8-bit Output status
OutputStatus= rfpal_msgCmdRead(hex2dec('32'), 0);
if (OutputStatus==0)
   fprintf('Output Status: RFOUT OFF\n'); % RFOUT Disabled
else
   fprintf('Output Status: RFOUT ON\n'); % RFOUT ON
end
 %Get 8-bit Output Mode
OutputMode= rfpal_msgCmdRead(hex2dec('08'), 0);
if (OutputMode==0)
   fprintf('Output Mode: RFOUT Disabled\n');
else
   fprintf('Output Mode: FW Control\n');
end
 %Get 16-bit Center Frequency
Center_frequency=rfpal_msgCmdRead(hex2dec('1A'), 1)/2;%2xCenter Frequency(MHz)
fprintf( 'Center_frequency %4d MHz\n',Center_frequency);
%Get 16-bit Signal Bandwidth
Bandwidth=rfpal_msgCmdRead(hex2dec('18'), 1)/2; %2xBandwidth(MHz)
fprintf( 'Bandwidth %2d MHz\n',Bandwidth);
%Get 16-bit Frequency Range
Frequency_Range=rfpal_msgCmdRead(hex2dec('10'), 0);
fprintf( 'Frequency_Range %2d\n',Frequency_Range);

%Get 16-bit Min Frequency Scan
MinFrequencyScan=rfpal_msgCmdRead(hex2dec('11'), 1)/2;%2xMinFrequencyScan(MHz)
fprintf( 'MinFrequencyScan %4d MHz\n',MinFrequencyScan);
%Get 16-bit Max Frequency Scan
MaxFrequencyScan=rfpal_msgCmdRead(hex2dec('13'), 1)/2;%2xMaxFrequencyScan(MHz)
fprintf( 'MaxFrequencyScan %4d MHz\n',MaxFrequencyScan);
%Get 8-bit Duty Cycled Feedback Mode
DCF_Mode= rfpal_msgCmdRead(hex2dec('17'), 0);
if (DCF_Mode==0)
   fprintf('Duty Cycled Feedback OFF\n');
else
   fprintf('Duty Cycled Feedback ON\n');
end
% Read and compute Average Coefficient
Norm_Factor= rfpal_msgCmdRead(hex2dec('33'), 0);
UnNorm_Coeff = double(rfpal_msgCmdRead(hex2dec('34'), 1));
```

```
Average_Coefficient = UnNorm_Coeff/Norm_Factor;
fprintf('Average_Coefficient  %4d \n',Average_Coefficient);
%Get 8-bit Status
status = rfpal_msgCmdRead(hex2dec('05'), 0);
state = bitand(status,hex2dec('3F'));   %Overall Status
warning = bitand(status,hex2dec('40')); %Warning Status
error = bitand(status,hex2dec('80'));   %Error Status
if (state == 0)
  fprintf( 'INIT\n');
elseif (state== 1)
  fprintf( 'FSA\n');
elseif (state == 3)
  fprintf( 'TRACK\n');
elseif (state== 6)
  fprintf('CAL\n');
elseif (state== 9)
  fprintf('PDET\n');
else
  fprintf('No Valid State\n');
end
if (error~=0) % There is an error
    error_code=rfpal_msgCmdRead(hex2dec('06'), 0);%Get 8-bit Error Code
    fprintf('Error %d \n',error_code);
end
if (warning~=0) % There is a warning
    warning_code=rfpal_msgCmdRead(hex2dec('07'), 0); %Get 8-bit Warning Code
    fprintf('Warning %d \n',warning_code);
    clear_warning;
end
end %End of function
```

8.3. SC1905 Clear Max PWR Cal Parameters Example Code (Optimized Mode)

This routine is used to clear from Smooth optimization mode to Optimized performance mode.

