LR1121 Transceiver

User Manual

Disclaimer

Long Range-Frequency Hopping Spread Spectrum (LR-FHSS) is a high link-budget, high-performance technology combining the benefits of a modulation employing low energy per bit and advanced frequency hopping schemes to achieve improved coexistence, spectral efficiency and sensitivity. Semtech Corp. holds patents directed to aspects of the LR-FHSS technology.

Your use of LR-FHSS software made available by Semtech Corp. or its affiliates does not grant any rights to their patents for LR-FHSS technology. Rights under Semtech patents may be available via various mechanisms, including by purchasing Semtech SX1261, SX1262, SX1268, LR1110, LR1120 or LR1121 semiconductor devices, or their authorized counterparts from Semtech or its affiliates, or their respective licensees.

Table of Contents

1. Introduction 12
    1.1 Scope 12
    1.2 Overview 12
2. System Processes 13
    2.1 System Modes 13
        2.1.1 Boot 14
        2.1.2 Standby 14
        2.1.3 Calibrations 15
        2.1.4 Power Down 16
        2.1.5 Sleep 17
        2.1.6 Reset 18
        2.1.7 RX Mode 19
        2.1.8 TX Mode 19
        2.1.9 FS Mode 20
    2.2 Startup Sequence 20
    2.3 Firmware Upgrade 21
        2.3.1 GetVersion 21
        2.3.2 EraseFlash 21
        2.3.3 WriteFlashEncrypted 22
    2.4 Mode Transitions & Timings 22
    2.5 Info Page 23
        2.5.1 EraseInfoPage 23
        2.5.2 WriteInfoPage 23
        2.5.3 ReadInfoPage 24
3. Host-Controller Interface 25
    3.1 Write Commands 25
    3.2 Read Commands 25
    3.3 Command Endianness 26
    3.4 Status Registers 27
        3.4.1 GetStatus 27
        3.4.2 Stat1 27
        3.4.3 Stat2 28
    3.5 BUSY 29
    3.6 Errors 30
        3.6.1 GetErrors 30
        3.6.2 ClearErrors 30
    3.7 Memory Access 31
        3.7.1 WriteRegMem32 31
        3.7.2 ReadRegMem32 31
        3.7.3 WriteRegMemMask32 32
        3.7.4 WriteBuffer8 32
        3.7.5 ReadBuffer8 33
        3.7.6 ClearRxBuffer 33
        3.7.7 GetRandomNumber 33

3.7.8 EnableSpiCrc 34
4. GPIOs 35
4.1 Interrupts 35
4.1.1 SetDioIrqParams 37
4.1.2 ClearIrq 37
4.2 RF Switch Control 38
4.2.1 SetDioAsRfSwitch 38
4.2.2 DriveDiosInSleepMode 39
4.3 Temperature Sensor 40
4.3.1 GetTemp 40
5. Power Distribution 41
5.1 Power Modes 41
5.1.1 SetRegMode 41
5.2 VBAT Measurement 42
5.2.1 GetVbat 42
5.3 Power-On-Reset and Brown-Out-Reset 43
5.4 Low Battery Detector 43
5.5 Over Current Protection 43
6. Clock Sources 44
6.1 RC Oscillators Clock References 44
6.2 High-Precision Clock References 44
6.2.1 32.768kHz Crystal 44
6.2.2 32MHz Crystal 44
6.2.3 32MHz TCXO 45
6.3 Commands 46
6.3.1 ConfigLfClock 46
6.3.2 SetTcxoMode 47
7. Radio 48
7.1 Overview 48
7.2 Commands 49
7.2.1 SetRfFrequency 49
7.2.2 SetRx 49
7.2.3 SetTx 50
7.2.4 AutoTxRx 51
7.2.5 SetRxTxFallbackMode 52
7.2.6 SetRxDutyCycle 53
7.2.7 StopTimeoutOnPreamble 55
7.2.8 GetRssiInst 55
7.2.9 GetStats 56
7.2.10 ResetStats 56
7.2.11 GetRxBufferStatus 57
7.2.12 SetRxBoosted 57
7.2.13 SetLoRaSyncWord 58
7.2.14 GetLoRaRxHeaderInfos 58
7.2.15 SetRssiCalibration 59
8. Modems 62

8.1 Modem Configuration 62
8.1.1 SetPacketType 63
8.1.2 GetPacketType 63
8.2 LoRa® Modem 64
8.2.1 LoRa Modulation Principle 64
8.2.2 LoRa Packet Format 65
8.2.3 Channel Activity Detection (CAD) 66
8.3 LoRa Commands 67
8.3.1 SetModulationParams 67
8.3.2 SetPacketParams 68
8.3.3 SetCad 68
8.3.4 SetCadParams 69
8.3.5 SetLoRaSynchTimeout 69
8.3.6 SetLoRaPublicNetwork 70
8.3.7 GetPacketStatus 70
8.4 (G)FSK Modem 71
8.4.1 (G)FSK Modulation Principle 71
8.4.2 (G)FSK Packet Engine 71
8.4.3 (G)FSK Packet Format 72
8.4.4 SX128x Compatibility 73
8.5 (G)FSK Commands 74
8.5.1 SetModulationParams 74
8.5.2 SetPacketParams 75
8.5.3 SetGfskSyncWord 76
8.5.4 SetPacketAdrs 76
8.5.5 SetGfskCrcParams 77
8.5.6 SetGfskWhitParams 77
8.5.7 GetPacketStatus 78
8.6 LR-FHSS Modulation 79
8.6.1 LR-FHSS Modulation Principle 79
8.7 LR-FHSS Commands 80
8.7.1 SetModulationParams 80
8.7.2 LrFhssBuildFrame 80
8.7.3 LrFhssSetSyncWord 82
8.7.4 Circuit Configuration for Long Range FHSS Transmission 82
8.8 Data Buffer 83
8.9 RSSI Functionality 83
9. Power Amplifiers 84
9.1 PA Supply Scheme 85
9.1.1 Low Power PA 86
9.1.2 High Power PA 87
9.2 PA Output Power 88
9.2.1 Low Power PA 88
9.2.2 High Power PA 89
9.2.3 High Frequency PA 90
9.3 PA Current Consumption 91

9.3.1 Low Power PA 91
9.3.2 High Power PA 92
9.3.3 High Frequency PA 94
9.4 Impedance Matching Networks 95
9.4.1 Multi-Band Operation 95
9.4.2 RF Switch Implementation 96
9.4.3 Direct-Tie Implementation 97
9.5 Commands 98
9.5.1 SetPaConfig 98
9.5.2 SetTxParams 99
10. Cryptographic Engine 100
10.1 Description 100
10.2 Cryptographic Keys Definition 100
10.3 Commands 102
10.3.1 CEStatus 102
10.3.2 CryptoSetKey 102
10.3.3 CryptoDeriveKey 103
10.3.4 CryptoProcessJoinAccept 104
10.3.5 CryptoComputeAesCmac 105
10.3.6 CryptoVerifyAesCmac 106
10.3.7 CryptoAesEncrypt01 107
10.3.8 CryptoAesEncrypt 108
10.3.9 CryptoAesDecrypt 109
10.3.10 CryptoStoreToFlash 110
10.3.11 CryptoRestoreFromFlash 110
10.3.12 CryptoSetParam 111
10.3.13 CryptoGetParam 111
11. LR1121 Provisioning 112
11.1 Description 112
11.2 Provisioning Commands 112
11.2.1 GetChipEui 112
11.2.2 GetSemtechJoinEui 113
11.2.3 DeriveRootKeysAndGetPin 114
11.3 Crypto Engine Use With LoRaWAN V1.1.x 116
11.4 Crypto Engine Use with LoRaWAN V1.0.x 117
12. Test Commands 118
12.1 Regulatory Overview 118
12.1.1 ETSI 118
12.1.2 FCC 118
12.2 Commands 119
12.2.1 SetTxCw 119
12.2.2 SetTxInfinitePreamble 119
13. List Of Commands 120
13.1 Register / Memory Access Operations 120
13.2 System Configuration / Status Operations 121
13.3 Radio Configuration / Status Operations 122

13.4 CryptoElement Configuration / Status Operations 124
13.5 Bootloader Commands 125
14. Revision History 126

List of Figures

Figure 1-1: LR1121 Block Diagram 12
Figure 2-1: LR1121 Modes and Transitions 13
Figure 2-2: Bootloader 14
Figure 3-1: Write Command Timing Diagram 25
Figure 3-2: Read Command Timing Diagram 25
Figure 3-3: GetVersion Write Capture 26
Figure 3-4: GetVersion Read Capture 26
Figure 3-5: BUSY Timing Diagram 29
Figure 5-1: LR1121 POR and BRN Functions 43
Figure 6-1: LR1121 Thermal Insulation on PCB Top Layer 45
Figure 6-2: TCXO Circuit Diagram 45
Figure 7-1: Radio 48
Figure 7-2: LR1121 Current Profile During RX Duty Cycle Operation 54
Figure 7-3: RX Duty Cycle Upon Preamble Detection 54
Figure 8-1: LoRa /(G)FSK / LR-FHSS Command Order 62
Figure 8-2: LoRa Signal Bandwidth 64
Figure 8-3: LoRa Packet Format 65
Figure 8-4: Fixed-Length Packet 72
Figure 8-5: Variable-Length Packet 72
Figure 8-6: LR-FHSS Spectral Plot Example 79
Figure 9-1: LR1121 Power Amplifiers 84
Figure 9-2: PA Block Diagram 85
Figure 9-3: Low Power PA VR_PA Voltage vs. *TxPower* *86*
Figure 9-4: High Power PA VR_PA Voltage vs. TxPower 87
Figure 9-5: Low Power PA Output Power vs. TxPower 88
Figure 9-6: HP PA Output Power vs. TxPower 89
Figure 9-7: High Frequency PA Output Power vs. TxPower 90
Figure 9-8: IDDTX vs TxPower, Low Power PA, DC-DC Configuration 91
Figure 9-9: IDDTX vs TxPower, Low Power PA, LDO Configuration 92
Figure 9-10: IDDTX vs TxPower, High Power PA, DC-DC Configuration 93
Figure 9-11: IDDTX vs TxPower, High Power PA, LDO Configuration 93
Figure 9-12: IDDTX vs TxPower, High Frequency PA, DC-DC Configuration 94
Figure 9-13: IDDTX vs TxPower, High Frequency PA, LDO Configuration 94
Figure 9-14: RF Switch, Double PA Operation 96
Figure 9-15: RF Switch, Single PA Operation (High Power PA Example) 96
Figure 9-16: Single Tie implementation: Only one PA Used (High Power PA Example) 97
Figure 9-17: Single Tie implementation: Both PAs Used (High Power PA Example) 97
Figure 11-1: Key Derivation Scheme For LoRaWAN 1.1.x 116
Figure 11-2: Key Derivation Scheme for LoRaWAN 1.0.x 117

List of Tables

Table 2-1: SetStandby Command 14
Table 2-2: CalibImage Command 15
Table 2-3: ISM Band Values 15
Table 2-4: Calibrate Command 16
Table 2-5: CalibParams Parameter 16
Table 2-6: SetSleep Command 17
Table 2-7: SleepConfig Parameter 17
Table 2-8: Sleep Mode Summary 17
Table 2-9: Reboot Command 18
Table 2-10: SetFsCommand 20
Table 2-11: GetVersion Command 21
Table 2-12: GetVersion Response 21
Table 2-13: EraseFlash Command 21
Table 2-14: WriteFlashEncrypted Command 22
Table 2-15: Mode Transitions and Timings 22
Table 2-16: EraseInfoPage Command 23
Table 2-17: WriteInfoPage Command 23
Table 2-18: ReadInfoPage Command 24
Table 2-19: ReadInfoPage Response 24
Table 3-1: GetStatus Command 27
Table 3-2: Stat1 Values 27
Table 3-3: Stat2 Values 28
Table 3-4: GetErrors Command 30
Table 3-5: GetErrors Response 30
Table 3-6: ClearErrors Command 30
Table 3-7: WriteRegMem32 Command 31
Table 3-8: ReadRegMem32 Command 31
Table 3-9: ReadRegMem32 Response 31
Table 3-10: WriteRegMemMask32 Command 32
Table 3-11: WriteBuffer8 Command 32
Table 3-12: ReadBuffer8 Command 33
Table 3-13: ReadBuffer8 Response 33
Table 3-14: ClearRxBuffer Command 33
Table 3-15: GetRandomNumber Command 33
Table 3-16: GetRandomNumber Response 33
Table 3-17: EnableSpiCrc Command 34
Table 4-1: Digital I/Os 35
Table 4-2: IrqToEnable Interruption Mapping 36
Table 4-3: SetDioIrqParams Command 37
Table 4-4: ClearIrq Command 37
Table 4-5: SetDioAsRfSwitch Command 38
Table 4-6: DriveDiosInSleepMode Command 39
Table 4-7: GetTemp Command 40
Table 4-8: GetTemp Response 40
Table 5-1: SetRegMode Command 41
Table 5-2: Power Regulation Options 41
Table 5-3: GetVbat Command 42
Table 5-4: GetVbat Response 42
Table 6-1: ConfigLfClock Command 46

Table 6-2: SetTcxoMode Command 47
Table 6-3: TCXO Supply Voltage Programming Values 47
Table 7-1: SetRfFrequency Command 49
Table 7-2: SetRx Command 49
Table 7-3: SetTx Command 50
Table 7-4: AutoTxRx Command 51
Table 7-5: SetRxTxFallbackMode Command 52
Table 7-6: SetRxDutyCycle Command 53
Table 7-7: StopTimeoutOnPreamble Command 55
Table 7-8: GetRssiInst Command 55
Table 7-9: GetRssiInst Response 55
Table 7-10: GetStats Command 56
Table 7-11: GetStats Response 56
Table 7-12: ResetStats Command 56
Table 7-13: GetRxBufferStatus Command 57
Table 7-14: GetRxBufferStatus Response 57
Table 7-15: SetRxBoosted Command 57
Table 7-16: SetLoRaSyncWord Command 58
Table 7-17: GetLoRaRxHeaderInfos Command 58
Table 7-18: GetLoRaRxHeaderInfos Response 58
Table 7-19: SetRssiCalibration Command 59
Table 7-20: Gain Tune Values 59
Table 7-21: Recommended Values for Reference EVK 59
Table 8-1: SetPacketType Command 63
Table 8-2: GetPacketType Command 63
Table 8-3: GetPacketType Response 63
Table 8-4: SetModulationParams Command 67
Table 8-5: SetPacketParams Command 68
Table 8-6: SetCad Command 68
Table 8-7: SetCadParams Command 69
Table 8-8: CadExitMode Parameter 69
Table 8-9: SetLoRaSynchTimeout Command 69
Table 8-10: SetLoRaPublicNetwork Command 70
Table 8-11: GetPacketStatus Command 70
Table 8-12: GetPacketStatus Response 70
Table 8-13: SetModulationParams Command 74
Table 8-14: Bandwidth Parameter 74
Table 8-15: SetPacketParams Command 75
Table 8-16: SetGfskSyncWord Command 76
Table 8-17: SetPacketAdrs Command 76
Table 8-18: SetGfskCrcParams Command 77
Table 8-19: SetGfskWhitParams Command 77
Table 8-20: GetPacketStatus Command 78
Table 8-21: GetPacketStatus Response 78
Table 8-22: SetModulationParams Command 80
Table 8-23: LrFhssBuildFrame Command 80
Table 8-24: Maximum user payload length, in bytes 81
Table 8-25: LrFhssSetSyncWord Command 82
Table 8-26: RSSI Information Origin and Meaning 83
Table 9-1: Optimized Settings for LP PA with the Same Matching Network 95
Table 9-2: Optimized Settings for HP PA with the Same Matching Network 95
Table 9-3: Optimized Settings for HF PA with the Same Matching Network 95
Table 9-4: SetPaConfig Command 98
Table 9-5: PaDutyCycle Parameter Allowed Ranges 98

Table 9-6: SetTxParams Command 99
Table 9-7: RampTime Values 99
Table 10-1: Cryptographic Keys Usage and Derivation 100
Table 10-2: CryptoSetKey Command 102
Table 10-3: CryptoSetKey Response 102
Table 10-4: CryptoDeriveKey Command 103
Table 10-5: CryptoDeriveKey Response 103
Table 10-6: CryptoProcessJoinAccept Command 104
Table 10-7: CryptoProcessJoinAccept Response 104
Table 10-8: CryptoComputeAesCmac Command 105
Table 10-9: CryptoComputeAesCmac Response 105
Table 10-10: CryptoComputeAesCmac Command Example 105
Table 10-11: CryptoComputeAesCmac Response Example 105
Table 10-12: CryptoVerifyAesCmac Command 106
Table 10-13: CryptoVerifyAesCmac Response 106
Table 10-14: CryptoAesEncrypt01 Command 107
Table 10-15: CryptoAesEncrypt01 Response 107
Table 10-16: CryptoAesEncrypt Command 108
Table 10-17: CryptoAesEncrypt Response 108
Table 10-18: CryptoAesDecrypt Command 109
Table 10-19: CryptoAesDecrypt Response 109
Table 10-20: CryptoStoreToFlash Command 110
Table 10-21: CryptoAesDecrypt Response 110
Table 10-22: CryptoRestoreFromFlash Command 110
Table 10-23: CryptoRestoreFromFlash Response 110
Table 10-24: CryptoSetParam Command 111
Table 10-25: CryptoSetParam Response 111
Table 10-26: CryptoGetParam Command 111
Table 10-27: CryptoGetParam Response 111
Table 11-1: GetChipEui Command 112
Table 11-2: GetChipEui Response 112
Table 11-3: GetSemtechJoinEui Command 113
Table 11-4: GetSemtechJoinEui Response 113
Table 11-5: DeriveRootKeysAndGetPin Command (Standard) 114
Table 11-6: DeriveRootKeysAndGetPin Response 114
Table 11-7: DeriveRootKeysAndGetPin Command (advanced) 115
Table 11-8: DeriveRootKeysAndGetPin Response (advanced) 115
Table 11-9: LoRaWAN 1.0.x vs. 1.1.x Security Correspondence Table 117
Table 12-1: ETSI Test Signals 118
Table 12-2: SetTxCw Command 119
Table 12-3: SetTxInfinitePreamble Command 119
Table 13-1: Register / Memory Access Operations 120
Table 13-2: System Configuration / Status Operations 121
Table 13-3: Radio Configuration / Status Operations 122
Table 13-4: CryptoElement Configuration / Status Operations 124
Table 13-5: Bootloader Commands 125
Table 14-1: Revision History 126

1. Introduction

1.1 Scope

This document provides complete information on how to use the LR1121 transceiver in an application. It covers both hardware and software aspects. For LR1121 functionalities and circuit specifications, refer to the LR1121 Datasheet.

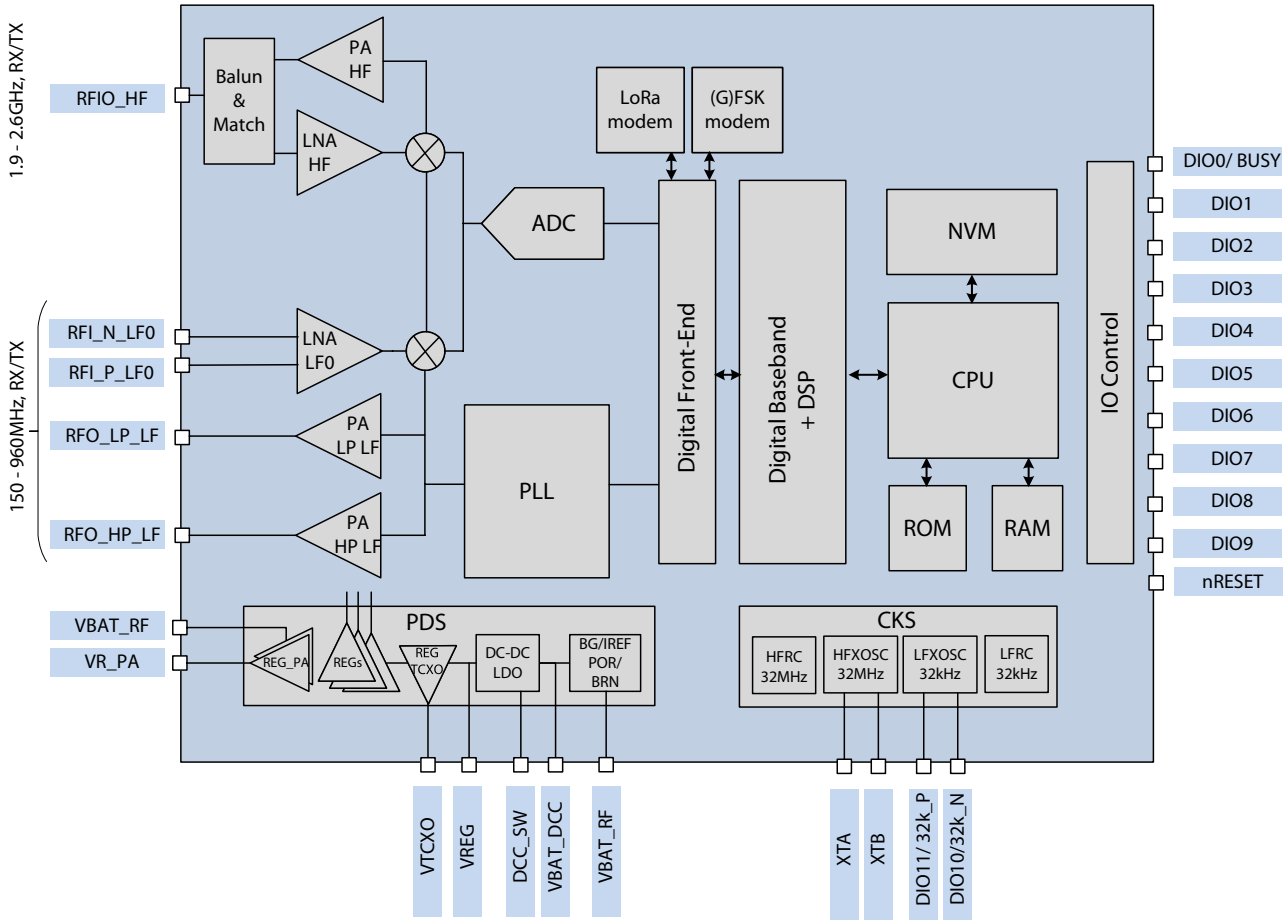

Figure 1-1: LR1121 Block Diagram

1.2 Overview

The LR1121 chip is a long range, ultra-low power transceiver that provides ISM sub-GHz, a global 2.4 GHz band, and 1.9GHz-2.2GHz S-Band communication targeting terrestrial and satellite connectivity. For LPWAN use cases, the LR1121 supports LoRa® and (G)FSK and Long Range-Frequency Hopping Spread Spectrum (LR-FHSS) modulations. The device is highly configurable over the 150MHz-960MHz, 1.9-2.1GHz Satellite band and 2.4GHz ISM bands to meet different application requirements utilizing the global LoRaWAN® Standard or proprietary protocols.