```
function SC1905clearMaxPWRCalParameters (h, freqSelect)
% Parameters:
%   h (in): RFPAL object
%   freqSelect (in): Frequency select (Optional parameter):
%   if = 0, then "A" frequency and both "A" and "B" max power cal parameters are cleared.
%   if = 1, then "B" frequency and only "B" max power cal parameters are cleared.
%   if not specified, then default value is 0.
if(nargin < 2)
    freqSelect = 0;
end
rfpal_eepromWriteStatus (h,0); % Set TESTSEL0 High and Unlock the EEPROM
pause(1);  % allow RFPAL enough time after reset to start message interface
if (freqSelect == 0)
    rfpal_msgSa(h,hex2dec('F3')); % Clear MaxPWRCalParameters A and B
else
% Set Frequency B to Optimize Mode. Doesn't clear all the parameters.
% Set B
    rfpal_msgSa(h,hex2dec('F4'));
end
pause(0.1); % allow firmware time to initially set flag
clearOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC3'),0);
while (clearOngoingFlg~=0)%Wait for clear ongoing flag to become zero
  clearOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC3'),0);
end
rfpal_eepromWriteStatus (h,3); % Lock the EEPROM
```

8.4. SC1905 Set Max PWR Cal Parameters (Smooth Mode Calibration)

```
function [cal_err] = SC1905SetMaxPWRCalParameters(h, freqSelect)
% Parameters:
%   cal_error (out): =1 if an error occur; =0 if calibration was OK
%   h (in): RFPAL object
%   freqSelect (in): Optional parameter, Frequency select:
%    If = 0, then "A" frequency and "A" max power cal parameters are stored
%    If ? 1, then "B" frequency and "B" max power cal parameters are stored
%    If not specified, then default value is 0.
% optional arguments
if(nargin < 2)
   freqSelect = 0;
end
fprintf(1, '"Clear the MaxPWRCalParameters"\n');
SC1905clearMaxPWRCalParameters(h, freqSelect); % IT IS IMPORTANT TO FIRST Clear the
MaxPWRCalParameters
rfpal_hardReset(h); % Reset and Set TESTSEL0 LOW
cal_err=0;
pause(1);  % allow RFPAL enough time after reset to start message interface
rfpal_eepromWriteStatus (h,0); % Set TESTSEL0 High and Unlock the EEPROM
if (freqSelect == 0)   % A frequency
   rfpal_msgSa(h,hex2dec('F5')); % Write MaxPWRCalParameters A
else   % B frequency
   rfpal_msgSa(h,hex2dec('F6')); % Write MaxPWRCalParameters B
end
pause(0.1); % allow firmware time to initially set flag
if (freqSelect == 0)   % A frequency
   calAOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC4'),0);
   while (calAOngoingFlg~=0) %Wait for cal A ongoing flag to become zero
      calAOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC4'),0);
   end
else  % B frequency
   calBOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC6'),0);
   while (calBOngoingFlg~=0) %Wait for cal B ongoing flag to become zero
      calBOngoingFlg = rfpal_msgCmdRead(h,hex2dec('DC6'),0);
   end
end
rfpal_eepromWriteStatus (h,3); % Lock the EEPROM
rfpal_hardReset(h); % % Reset and Set TESTSEL0 LOW
end
```

8.5. Read Cost Example Code

```
function [cal_err Average_CostFunction] = Read_Cost_Function_Example_Code()
% Parameters:
%   cal_error (out): =1 if an error occur; =0 if calibration was OK
cal_err=0;
iteration=30; %Recommended value for more accurate measurement.
Average_CostFunction=0; %Initialize to zero.
% Check status
status = rfpal_msgCmdRead(hex2dec('05'), 0);
for i=1:iteration
   TimeOut=30;
   while ((status~=3) && (status<128) && (TimeOut>0)) % Wait for TRACK and make sure there
is no Error and no Time Out
      pause(5) % Wait 5s
      status = rfpal_msgCmdRead(hex2dec('05'), 0); % Check status
      fprintf(1, 'Not in TRACK yet, please wait\n');
      TimeOut=TimeOut-1;
   end
   if (status>127)
      fprintf(1, 'Chip Error. Make sure the EEPROM is not corrupted. Check the checksum\n');
      cal_err=1; % If a Chip error is reported, this needs to be fixed first.
      % Check the customer Configuration Parameters checksum was computed correctly.
   elseif (TimeOut==0)
      cal_err=1; % Increase TimeOut or check that system is working correctly
   else
      %Freeze adaptation
      rfpal_msgCmdWrite(hex2dec('23'),0);
      %Reading byte 2 @ 0x213 of cost function
      Cost_function_bytes = double(rfpal_msgCmdRead(hex2dec('20D'), 1));
      if (Cost_function_bytes>hex2dec('7FFF')) % If Negative Value
Cost_function_Vector(i)= - double(bitxor(Cost_function_bytes- 1,hex2dec('FFFF')));
      else                              % Positive Value
         Cost_function_Vector(i)=  Cost_function_bytes;
      end
      Average_CostFunction= double(Average_CostFunction + Cost_function_Vector(i));
      %Un-freeze adaptation
      rfpal_msgCmdWrite(hex2dec('23'),1);
      pause(0.2); %Add some delay between measurements.
   end
end
% Compute Average Value
Average_CostFunction = double(Average_CostFunction/iteration);
%Following is for debug purpose only
min_cost=min(Cost_function_Vector)
max_cost=max(Cost_function_Vector)
Delta = max_cost - min_cost
End
```

8.6. Read PMU CCDF Example Code

```
function [RFIN RFFB] = SC1905_Read_PMU_CCDF(h, DutyCycle)
% Parameters:
%   h (in): RFPAL object. Internal Parameter
%   Duty Cycle (Percent value) it is possible, but less accurate, to store
%   this value in EEPROM parameter PMUDutyCycleFactor and remove the Duty cycle factor %
in this function. So, it is recommended to take the duty cycle factor into account %   %   in the
host software.
%   PMU RFIN and RFFB Reference Offset are EEPROM parameters and are applied to the
%   values reported by the firmware
%   Function for SC1905 FW>4.1
%   RFIN and RFFB include all the PMU and CCDF values reported in the PMU
%   tab of the GUI
if nargin<2
   DutyCycle=100; %100%
end
CCDF_Per_format = 2^13;
% To convert Scratch dBN format to dBm values needs
%   1. To multiply by 3/1024 due to value format 6.10 signed
%   2. Take into account the waveform Duty Cycle: -10*log10(DutyCycle/100)

% Read RFFI_PMU Signed 6.10 signed Value from Internal Memory through the message
protocol
RFIN_PMU_bytes  = double(rfpal_msgCmdRead(h, hex2dec('247'), 1)); %Address 0x247 = 583
% RFIN RMS Power (dBm/40ms) over a 40ms measurement window. Updated every 300ms
RFIN.RMS  = 3.01*Read16B_signed_Scratch(RFIN_PMU_bytes)/1024-10*log10(DutyCycle/100);
RFFB_PMU_bytes = double(rfpal_msgCmdRead(h, hex2dec('245'), 1)); %Address 0x245 = 581
% RFFB RMS Power (dBm/40ms) over a 40ms measurement window. Updated every 300ms
RFFB.RMS=                              3.01*Read16B_signed_Scratch(RFFB_PMU_bytes)/1024-
10*log10(DutyCycle/100);
% RFIN Highest 10ns average values over the 40ms average window.
% Updated every 300ms
RFIN_PeakPower=rfpal_msgCmdRead(h, hex2dec('FB7'), 1);
RFIN.PeakPower_10ns          =          3.01*Read16B_signed_Scratch(RFIN_PeakPower)/1024-
10*log10(DutyCycle/100);
% RFFB Highest 10ns average values over the 40ms average window.
% Updated every 300ms
RFFB_PeakPower=rfpal_msgCmdRead(h, hex2dec('FB9'), 1);
RFFB.PeakPower_10ns          =          3.01*Read16B_signed_Scratch(RFFB_PeakPower)/1024-
10*log10(DutyCycle/100);
% RFIN Highest 40µs average values over the 40ms average window.
% Updated every 300ms
RFIN_MAX_RMS  = rfpal_msgCmdRead(h, hex2dec('4B'), 1);
RFIN.MAX_RMS                =                3.01*Read16B_signed_Scratch(RFIN_MAX_RMS)/1024-
10*log10(DutyCycle/100);
% RFFB Highest 40µs average values over the 40ms average window.
% Updated every 300ms
RFFB_MAX_RMS=rfpal_msgCmdRead(h, hex2dec('47'), 1);
RFFB.MAX_RMS                =                3.01*Read16B_signed_Scratch(RFFB_MAX_RMS)/1024-
10*log10(DutyCycle/100);
```