Besides the world-wide sub-GHz, 1.9-2.1GHz Satellite band and 2.4GHz transceiver capabilities, the LR1121 features a very low power multi-band front-end.

2. System Processes

2.1 System Modes

The LR1121 operating modes are shown in Figure 2-1: LR1121 Modes and Transitions:

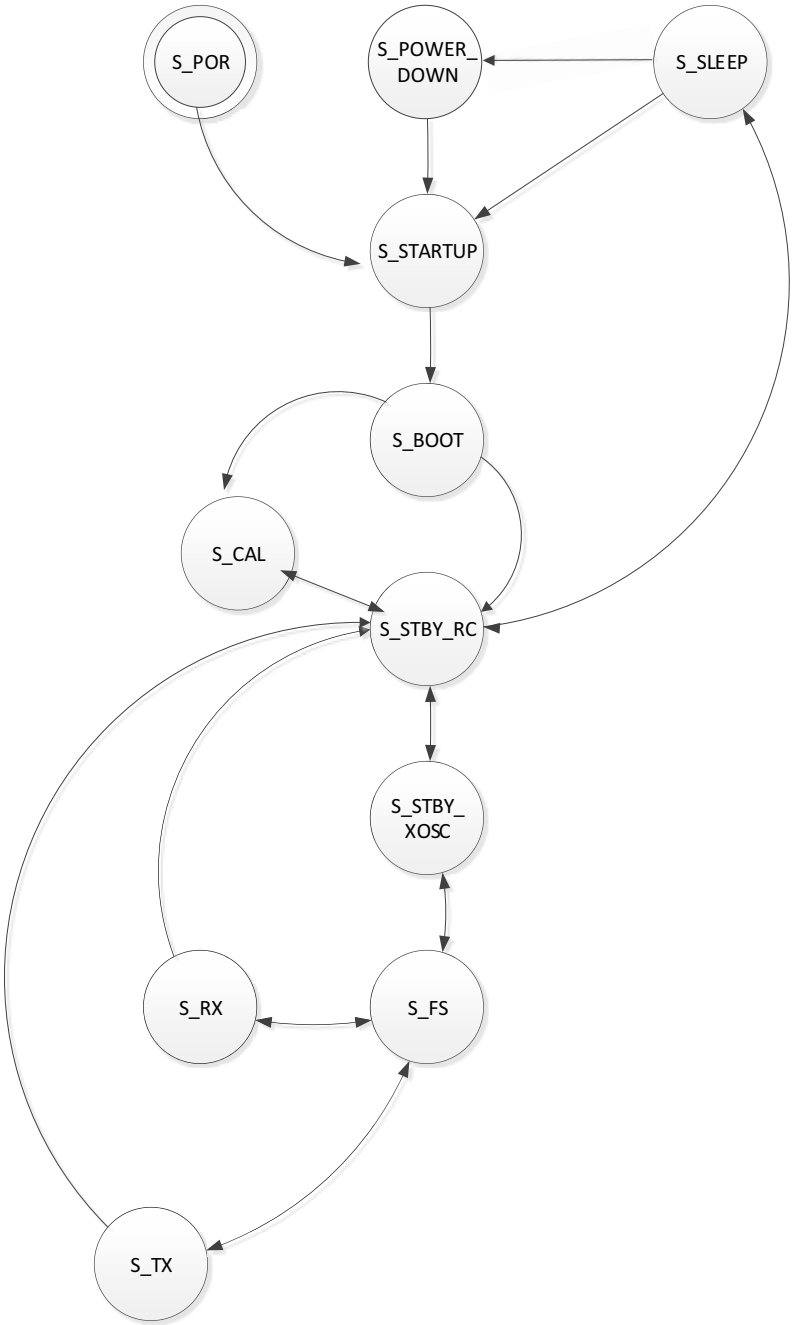

Figure 2-1: LR1121 Modes and Transitions

2.1.1 Boot

The bootloader is the first piece of software executed after power-on reset, and is located at the beginning of flash memory. The detailed description in AN1200.57 LR1110 Program Memory Update applies to all LR11xx devices.

Two main tasks are assigned to the bootloader:

- The first task checks if a valid firmware is loaded before jumping there.
- The second task loads a firmware: it receives encrypted chunks of data sent through the SPI and performs an on-the-fly decryption before writing data in flash.

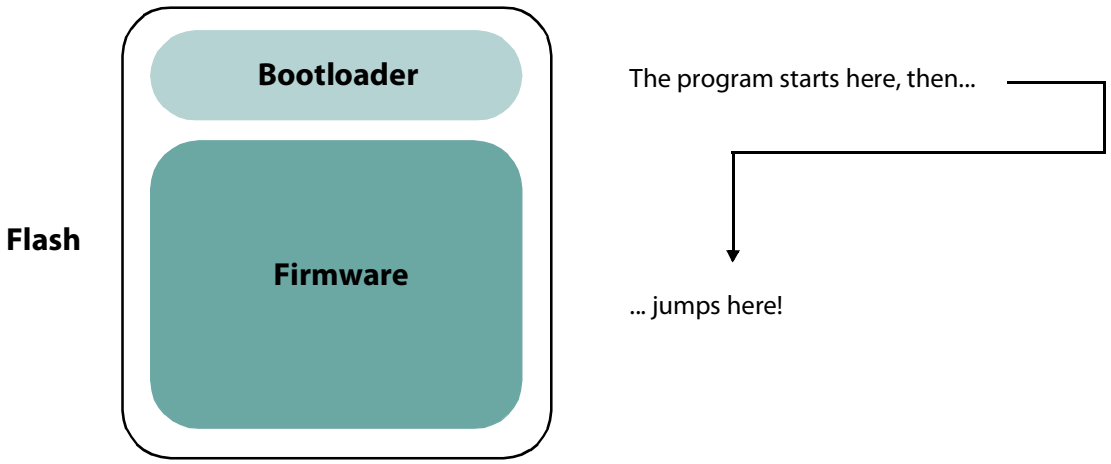

Figure 2-2: Bootloader

2.1.2 Standby

This is the default mode of the LR1121. It is the return state from all other modes (except for specific fall-back options), and the mode from which transitions to other modes are possible. All commands to configure the device should be issued in this mode.

Two clocks are available: either the internal 32MHz RC oscillator (Standby RC mode), or an external 32MHz crystal/TCXO (Standby Xosc mode). The RC clock is used by default for all automatic mode transitions. The crystal/TCXO clock allows faster transitions to other modes at the expense of a higher power consumption.

2.1.2.1 SetStandby

Command `SetStandby(...)` sets the device in standby mode with the chosen 32MHz oscillator.

Table 2-1: SetStandby Command

Byte	0	1	2
Data from Host	0x01	0x1C	StdbyConfig
Data to Host	Stat1	Stat2	IrqStatus(31:24)

StdbyConfig selects the oscillator used in standby mode:

- 0x00: Selects internal RC oscillator (Standby RC mode)
- 0x01: Selects external Xtal/TCXO oscillator (Standby Xosc mode)

2.1.3 Calibrations

During the startup sequence, the device firmware calibrates the low and high frequency RC oscillators, the PLL, the ADC, and the image rejection of the mixer at 915MHz. After the calibration procedure the device is set in Standby RC mode.

If operating at another frequency, the image calibration procedure must be restarted using command `CalibImage(...)`. An image calibration is advised after large temperature variations and optimal image rejection using command `CalibImage(...)`.

Image calibration is necessary if there is a frequency change > 10MHz, or a temperature change > 10°C.

2.1.3.1 CalibImage

The `CalibImage(...)` command launches an image calibration for the given range of frequencies *Freq1* and *Freq2* on the RFI_N/P_LF sub-GHz path.

Table 2-2: CalibImage Command

Byte	0	1	2	3
Data from Host	0x01	0x11	Freq1	Freq2
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

By default, the image calibration is made in the band 902 - 928MHz. Nevertheless, it is possible to request the device to perform a new image calibration at other frequencies. The frequencies are given in 4MHz steps (Ex: 900MHz -> 0xE1).

The calibration is valid for all frequencies between the two parameters. Typically, the user selects the parameters *Freq1* and *Freq2* from Table 2-3: ISM Band Values. The same frequency may be provided as *Freq1* and *Freq2* to perform a single calibration for the *Freq1* / *Freq2* value.

Table 2-3: ISM Band Values

Frequency Band [MHz]	Freq1	Freq2
430-440	0x6B	0x6E
470-510	0x75	0x81
779-787	0xC1	0xC5
863-870	0xD7	0xDB
902-928	0xE1 (default)	OxE9 (default)

The command operates in any mode. At the end of the calibration procedure, the device returns to Standby RC.

For other frequency bands, *Freq1* and *Freq2* values can be obtained using the following formula:

- *Freq1* = floor((fmin_mhz - 1)/4)
- *Freq2* = ceil((fmax_mhz + 1)/4)

In the case of POR, or when the device is recovering from power-down or sleep mode without retention, the image calibration is performed as part of the initial calibration process and for optimal image rejection in the band 902 - 928MHz. If a TCXO is fitted, the calibration fails.

2.1.3.2 Calibrate

The `Calibrate(...)` command calibrates the requested blocks defined by the *CalibParams* parameter.

Table 2-4: Calibrate Command

Byte	0	1	2
Data from Host	0x01	0x0F	CalibParams
Data to Host	Stat1	Stat2	IrqStatus(31:24)

Table 2-5: CalibParams Parameter

Bits	(7:6)	5	4	3	2	1	0
Name	RFU	PLL_TX	IMG	ADC	PLL	HF_RC	LF_RC

Command operates in any mode. At the end of the calibration procedure, the device returns to Standby RC.

2.1.4 Power Down

This is the lowest power consumption mode of the device. In this mode:

- All clocks are stopped, therefore no RTC is available
- There is no data retention, so device reconfiguration is necessary when leaving power down mode
- The BUSY signal is set to high, indicating to the host that the device is not ready to accept a command
- The device is put in power down mode with the `SetSleep(...)` command (refer to sleep mode description)

The device can exit this mode if NSS activity is detected.

Exiting this mode, the device performs a firmware restart, and sets the BUSY signal to low, indicating that the startup phase has been performed successfully and that the device is ready to accept a command.

2.1.5 Sleep

Sleep mode configures the LR1121 into a low power consumption mode between radio operations, while retaining the configuration register values and storing the firmware data in RAM. When the LR1121 is in sleep mode, the BUSY signal is set to 1 with a pull-up, all MISO SPI signals are high-Z, and all DIOs are in Hi-Z mode (see DriveDiosInSleepMode).

An optional 32kHz source can run either on the internal RC oscillator, or on the internal 32.768kHz oscillator driving an external crystal. The 32.768kHz crystal oscillator allows a faster transition to standby mode, at the expense of higher power consumption. In both cases, the RTC uses the 32kHz clock source to allow an automatic wake-up from Sleep mode.

2.1.5.1 SetSleep

`SetSleep(...)` puts the device in Powerdown or Sleep mode, and configures the timeout for automatic wake-up.

Table 2-6: SetSleep Command

Byte	0	1	2	3	4	5	6
Data from Host	0x01	0x1B	SleepConfig	SleepTime(31:24)	SleepTime(23:16)	SleepTime(15:8)	SleepTime(7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00

Table 2-7: SleepConfig Parameter

SleepConfig bit	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Definition	RFU	RFU	RFU	RFU	RFU	RFU	Wakeup	Retention

- *SleepConfig* defines in which sleep mode the device is put, and if it wakes up after a given time on the RTC event:
  - *Retention* (bit 0) defines if the device configuration and firmware data are retained:
    - 1: 8kB of memory used for device state and firmware data retention
    - 0: No data retention (Power Down mode)
  - *Wakeup* (bit 1) determines if the device wakes up after a given time on the RTC event:
    - 1: Automatic wake-up enabled. Device automatically goes in Standby mode with RC oscillator, at end of *SleepTime* timer. Configure the 32kHz clock source using command `ConfigLfClock (...)` for modem applications.
    - 0: Automatic wake-up disabled
  - Other bits are RFU and should be set to 0
- *SleepTime*: sleep time in number of 32.768kHz clock cycles, prior to automatic wake-up. Therefore, the sleep time can vary from 0ms to 36.4 hours in steps of 30.52us

The device exits this mode upon the falling edge on the NSS signal even when automatic wakeup is enabled. Exiting this mode, the device performs a firmware restart. When the BUSY signal is set to low, it indicates that the startup phase has been performed successfully, and that the device is ready to accept a command.

The following table summarizes the sleep modes according to Retention and Wakeup bits configuration, with their current consumption (RC /XTAL**)** and Standby transitions times (indicative values, for comparison only).

Table 2-8: Sleep Mode Summary

Retention	Wakeup	Datasheet	Indicative Consumption (uA)	Indicative Transition to Stby (ms)
0	0	Powerdown	IDDPDN	30
0	1	Sleep	IDDSL1 / IDDSL2	30
1	0	RFU	-	-
1	1	Sleep w/ 8kB retention	IDDSL3A / IDDSL4A	<1

2.1.6 Reset

Four reset sources are available to trigger a LR1121 restart and execute the startup sequence: Power-On-Reset / Brown-Out Reset (POR/BRN), NRESET, and `Reboot(...)` command.

The BUSY signal is kept high during each one of the reset procedures, and returns to low when the restart procedure is finished. At the end of the restart procedure, the device is ready to accept commands, it goes into Standby mode with RC oscillator on (STBY_RC). All device context is lost during this operation, so the device must be re-configured and recalibrated. POR/BRN and NRESET also trigger an authentication of the internal firmware.

2.1.6.1 Power-On-Reset and Brown-Out Reset

The LR1121 performs a restart if either of the following occurs:

- The battery voltage rises above the Power-On-Reset (POR) level
- The battery voltage temporarily drops below the Brown-Out Reset (BRN) level

Both POR and BRN trigger a full restart of the internal firmware. The *Status* field of the *Stat2* status variable indicates if a POR or BRN occurred.

Please refer to 5.3 Power-On-Reset and Brown-Out-Reset for addition information on the POR and BRN.

2.1.6.2 NRESET

Putting the NRESET signal to low for at least 100µs restarts the LR1121. The restart is equivalent to a Power-On Reset, and the device follows the same restart sequence.

2.1.6.3 Reboot

Command `Reboot(...)` triggers a restart of the LR1121 firmware.

Table 2-9: Reboot Command

Byte	0	1	2
Data from Host	0x01	0x18	StayInBootLoader
Data to Host	Stat1	Stat2	IrqStatus (31:24)

StayInBootLoader determines the type of reboot:

- 0: Performs a software restart
- 3: The boot-loader does not execute the firmware in flash, but allows firmware upgrades
- Other values are RFU

The 32kHz clock's configuration is kept on a Reboot. Command `ConfigLfClock(...)` modifies the 32kHz clock configuration.

2.1.7 RX Mode

RX mode receives incoming RF packets on the RFI_N_LF0/RFI_P_LF0 pins in the sub-GHz band (150-960MHz) , and the RFIO_HF pin in the 1.9-2.1GHz Satellite band and 2.4GHz band (2400-2500MHz), using one of the modems (LoRa or (G)FSK). The device enters RX mode using command `SetRx(...)`. At packet reception, an RX_DONE interrupt is generated, and the received data is stored in the device data buffer. The RX operation can be automatically terminated after a packet reception, duty-cycled or infinite, based on the application requirements.

While in RX mode, the LR1121 operates in different sub-modes:

- Continuous mode, the device remains in RX mode and looks for incoming packets until the host requests a different mode.
- Single mode, the device automatically returns to a configured mode (Standby RC by default) after a packet reception.
- Single with timeout mode, the device automatically returns to a configured mode (Standby RC by default) after a packet reception or after the given timeout. If a sync word (G)FSK or a LoRa header is detected, the timeout is stopped.
- RX Duty Cycle mode, the device goes periodically into RX mode to receive a packet before going back to Sleep mode, until a packet is received.
- AutoTx mode (auto transmits a packet a given time after packet reception), the device goes into an intermediary mode for the requested time after a packet reception, before entering TX mode to transmit the packet.

2.1.8 TX Mode

TX mode transmits RF packets using the selected sub-GHz PA on the RFO_LP_LF or RFO_HP_LF pins in the sub-GHz band (150-960MHz) , and the RFIO_HF pin in the 1.9-2.1GHz Satellite band and 2.4GHz band (2400-2500MHz), using the modems (LoRa, (G)FSK, or LR-FHSS).

After ramping-up the PA, the LR1121 transmits the data buffer at the given frequency, PA, output power and packet and modulation configurations. When the last bit of the packet has been sent, a TX_DONE interrupt is generated, the PA regulator is ramped down, the selected PA is switched OFF and the device goes back to Standby RC or Xosc mode, depending on the *FallBackMode* configuration.

In TX mode, the BUSY signal goes low as soon as the PA has ramped-up and transmission of the preamble starts.

While in TX mode, the LR1121 operates in different sub-modes:

- Single mode, the device automatically returns to a configured mode (Standby RC by default) after a packet transmission.
- Single mode with timeout, the device automatically returns to a configured mode (Standby RC by default) after a packet transmission or after the given timeout.
- AutoRX mode, (automatically goes into RX mode a given time after transmitting a packet) the device goes into an intermediary mode for the requested time after a packet transmission, before entering RX mode for reception of a packet or until the configured timeout.
- Continuous Wave mode (CW mode), the device indefinitely transmits an unmodulated carrier at the predefined frequency until another command is issued to change the mode.
- Infinite preamble mode: the device indefinitely transmits an infinite preamble of the configured modulation.

2.1.9 FS Mode

Frequency Synthesis (FS) mode is an intermediate mode between standby mode and the RX or TX modes, where the PLL and the associated regulators are switched on. The BUSY signal goes low as soon as the PLL is locked.

2.1.9.1 SetFs

Command `SetFs(...)` sets the device in Frequency Synthesis mode.

Table 2-10: SetFsCommand

Byte	0	1
Data from Host	0x01	0x1D
Data to Host	Stat1	Stat2

2.2 Startup Sequence

At power-up or after a reset, the device initiates its startup phase.

- The BUSY signal is set to high, indicating that the device is busy and cannot accept a command.
- When the power management unit and RC oscillator become available, the embedded CPU starts and executes the internal firmware.
- At the end of the startup sequence, the device is set in Standby RC mode, the BUSY signal goes low and the device accepts commands.

2.3 Firmware Upgrade

The LR1121 can be upgraded with a new firmware image, supplied by Semtech. Complete details are provided in the Application note AN1200.57 "LR1110: Upgrade of the Program Memory", available from the Semtech website. The related commands are described hereafter.

2.3.1 GetVersion

Command `GetVersion()` returns the version of the LR1121.

Table 2-11: GetVersion Command

Byte	0	1
Data from Host	0x01	0x01
Data to Host	Stat1	Stat2

Table 2-12: GetVersion Response

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	HW Version	Use Case	FW Major	FW Minor

- *HW Version* is the version of the LR1121 hardware
- *Use Case* specifies the usage of the device:
  - 0x01: LR1110
  - 0x02: LR1120
  - 0x03: LR1121
  - 0xDF: Bootloader mode
- *FW Major* + *FW Minor i*s the version of the LR1121 internal firmware stored in flash memory

2.3.2 EraseFlash

In bootloader mode **only**, the `EraseFlash(...)` command must be used before a new image is written to the device.

If executed in another mode, the command status in *Stat1* is set to *P_ERR*.