```
% RFIN Lowest 40µs average values over the 40ms average window.
% Updated every 300ms
RFIN_MIN_RMS  = rfpal_msgCmdRead(h, hex2dec('4D'), 1);
RFIN.MIN_RMS = 3.01*Read16B_signed_Scratch(RFIN_MIN_RMS)/1024-
10*log10(DutyCycle/100);
% RFFB Lowest 40µs average values over the 40ms average window.
% Updated every 300ms
RFFB_MIN_RMS=rfpal_msgCmdRead(h, hex2dec('49'), 1);
RFFB.MAX_RMS = 3.01*Read16B_signed_Scratch(RFFB_MIN_RMS)/1024-
10*log10(DutyCycle/100);
% Peak PAR (dB) = Peak Power (dBm/10ns) - RMS Power (dBm/40ms).
% Computed by Host.
RFIN.Peak_PAR = RFIN.PeakPower_10ns - RFIN.RMS;
RFFB.Peak_PAR = RFFB.PeakPower_10ns - RFFB.RMS;

%===== RFIN CCDF Parameters ================
% CCDF(dB) is the threshold(dB) for RFPAL to find the percentage of samples
% which its power level is above RMS Power (dBm/40ms) + CCDF(dB)
% Automatic mode will set CCDF1(dB)= ~Peak_PAR(dB)-0.25dB,
% CCDF2(dB)=~Peak_PAR(dB)-1dB and CCDF3(dB)= ~Peak_PAR(dB)-2dB.
RFIN_CCDF1_dB = rfpal_msgCmdRead(h, hex2dec('51'), 1);
RFIN.CCDF1_dB = 3.01*Read16B_signed_Scratch(RFIN_CCDF1_dB)/1024;

RFIN_CCDF2_dB = rfpal_msgCmdRead(h, hex2dec('53'), 1);
RFIN.CCDF2_dB = 3.01*Read16B_signed_Scratch(RFIN_CCDF2_dB)/1024;

RFIN_CCDF3_dB = rfpal_msgCmdRead(h, hex2dec('55'), 1);
RFIN.CCDF3_dB = 3.01*Read16B_signed_Scratch(RFIN_CCDF3_dB)/1024;

% Percentage value read from scratch
RFIN_CCDF1_Per = rfpal_msgCmdRead(h, hex2dec('45'), 1);
RFIN_CCDF2_Per = rfpal_msgCmdRead(h, hex2dec('61'), 1);
RFIN_CCDF3_Per = rfpal_msgCmdRead(h, hex2dec('57'), 1);

% Percentage display on GUI.  Need to adjust format from scratch values
% CCDF(%)percentage of samples which its power level is
% above RMS Power(dBm/40ms) + CCDF(dB). Updated every 300ms
RFIN.CCDF1_Per = double(RFIN_CCDF1_Per)/CCDF_Per_format;
RFIN.CCDF2_Per = double(RFIN_CCDF2_Per)/CCDF_Per_format;
RFIN.CCDF3_Per = double(RFIN_CCDF3_Per)/CCDF_Per_format;

%===== RFFB CCDF Parameters ================
% CCDF(dB) is the threshold(dB) for RFPAL to find the percentage of samples
% which its power level is above RMS Power (dBm/40ms) + CCDF(dB)
% Automatic mode will set CCDF1(dB)= ~Peak_PAR(dB)-0.25dB,
% CCDF2(dB)=~Peak_PAR(dB)-1dB and CCDF3(dB)= ~Peak_PAR(dB)-2dB.
RFFB_CCDF1_dB=rfpal_msgCmdRead(h, hex2dec('2E'), 1);
RFFB.CCDF1_dB = 3.01*Read16B_signed_Scratch(RFFB_CCDF1_dB)/1024;

RFFB_CCDF2_dB=rfpal_msgCmdRead(h, hex2dec('4F'), 1);
```

```
RFFB.CCDF2_dB = 3.01*Read16B_signed_Scratch(RFFB_CCDF2_dB)/1024;

RFFB_CCDF3_dB=rfpal_msgCmdRead(h, hex2dec('5F'), 1);
RFFB.CCDF3_dB = 3.01*Read16B_signed_Scratch(RFFB_CCDF3_dB)/1024;

% Percentage value read
RFFB_CCDF1_Per = rfpal_msgCmdRead(h, hex2dec('59'), 1);
RFFB_CCDF2_Per = rfpal_msgCmdRead(h, hex2dec('5B'), 1);
RFFB_CCDF3_Per = rfpal_msgCmdRead(h, hex2dec('5D'), 1);

% Percentage display on GUI.  Need to adjust format from scratch values
% CCDF(%)percentage of samples which its power level is
% above RMS Power(dBm/40ms) + CCDF(dB). Updated every 300ms
RFFB.CCDF1_Per = double(RFFB_CCDF1_Per)/CCDF_Per_format;
RFFB.CCDF2_Per = double(RFFB_CCDF2_Per)/CCDF_Per_format;
RFFB.CCDF3_Per = double(RFFB_CCDF3_Per)/CCDF_Per_format;
```

8.7. Set CCDF Mode Example Code

```
function [Err] = SC1905_Set_CCDF_Mode(h, CCDF_Mode)
% CCDF_Mode 0= Automatic; 1= Manual Mode
% CCDF Mode: 0= Automatic or 1=Manual.
% Automatic mode will set CCDF1_dB= Peak_PAR_dB-0.25dB, CCDF2_dB= Peak_PAR(dB)-
1dB and CCDF3_dB= Peak_PAR(dB)-2dB.
% After selecting Manual mode, these thresholds need to be adjusted as needed after each reset
Current_CCDF_Mode = rfpal_eepromRead(h, hex2dec('FD3B'), 1);

RFIN_CCDF1_dB = 8.3;
RFIN_CCDF2_dB = 8.1;
RFIN_CCDF3_dB = 8;
RFFB_CCDF1_dB = 8.2;
RFFB_CCDF2_dB = 8.1;
RFFB_CCDF3_dB = 8;

if (Current_CCDF_Mode ~= CCDF_Mode)
   CustomerConfigParameters = rfpal_eepromRead(h, hex2dec('FC00'), 1024);
   CustomerConfigParameters(316)= CCDF_Mode;
   checksum = double(0);
   for i=1:1023
      checksum = double(checksum + double(CustomerConfigParameters(i)));
   end
   CustomerConfigParameters(1024) = uint8(mod(checksum,256));
   rfpal_eepromWriteStatus (h,0); % Set TESTSEL0 High and Unlock the EEPROM
   rfpal_eepromWrite(h,hex2dec('FD3B'),CustomerConfigParameters(316));
   rfpal_eepromWrite(h,hex2dec('FFFF'),CustomerConfigParameters(1024));
   rfpal_eepromWriteStatus (h,3); % Lock the EEPROM
   rfpal_hardReset(h); % Reset and Set TESTSEL0 LOW
end
if (CCDF_Mode==1) %Only Needed for Manual Mode
   % Need to adjust Format to write to Scratch
   RFIN_CCDF1_dBN = double(1024*RFIN_CCDF1_dB/3.01);
   RFIN_CCDF2_dBN = double(1024*RFIN_CCDF2_dB/3.01);
   RFIN_CCDF3_dBN = double(1024*RFIN_CCDF3_dB/3.01);
   % Need to adjust Format to write to Scratch
   RFFB_CCDF1_dBN = double(1024*RFFB_CCDF1_dB/3.01);
   RFFB_CCDF2_dBN = double(1024*RFFB_CCDF2_dB/3.01);
   RFFB_CCDF3_dBN = double(1024*RFFB_CCDF3_dB/3.01);
   % Write to Scratch RFIN CCDF Threshold
   rfpal_msgCmdWrite(h, hex2dec('51'), RFIN_CCDF1_dBN, 1);
   rfpal_msgCmdWrite(h, hex2dec('53'), RFIN_CCDF2_dBN, 1);
   rfpal_msgCmdWrite(h, hex2dec('55'), RFIN_CCDF3_dBN, 1);
   % Write to Scratch RFFB CCDF Threshold
   rfpal_msgCmdWrite(h, hex2dec('2E'), RFFB_CCDF1_dBN, 1);
   rfpal_msgCmdWrite(h, hex2dec('4F'), RFFB_CCDF2_dBN, 1);
   rfpal_msgCmdWrite(h, hex2dec('5F'), RFFB_CCDF3_dBN, 1);
end
```

8.8. Get RFIN and RFFB PSD Example Code