Table 2-13: EraseFlash Command

Byte	0	1
Data from Host	0x80	0x00
Data to Host	Stat1	Stat2

2.3.3 WriteFlashEncrypted

The `WriteFlashEncrypted()` command writes a new firmware image to the LR1121. More details are supplied in the AN1200.57 Application Note.

Table 2-14: WriteFlashEncrypted Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x80	0x03	Offset (31:24)	Offset (23:16)	Offset (15:8)	Offset (7:0)	Data1 (31:24)	Data1 (23:16)
Data to Host	Stat1	Stat2	0x00	0x00	0x00	0x00	0x00	0x00

Byte	8	9	10	11			...	4*N+5
Data from Host	Data1 (15:8)	Data1 (7:0)	Data2 (31:24)	Data2 (23:16)	Data2 (15:8)	Data2 (7:0)	...	DataN (7:0)
Data to Host	0x00	0x00	0x00	0x00	0x00	0x00	...	0x00

Where N must range from 1 to 32 inclusive.

2.4 Mode Transitions & Timings

Table 2-15: Mode Transitions and Timings lists the main mode transitions of the LR1121. Please refer to Figure 2-1: LR1121 Modes and Transitions for a representation of the LR1121 modes and mode transitions.

Table 2-15: Mode Transitions and Timings

Transition	T_{SW} Mode Typical value (µs)
POR to STBY_RC	237e3
SLEEP to STBY_RC (no data retention)	
if XOSC = XTAL	52e3
if XOSC = TCXO	10e3
SLEEP to STBY_RC (with data retention)	<1000
STBY_RC to STBY_XOSC	43
STBY_XOSC to FS for sub-GHz	50
STBY_XOSC to FS for 2.4 GHz band	80
STBY_XOSC to TX for sub-GHz	142
STBY_XOSC to TX for 2.4 GHz band	172
FS to RX (LoRa®, (G)FSK)	39
FS to TX	102

Table 2-15: Mode Transitions and Timings (Continued)

Transition	T_{SW}Mode Typical value (µs)
RX to FS	25
RX to TX	118

2.5 Info Page

The LR11xx flash memory contains an Info Page of 512 32-bit words, on which the host controller can store user data.

Note: the maximum number of LR11x flash memory R/W accesses is 10k cycles.

2.5.1 EraseInfoPage

The command `EraseInfoPage(...)` erases the requested Info Page in the LR11xx flash memory.

Table 2-16: EraseInfoPage Command

Byte	0	1	2
Data from Host	0x01	0x21	InfoPage
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *InfoPage*: can only be 1
- Other values are RFU

2.5.2 WriteInfoPage

The command `WriteInfoPage(...)` writes a block of 32-bit words in the Info Page flash memory space starting at a specific word address.

Table 2-17: WriteInfoPage Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x01	0x22	InfoPage	InfoWordAddr (15:8)	InfoWordAddr (7:0)	data1 (31:24)	data1 (23:16)	data1 (15:8)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0

Byte	8	9	10	...	N*4+4
Data from Host	data1 (7:0)	data2 (31:24)	data2 (23:16)	...	dataN (7:0)
Data to Host	0	0	0	...	0

- *InfoPage*: can only be 1. Other values are RFU.
- *InfoWordAddr*: starting address of the data block in the Info Page. The address is auto incremented after each data word so that data is stored in contiguous memory locations.
- *data*: data block to be written in the Info Page. Data length should be a multiple of 4. The maximum value for N is 64.

2.5.3 ReadInfoPage

The command `ReadInfoPage(...)` reads a block of 32-bit words in Info Page flash memory space starting at a specific word address.

Table 2-18: ReadInfoPage Command

Byte	0	1	2	3	4	5
Data from Host	0x01	0x23	InfoPage	InfoWordAddr (15:8)	InfoWordAddr (7:0)	InfoReadLen
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

Table 2-19: ReadInfoPage Response

Byte	0	1	2	3	4	5	6	...	N*4
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00	0x00	...	0x00
Data to Host	Stat1	data1 (31:24)	data1 (23:16)	data1 (15:8)	data1 (7:0)	data2 (31:24)	data2 (23:16)	...	data2 (7:0)

- *InfoPage*: can only be 1. Other values are RFU.
- *InfoWordAddr*: starting address of the data block in the Info Page. The address is auto incremented after each data word so that data is stored in contiguous memory locations.
- *InfoReadLen*: number of words to read. The maximum value for *InfoReadLen* is 64.

Returns CMD_PERR if the user tries to read data after the end of page.

3. Host-Controller Interface

The LR1121 exposes an API which allows the Host controller to communicate with the LR1121 through a set of SPI commands / responses. The BUSY signal is used as a handshake to indicate if the LR1121 is ready to accept a command. Therefore, it is necessary to check the status of BUSY prior to sending a command.

3.1 Write Commands

During write commands, the LR1121 returns the status registers and the interrupt registers to the host on the MOSI pin, depending on the length of the command opcode and arguments.

The host sends a 16-bit opcode followed by the required arguments.

The BUSY signal is automatically asserted on the falling edge of the NSS.

Once the LR1121 finishes processing the command, the BUSY signal is de-asserted to indicate that the device is ready to accept another command.

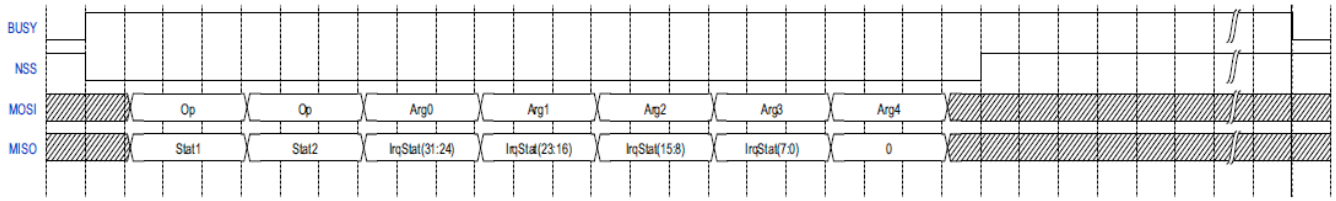

Figure 3-1: Write Command Timing Diagram

3.2 Read Commands

Specific Read commands retrieve data from LR1121, such as internal status results.

The host sends a 16-bit opcode, followed by arguments if required.

The BUSY signal is automatically asserted on the falling edge of the NSS.

Once the LR1121 has finished preparing the requested data, the BUSY signal is de-asserted.

The host can then read back the data by sending NOPs (0x00 bytes) to shift out the data on the SPI.

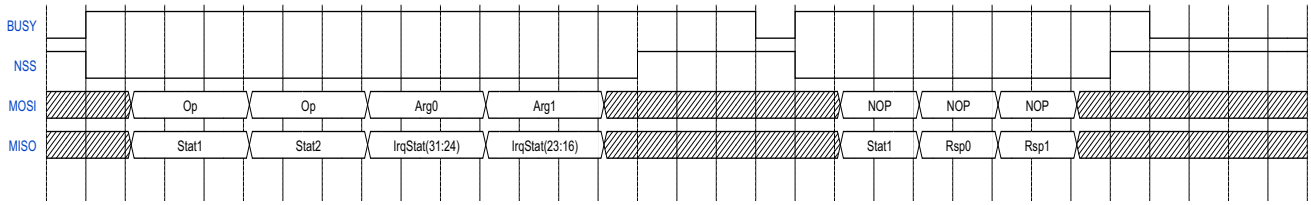

Figure 3-2: Read Command Timing Diagram

3.3 Command Endianness

The following figures are examples of an SPI transaction for command `GetVersion(...)`.

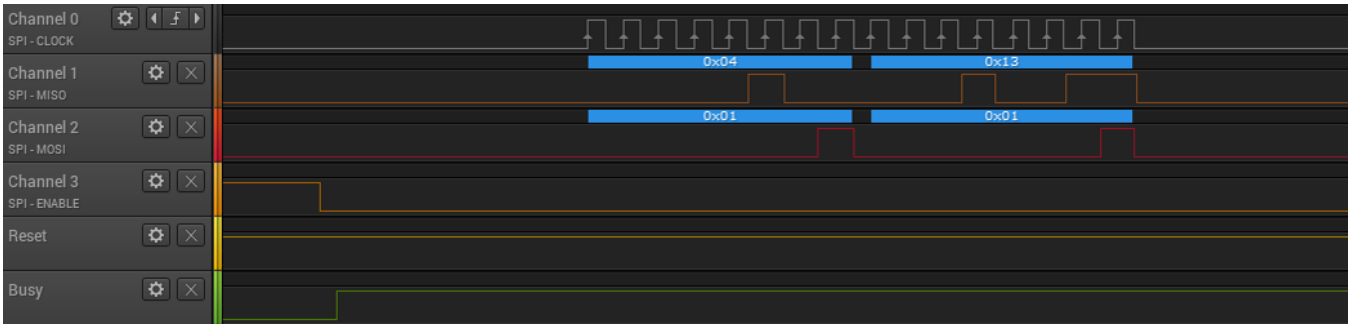

Figure 3-3: GetVersion Write Capture

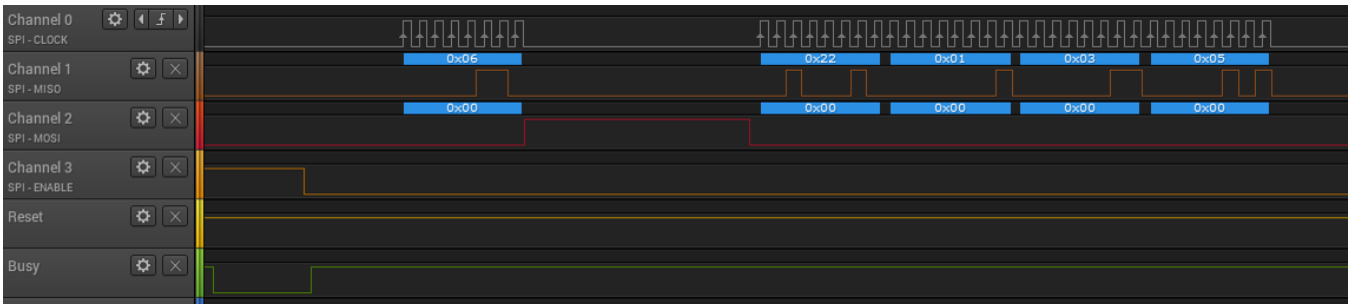

Figure 3-4: GetVersion Read Capture

3.4 Status Registers

The LR1121 features 2 status variables *Stat1* and *Stat2*, which determine the status of the LR1121 (the last command sent, of the device interrupts, of the device operating mode, and of the bootloader) without the need for the host to send a specific command. Command `GetStatus(...)` returns the status registers.

Stat1 and *Stat2* are always sent when the host issues a command. Only Stat1 is sent back when retrieving data from the LR1121.

3.4.1 GetStatus

Command `GetStatus(...)` returns the LR1121 status flags *stat1* and *stat2*, and the LR1121 interrupt flags. It then clears the stat2 *ResetStatus* field.

Note that there is an alternate method for retrieving this status information: If a sequence of zeros (NOP) is written to the MOSI signal, the LR1121 clocks out either the response to the last command, or the status information if no response is pending. If the SPI read command is designed to write zero /NOP on the MOSI signal, then this provides a way to obtain the status information by using an ordinary SPI read command. Note, however, that this method does not clear the ResetStatus field. See stat1/CMD_DAT for more information.

Table 3-1: GetStatus Command

Byte	0	1	2	3	4	5
Data from Host	0x01	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

3.4.2 Stat1

Table 3-2: Stat1 Values

Bits	(7:4)	(3:1)	(0)
Name	RFU	Command Status	Interrupt Status

- *Command Status* indicates the status of the last command sent by the host:
  - 0: CMD_FAIL: The last command could not be executed
  - 1: CMD_PERR: The last command could not be processed (wrong opcode, arguments). It is possible to generate an interrupt on DIO if a command error occurred
    Any attempt to use Wi-Fi or GNSS functions returns this code
  - 2: CMD_OK: The last command was processed successfully
  - 3: CMD_DAT: The last command was successfully processed, and data is currently transmitted instead of IRQ status
  - 4-7: RFU
- *Interrupt Status* indicates if an LR1121 system interrupt was raised:
  - 0: No interrupt active
  - 1: At least 1 interrupt active

3.4.3 Stat2

Table 3-3: Stat2 Values

Bits	(7:4)	(3:1)	(0)
Name	Reset Status	Chip Mode	Bootloader

- *Reset Status* indicates the origin of a LR1121 reset:
  - 0: Cleared (no active reset)
  - 1: Analog reset (Power On Reset, Brown-Out Reset)
  - 2: External reset (NRESET pin)
  - 3: System reset
  - 4: Watchdog reset
  - *5*: Wakeup NSS toggling
  - 6: RTC restart
  - 7: RFU
- *Chip Mode* indicates the current mode of the LR1121:
  - 0: Sleep
  - 1: Standby with RC Oscillator
  - 2: Standby with external Oscillator
  - 3: FS
  - 4: RX
  - 5: TX
  - 6: RFU
  - 7: RFU
- *Bootloader*:
  - 0: currently executes from boot-loader
  - 1: currently executes from flash.The *ResetStatus* field is cleared on the first `GetStatus()` command after a reset. It is not cleared by any other command

3.5 BUSY

DIO0 is used for the BUSY signal: the BUSY signal is set high when a command is being processed, and when the device is not ready to accept a new command. The timing diagram of the BUSY signal is as follows:

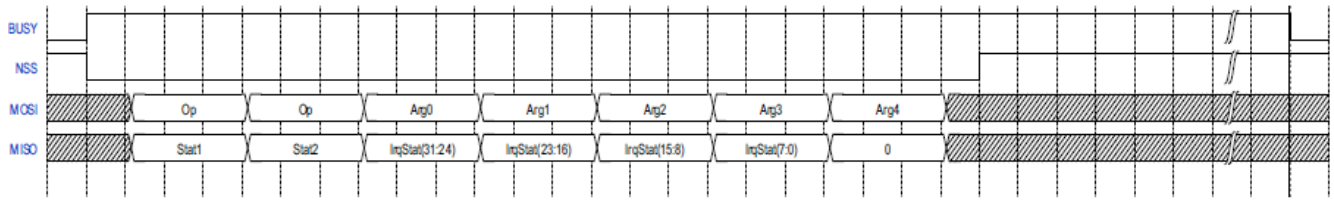

Figure 3-5: BUSY Timing Diagram

The amount of time the BUSY signal stays high after the end of rising edge of NSS ($T_{SW\ Mode}$) depends on the nature of the command.

The most common switching times $T_{SW\ Mode}$ are indicated in Section 2.4 "Mode Transitions & Timings" on page 22 .

3.6 Errors

3.6.1 GetErrors

Command `GetErrors(...)` returns the pending errors that occurred since the last *ClearErrors(...)*, or the circuit startup.

It is possible to generate an interrupt on DIO9 or DIO11 when an error occurs. The error cannot be masked.

Table 3-4: GetErrors Command

Byte	0	1
Data from Host	0x01	0x0D
Data to Host	Stat1	Stat2

Table 3-5: GetErrors Response

Byte	0	1	2
Data from Host	0x00	0x00	0x00
Data to Host	Stat1	ErrorStat(15:8)	ErrorStat(7:0)

ErrorStat contains all the possible error flags that could occur during chip operations:

- Bit 0: LF_RC_CALIB_ERR. Calibration of low frequency RC was not done. To fix it redo a calibration.
- Bit 1: HF_RC_CALIB_ERR. Calibration of high frequency RC was not done. To fix it redo a calibration.
- Bit 2: ADC_CALIB_ERR. Calibration of ADC was not done. To fix it redo a calibration.
- Bit 3: PLL_CALIB_ERR. Calibration of maximum and minimum frequencies was not done. To fix it redo the PLL calibration.
- Bit 4: IMG_CALIB_ERR. Calibration of the image rejection was not done. To fix it redo the image calibration.
- Bit 5: HF_XOSC_START_ERR. High frequency XOSC did not start correctly. To fix it redo a reset, or send `SetTcxoCmd(...)` if a TCXO is connected and redo calibrations.
- Bit 6: LF_XOSC_START_ERR. Low frequency XOSC did not start correctly. To fix it redo a reset.
- Bit 7: PLL_LOCK_ERR. The PLL did not lock. This can come from too high or too low frequency configuration, or if the PLL was not calibrated. To fix it redo a PLL calibration, or use other frequencies.
- Bit 8: RX_ADC_OFFSET_ERR. Calibration of ADC offset was not done. To fix it redo a calibration.
- Bit 9-15: RFU.

3.6.2 ClearErrors

Command `ClearErrors(...)` clears all errors flags pending in the device. The error flags cannot be cleared individually.

Table 3-6: ClearErrors Command

Byte	0	1
Data from Host	0x01	0x0E
Data to Host	Stat1	Stat2

3.7 Memory Access

3.7.1 WriteRegMem32

Command `WriteRegMem32(...)` writes blocks of 32-bit words in register/memory space starting at a specific address.

Table 3-7: WriteRegMem32 Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x01	0x05	Addr (31:24)	Addr (23:16)	Addr (15:8)	Addr (7:0)	Data1 (31:24)	Data1 (23:16)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00

Byte	8	9	10	...	4*N +5
Data from Host	Data1 (15:8)	Data1 (7:0)	Data2 (31:24)	...	DataN (7:0)
Data to Host	0x00	0x00	0x00	...	0x00

- The address is auto incremented after each data byte so that data is stored in contiguous register/memory locations.
- The value of N is maximum 64.

3.7.2 ReadRegMem32

Command `ReadRegMem32(...)` reads blocks of 32-bit words in register/memory space starting at a specific address.

Table 3-8: ReadRegMem32 Command

Byte	0	1	2	3	4	5	6
Data from Host	0x01	0x06	Addr (31:24)	Addr (23:16)	Addr (15:8)	Addr (7:0)	Len
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00

Table 3-9: ReadRegMem32 Response

Byte	0	1	2	3	4	5	...	4*N
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	Data1 (31:24)	Data1 (23:16)	Data1 (15:8)	Data1 (7:0)	Data2 (31:24)	...	DataN (7:0)

- The address is auto incremented after each data byte so that data is read from contiguous register locations
- *Len* is the number of words to read, and is maximum 64

3.7.3 WriteRegMemMask32

Command `WriteRegMemMask32(...)` reads/modifies/writes the masked bits (Mask bits = 1) of a single 32-bit word in register/memory space at the specified address.

Table 3-10: WriteRegMemMask32 Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x01	0x0C	Addr (31:24)	Addr (23:16)	Addr (15:8)	Addr (7:0)	Mask (31:24)	Mask (23:16)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00

Byte	8	9	10	11	12	13
Data from Host	Mask (15:8)	Mask (7:0)	Data (31:24)	Data (23:16)	Data (15:8)	Data (7:0)
Data to Host	0x00	0x00	0x00	0x00	0x00	0x00

3.7.4 WriteBuffer8

Command `WriteBuffer8`(...) writes a block of bytes into the radio TX buffer.

Table 3-11: WriteBuffer8 Command

Byte	0	1	2	3	4	5	6	...	N+1
Data from Host	0x01	0x09	Data1	Data2	Data3	Data4	Data5	...	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	...	0x00

- *Data*: N bytes of data. The value of N is maximum 255

3.7.5 ReadBuffer8

Command `ReadBuffer8(...)` reads a block of *Len* bytes in the radio RX buffer starting at a specific *Offset*. RX buffer must be implemented as a ring buffer.

Table 3-12: ReadBuffer8 Command

Byte	0	1	2	3
Data from Host	0x01	0x0A	Offset (7:0)	Len (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

Table 3-13: ReadBuffer8 Response

Byte	0	1	2	3	...	N
Data from Host	0x00	0x00	0x00	0x00	...	0x00
Data to Host	Stat1	Data1	Data2	Data3	...	DataN

3.7.6 ClearRxBuffer

Command `ClearRxBuffer(...)` clears all data in the radio RX buffer. It writes '0' over the whole Rx buffer. It is mainly used for debug purposes to ensure the data in the RX buffer is not from the previous packet.

Table 3-14: ClearRxBuffer Command

Byte	0	1
Data from Host	0x01	0x0B
Data to Host	Stat1	Stat2

3.7.7 GetRandomNumber

This command gets a 32-bit random number. This is not used for security purposes.

Table 3-15: GetRandomNumber Command

Byte	0	1
Data from Host	0x01	0x20
Data to Host	Stat1	Stat2

Table 3-16: GetRandomNumber Response

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	RandomNo(31:24)	RandomNo(23:16)	RandomNo(15:8)	RandomNo(7:0)

3.7.8 EnableSpiCrc

This command enables / disables an 8-bit CRC on the Serial Peripheral Interface.

CRC generation uses a polynomial generator 0x65 (reversed reciprocal), initial value 0xFF. The CRC is calculated on all data received on the MOSI signal (including Opcode) and on all data sent on the MISO signal (including all status).
This command is always protected by the CRC:

- To enable the CRC, the CRC must already be appended in this command. As an example: the whole command to enable is 0x01 0x28 0x01 0x20.
- To disable the CRC, the CRC must be appended to the command since it is still enabled. As an example: the whole command to disable is 0x01 0x28 0x00 0x1C.