```
function [psd_point]= SC1905_Get_PSD(h, PSD_select, FreqSpan)
% Parameters: h (in): RFPAL object
%   PSD_select: 1 for RFFB PSD capture and 2 for RFIN PSD capture
%   FreqSpan: Frequency Span in MHz for PSD measurement.
%   0 or 1 = 100MHz (Default); 2 = 50MHz; 3 = 25MHz
if nargin<2
    PSD_select=1; %Get RFFB PSD
    FreqSpan=0; % For 100MHz Span
elseif nargin <3
    FreqSpan=0; % For 100MHz Span
end
psd_point = zeros(1,256);
if (PSD_select>2)
    PSD_select=2; %Get RFIN PSD for value>2
end
%==========Optional=============================================
Center_frequency=rfpal_msgCmdRead(h, hex2dec('1A'), 1)/2;%2xCenter Frequency(MHz)
rfpal_msgCmdWrite(h, hex2dec('CEC'), 2*Center_frequency,1);
rfpal_msgCmdWrite(h, hex2dec('BC8'), FreqSpan,0); % If not set, use 100MHz (Default)
%==========End Optional=========================================
rfpal_msgCmdWrite(h, hex2dec('02C'), PSD_select,0);
PSD_ready=rfpal_msgCmdRead(h, hex2dec('02C'),0);
while (PSD_ready>0)
    PSD_ready=rfpal_msgCmdRead(h, hex2dec('02C'),0);
    pause(1)
end
rfpal_msgSa(h,hex2dec('CD')); %Set offset to enable Extend Scratch Readable Access
for psd_Index=1:256
    %PSD points need to be spectrally inverted
    psd_point_address = hex2dec('1340')-hex2dec('800')+(256-psd_Index)*2;
    psd_bytes = double(rfpal_msgCmdRead(h, psd_point_address, 1));
    psd_point(psd_Index)  = 3.01*Read16B_signed_Scratch(psd_bytes)/1024;
end
rfpal_msgSa(h,hex2dec('CE')); %Remove offset to disable Extend Scratch Readable Access
if (PSD_select==1)
    figure(1);
else
    figure(2);
end
plot(psd_point,'r');
xlabel('PSD Bin Number')
ylabel('PSD Bin Power (dB)')
if (PSD_select==1)
    title('PSD of RFFB Signal')
else
    title('PSD of RFIN Signal')
end
```

8.9. Read EEPROM Customer Configuration Parameters

```
function [customerConfigParameters]= SC1905_Read_customerConfigParameters(h)
% Parameters:
%   h (in): RFPAL object
%  customerConfigParameters(out): EEPROM Customer Configuration Parameters

cfg = rfpal_eepromRead(h, hex2dec('FC00'), 1024)
%Frequency band information
customerConfigParameters.minFreq = (double(cfg(2))*256+double(cfg(1)))/2;
customerConfigParameters.maxFreq = (double(cfg(4))*256+double(cfg(3)))/2;
customerConfigParameters.band=cfg(5);
% Smooth Calibration for Frequency A
CAL1A =(double(cfg(29))*256+double(cfg(28)));
if (CAL1A>hex2dec('7FFF')) %Negative Value
   customerConfigParameters.CAL1A = - double(3.01*bitxor(CAL1A-
1,hex2dec('FFFF'))/1024); %2s complement for negative values
else % Positive Value
   customerConfigParameters.CAL1A = double(3.01*CAL1A/1024);
end
customerConfigParameters.CAL2A_PDET_Index=cfg(30);
CAL3A=double(cfg(32))*256+double(cfg(31));
if (CAL3A>hex2dec('7FFF'))
   customerConfigParameters.CAL3A=-double((bitcmp(CAL3A)+1));
else
   customerConfigParameters.CAL3A=CAL3A;
end
customerConfigParameters.CAL4A_CorrVGA_Index=cfg(33);
customerConfigParameters.CAL5A_PDET_DC_DAC=cfg(34);
customerConfigParameters.CAL6A_Fine_PDET_Index=cfg(56);
customerConfigParameters.CAL7A_EDET_Index=cfg(57);
customerConfigParameters.CAL8A_CORR_Multi_DC_DAC=cfg(58:58+24);
CAL9A =(double(cfg(83))*256+double(cfg(82)));
if (CAL9A>hex2dec('7FFF')) %Negative Value
   customerConfigParameters.CAL9A = -double(3.01*bitxor(CAL9A-
1,hex2dec('FFFF'))/1024); %2s complement for negative values
else % Positive Value
   customerConfigParameters.CAL9A = double(3.01*CAL9A/1024);
end
customerConfigParameters.CAL10A_Freq = (double(cfg(85))*256+double(cfg(84)))/2;
customerConfigParameters.MaxPWRCalCoeffA = cfg(126:175);
customerConfigParameters.NotFirstMaxPwrCal = cfg(97+1);
% Smooth Calibration for Frequency B
CAL1B =(double(cfg(87))*256+double(cfg(86)));
if (CAL1B>hex2dec('7FFF')) % Negative Value
   %2s complement for negative values
  customerConfigParameters.CAL1B_Power = - double(3.01*bitxor(CAL1B-
1,hex2dec('FFFF'))/1024);
else % Positive Value
   customerConfigParameters.CAL1B_Power = double(3.01*CAL1B/1024);
end
```

```
customerConfigParameters.CAL2B_PDET_Index=cfg(88);
CAL3B=double(cfg(90))*256+double(cfg(89));
if (CAL3B>hex2dec('7FFF'))
    customerConfigParameters.CAL3B=-double((bitcmp(CAL3B)+1));
else
    customerConfigParameters.CAL3B=CAL3B;
end
customerConfigParameters.CAL4B_CorrVGA_Index=cfg(91);
customerConfigParameters.CAL5B_PDET_DC_DAC=cfg(92);
customerConfigParameters.CAL6B_Fine_PDET_Index=cfg(100);
customerConfigParameters.CAL7B_EDET_Index=cfg(101);
customerConfigParameters.CAL8B_CORR_Multi_DC_DAC=cfg(102:102+24);
CAL9B =(double(cfg(94))*256+double(cfg(93)));
if (CAL9B>hex2dec('7FFF')) %Negative Value
    %2s complement for negative values
   customerConfigParameters.CAL9B = - double(3.01*bitxor(CAL9B-1,hex2dec('FFFF'))/1024);
else                        % Positive Value
    customerConfigParameters.CAL9B = double(3.01*CAL9B/1024);
end
customerConfigParameters.CAL10B_Freq = (double(cfg(96))*256+double(cfg(95)))/2;
customerConfigParameters.MaxPWRCalCoeffB = cfg(176:225);
if (cfg(37)==1)
    customerConfigParameters.PDET_Temp_Comp_Flag = 'DISABLED';
else
    customerConfigParameters.PDET_Temp_Comp_Flag = 'ENABLED';
end