Table 3-17: EnableSpiCrc Command

Byte	0	1	2	3
Data from Host	0x01	0x28	Enable	CRC
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

- *Enable*: enables / disables an 8-bit CRC on the SPI interface:
  - 0: Disabled. No CRC is expected or sent on the SPI (default)
  - 1: Enabled. A CRC is expected and sent on the SP.
  - Other values are RFU
- *CRC*: cyclic redundancy check

4. GPIOs

The LR1121 features 13 digital I/Os.

Table 4-1: Digital I/Os

I/O	Description
DIO0	Used for BUSY signalling, and is mandatory to properly handle the host controller interface.
DIO1 to DIO4	Dedicated to the SPI interface signals NSS, SCK, MOSI and MISO respectively.
DIO5, DIO6, DIO7	Can control external RF switches or LNAs on the RFIO-HF and sub-GHz RF paths.
DIO8	Can control external RF switches or LNAs on the RFIO-HF and sub-GHz RF paths. Can be used as a 32.768kHz source to host controller if a 32.768kHz crystal oscillator is connected to DIO10 and DIO11.
DIO9	Dedicated to LR1121 interrupts. It is recommended to connect DIO9 to the host controller for the lowest-power applications. DIO11 can be used as another interrupt pin if no 32.768kHz crystal oscillator is used.
DIO10	Can control external RF switches or LNAs on the RFIO-HF and sub-GHz RF paths. Can be used as connection pins for an external 32.768kHz crystal oscillator as an RTC source.
DIO11	Can be used as connection pins for an external 32.768kHz crystal oscillator as an RTC source. Can be used as an input pin if the 32.768kHz signal is fed by the host controller. In this case DIO10 must be left unconnected. Can be used as another interrupt pin if no 32.768kHz crystal oscillator is used.
NRESET	Can cancel on-going functions of the LR1121, and reset all HW and FW. Although a device restart is also possible through host controller commands, it is recommended to allow the host controller to control this signal.

4.1 Interrupts

The LR1121 features numerous interrupt sources, allowing the host to react to a large variety of events in the LR1121 system without the need to poll registers, therefore allowing power-optimized applications.

The LR1121 interrupts are multiplexed on the DIO9 and/or DIO11 pin. When the application receives an interrupt, it can determine the source by using command `GetStatus(...)`. The interrupt can then be cleared using the `ClearIrq(...)` command.

Command `SetDioIrqParams(...)` configures which interrupt signal should be activated on the DIO9 and/or DIO11 interrupt pins.

The status of the LR1121 interrupts can be read using command `GetStatus(...)`, but they are also returned by the LR1121 to the host on the MISO signal during the SPI transactions via the *IrqStatus* bytes, simultaneously with the command arguments. Therefore, the number of *IrqStatus* sent by the LR1121 during the SPI commands depends on the number of arguments. Refer to Section 3. Host-Controller Interface for additional information.

The interrupts mapping table *IrqToEnable* is as follows:

Table 4-2: IrqToEnable Interruption Mapping

Bit	Interrupt	Description
0	RFU	RFU
1	RFU	RFU
2	TxDone	Packet transmission completed
3	RxDone	Packet received
4	PreambleDetected	Preamble detected
5	SyncWordValid / HeaderValid	Valid sync word / LoRa® header detected
6	HeaderErr	LoRa header CRC error
7	Err	Packet received with error. LoRa: Wrong CRC received (G)FSK: CRC error
8	CadDone	LoRa Channel activity detection finished
9	CadDetected	LoRa Channel activity detected
10	Timeout	RX or TX timeout
11	LrFhssHop	LR-FHSS intra-packet hopping
12-18	RFU	RFU
19-20	RFU	RFU
21	LBD	Low Battery Detection
22	CmdError	Host command error
23	Error	An error other than a command error occurred (see GetErrors)
24	FskLenError	IRQ raised if the packet was received with a length error
25	FskAddrError	IRQ raised if the packet was received with an address error
26-31	-	RFU

4.1.1 SetDioIrqParams

Command `SetDioIrqParams(...)` configures which interrupt signal should be activated on the DIO9 and/or DIO11 interrupt pin (referred to as IRQ pin 1 and/or 2).

Table 4-3: SetDioIrqParams Command

Byte	0	1	2	3	4	5
Data from Host	0x01	0x13	Irq1ToEnable (31:24)	Irq1ToEnable (23:16)	Irq1ToEnable (15:8)	Irq1ToEnable (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

Byte	6	7	8	9
Data from Host	Irq2ToEnable (31:24)	Irq2ToEnable (23:16)	Irq2ToEnable (15:8)	Irq2ToEnable (7:0)
Data to Host	0x00	0x00	0x00	0x00

4.1.2 ClearIrq

The `ClearIrq(...)` command clears the selected interrupt signals by writing a 1 in the respective bit.

Table 4-4: ClearIrq Command

Byte	0	1	2	3	4	5
Data from Host	0x01	0x14	IrqToClear (31:24)	IrqToClear (23:16)	IrqToClear (15:8)	IrqToClear (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

The *IrqToClear* is identical to *IrqToEnable* assignment.

4.2 RF Switch Control

4.2.1 SetDioAsRfSwitch

DIO5, DIO6, DIO7, DIO8 and DIO10 can control external RF switches or LNAs on the Sub-GHz and RFIO_HF RF paths using the `SetDioAsRfSwitch(...)` command.

Only the lowest 5 bits of all the configurations as well as the enable are taken into account.

Each Cfg bit corresponds to the state of the RFSW output for that particular mode:

Table 4-5: SetDioAsRfSwitch Command

Byte	0	1	2	3	4	5	6	7	8	9
Data from Host	0x01	0x12	RfSw Enable	RfSw StbyCfg	RfSw RxCfg	RfSw TxCfg	RfSw TxHPCfg	RfSw TxHfCfg	-RFU	-RFU
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00	0x00

- *RfswEnable* value indicates which switch is used (1) and which is not (0):
  - Bit 0 - RFSW0 Enabled (DIO5 pin)
  - Bit 1 - RFSW1 Enabled (DIO6 pin)
  - Bit 2 - RFSW2 Enabled (DIO7 pin)
  - Bit 3 - RFSW3 Enabled (DIO8 pin)
  - Bit 4 - RFSW4 Enabled (DIO10 pin)
- *RfSwStbyCfg:* Each bit indicates the state of the relevant RFSW DIO when in standby mode (bits 5:7 RFU)
- *RfSwRxCfg:* Each bit indicates the state of the relevant RFSW DIO when in RX mode
- *RfSwTxCfg:* Each bit indicates the state of the relevant RFSW DIO when in low power TX mode
- *RfSwTxHPCfg:* Each bit indicates the state of the relevant RFSW DIO when in high power TX mode
- *RfSwTxHfCfg*: Each bit indicates the state of the relevant RFSW DIO when in radio high frequency TX mode
- Bits 8:9: RFU

By default, no DIO is used as RF switch: all RFSW outputs are in High-Z state.

This command only works with the chip in Standby RC mode, otherwise it returns a CMD_FAIL on the next `GetStatus` command.

4.2.2 DriveDiosInSleepMode

Enables or disables the addition of pull ups or pull downs resistors on the configured RF switch and IRQ line DIOs. This command allows to save power consumption in an application where the RF switches are supplied by the LR1121 DIOs, when the LR1121 is in sleep mode.

Table 4-6: DriveDiosInSleepMode Command

Byte	0	1	2
Data from Host	0x01	0x2A	Enable
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *Enable* value indicates which switch is used for pull-up/down (1) and which is not (0):
  - 0: No pull-up or pull-down is configured by the FW (default).
  - 1: A pull-up or pull-down is added by the FW on the configured RF switch and IRQ DIOS when going to sleep modes (all sleep modes), depending on the state of the DIO in config RC mode.

On wake-up from sleep mode, the according DIOs are re-configured in push/pull mode, and pull-up/down are removed.

Note: When going to sleep mode without retention (retention = 0), all pending IRQs are cleared before going to sleep mode.

4.3 Temperature Sensor

The LR1121 has a built-in temperature sensor which gives an indication of the internal device temperature.

4.3.1 GetTemp

The temperature measurement can be triggered using command `GetTemp(...)`.

Table 4-7: GetTemp Command

Byte	0	1
Data from Host	0x01	0x1A
Data to Host	Stat1	Stat2

Table 4-8: GetTemp Response

Byte	0	1	2
Data from Host	0x00	0x00	0x00
Data to Host	Stat1	Temp(15:8)	Temp(7:0)

The temperature value is a function of an internal reference voltage (typ. 1.35V), and a typical temperature characteristic (typ -1.7mV/°C), and can be approximated using the following formula:

$$\mathbf{Temperature(degC)} \sim 25 + \frac{1000}{-1.7\text{mV/°C}} \times (\text{Temp(10:0)/2047*1.35 - 0.7295})$$

NOTE: `GetTemp()` **uses XOSC mode to get the temperature, so if a TCXO is connected to the LR1121, it must be configured using *SetTcxoMode* before calling** `GetTemp()`**.**

5. Power Distribution

5.1 Power Modes

Two power modes are available:

- DC-DC converter for low power applications
- LDO for low-cost or small size applications

5.1.1 SetRegMode

Command `SetRegMode(...)` defines which regulator should be used.

Table 5-1: SetRegMode Command

Byte	0	1	2
Data from Host	0x01	0x10	RegMode
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *RegMode* defines if the DC-DC converter has to be switched ON:
  - 0: Do not switch on the DC-DC converter in any mode (Default)
  - 1: Automatically switch on the DC-DC converter, depending on the mode as per Table 5-2
  - Other values are RFU

This command only works with the device in Standby RC mode, otherwise it returns CMD_FAIL on the next GetStatus command.

The following table illustrates the power regulation options for different modes and user settings.

Table 5-2: Power Regulation Options

Circuit Mode	Sleep	STDBY_RC	STDBY_XOSC	FS	RX	TX
Regulator Type = 0	-	LDO	LDO	LDO	LDO	LDO
Regulator Type = 1	-	LDO	DC-DC + LDO	DC-DC + LDO	DC-DC + LDO	DC-DC + LDO

5.2 VBAT Measurement

5.2.1 GetVbat

`GetVbat(...)` command monitors the battery supply voltage, it returns the Vbat voltage as a function of a reference voltage:

Table 5-3: GetVbat Command

Byte	0	1
Data from Host	0x01	0x19
Data to Host	Stat1	Stat2

Table 5-4: GetVbat Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	Vbat(7:0)

$$\mathbf{VBAT(V)} = \left(\left(\frac{5 \times \text{VBat(7:0)}}{255}\right) - 1\right) 1.35$$

5.3 Power-On-Reset and Brown-Out-Reset

The LR1121 features both POR and BRN features.

- POR and BRN ensure a proper startup of the circuit, maintaining the internal blocks reset until a safe battery voltage level is reached, for example at battery insertion.
- The BRN triggers a device reset if the battery voltage goes below the safe operation threshold of 1.7V (typically)
- The POR/BRN detector has a 50mV hysteresis
- A POR resets the statistics

Refer to Figure 5-1: LR1121 POR and BRN Functions for an illustration of the POR and BRN functions.

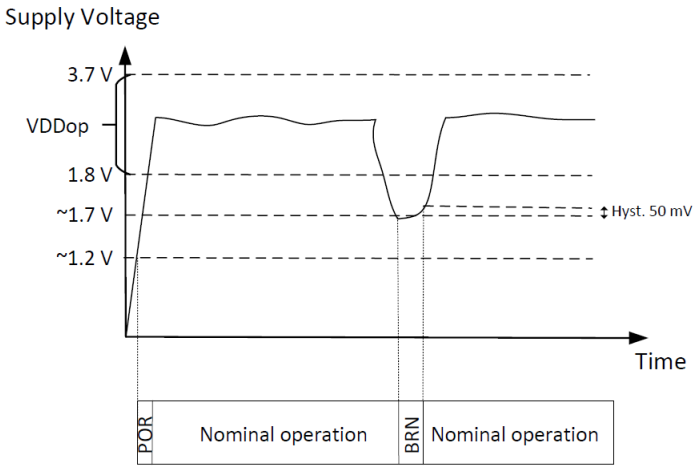

Figure 5-1: LR1121 POR and BRN Functions

5.4 Low Battery Detector

The Low Battery Detector (LBD) detects when the supply voltage VBAT drops below 1.88V (typ). The LBD indication is given through an interrupt signal, hence minimizing the host activity in critical supply voltage conditions. The LBD IRQ is activated though command `SetDioIrqParams()`.

5.5 Over Current Protection

The LR1121 has a built-in Over Current Protection (OCP) block which prevents surge currents when the device is used at its highest power levels, thus protecting the battery that may power the application. The current clamping values are trimmable by register access. The OCP threshold should be configured just below the battery requirement but at least 25% higher than the PA output stage current.

The default OCP values are:

- 60mA for the low power PA (LP PA)
- 150mA for the high power PA (HP PA)
- 50mA for the high frequency PA (HF PA)

6. Clock Sources

The LR1121 uses both low frequency (32kHz) and high frequency (32MHz) clock sources. For each frequency, the clock signal can be obtained by either an RC oscillator, or a crystal oscillator. RC oscillators allow optimized power consumption and faster switching times. Crystal oscillators provide a more precise frequency, in cases when frequency accuracy is needed. RF operations require a 32MHz high precision clock reference, which can be provided by either an external crystal oscillator or by a TCXO.

The clock requirements for LoRa Edge™ devices are summarized in AN1200.74 LoRa Edge Clock Requirements.

6.1 RC Oscillators Clock References

Two RC oscillators are available (refer to the LR1121 datasheet for the crystal/TCXO choice criteria):

- The 32.768kHz RC oscillator (RTC) can be used by the circuit in Sleep mode to wake-up the device when performing periodic or duty cycled operations. Several commands make use of this RTC to generate time-based events.
- The 32MHz RC oscillator is enabled for all SPI communication to permit configuration of the device without the need to start the 32MHz crystal oscillator.

6.2 High-Precision Clock References

This section provides relevant information about the reference clocks. For additional guidance on clock external clock sources (crystal and TCXO) refer to AN1200.59 Selecting the Optimal Reference Clock.

6.2.1 32.768kHz Crystal

A 32.768kHz crystal oscillator can be used instead of the (default) 32.768kHz RC oscillator as a low frequency clock source using command `ConfigLfClock(...)`.

6.2.2 32MHz Crystal

A 32MHz crystal oscillator is the cheapest and lowest power consuming approach to provide the 32MHz clock reference to the LR1121. The crystal loading capacitance are integrated, minimizing the overall BOM cost and optimizing the PCB space. In case of crystal operation, the VTCXO pin should be left unconnected.

6.2.2.1 Frequency Drift During Packet Transmission and Thermal Insulation

The transmission of RF packets at high RF power by the LR1121 generates significant heat which may transfer to the 32MHz crystal through the PCB. For long packet durations, this generates a frequency drift which may induce receive errors in the peer device if no precautions are taken during PCB design. This effect is described in more detail in the SX1261/62 application note AN1200.37 Recommendations for Best Performance.

For example, in LoRa modulation when the Low Data Rate (LDRO) is used, a frequency drift rate of the transmitted packet of 120Hz/s typically implies a 3dB sensitivity reduction in the LR1121 receiver for all SF and BW. This maximum drift rate is 110Hz/s when the LDRO is not used. Therefore, the frequency drift during packet transmission has to be kept below this maximum value in order to ensure best packet reception. A design using a TCXO is not subject to frequency drift during the packet transmission.

Implementing cuts in the PCB's ground plane layers reduces heat transfer between the LR1121 and the 32MHz crystal, as shown in Figure 6-1: LR1121 Thermal Insulation on PCB Top Layer.

Refer to the LR1121 PCB reference design on www.semtech.com for an implementation example in PCB design.

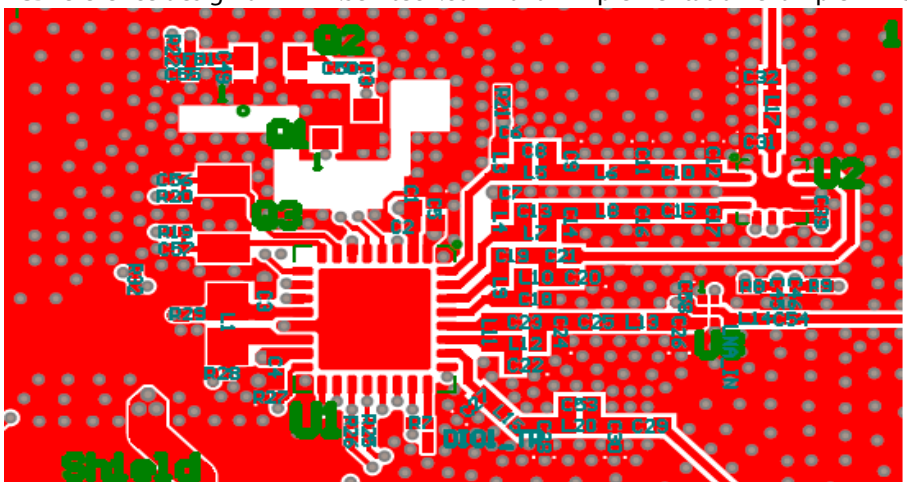

Figure 6-1: LR1121 Thermal Insulation on PCB Top Layer

6.2.3 32MHz TCXO

In environments with extreme temperature variation, it may be required to use a TCXO (Temperature Compensated Crystal Oscillator) to achieve better frequency accuracy. When a TCXO is used:

- The TCXO should be connected to pin XTA, through a 220Ω resistor and a10pF DC-cut capacitor. AN1200.59 provides guidelines for the selection of 32MHz TCXOs and associated components.
- Pin XTB should be left open.
- The TCXO is supplied by the internal regulator on the VTCXO pin.
- The regulated VTCXO is programmable from 1.6 to 3.3V using command `SetTcxoMode()`.
- VBAT should always be 200mV higher than the programmed voltage to ensure proper operation.
- The nominal current drain is 1.5mA, but the regulator can support up to 4mA of load.
- Clipped-sine output TCXO are required, with the output amplitude not exceeding 1.2V peak-to-peak.

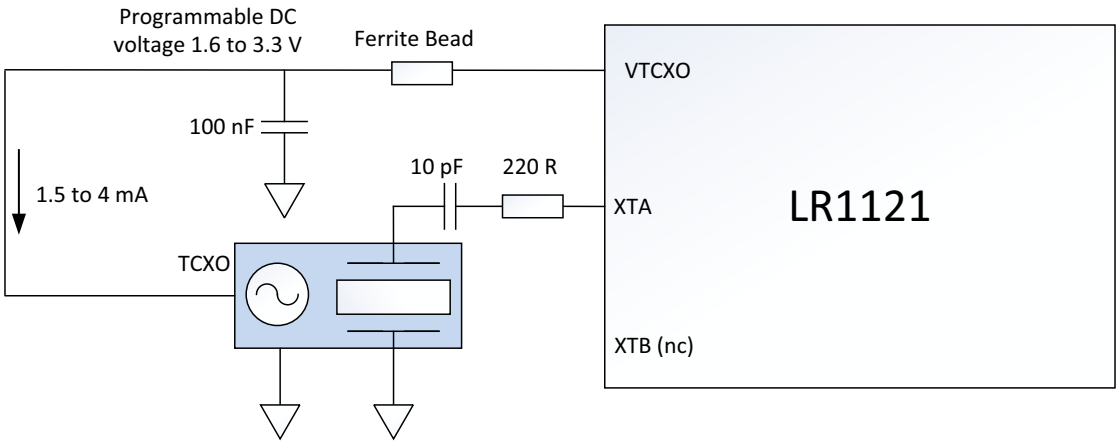

Figure 6-2: TCXO Circuit Diagram

6.3 Commands

6.3.1 ConfigLfClock

Configures the 32kHz source.

Table 6-1: ConfigLfClock Command

Byte	0	1	2
Data from Host	0x01	0x16	LfClkConfig
Data to Host	Stat1	Stat2	IrqStatus (31:24)

LfClkConfig parameter:

- bits 0-1: LF clock selection:
  - 0: Use 32.768kHz RC oscillator
  - 1: Use 32.768kHz crystal oscillator
  - 2: Use externally provided 32.768kHz signal on DIO11 pin
  - 3: RFU
- bit 2: When to release BUSY signal:
  - 0: Wait for Xtal 32k ready
  - 1: Wait for Xtal 32k to be ready before releasing the BUSY signal
- bits 3-7: RFU.

6.3.2 SetTcxoMode

Configures the chip for a connected TCXO.

The TCXO must be configured using `SetTcxoMode(...)` before calling `GetTemp()`.

Table 6-2: SetTcxoMode Command

Byte	0	1	2	3	4	5
Data from Host	0x01	0x17	RegTcxoTune	Delay (23:16)	Delay (15:8)	Delay (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *RegTcxoTune* tunes the output voltage on the TCXO supply pin VTCXO, according to Table 6-3: TCXO Supply Voltage Programming Values.

Table 6-3: TCXO Supply Voltage Programming Values

RegTcxoTune	TCXO Supply Voltage (typ)
0x00	1.6V
0x01	1.7V
0x02	1.8V
0x03	2.2V
0x04	2.4V
0x05	2.7V
0x06	3.0V
0x07	3.3V

- *Delay* represents the maximum duration for the 32MHz oscillator to start and stabilize (in 30.52us steps). If the 32MHz oscillator from the TCXO is not detected internally at the end the delay period, the device internal firmware triggers a HF_XOSC_START_ERR error.
  - 0: Disables TCXO mode (default value).
  - 1: Sets TCXO mode. A complete Reset of the chip is required to get back to normal XOSC operation

Command only operates in Standby RC mode, otherwise it returns CMD_FAIL on the next `GetStatus()` command.

7. Radio

7.1 Overview

The LR1121 is a half-duplex RF transceiver capable of handling constant envelope modulation schemes such as LoRa, (G)FSK, and LR-FHSS. The LoRa and (G)FSK modulations are fully compatible with the SX1261/62/68 family for sub-GHz, and can be configured to be compatible with the SX1280/81 family for 2.4GHz.

The radio system is shown in Figure 7-1: Radio below. It is composed of the frequency synthesizer (also referred as PLL), three TX paths (High Power, Low Power and High Frequency), and two RX paths, followed by a high-bandwidth ADC. Both the ADC and the PLL are tied to the digital subsystem and to the LoRa and (G)FSK modems, and the LR-FHSS modulator.

The PAs are described in a dedicated section, as well as the LoRa and (G)FSK modems, and the LR-FHSS modulation.

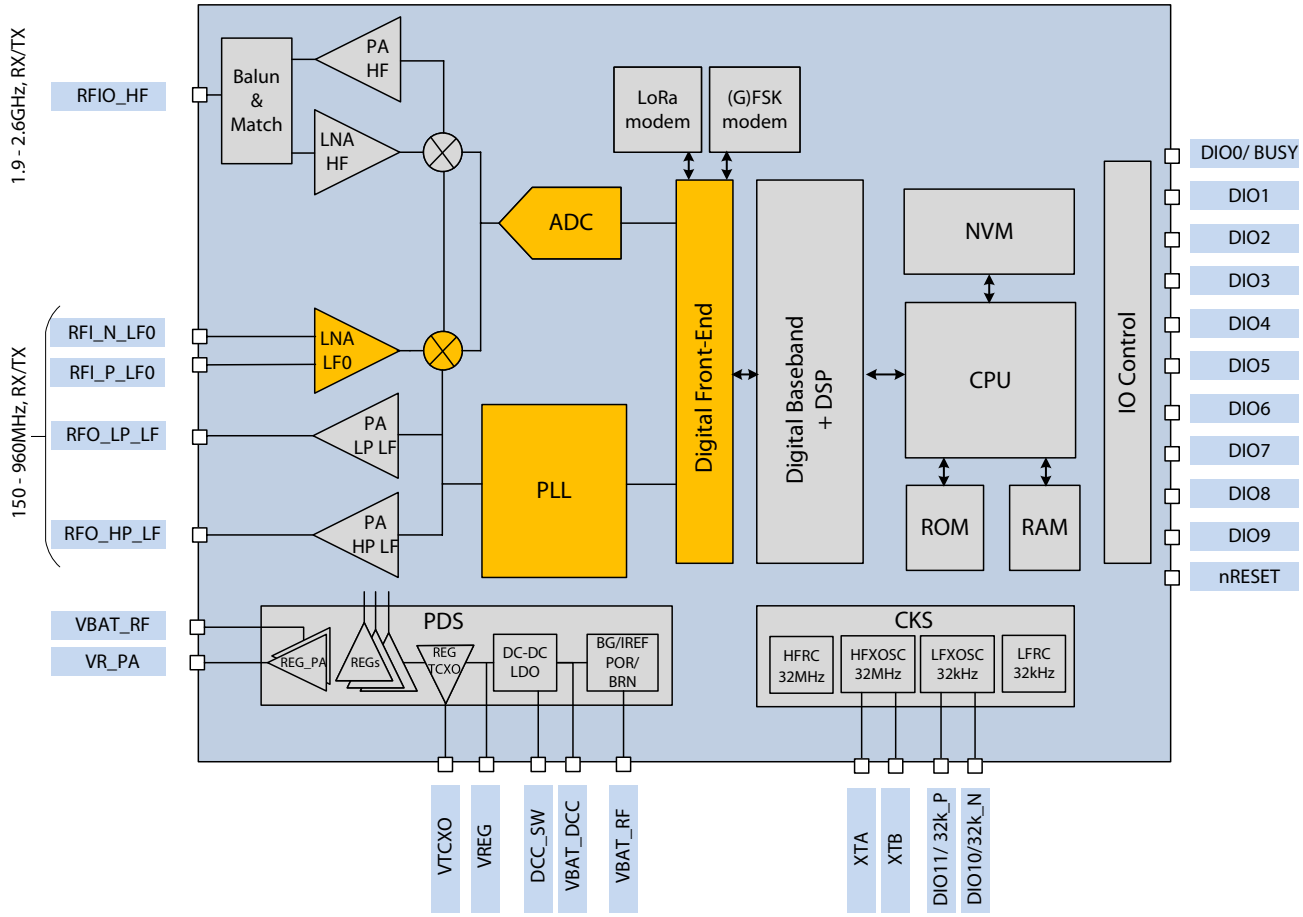

Figure 7-1: Radio

The LR1121 frequency synthesizer allows a continuous operation in the 150MHz-2500MHz frequency range. It is shared between the sub-GHz radioand HF radio, therefore no simultaneous sub-GHz /HF radio operation is possible.

The LR1121 frequency synthesizer is clocked by a 32MHz reference, provided by either a crystal oscillator, or a TCXO. Refer to Section 6. "Clock Sources" on page 44 for details.

7.2 Commands

7.2.1 SetRfFrequency

Command `SetRfFrequency(...)` sets the RF (PLL) frequency of the radio. In RX mode, the frequency is internally down-converted to IF Frequency.

Table 7-1: SetRfFrequency Command

Byte	0	1	2	3	4	5
Data from Host	0x02	0x0B	RfFreq (31:24)	RfFreq (23:16)	RfFreq (15:8)	RfFreq (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *RfFreq*: RF Frequency of the radio in Hz. All frequency dependent parameters are automatically recomputed by the LR1121 firmware when processing this command.

7.2.2 SetRx

Command `SetRx(...)` sets the sub-GHz radio in RX mode. The sub-GHz path is selected for *RfFreq* frequencies below or equal to 1.50GHz. Above this value the HF path is selected. If no packet is received after the defined *RxTimeout*, the device goes back to Standby RC mode.

Table 7-2: SetRx Command

Byte	0	1	2	3	4
Data from Host	0x02	0x09	RxTimeout (23:16)	RxTimeout (15:8)	RxTimeout (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)

- *RxTimeout* is expressed in periods of the 32.768kHz RTC. The maximum timeout value corresponds to 512s. Values 0x000000 and 0xFFFFFF disable the timeout function.
  - *0x000000* sets the device in RX mode until a reception occurs. After packet reception, the device returns to Standby mode.
  - *0xFFFFFF* sets the device in RX mode until the host sends a command to change the mode. The device can receive several packets. Each time a packet is received, a packet done indication is given to the host and the device automatically searches for a new packet.

If the timer is active, the radio stops the reception at the end of the timeout period, unless a preamble or a Header has been detected as defined by the *StopTimeoutOnPreamble* configuration.

If no packet type was configured, or the packet type does not allow RX operations, this command fails.

The BUSY signal goes low after the device is set into RX mode.

7.2.3 SetTx

Command `SetTx(...)` sets the sub-GHz radio or the HF radio in TX mode, triggering RF packet transmission, and starting the RTC with the given *TxTimeout* value. There is no PA selection based on the RfFreq configuration. For details on PA choice and operation refer to Section 9.

If the RTC event fires before the end of transmission, it triggers a TIMEOUT IRQ, and stops the transmission. Otherwise, at the end of the packet transmission, a TX_DONE interrupt is generated.

After a TIMEOUT IRQ or TX_DONE IRQ, the device goes back to STBY_RC (default), STBY_XOSC or FS depending on the *FallBackMode* configuration.

Table 7-3: SetTx Command

Byte	0	1	2	3	4
Data from Host	0x02	0x0A	TxTimeout (23:16)	TxTimeout (15:8)	TxTimeout (7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)

- *TxTimeout* is expressed in periods of the 32.768kHz RTC. The maximum value corresponds to 512 s. 0x000000 disables the timeout function.

If no packet type was configured, or the packet type does not allow Tx operations, the command fails.

The BUSY signal goes to 0 after the device is set into TX mode.

7.2.4 AutoTxRx

Command `AutoTxRx(...)` automatically performs the transition to RX mode after a packet transmission, or to TX mode after a packet reception. After the second mode, the device goes back to Standby RC mode.

If AutoTxRx mode is enabled, and a:

- `SetTx(...)` command is sent to the device, the device goes to RX mode after TX_DONE and the given delay. *Timeout* is used as the *RxTimeout* of the auto RX.
- `SetRx(...)` command is sent to the chip, the chip goes to TX mode after RX_DONE and the given delay. *Timeout* is used as the *TxTimeout* of the auto TX.

If an Rx Duty Cycle is started, this mode is not used.

Table 7-4: AutoTxRx Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x02	0x0C	Delay (23:16)	Delay (15:8)	Delay (7:0)	Intermediary Mode	Timeout (23:16)	Timeout (15:8)	Timeout (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

Delay defines the transition time between the TX and RX mode, expressed in periods of the 32.768kHz RTC. The maximum Delay value corresponds to 512s.

  - 0x000000: Performs a direct TX to RX or RX to TX transition, without going through the IntermediaryMode.
  - 0xFFFFFF: Disables the AutoTxRx function. The AutoTxRx function is disabled by default.
- *IntermediaryMode*: device mode inbetween TX and RX modes.
  - 0x00: Sleep mode.
  - 0x01: Standby RC mode.
  - 0x02: Standby Xosc mode.
  - 0x03: FS mode.
- *Timeout* defines the timeout of the second mode, after the automatic transition. It is expressed in periods of the 32.768kHz RTC. The maximum timeout value corresponds to 512s.
  - 0x000000: Disables the timeout function.

7.2.5 SetRxTxFallbackMode

Command `SetRxTxFallbackMode(...)` defines what mode the device goes into after a packet transmission or a packet reception. If an *Rx Duty Cycle* is started or an *AutoRxTx* is configured, this mode is not used.

Table 7-5: SetRxTxFallbackMode Command

Byte	0	1	2
Data from Host	0x02	0x13	FallbackMode
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *FallbackMode* values*:*
  - 0x01: Standby RC mode (default value).
  - 0x02: Standby Xosc mode.
  - 0x03: FS mode.
  - Other values are RFU.

The fallback mode is also used for an Rx Duty Cycle after the RX_DONE interrupt, or for an AutoRxTx after the RX to TX, or when the TX to RX sequence is completed.

7.2.6 SetRxDutyCycle

Command `SetRxDutyCycle(...)` periodically opens RX windows. Between RX windows, the device goes in Sleep mode (with retention). The clock source for the RTC has to be configured with a command before entering Duty Cycle mode.

Table 7-6: SetRxDutyCycle Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x02	0x14	RxPeriod (23:16)	RxPeriod (15:8)	RxPeriod (7:0)	Sleep Period (23:16)	Sleep Period (15:8)	Sleep Period (7:0)	Mode
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

- *RxPeriod* defines the maximum RX window duration, expressed in periods of the 32.768kHz RTC. The maximum Delay value corresponds to 512s.
- *SleepPeriod* defines the duration of the Sleep period between the RX windows. It is expressed in periods of the 32.768kHz RTC. The maximum Delay value corresponds to 512s.
- *Mode* selects the device mode during the RX windows:

  0: Configures the device in RX mode during RX windows. Available for (G)FSK and LoRa packet types.

  1: Configures the device in CAD mode during RX windows. Available only for LoRa packet types. Returns CMD_FAIL for (G)FSK packet types.

  The *Mode* parameter is optional, and is set to 0 if not sent.

When this command is sent in Standby mode, the context (device configuration) is saved and the device enters in a loop defined by the following steps, and depicted in Figure 7-2: LR1121 Current Profile During RX Duty Cycle Operation .

1. The device enters RX and listens for an incoming RF packet for a period of time defined by *RxPeriod*.
2. Upon preamble detection, the timeout is stopped and restarted with the value 2 * *RxPeriod* + *SleepPeriod*, as shown in Figure 7-3: RX Duty Cycle Upon Preamble Detection.
3. If no packet is received during the RX window, the device goes into Sleep mode with context saved for a period of time defined by *SleepPeriod*.
4. At the end of the Sleep window, the device automatically restarts the process of restoring context and enters the RX mode, and so on. At any time, the host can stop the procedure.

The loop is terminated if either:

- A packet is detected during the RX window, at which moment the chip interrupts the host via the RX_DONE flag and returns to the configured Fallback mode (refer to Section 7.2.5 "SetRxTxFallbackMode " on page 52).
- The host issues a `SetStandby(...)` command during the RX window.
- The device is woken up from Sleep mode with a falling edge of NSS. In this case, the user should send the `SetStandby(...)` command to avoid race conditions if the NSS falling edge was issued during the boot phase.

If an `RxDutyCycle(...)` is started, *AutoRxTx* or *SetRxTxFallback* modes are not used.

`StopTimeoutOnPreamble(...)` has no effect on this mode.

Note: The `RxDutyCycle(...)` command returns CMD_FAIL in the next command status if the packet type has not been set.

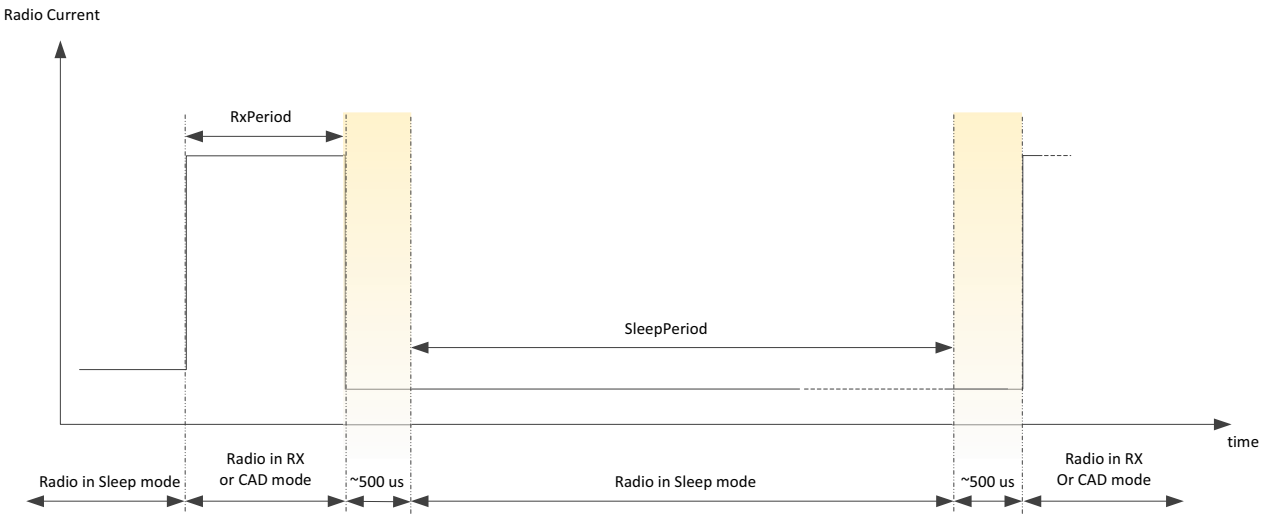

Figure 7-2: LR1121 Current Profile During RX Duty Cycle Operation

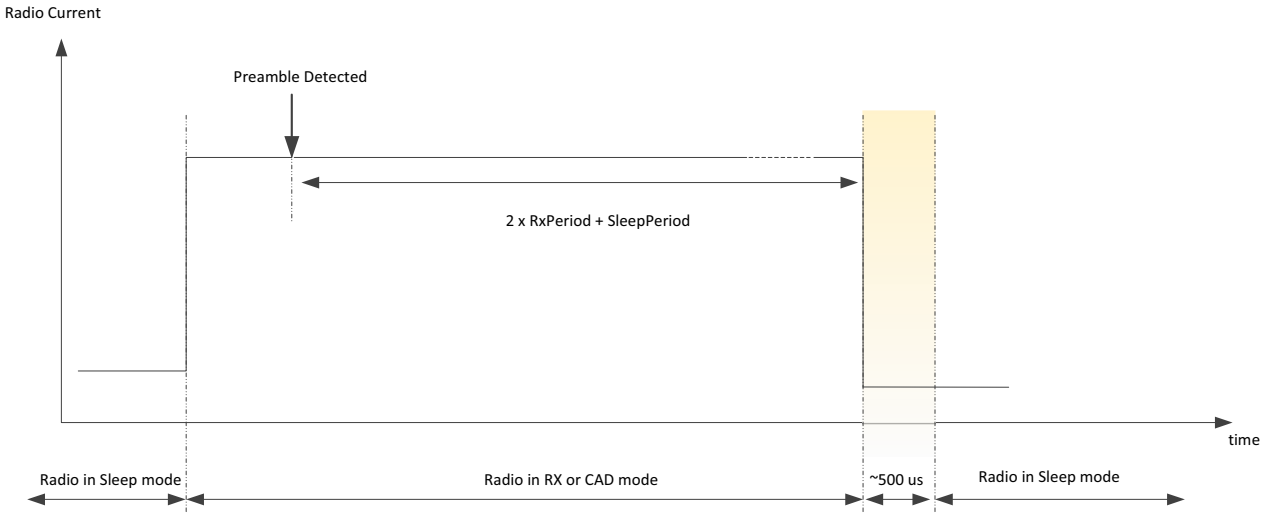

Figure 7-3: RX Duty Cycle Upon Preamble Detection

7.2.7 StopTimeoutOnPreamble

Command `StopTimeoutOnPreamble(...)` defines if the RX timeout should be stopped on Syncword / Header detection or on PreambleDetection.

Table 7-7: StopTimeoutOnPreamble Command

Byte	0	1	2
Data from Host	0x02	0x17	StopOnPreamble
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *StopOnPreamble* values*:*
  - 0x00: Stop on Syncword/Header detection (default value).
  - 0x01: Stop on Preamble detection.

7.2.8 GetRssiInst

Command `GetRssiInst(...)` returns the instantaneous RSSI value at the precise time when the command is sent. Therefore if no RF packet is present, the RSSI value returned by command `GetRssiInst(...)` corresponds to the RF noise.

Table 7-8: GetRssiInst Command

Byte	0	1
Data from Host	0x02	0x05
Data to Host	Stat1	Stat2

Table 7-9: GetRssiInst Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	Rssi

The RSSI is calculated using the following formula: RSSI (dBm) = *-Rssi*/2.

7.2.9 GetStats

Command `GetStats(...)` returns the internal statistics of the received RF packets:

Table 7-10: GetStats Command

Byte	0	1
Data from Host	0x02	0x01
Data to Host	Stat1	Stat2

Table 7-11: GetStats Response

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	NbPkt Received (15:8)	NbPkt Received (7:0)	NbPkt CrcError (15:8)	NbPkt CrcError (7:0)	Data1 (15:8)	Data1 (7:0)	Data2 (15:8)	Data2 (7:0)

- *NbPktReceived* is the total number of received packets.
- *NbPktCrcError* is the total number of received packets with a CRC error.
- *Data1* is PacketType dependant:
  - (G)FSK mode: *Data1 = NbPacketLengthError(15:0*): number of packet with a length error.
  - LoRa mode: *Data1 = NbPktHeaderErr(15:0):* number of packets with a Header error.
- *Data2* is PacketType dependant:
  - (G)FSK mode: *Data2* = 0x00.
  - LoRa mode: *Data2 = NbPktFalseSync(15:0):* number of false synchronizations.

Statistics are reset on a Power On Reset, power down, or by command `ResetStats(...)`.

7.2.10 ResetStats

Command `ResetStats(...)` resets the internal statistics of the received RF packets:

Table 7-12: ResetStats Command

Byte	0	1
Data from Host	0x02	0x00
Data to Host	Stat1	Stat2

7.2.11 GetRxBufferStatus

Command `GetRxBufferStatus(...)` returns the length of the last packet received and the offset in the RX buffer of the first byte received:

Table 7-13: GetRxBufferStatus Command

Byte	0	1
Data from Host	0x02	0x03
Data to Host	Stat1	Stat2

Table 7-14: GetRxBufferStatus Response

Byte	0	1	2
Data from Host	0x00	0x00	0x00
Data to Host	Stat1	PayloadLengthRX	RxStartBufferPointer

- *PayloadLengthRX* is the Payload length of the last packet received, in bytes.
- *RxStartBufferPointer* is the offset in the RX buffer of the first byte received.

7.2.12 SetRxBoosted

Command `SetRxBoosted(...)` sets the device in RX Boosted mode, allowing a ~2dB increased sensitivity, at the expense of a ~2mA higher current consumption in RX mode.

Table 7-15: SetRxBoosted Command

Byte	0	1	2
Data from Host	0x02	0x27	RxBoosted
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *RxBoosted*: Sets the Rx Boosted mode.
  - 0: RX Boosted mode deactivated.
  - 1: RX Boosted mode activated.
  - Other values are RFU.