% ATE Calibration Offset Zone Written
customerConfigParameters.ATE_CalibrationOffsetZoneWritten = cfg(436);

%============== Wideband Operation Parameters==============
  % Linearization Operation Mode
customerConfigParameters.LinearizerOperationMode = cfg(351);
% Customer Definable Guard Bin, if set to zero, use default value
customerConfigParameters.CustomerGuardBinEeprom = cfg(99);
% Parameters used for Wideband Mode
customerConfigParameters.SemMeasBw_MHz = double(AdvConfParam(17))/2; %2xSEM
Measurement Bandwidth in MHz
customerConfigParameters.LowerSemFreqA_MHz =
double(Read8B_signed_EEPROM(cfg(18)))/2; %2xLowerOffsetA in MHz from the Lower edge
of the signal
customerConfigParameters.UpperSemFreqA_MHz =
double(Read8B_signed_EEPROM(cfg(242)))/2; %2xUpperOffsetA in MHz from the UpperLower
edge of the signal
customerConfigParameters.LowerSemFreqB_MHz =
double(Read8B_signed_EEPROM(cfg(241)))/2; %2xLowerOffsetB in MHz from the Lower edge
of the signal
customerConfigParameters.UpperSemFreqB_MHz =
double(Read8B_signed_EEPROM(cfg(243)))/2; %2xUpperOffsetB in MHz from the Lower edge
of the signal
```

```
% PMU & CCDF Parameters
customerConfigParameters.RFFB_Reference_Offset =
3.01*Read16B_signed_EEPROM(cfg(324),cfg(325))/1024;
customerConfigParameters.RFIN_Reference_Offset =
3.01*Read16B_signed_EEPROM(cfg(326),cfg(327))/1024;
customerConfigParameters.CCDF_Mode = cfg(316);
% SEM Parameters
customerConfigParameters.SemMeasBw_MHz = double(cfg(17))/2; %2xSEM Measurement
Bandwidth in MHz
customerConfigParameters.LowerSemFreqA_MHz =
double(Read8B_signed_EEPROM(cfg(18)))/2; %2xLowerOffsetA in MHz from the Lower edge
of the signal
customerConfigParameters.UpperSemFreqA_MHz =
double(Read8B_signed_EEPROM(cfg(242)))/2; %2xUpperOffsetA in MHz from the UpperLower
edge of the signal
customerConfigParameters.LowerSemFreqB_MHz =
double(Read8B_signed_EEPROM(cfg(241)))/2; %2xLowerOffsetB in MHz from the Lower edge
of the signal
customerConfigParameters.UpperSemFreqB_MHz =
double(Read8B_signed_EEPROM(cfg(243)))/2; %2xUpperOffsetB in MHz from the Lower edge
of the signal
customerConfigParameters.CustomerGuardBandEeprom = cfg(99);
% Customer Configuration Parameter Checksum
customerConfigParameters.checksum=cfg(1024);
```

8.10. Convert 16-bit Signed Values from EEPROM Example Code

```
function [Signed_16Bits_value]= Convert16B_signed_EEPROM(LSB, MSB)
%EEPROM is little Endian LSB MSB
Value= double(MSB)*256+double(LSB);
if (Value>hex2dec('7FFF'))
   Signed_16Bits_value=double(Value)-65536;
else
   Signed_16Bits_value=double(Value);
end
```

8.11. Convert 8-bit Signed Values from EEPROM Example Code

```
function [Signed_8Bits_value]= Read8B_signed_EEPROM(value)
if (value>hex2dec('7F'))
   Signed_8Bits_value=double(value)-256;
else
   Signed_8Bits_value=double(value);
end
```

8.12. Convert 16-bit Signed Values from Scratch Example Code

```
function [Signed_16Bits_value]= Convert16B_signed_Scratch(value)
%Scratch is big Endian
if (value>hex2dec('7FFF')) % Negative value
   Signed_16Bits_value=double(value)-65536;
else % Positive value
   Signed_16Bits_value=double(value);
end
```

9. Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	11/18	Initial release	—