7.2.13 SetLoRaSyncWord

This command sets the SetLoRaSyncWord.

Table 7-16: SetLoRaSyncWord Command

Byte	0	1	2
Data from Host	0x02	0x2B	Syncword
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *Syncword*: Sets the SetLoRaSyncWord. Valid for all SFs. Example values are:
  - 0x12: Private Network (default).
  - 0x34: Public Network.

7.2.14 GetLoRaRxHeaderInfos

Command `GetLoRaRxHeaderInfos(...)` returns the information coded in the last received packet header (explicit header mode), or the configured *coding_rate* and *crc_type* settings:

Table 7-17: GetLoRaRxHeaderInfos Command

Byte	0	1
Data from Host	0x02	0x30
Data to Host	Stat1	Stat2

Table 7-18: GetLoRaRxHeaderInfos Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	Infos

- *Infos:*
  - bits 7-5: 0 RFU
  - bits 4: 1 = CRC ON, 0 = CRC OFF
  - bit 3: 0 RFU
  - bits 2-0: coding rate (see 8.3.1 SetModulationParams command for values)

7.2.15 SetRssiCalibration

Command `SetRssiCalibration(...)` sets the gain offset for the on-chip power estimation:

Table 7-19: SetRssiCalibration Command

Byte	0	1	2	3	4	5	6
Data from Host	0x02	0x29	Tune G5(7:4) G4(3:0)	Tune G7(7:4) G6(3:0)	Tune G9(7:4) G8(3:0)	Tune G11(7:4) G10(3:0)	Tune G13(7:4) G12(3:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0

Byte	7	8	9	10	11	12
Data from Host	Tune G13hp2(7:4) G13hp1(3:0)	Tune G13hp4(7:4) G13hp3(3:0)	Tune G13hp6(7:4) G13hp5(3:0)	Tune G13hp7(3:0)	GainOffset (15:8)	GainOffset (7:0)
Data to Host	0	0				

- `TuneGx(...)` (arguments 2 to 10): RSSI Gain Tune values. A tune is a 4-bit signed value, where 1lsb = 0.5dB.

Table 7-20: Gain Tune Values

Tune	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value (dBm)	0	0.5	1	1.5	2	2.5	3	3.5	-4	-3.5	-3	-2.5	-2	-1.5	-1	-0.5

- *GainOffset*: Global offset added to the Gain Tune values. The offset is a 12-bit signed value, where 1lsb = 0.5dB.

The power seen by the LR1121 analog front-end is affected by external components such as the matching network, or RF switches. An incorrect RSSI results in a sensitivity degradation in (G)FSK mode and an incorrect gain selection in LoRa and GFSK mode. An incorrect gain can result in a missed detection (packet loss) or decreased resistance to interference.

By default, the chip is calibrated for the 868-915MHz band on the LR1121 EVK.

Table 7-21: Recommended Values for Reference EVK

Frequency	Gain Offset	G4 Tune	G5 Tune	G6 Tune	G7 Tune	G8 Tune	G9 Tune	G10 Tune	G11 Tune	G12 Tune	G13 Tune	G13hp1 Tune	G13hp2 Tune	G13hp3 Tune	G13hp4 Tune	G13hp5 Tune	G13hp6 Tune	G13hp7 Tune
below 600MHz	0	12	12	14	0	1	3	4	4	3	6	6	6	6	6	6	6	6
from 600MHz to 2GHz	0	2	2	2	3	3	4	5	4	4	6	5	5	6	6	6	7	6
above 2GHz	2030	6	7	6	4	3	4	14	12	14	12	12	12	12	8	8	9	9

Gain Tune and Gain Offset values can be determined using an RF generator with the LR1121 hardware implementation as follows:

1. Setup the device in GFSK mode and set the desired RF frequency.
2. Setup the device in manual gain mode (disable the AGC (Auto Gain Control)).
3. Use SetRssiCalibration API to set all tunes and offsets to 0.
4. Loop through all the different gains, including LNA boost mode (high power mode):
   a. Use the generator to emit a continuous tone at a specific output power gen_pwr (gain dependant), and account for all cable losses. A direct connection is recommended.
   b. Read the instantaneous RSSI (rssi_inst).
   c. Calculate the rssi_error = rssi_inst - gen_pwr.
   d. Log the rssi_error in a file.
5. Find the common offset between all errors, and use it as the new global offset.
6. Find the individual errors (relative to the new global offset) and use them as the tunes.
7. Call `SetRssiCalibration()`, and go though the main loop again to check that the instantaneous RSSI is now calibrated.

Some pseudo-code can be found hereafter:

```
        uint8_t gain[] = { 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 };
    // NB: for the 2G4 path, the max gain is 16 - 17-20 can be ignored.
        float power[] = { -35.0, -41.0, -45.0, -50.0, -53.5, -60.0, -65.0, -69.5, -75.0,
    -81.0, -82.5, -83.5, -84.0, -85.0, -86.0, -86.5, -87.0 };


    /* Configure the chip at the system level */
    lr11xx_system_reset( context );
    lr11xx_system_set_reg_mode( context, reg_mode );
    lr11xx_system_set_dio_as_rf_switch( context, rf_switch_cfg );
    lr11xx_system_set_tcxo_mode( context, tune, timeout );  // Optional - only if there is a
TCXO
    lr11xx_system_clear_errors( context );
    lr11xx_system_calibrate( context, 0x3F );  // 0x3F to enable all fields

    /* Configure the chip at the modem level */
    lr11xx_radio_set_pkt_type( context, LR11XX_RADIO_PKT_TYPE_GFSK );
    lr11xx_radio_set_rf_freq( context, freq_in_hz );

    lr11xx_system_calibrate_image_in_mhz( context, freq1_in_mhz, freq2_in_mhz );

    lr11xx_radio_set_gfsk_mod_params( context, mod_params );  // Rx BW has to be set to
LR11XX_RADIO_GFSK_BW_234300 - other modulation parameters can be anything
    lr11xx_radio_set_gfsk_pkt_params( context, pkt_params );  // Packet parameters can be
anything

/* Configure the chip to be controlled manually */
    lr11xx_regmem_write_regmem32_mask( context, 0x00F20214, 0x00080000, 0x00080000 );
    lr11xx_regmem_write_regmem32_mask( context, 0x00F20230, 0x71110000, 0x71100000 );
```

```
    lr11xx_radio_set_rssi_calibration( context, rssi_cal_table );  // All parrameters of
rssi_cal_table set to 0
    lr11xx_radio_set_rx_with_timeout_in_rtc_step( context, 0xFFFFFF );

    for( int i = 0; i++; i < 17 )  // 17 is the number of elements in gain array
    {
        const uint8_t gain_step = MIN( gain[i], 13 );
        const uint8_t lna_boost = ( gain > 13 ) ? gain - 13 : 0;

       lr11xx_regmem_write_regmem32_mask( context, 0x00F20214, 0x00F00000, gain_step << 20 );
       lr11xx_regmem_write_regmem32_mask( context, 0x00F3008C, 0x00070000, lna_boost << 16 );

/* Wait for 1 ms */

/* Insert here a control for your test equipment to generate a tone at RF frequency set to
freq_in_hz with an output power set to power[i] dBm */

        lr11xx_radio_get_rssi_inst( context, rssi_in_dbm );

/* Add a way to log (gain[i], power[i], rssi_in_dbm) triplet to be able to compute offset and
tunes for the RSSI calibration */
    }
```

8. Modems

8.1 Modem Configuration

The LR1121 contains different modems capable of handling different constant envelope modulations. For the LoRa®/(G)FSK modems and LR-FHSS modulation, the suitable command order is the following:

- The user must specify the modem to be used in command `SetPacketType(...)`
- `SetModulationParams(...)` configures the modem parameters (SF, BW, CR and LDRO)
- `SetPacketParams(...)` defines the RF packet parameters (Payload length, Implicit/explicit mode, …). This command is not needed for LR-FHSS
- `SetPaConfig(...)` configures the PA settings used for RF packet transmission (which PA, supply mode…)
- `SetTxParams(...)` defines the PA parameters (output power, ramp time).

The suitable command order is the following:

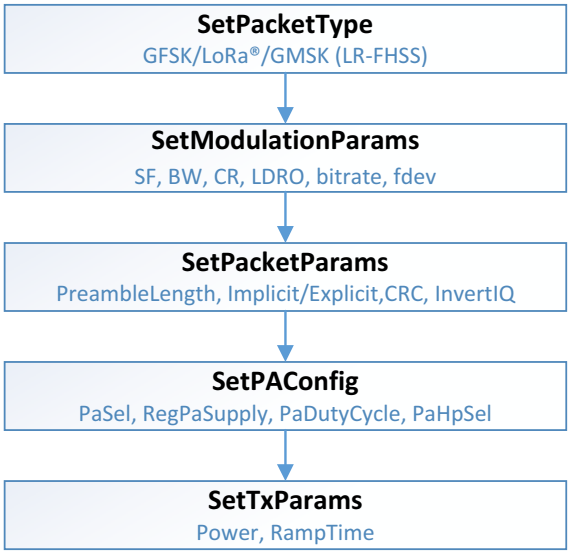

Figure 8-1: LoRa /(G)FSK / LR-FHSS Command Order

8.1.1 SetPacketType

The `SetPacketType(...)` command defines which modem is to be used.

Table 8-1: SetPacketType Command

Byte	0	1	2
Data from Host	0x02	0x0E	PacketType
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *PacketType* defines the modem to be used for the next RF transactions:
  - 0x00: None (default)
  - 0x01: (G)FSK
  - 0x02: LoRa
  - 0x04: GMSK (LR-FHSS)
  - Other values are RFU

This command is the first one to be called before going to RX or TX and before defining modulation and packet parameters.

This command only works with the device in Standby RC, Standby Xosc or Fs mode, otherwise it returns CMD_FAIL in the status of the next command.

8.1.2 GetPacketType

Command `GetPacketType(...)` returns the current protocol of the radio.

Table 8-2: GetPacketType Command

Byte	0	1
Data from Host	0x02	0x02
Data to Host	Stat1	Stat2

Table 8-3: GetPacketType Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	PacketType

- *PacketType* corresponds to the modem used for the following RF transactions:
  - 0: None
  - 1: (G)FSK
  - 2: LoRa
  - 4: LR-FHSS
  - Other values are RFU

8.2 LoRa® Modem

8.2.1 LoRa Modulation Principle

The LoRa modem uses a proprietary spread spectrum modulation, which permits an increased link budget and increased immunity to in-band interference compared to legacy modulation techniques. It has the capability to receive signals with negative SNR that increases sensitivity as well as link budget and range of the LoRa receiver.

8.2.1.1 Spreading Factor (SF)

The spread spectrum LoRa modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which a spread symbol (containing 2^{SF} chips) is sent is referred to as the symbol rate (Rs). The ratio between the nominal symbol rate and chip rate is the Spreading Factor and it defines the number of bits sent per symbol.

Note: The spreading factor must be known in advance on both transmit and receive sides of the link as different spreading factors are orthogonal to each other.

8.2.1.2 LoRa Bandwidth (BWL)

The LoRa modem operates at a programmable bandwidth (BWL) around a programmable central frequency fRF. The LoRa modem bandwidth always refers to the double side band (DSB), as shown in Figure 8-2: LoRa Signal Bandwidth.

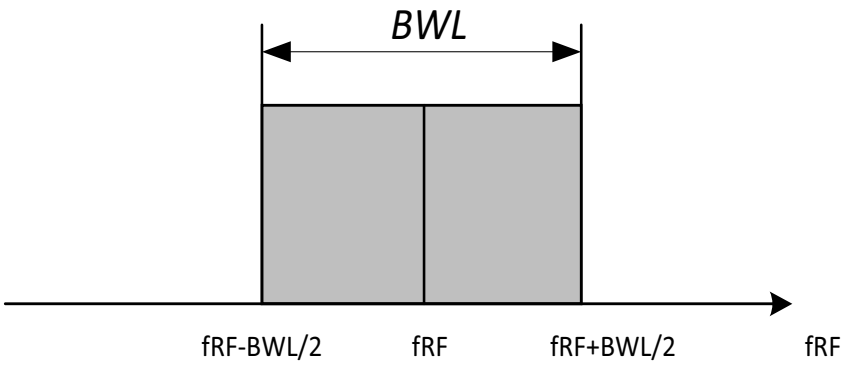

Figure 8-2: LoRa Signal Bandwidth

An increase in signal bandwidth permits the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity.

Note: Most countries have regulatory constraints on the permissible occupied bandwidth that only allow usage of a subset of BWL.

8.2.1.3 Coding Rate (CR)

To further improve the robustness of the link, the LoRa modem employs cyclic error coding to perform forward error detection and correction. Such error coding incurs a transmission overhead.

8.2.1.4 Low Data Rate Optimization (LDRO)

LDRO increases the robustness of the LoRa link at low effective data rates, improving the sensitivity level and increasing the robustness towards frequency drift and Doppler events. Its use is mandated with spreading factors of 11 and 12 at 125kHz bandwidth, and SF12 at 250kHz BW.

8.2.1.5 LoRa Symbol Rate (Rs)

With a knowledge of the key parameters that can be controlled by the user we define the LoRa symbol rate as:

$$Rs = \frac{BWL}{2^{SF}}$$

where BWL is the programmed bandwidth and SF is the spreading factor. The transmitted signal is a constant envelope signal. Equivalently, one chip is sent per second per Hz of bandwidth.

8.2.2 LoRa Packet Format

The LoRa modem employs two packet formats: explicit and implicit. The explicit packet contains additional information, it includes a short header that contains information about the number of bytes, coding rate and whether a CRC is used in the packet.

In certain scenarios, where the payload, coding rate and CRC presence are fixed or known in advance, it may be advantageous to reduce transmission time by invoking implicit header mode. In this mode the header is removed from the packet. In this case the payload length, error coding rate and presence of the payload CRC must be manually configured identically on both sides of the radio link.

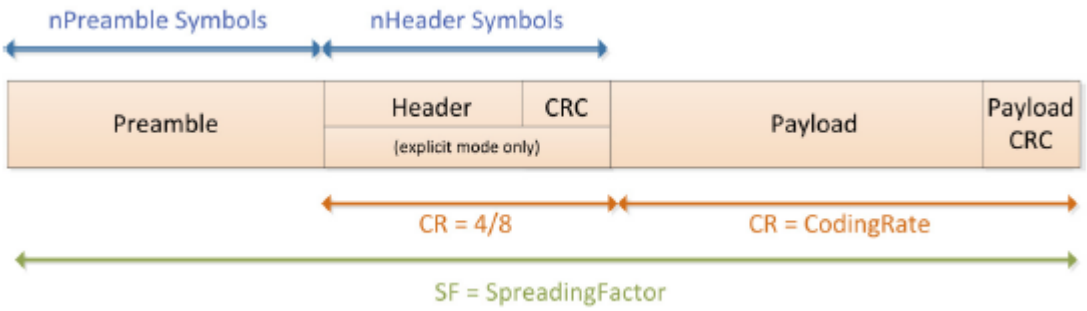

Figure 8-3: LoRa Packet Format

8.2.2.1 Preamble

The LoRa packet starts with a preamble sequence, used to synchronize the receiver with the incoming signal. The transmitted preamble length may vary from 1 to 65535 symbols. This permits the transmission of near arbitrarily long preamble sequences. In order to optimize the packet reception, it is advised to use a minimum preamble length of 12 with SF5 and SF6, and of 8 for other SF.

The receiver undertakes a preamble detection process that periodically restarts. For this reason the preamble length should be configured as identical to the transmitter preamble length. Where the preamble length is not known, or can vary, the maximum preamble length should be programmed on the receiver side.

8.2.2.2 Header (explicit packets only)

The header provides payload information:

- The payload length in bytes
- The forward error correction coding rate
- An optional 16-bit CRC for the payload

The header is transmitted with maximum error correction code (4/8). It also has its own CRC to allow the receiver to discard invalid headers.

8.2.2.3 Payload

The packet payload is a variable-length field that contains the actual data coded at a rate specified in the header in explicit mode or in the register settings in implicit mode. An optional CRC may be appended.

8.2.2.4 LoRa Packet Time On Air

The Time On Air of the LoRa packet is shown in the LR1121 drivers.

8.2.3 Channel Activity Detection (CAD)

Used only in LoRa packets, the Channel Activity Detection (CAD) is a LoRa specific mode of operation where the device searches for the presence of a LoRa preamble signal.

At the end of the search period, the device triggers the IRQ CADdone. If a valid signal has been detected it also generates the IRQ CadDetected. A minimum of 2 symbols is recommended to perform a CAD.

After the search has completed, the device returns to STDBY_RC mode. The length of the search is configured via command `SetCadParams(...)`.

8.3 LoRa Commands

8.3.1 SetModulationParams

Command `SetModulationParams(...)` configures the modulation parameters for the selected modem. Since the parameters are modem dependent, the description hereafter is valid only for the LoRa modem.

Table 8-4: SetModulationParams Command

Byte	0	1	2	3	4	5
Data from Host	0x02	0x0F	SF	BWL	CR	LowDataRateOptimize
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus (7:0)

- *SF* defines the spreading factor (values other than those below are RFU). SF5 and SF6 are compatible with the SX126x device family. SF6 can be made compatible with the SX127x family in implicit mode via a register setting[1].
  - 0x05: SF5
  - 0x06: SF6
  - 0x07:SF7
  - 0x08: SF8
  - 0x09: SF9
  - 0x0A: SF10
  - 0x0B: SF11
  - 0x0C: SF12
- *BWL* defines the LoRa modulation bandwidth (values other than those below are RFU):
  - 0x03: LoRa_BW_62, LoRa Bandwidth 62.5kHz
  - 0x04: LoRa_BW_125, LoRa Bandwidth 125kHz
  - 0x05: LoRa_BW_250, LoRa Bandwidth 250kHz
  - 0x06: LoRa_BW_500, LoRa Bandwidth 500kHz
  - 0x0D: BW203 (2.4GHz band only)
  - 0x0E: BW406 (2.4GHz band only)
  - 0x0F: BW812 (2.4GHz band only)
- *CR* configures the Coding Rate (values other than those below are RFU):
  - 0x01: Short Interleaver CR = 4/5 Overhead Ratio 1.25
  - 0x02: Short Interleaver CR = 4/6 Overhead Ratio 1.5
  - 0x03: Short Interleaver CR = 4/7 Overhead Ratio 1.75
  - 0x04: Short Interleaver CR = 4/8 Overhead Ratio 2
  - 0x05: Long Interleaver CR = 4/5[2] Overhead Ratio 1.25
  - 0x06: Long Interleaver CR = 4/6[2] Overhead Ratio 1.5
  - 0x07: Long Interleaver CR = 4/8[2] Overhead Ratio 2
- *LowDataRateOptimize* reduces the number of bits per symbol:
  - 0x00: LowDataRateOptimize off
  - 0x01: LowDataRateOptimize on

1.Set bit 18 of register at address 0xf20414 to 1

2.Long Interleaver (CR = 4/5, 4/6 and 4/8) is supported for packets of minimum Payload length of 8 bytes, and maximum Payload length of 253 bytes if the CRC is activated (255 bytes if the CRC is deactivated).

8.3.2 SetPacketParams

Command `SetPacketParams(...)` configures the parameters of the RF packet for the selected modem. Since the parameters are modem dependent, the description hereafter is valid only for the LoRa modem.

Table 8-5: SetPacketParams Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x02	0x10	PbLengthTX (15:8)	PbLengthTX (7:0)	HeaderType	PayloadLen	CRC	InvertIQ
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00

- *PbLengthTX* defines the length of the LoRa packet preamble.
  - Coded on 2 bytes, from 0x0001 (1) to 0xFFFF (65535). Minimum of 12 with SF5 and SF6, and of 8 for other SF advised.
- *HeaderType* defines if the header is explicit or implicit:
  - 0x00: Explicit header (default)
  - 0x01: Implicit header
- *PayloadLen* defines the size of the payload (bytes) to transmit or the maximum size of the payload that the receiver can accept.
  - In explicit header mode:

    0: Reception of any payload length between 0 and 255 bytes allowed.

    N: Reception of any payload length between 1 and N bytes accepted. Payload lengths of 0 or > N are rejected and result in a HeaderErr IRQ.
  - In implicit header mode, *PayloadLen* configures the exact length of the payload to be transmitted or received.
- *CRC* defines if the CRC is OFF or ON:
  - 0x00: OFF
  - 0x01: ON
- *InvertIQ* defines if the I and Q signals are inverted.
  - 0x00: Not inverted
  - 0x01: Inverted

This command fails if no packet type has been set.

8.3.3 SetCad

Command `SetCad(...)` activates the CAD feature.

Table 8-6: SetCad Command

Byte	0	1
Data from Host	0x02	0x18
Data to Host	Stat1	Stat2

8.3.4 SetCadParams

Command `SetCadParams(...)` defines the LoRa CAD parameters.

Table 8-7: SetCadParams Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x02	0x0D	Symbol Num	DetPeak	DetMin	CadExit Mode	Timeout (23:16)	Timeout (15:8)	Timeout (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

- *SymbolNum* defines the number of symbols used for the CAD detection.
- *DetPeak* and *DetMin* define the sensitivity of the LoRa modem when trying to correlate to actual LoRa preamble symbols. These two settings depend on the LoRa spreading factor, the Bandwidth, and the number of symbols used to validate the detection. Choosing the right value must be carefully tested to ensure a good detection at sensitivity level, and also to limit the number of false detections.
- *CadExitMode* defines the action to be performed after a CAD operation.

Table 8-8: CadExitMode Parameter

Value	CadExitMode	Operation
0x00	CAD_ONLY	The chip performs a CAD operation in LoRa. Once done and whatever the activity on the channel, the device goes back to STBY_RC mode.
0x01	CAD_RX	The device performs a CAD operation and if an activity is detected, it stays in RX until a packet is detected or the timer reaches the timeout defined by *Timeout* *31.25us
0x10	CAD_LBT	The device performs a CAD operation and if no activity is detected, it goes into TX mode with the defined *Timeout* as timeout parameter.

- *Timeout* is only used when the CAD is performed with *cadExitMode* = CAD_RX or CAD_LBT.
  - If *cadExitMode* = CAD_RX, see 7.2.2 SetRx for *Timeout* definition
  - If *cadExitMode* = CAD_LBT, see 7.2.3 SetTx for *Timeout* definition

8.3.5 SetLoRaSynchTimeout

Command `SetLoRaSynchTimeout(...)` configures the LoRa modem to issue an RX timeout after exactly *SymbolNum* symbols if no packet was detected by then.

Table 8-9: SetLoRaSynchTimeout Command

Byte	0	1	2
Data from Host	0x02	0x1B	SymbolNum
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *SymbolNum:* 0x00: No timeout (default value)

8.3.6 SetLoRaPublicNetwork

Command `SetLoRaPublicNetwork(...)` sets the LoRa modem syncword to public or private.

Table 8-10: SetLoRaPublicNetwork Command

Byte	0	1	2
Data from Host	0x02	0x08	PublicNetwork
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *PublicNetwork:*
  - 0x00: Private network (default)
  - 0x01: Public network
  - Other values are RFU

8.3.7 GetPacketStatus

Command `GetPacketStatus(...)` gets the status of the last received packet. Since the returned values are modem dependent, the description hereafter is valid only for the LoRa modem.

Table 8-11: GetPacketStatus Command

Byte	0	1
Data from Host	0x02	0x04
Data to Host	Stat1	Stat2

Table 8-12: GetPacketStatus Response

Byte	0	1	2	3
Data from Host	0x00	0x00	0x00	0x00
Data to Host	Stat1	RssiPkt	SnrPkt	SignalRssiPkt

- *RssiPkt* defines the average RSSI over the last packet received. RSSI value in dBm is –RssiPkt/2.
- *SnrPkt* is an estimation of SNR on last packet received. Expressed in two's complement format multiplied by 4. Actual SNR in dB is SnrPkt/4.
- *SignalRssiPkt* is an estimation of RSSI of the LoRa signal (after despreading) on last packet received, in two's complement format [negated, dBm, fixdt(0,8,1)]. Actual RSSI in dB is -SignalRssiPkt/2.

Additional information on the RSSI can be found in section Section 8.9 "RSSI Functionality" on page 83.

8.4 (G)FSK Modem

8.4.1 (G)FSK Modulation Principle

The (G)FSK modem can transmit and receive 2-FSK modulated packets over data rates ranging from 0.6kbps to 300kbps.

Both the bit rate (*Bitrate)* and frequency deviation (*Fdev*) are directly configured using command `SetModulationParams()`.

Additionally, in transmission mode, several shaping filters can be applied to the signal in packet mode or in continuous mode. In reception mode, the user needs to select the best reception bandwidth depending on its conditions. To ensure correct demodulation, the following limit must be respected for the bandwidth:

(2 x Fdev + BR) < BWF

where the bandwidth BWF ranges from 4.8kHz to 467kHz.

The bandwidth must be chosen so that:

Bandwidth [DSB] ≥ BR + 2 x frequency deviation + frequency error

where the frequency error is twice the frequency error of the crystal oscillator used.

8.4.2 (G)FSK Packet Engine

The LR1121 is designed for packet-based communication. The packet controller block is responsible for assembling received data bit-streams into packets and storing them in the data buffer. It also performs bit-stream decoding operations such as de-whitening & CRC-checks on the received bit-stream.

On the transmit side, the packet handler can construct a packet and send it bit by bit to the modulator for transmission. It can whiten the payload and append the CRC-checksum to the end of the packet. The packet controller only works in half-duplex mode i.e. either in transmit or receive at a time.

The packet controller is configured using command `SetPacketParams(...)`. This function can be called only after defining the protocol.

Preamble Detection in Receiver Mode

The LR1121 can gate the reception of a packet if an insufficient number of alternating preamble symbols (usually referred to 0x55 or 0xAA in hexadecimal form) has been detected. The parameter *PreambleDetectorLength* in command `SetPacketParams(...)` allows the user to select a value ranging from:

- "Preamble detector length off": where the radio performs no gating and locks directly on the following Syncword, to
- "Preamble detector length 32 bits": where the radio expects to receive 32 bits of preamble before the following Syncword. In this case, if the 32 bits of preamble are not detected, the radio either drops the reception in RxSingle mode, or restarts its tracking loop in RxContinuous mode.

To achieve best performance, it is recommended to set *PreambleDetectorLength* to "Preamble detector length 8 bits" or "Preamble detector length 16 bits" depending on the complete size of preamble sent by the transmitter.

Note: To detect packets correctly, the *PreambleDetectorLength* must always be smaller than the size of the following Syncword. If the *PreambleDetectorLength* is greater than the following Syncword length (typically when no Syncword is used) the user should fill some of the Syncword bytes with 0x55.

8.4.3 (G)FSK Packet Format

The (G)FSK packet format provides a conventional packet format for application in proprietary NRZ coded, low energy communication links. The packet format has built in facilities for CRC checking of the payload, dynamic payload size and packet acknowledgement. Whitening based upon pseudo random number generation can be enabled. Two principle packet formats are available in the (G)FSK protocol: fixed-length and variable-length.

8.4.3.1 Fixed-Length Packet

If the packet length is fixed and known on both sides of the link then knowledge of the packet length does not need to be transmitted over the air. Instead the packet length can be written to the parameter *PacketLength* which determines the packet length in bytes:

- 0 to 255 if address filtering is not activated
- 0 to 254 if address filtering is activated

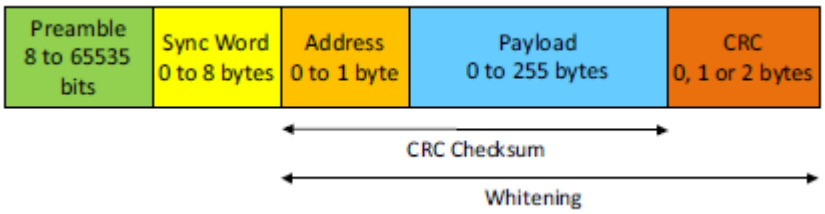

Figure 8-4: Fixed-Length Packet

It is usually recommended to use a minimum of 16 bits for the preamble to guarantee a valid reception of the packet on the receiver side.
The CRC operation, packet length and preamble length are defined in command `SetPacketParams(...)`.

8.4.3.2 Variable-Length Packet

This field encodes the payload length in bytes.

If the packet is of uncertain or variable size, information about the payload length must be transmitted within the packet.

The format of the variable-length packet is shown in the figure below:

- the packet length is 0 to 254 bytes if address filtering is activated
- the Length field is 1 byte long, except for SX128x compatibility, when it is 9 bits (see 8.4.4 SX128x Compatibility)

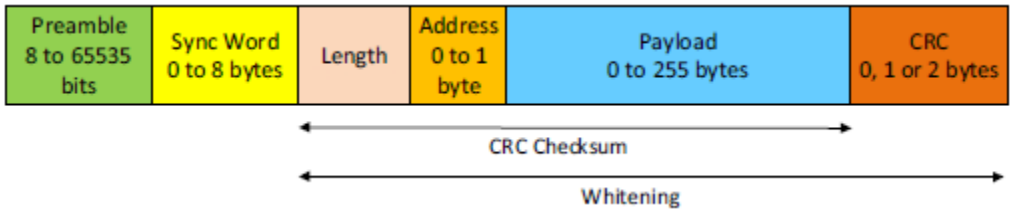

Figure 8-5: Variable-Length Packet

8.4.3.3 Setting The Packet Length Or Node Address

The packet length and Node or Broadcast address are not considered part of the payload and are added automatically in hardware. The packet length is added automatically in the packet when the *PacketType* field is set to variable size in command `SetPacketParams(...)`.

The node or broadcast address can be enabled by the *AddrComp* field in command `SetPacketParams(...)`. This field allows the user to enable and select an additional packet filtering at the payload level.

8.4.3.4 Whitening

The whitening is based around the 9-bit LFSR polynomial x^9+x^5+1 for sub-GHz and x^7 + x^4 + 1 for HF which generates a random sequence. With this structure, the LSB at the output of the LFSR is XORed with the MSB of the data. At the initial stage, command `SetGfskWhitParams(...)` sets the whitening Seed.

The payload (including the payload length, the Node or Broadcast address and CRC checksum when needed) is then XORed with this random sequence to generate the whitened payload.

The data is de-whitened on the receiver side by XORing with the same random sequence. This process limits the number of consecutive 1's or 0's to 9.

Note that data whitening is only required when the user data has high correlation with long strings of 0's and 1's. If the data is already random then the whitening is not required.

8.4.3.5 CRC

The LR1121 offers full flexibility to select the CRC polynomial and initial value of the selected polynomial. The user can also select a complete inversion of the computed CRC to comply with some international standards.

The CRC can be enabled and configured in the *CrcType* field in command `SetPacketParams(...)`. This field allows the user to enable and select the length and configuration of the CRC.

Command `SetGfskCrcParams(...)` configures the CRC polynomial and initial value.

8.4.4 SX128x Compatibility

The following limitations apply when using the SX128x 9-bit header PacketType 0x02:

- Payload length: payload length is limited to max. 254 - CrcLength (Ex: 252 bytes with a 16-bit crc)
- CRC length: 0,1 or 2 bytes
- Syncword length: only allowed syncword length values are 8,16,24, 32 or 40 bits
- AddrComp: has to be disabled (feature not present on SX128x)
- DcFree has to be either disabled (0x00) or SX128x compatible (0x03). 0x01 is not accepted

In order to make the LR1121 to be air-compatible with the SX128x:

- Set the whitening seed to 0x0001 with the SetGfskWhitParams command (default SX128x value)
- SetGfskCrcParams: (default SX128x values):
  - 1 byte crc: seed = 0xFF, poly = 0x07
  - 2 byte crc: seed = 0xFFFF, poly = 0x1021

Note that for transmitted frames, 7 additional bits are transmitted and for received frames 23 additional bits are received (delaying the RX_DONE IRQ).

8.5 (G)FSK Commands

8.5.1 SetModulationParams

Command `SetModulationParams(...)` configures the modulation parameters for the selected modem. Since the parameters are modem dependent, the description hereafter is valid only for the (G)FSK modem.

Table 8-13: SetModulationParams Command

Byte	0	1	2	3	4	5	6	7	8	9	10	11
Data from Host	0x02	0x0F	Bitrate (31:24)	Bitrate (23:16)	Bitrate (15:8)	Bitrate (7:0)	Pulse Shape	BWF	Fdev (31:24)	Fdev (23:16)	Fdev (15:8)	Fdev (7:0)
Data to Host	Stat1	Stat2	Irq Status (31:24)	Irq Status (23:16)	Irq Status (15:8)	Irq Status (7:0)	0x00	0x00	0x00	0x00	0x00	0x00

- *BitRate* defines the (G)FSK bit rate in bits per second. It ranges from 600b/s to 300kb/s (default 4.8kb/s).
- *PulseShape* defines the filtering applied to the (G)FSK packet:
  - 0x00: No filter applied
  - 0x08: Gaussian BT 0.3
  - 0x09: Gaussian BT 0.5
  - 0x0A: Gaussian BT 0.7
  - 0x0B: Gaussian BT 1
- *BWF defines the bandwidth*

Table 8-14: Bandwidth Parameter

BWF	Description	BWF	Description
0x1F	RX_BW_4800 (4.8kHz DSB)	0x0C	RX_BW_58600 (58.6kHz DSB)
0x17	RX_BW_5800 (5.8kHz DSB)	0x1B	RX_BW_78200 (78.2kHz DSB)
0x0F	RX_BW_7300 (7.3kHz DSB)	0x13	RX_BW_93800 (93.8 Hz DSB)
0x1E	RX_BW_9700 (9.7kHz DSB)	0x0B	RX_BW_117300 (117.3kHz DSB)
0x16	RX_BW_11700 (11.7kHz DSB)	0x1A	RX_BW_156200 (156.2kHz DSB)
0x0E	RX_BW_14600 (14.6kHz DSB)	0x12	RX_BW_187200 (187.2kHz DSB)
0x1D	RX_BW_19500 (19.5kHz DSB)	0x0A	RX_BW_234300 (232.3kHz DSB)
0x15	RX_BW_23400 (23.4kHz DSB)	0x19	RX_BW_312000 (312kHz DSB)
0x0D	RX_BW_29300 (29.3kHz DSB)	0x11	RX_BW_373600 (373.6kHz DSB)
0x1C	RX_BW_39000 (39kHz DSB)	0x09	RX_BW_467000 (467kHz DSB)
0x14	RX_BW_46900 (46.9kHz DSB)	-	-

- *Fdev* defines the frequency deviation (Hz)

8.5.2 SetPacketParams

Command `SetPacketParams(...)` configures the parameters of the RF packet for the selected modem. Since the parameters are modem dependent, the description hereafter is valid only for the (G)FSK modem.

Table 8-15: SetPacketParams Command

Byte	0	1	2	3	4	5	6	7	8	9	10
Data from Host	0x02	0x10	PblLength TX(15:8)	PblLength TX(7:0)	Pbl Detect	Sync WordLen	Addr Comp	Packet Type	Payload Len	Crc Type	DcFree
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00	0x00	0x00

- *PblLengthTX* defines the (G)FSK packet preamble length in bits. Coded on 2 bytes, from 0x0008 (8 bits) to 0xFFFF (65535 bits).
- *PblDetect* defines the preamble detector length. The preamble detector acts as a gate to the packet controller, when not 0x00 (preamble detector length off), the packet controller only becomes active if a certain number of preamble bits have been successfully detected by the radio.
  - 0x00: Preamble detector length off
  - 0x04: Preamble detector length 8 bits
  - 0x05: Preamble detector length 16 bits
  - 0x06: Preamble detector length 24 bits
  - 0x07: Preamble detector length 32 bits
- *SyncWordLen* defines the length of the Syncword in bits. The Syncword is programmed by `SetGfskSyncWord(...)`.
- *AddrComp* allows conditioning the packet reception to a predefined peer device address. Node address and broadcast address can be set with the `SetPacketAdrs(...)` command. If the address comparison fails then the packet reception is aborted and the adrsErr flag is set.
  - 0x00: Address Filtering Disabled
  - 0x01: Rx & Tx: Address Filtering activated on Node address
  - 0x02: Address Filtering activated for Rx on Node & broadcast addresses, for Tx on Node address
- *PacketType* defines the length of the incoming packet.
  - 0x00: Packet length is known on both sides, the size of the payload is not added to the packet
  - 0x01: The packet is of variable size, *PayloadLen* is the size of the packet (with header encoded on 8 bits)
  - 0x02: The packet is of variable size, *PayloadLen* is the size of the packet - compatible with SX128x chip (with header encoded on 9 bits)
- *PayloadLen* defines the size of the payload (bytes) to transmit or the maximum payload size the receiver can accept.
  - In explicit header mode:
    - 0: Reception of any payload length between 0 and 255 bytes allowed
    - N: Reception of any payload length between 1 and N bytes accepted. Payload lengths of 0 or > N are rejected and result in a HeaderErr IRQ
  - In implicit header mode, PayloadLen configures the exact length of the payload to be transmitted or received.
- *CrcType* defines the packet CRC. The CRC can be fully configured and the polynomial used, as well as the initial values can be entered directly through command `SetGfskCrcParams(...)`:
  - 0x01: CRC_OFF (No CRC)
  - 0x00: CRC_1_BYTE (CRC computed on 1 byte)
  - 0x02: CRC_2_BYTE (CRC computed on 2 bytes)
  - 0x04: CRC_1_BYTE_INV (CRC computed on 1 byte and inverted)
  - 0x06: CRC_2_BYTE_INV (CRC computed on 2 bytes and inverted)
- *DcFree* enables whitening on the RF packet:
  - 0x00: No encoding
  - 0x01: SX127x/SX126x/LR11xx compatible whitening enable
  - 0x03: SX128x compatible whitening enable

8.5.3 SetGfskSyncWord

Command `SetGfskSyncWord(...)` configures the Syncword of the (G)FSK packet.

Table 8-16: SetGfskSyncWord Command

Byte	0	1	2	3	4	5	6	7	8	9
Data from Host	0x02	0x06	Syncword (63:56)	Syncword (55:48)	Syncword (47:40)	Syncword (39:32)	Syncword (31:24)	Syncword (23:16)	Syncword (15:8)	Syncword (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00	0x00

By default, the Syncword is set to 0x9723522556536564.

For reception only, the GFSK Syncword must be a multiple of 8 bits. If it is not, the Syncword length must still be configured as a multiple of 8 bits, and the Sync word must have filler added at the beginning. For example, if the syncword length was 30 bits, the syncword length should be configured to 32 bits, and "01b" or "10b" should be added to the beginning of the Syncword.

8.5.4 SetPacketAdrs

Command `SetPacketAdrs(...)` sets the Node address and Broadcast address used for (G)FSK packet reception/transmission when filtering is enabled (AddrComp 0x01, or 0x02).

Table 8-17: SetPacketAdrs Command

Byte	0	1	2	3
Data from Host	0x02	0x12	NodeAddr	BroadcastAddr
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

- NodeAddr: Default 0x00
- BroadcastAddr: Default 0x00

If the address comparison fails then the packet reception is aborted and the adrsErr flag is set.

8.5.5 SetGfskCrcParams

Command `SetGfskCrcParams(...)` configures the CRC polynomial and initial value, so they do not need to be controlled manually.

Table 8-18: SetGfskCrcParams Command

Byte	0	1	2	3	4	5	6	7	8	9
Data from Host	0x02	0x24	InitValue (31:24)	InitValue (23:16)	InitValue (15:8)	InitValue (7:0)	Poly (31:24)	Poly (23:16)	Poly (15:8)	Poly (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00	0x00

- *InitValue*: Initial value of the configured CRC polynomial (default 0x1D0F)
- *Poly*: CRC polynomial (default Ox1021)

This flexibility permits the user to select any standard CRC or to use their own CRC, allowing a specific detection of a given packet. Examples:

To use the IBM CRC configuration, the user must select:

- 0xFFFF for the initial value
- 0x8005 for the CRC polynomial
- 0x02 (CRC_2_BYTE) for the field *CrcType* in command `SetPacketParams(...)`

For the CCITT CRC configuration the user must select:

- 0x1D0F for the initial value
- 0x1021 for the CRC polynomial
- 0x06 (CRC_2_BYTE_INV) for the field *CrcType* in command `SetPacketParams(...)`

8.5.6 SetGfskWhitParams

Command `SetGfskWhitParams(...)` sets the whitening Seed:

Table 8-19: SetGfskWhitParams Command

Byte	0	1	2	3
Data from Host	0x02	0x025	Seed (15:8)	Seed (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

- *Seed* default value is 0x0100

8.5.7 GetPacketStatus

Command `GetPacketStatus(...)` gets the status of the last received packet. Since the returned values are modem dependent, the description hereafter is valid only for the (G)FSK modem.

Table 8-20: GetPacketStatus Command

Byte	0	1
Data from Host	0x02	0x04
Data to Host	Stat1	Stat2

Table 8-21: GetPacketStatus Response

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	RxStatus (31:24)	RxStatus (23:16)	RxStatus (15:8)	RxStatus (7:0)

- *RxStatus* bits are as follows:
  - Bits 31:24: *RssiSync*, RSSI values latched upon the detection of sync address. Negated, dBm, ufix(8,1).
  - Bits 23:16 *RssiAvg*, RSSI average over the payload of the received packet. Latched upon the packet_done IRQ. Negated, dBm, ufix(8,1).
  - Bits 15:8 *RxLen*, Length of the received packet
  - Bit 7: RFU
  - Bit 6: RFU
  - Bit 5: *Adrserr*, Address filtering status of current packet. Asserted when the received packet address byte doesn't match the configured node_adrs(7:0) or broadcast(7:0) address according to the adrs_comp(1:0) configuration.
  - Bit 4: *Crcerr*, CRC check status of the current packet. Only applicable in Rx when the CRC check is not disabled. The packet is available in FIFO.
  - Bit 3: *Lenerr*, Length filtering status of the current packet. Asserted when the length of the received packet is greater than the Max length defined in the payload_len(7:0) field. Only applicable in Rx for variable length packets.
  - Bit 2: *Aborterr*, Abort status of the current packet. Asserted when the current packet is aborted with the pkt_abort_p register field. Applicable in both Rx and Tx.
  - Bit 1: *PktRcvd*, Packet reception status. Indicates that the packet reception is done. Only show the completion of the receive process and not the packet validity. Only applicable in Rx.
  - Bit 0: *PktSent*, Packet transmission status. Indicates that the packet transmission is done. Only applicable in Tx.

8.6 LR-FHSS Modulation

LR-FHSS is a frequency hopping spread spectrum modulation, perfectly suited for situations where long-range communication is required for a large number of devices in the same area. LR-FHSS can nearly equate the link-budget of LoRa SF12 BW125kHz, allowing many devices to coexist on the same spectral allocation. LR-FHSS implementation in the LR1121 is transmit only. The LR1121 fully controls the following functionalities:

- constructing frames, including preamble, sync headers, payload, integrity check, whitening, interleaving and forward error correction calculations
- calculation of hopping sequences
- handling of the whole frequency hopping mechanism

8.6.1 LR-FHSS Modulation Principle

The LR-FHSS is a low data rate modulation derived from a GMSK modulation, with intra-packet frequency hopping sequences configurable for each device. Only 488.28215Hz data rate is currently supported. The payload uses convolution encoding to perform forward error detection and correction, implying a transmission overhead.

Spectrally speaking, the LR-FHSS packet is split into several frames, sent over pseudo-randomly distributed frequencies spaced by a configurable step size (Grid). The total spectral occupation of the LR-FHSS packet corresponds to the Bandwidth parameter. LR-FHSS modulation is represented in Figure 8-6: LR-FHSS Spectral Plot Example.

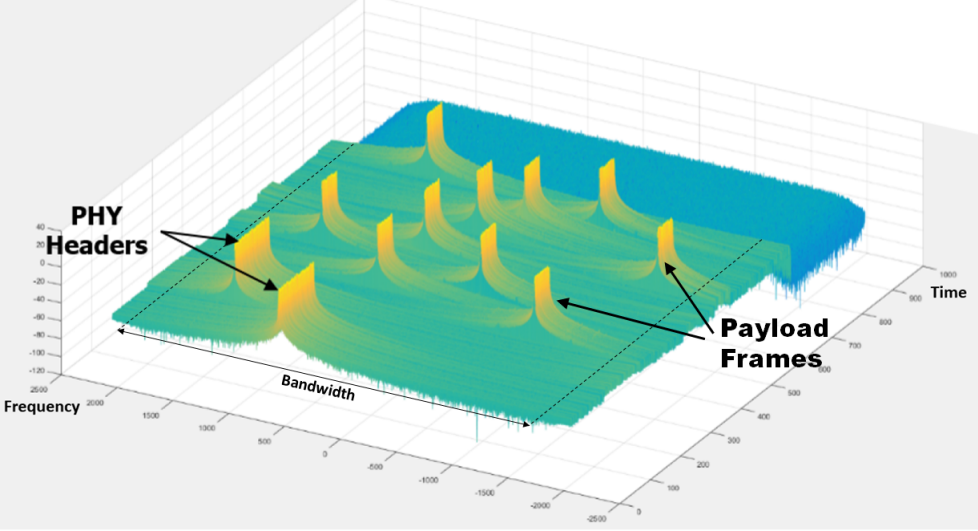

Figure 8-6: LR-FHSS Spectral Plot Example

Each payload frame, except the last frame, contains exactly 50 encoded bits, and are then of equal duration in time. The last frame may contain less than 50 bits. The number of frames varies therefore with the payload size and the Coding Rate.

The Payload frames are preceded with one to four synchronization sequences (Headers) to allow robust frequency and timing synchronization. Even if receiving a single Header is sufficient to be able to decode the payload, a minimum of 2 Headers improves immunity against radio interference. If multiple headers are used, they are sent on different frequencies. Each header contains exactly 114 encoded bits, and are then of equal duration in time.

Additional information is provided in AN1200.58 Long Range FHSS Demo and AN1200.64 Long Range FHSS System Performance.

8.7 LR-FHSS Commands

8.7.1 SetModulationParams

Command `SetModulationParams(...)` configures the modulation parameters for the selected modem. Since the parameters are modem dependent, the description hereafter is valid only for the LR-FHSS modulation.

Table 8-22: SetModulationParams Command

Byte	0	1	2	3	4	5	6
Data from Host	0x02	0x0F	BitRate (31:24)	BitRate (23:16)	BitRate (15:8)	BitRate (7:0)	PulseShape
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0

- *BitRate* defines the LR-FHSS Bit rate in bits per second. Bitrate in bits/s is *Bitrate*(30:0)/256, with *Bitrate*(31) = 1. For example, 488.28215Hz bit rate will be coded *Bitrate* = 125000 with *Bitrate*(31) = 1.

  In the current LR-FHSS implementation, only 488.28215 Hz is supported, other values are RFU.
- *PulseShape* defines the filtering applied to the LR-FHSS packet:
  - 0x0B: Gaussian BT 1. Other values are RFU

8.7.2 LrFhssBuildFrame

Command `LrFhssBuildFrame(...)` encodes the given payload and configures the internal hopping table

Table 8-23: LrFhssBuildFrame Command

Byte	0	1	2	3	4	5	6	7
Data from Host	0x02	0x2C	HeaderCount	CR	ModType	Grid	Hopping	BW
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0

Byte	8	9	10	11	12	...	...
Data from Host	HopSequence (8)	HopSequence (7:0)	DeviceOffset	Payload[0]	Payload[1]	...	Payload[N]
Data to Host	0	0	0	0	0	...	0

- *HeaderCount*: number of Headers for the LR-FHSS packet: 1, 2, 3 or 4
- *CR*: configures the Coding Rate:
  - 0x00: CR = 5/6
  - 0x01: CR = 2/3
  - 0x02: CR = 1/2
  - 0x03: CR = 1/3
- *ModType*: 0 = GMSK modulation, 488.28125bps data rate

- *Grid*: frequency hopping grid step size:
  - 0x00: 25.390625kHz for FCC use case
  - 0x01: 3.90625kHz for non-FCC use case
- *Hopping*: configures intra-packet hopping:
  - 0x00: no hopping (test purpose)
  - 0x01: hopping
- BW: bandwidth occupied by the frequency hopping pattern:
  - 0x00: 39.06kHz
  - 0x01: 85.94kHz
  - 0x02: 136.72kHz
  - 0x03: 183.59kHz
  - 0x04: 335.94kHz
  - 0x05: 386.72kHz
  - 0x06: 722.66kHz
  - 0x07: 773.44kHz
  - 0x08: 1523.4kHz
  - 0x09: 1574.2kHz
  - Other values are RFU
- *HopSequence*: 9-bit integer, seed of the pseudo-random frequency hopping sequence:
  - value in [0; 383] for Grid = 0x00, BW in {0x06, 0x07, 0x08, 0x09}
  - value in [0; 383] for Grid = 0x01, BW in {0x00, 0x01, 0x02, 0x03}
  - value in [0; 511] for Grid = 0x01, BW in {0x04, 0x05, 0x06, 0x07, 0x08, 0x09}
- *DeviceOffset*: per device frequency offset to reduce the risk of packet collision. Frequency offset (Hz) = DeviceOffset * 488.28125Hz:
  - signed value in [-26; 25] for Grid = 25.390625kHz
  - signed values in [-4; 3] for Grid = 3.90625kHz

  For a FCC use case, *DeviceOffset* shall not be changed once configured.
- *Payload*: actual payload to encode. Returns CMD_OK if the parameters are valid and if the payload can be encoded, CMD_PERR otherwise.

The maximum supported packet length is 255 coded bytes. The maximum user payload length is given by the Table 8-24: Maximum user payload length, in bytes.

Table 8-24: Maximum user payload length, in bytes

CR	HeaderCount = 1	HeaderCount = 2	HeaderCount = 3	HeaderCount = 4
CR = 5/6	189	178	167	155
CR = 2/3	151	142	133	123
CR = 1/2	112	105	99	92
CR = 1/3	74	69	65	60

A FCC use case corresponds to a use case where BW = 0x08/0x09, Hopping = 0x01, Grid = 0x00.

Note: This command does not send the LR-FHSS packet. The transmission will be triggered using `SetTx(...)` command.

8.7.3 LrFhssSetSyncWord

Command `LrFhssSetSyncWord(...)` sets the LR-FHSS Syncword

Table 8-25: LrFhssSetSyncWord Command

Byte	0	1	2	3	4	5
Data from Host	0x02	0x2D	Syncword[0]	Syncword[1]	Syncword[2]	Syncword[3]
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

Syncword: LR-FHSS Syncword. Default value is {0x2C, 0x0F, 0x79, 0x95}.

8.7.4 Circuit Configuration for Long Range FHSS Transmission

After power up (battery insertion or hard reset) the LR1121 runs automatically a calibration procedure and goes to STDBY_RC mode. This is indicated by a low state on BUSY pin.

From STDBY_RC, the sequence of operations needed to transmit a Long Range FHSS packet are:

1. If not in STDBY_RC mode, then set the circuit in this mode with the command `SetStandby(...)`.
2. Set the basic configuration parameters such as RF switch, PA configuration, OCP, TxParams, RfFrequency...
3. Define the modulation (LR-FHSS) with the command `SetPacketType(...)`.
4. Set the modulation parameters (`SetModulationParams(...)` command) with the proper Bit Rate (488.28125 bps), the proper Pulse Shape.
5. Call the command `LrFhssBuildFrame(...)`.
6. Activate the IRQ: *TxDone*.
7. Set the circuit in transmitter mode to start transmission with the command `SetTx(...)`.
8. Wait for *TxDone* interrupt.

If configured, *LrFhssHop* IRQ is asserted at each frequency hop inside the LR-FHSS packet, after each PA ramp-up.

8.8 Data Buffer

The LR1121 is equipped with two 255 byte RAM data buffers which are accessible in all modes except sleep mode. One buffer stores the received payload data, while the other contains the payload data to be transmitted.

The LR1121 automatically controls the data pointers, which means that no base address handling by the user is necessary.

- Data Buffer in Receive Mode:
  - The received payload data are stored in the RX buffer:
    - They can be read back using command `ReadBuffer8(...)` (see 3.7.5 ReadBuffer8)
    - `GetRxbufferStatus(...)` reads the pointer to the first byte of the last packet received and the packet length (see 7.2.11 GetRxBufferStatus).
  - `ClearRxBuffer(...)` clears all the data in the LR1121 RX buffer (see 3.7.6 ClearRxBuffer)
- Data Buffer in Transmit Mode:
  - The payload data to be transmitted shall be written the Tx Buffer using command `WriteBuffer8(...)` (see 3.7.4 WriteBuffer8)

8.9 RSSI Functionality

The RSSI information of the LR1121 is available either in the radio chain, or after demodulation at the end of the reception stage. A summary of the RSSI information and their meaning is summarized in table Table 8-26: RSSI Information Origin and Meaning below:

Table 8-26: RSSI Information Origin and Meaning

Command	Modem	Name	Description
`GetRssiInst(...)`	All	*RssiInst*	Instantaneous RSSI
`GetPacketStatus(...)`	(G)FSK	*RssiSync*	Instantaneous RSSI value latched in the (G)FSK demodulator, upon detection of sync address
		RssiAvg	Average RSSI value over the whole payload of the received packet, determined in the (G)FSK demodulator.
	LoRa®	*RssiPkt*	Measurement of the mean energy at the input of the modem over the last packet received.
		SignalRssiPkt	Estimation of the mean energy of the LoRa signal over the last packet received. Equivalent to RssiPkt - environment noise

Refer to each command description for implementation details on the various RSSI fields.

9. Power Amplifiers

The LR1121 features 2 power amplifiers for sub-GHz operation, and one power amplifier for HF (2.4GHz) operation:

- A high power PA (HP PA), optimized for +22dBm operation
- A low power PA (LP PA), optimized for +14dBm operation, capable of +15dBm output power
- A high frequency PA (HF PA), optimized for +13dBm operation in the 2.4GHz band

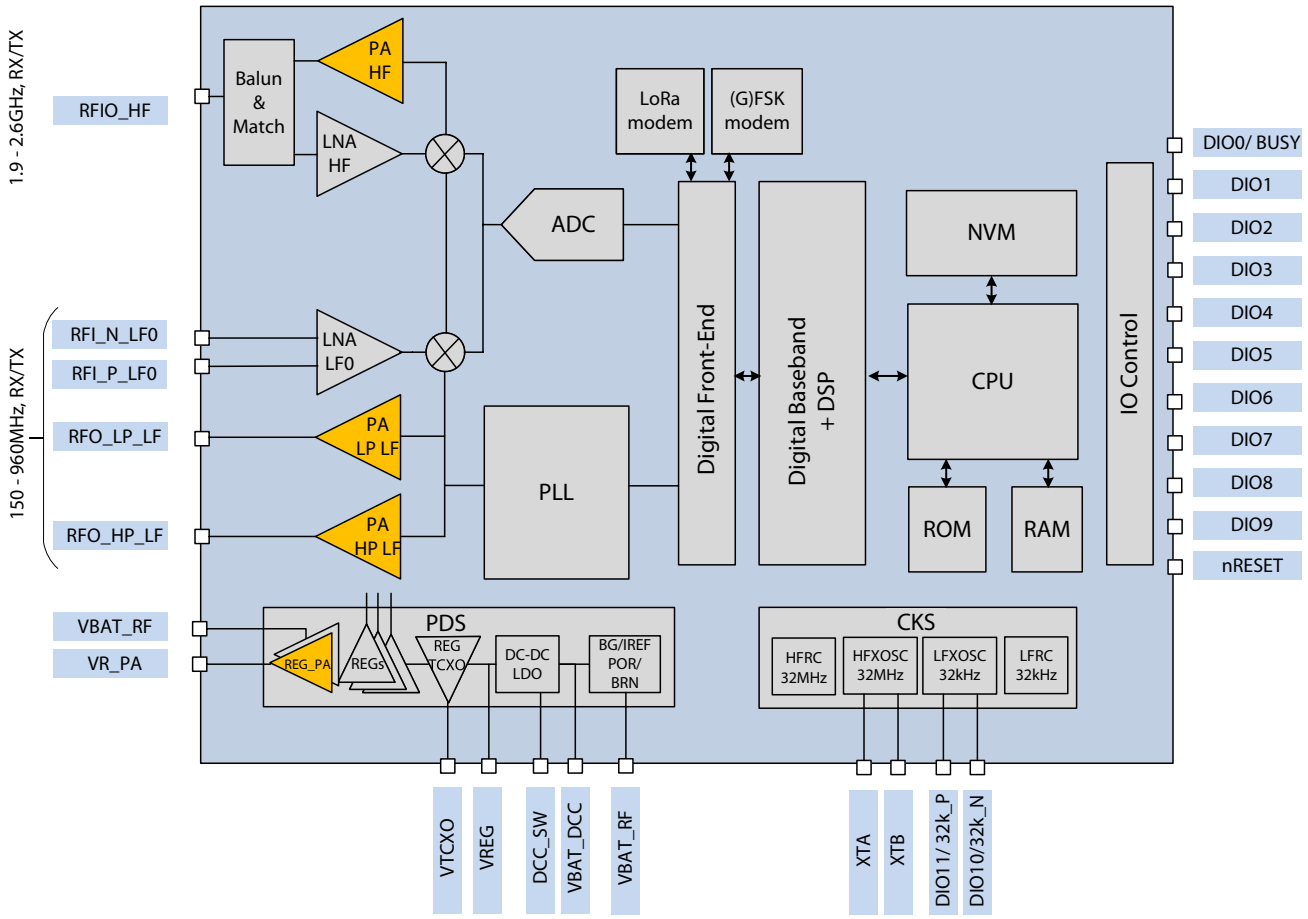

Figure 9-1: LR1121 Power Amplifiers

The PA is configured using two commands: `SetPaConfig(...)` and `SetTxParams(...)`.

The `SetPaConfig(...)` command is used to:

- Select the PA to be used (HP, LP or HF)
- Select the supply of the PA (VBAT or VREG)
- Select the duty cycle of either PA
- Select the size of the PA (only applicable to the high power PA)

The `SetTxParams(...)` command is used to:

- Control the supply voltage of the PA (VR_PA) and output power
- Choose the ramp time at the start / stop of TX

The user should configure the PA corresponding to the output power and current consumption requirements of the application. No automatic frequency limitation is performed during the PA selection: the frequency selection has to be performed according to the capability offered by the external matching network component values.

9.1 PA Supply Scheme

The PA supply scheme is depicted in Figure 9-2: PA Block Diagram.

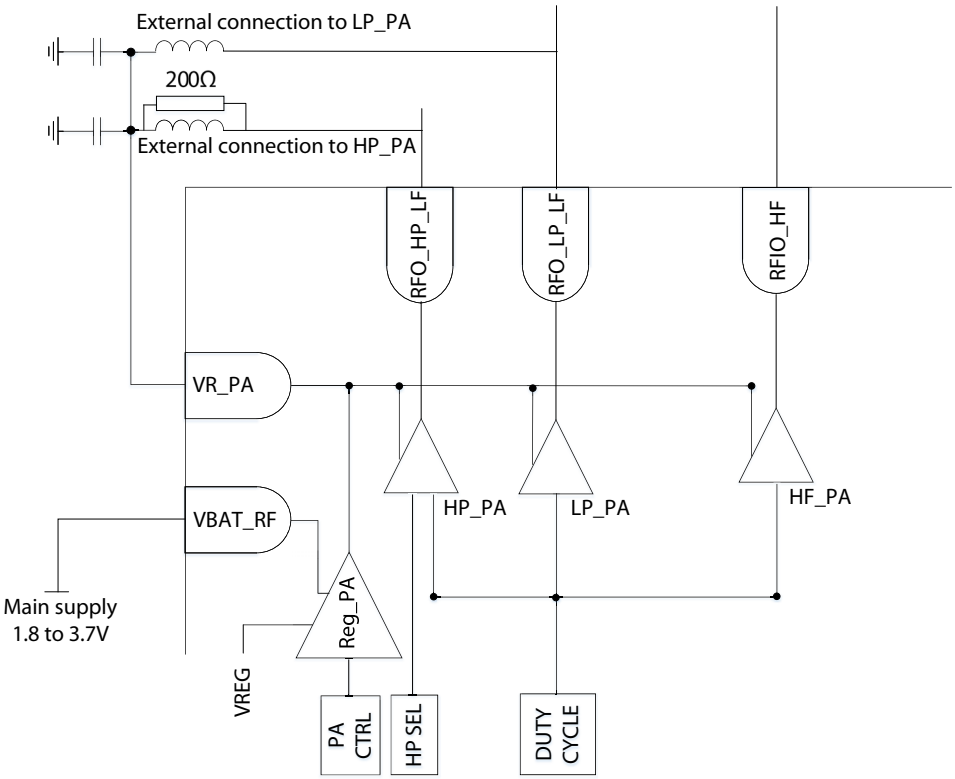

Figure 9-2: PA Block Diagram

The PA regulator (Reg_PA) supplies the high power PA (HP PA), the low power PA (LP PA), and the high-frequency PA (HF PA) through the VR_PA pin. Both the HP PA and the LP PA require a high-Q choke inductor connected externally between their respective outputs, and VR_PA to provide the bias and control the output power. No external choke inductor is necessary for the HF PA. It is best practice to place a 200 Ohm resistor in parallel to the HP PA choke inductor.

The PA regulator is internally connected to the DC-DC /LDO output for the low power PA and high frequency PA, allowing a +15dBm and +13dBm operation in both DCDC or LDO configurations. For the high power PA, it is connected to the VBAT_RF pin, therefore to the main supply voltage. This means that the maximum output power generated by the high power PA depends on the VBAT voltage.

The TX main supply can switch between the battery VBAT and the internal regulator VREG, according to the PA use case. When operating with VR_PA above 1.35V (e.g. in the case of high power), the battery supply VBAT must be chosen. When operating with VR_PA below 1.35V (e.g. in the case of low power PA), either supply can be chosen. However, it is better to choose the internal regulator VREG whenever the required VR_PA is 1.35V or below, in order to benefit from the Buck converter.

The LR1121 incorporates a precise duty cycle trimmer shared between the three power amplifiers. This duty cycle trimmer can be used to trade-off the output power, efficiency, and harmonic emission to address the different regional standard requirements. The duty cycle trimmer is dependent on the impedance presented to the PA, and thus might vary from one PCB design to another.

9.1.1 Low Power PA

For maximum efficiency, the low power PA should be operated with a maximum VR_PA near 1.35V.

To get VR_PA = 1.35V the user should set *TxPower* = 14. At this setting:

- The PA can deliver up to 15dBm output power, constant over the specified battery supply range
- The actual maximum output power can be set according to the duty cycle setting (*PaDutyCycle)*
- If needed, the output power can be decremented in steps of 1dB from maximum by using *TxPower* < 14

The VR_PA variation over the programmed power *TxPower* for different supply voltages and *PaDutyCycle* conditions is depicted in Figure 9-3: Low Power PA VR_PA Voltage vs. *TxPower* below (valid for VBAT and VREG supplies, in both LDO or DCDC configurations).

Note: All figures in this chapter are indicative and typical, and are not a specification. These figures only highlight the behavior of the PA over the various parameters and conditions.

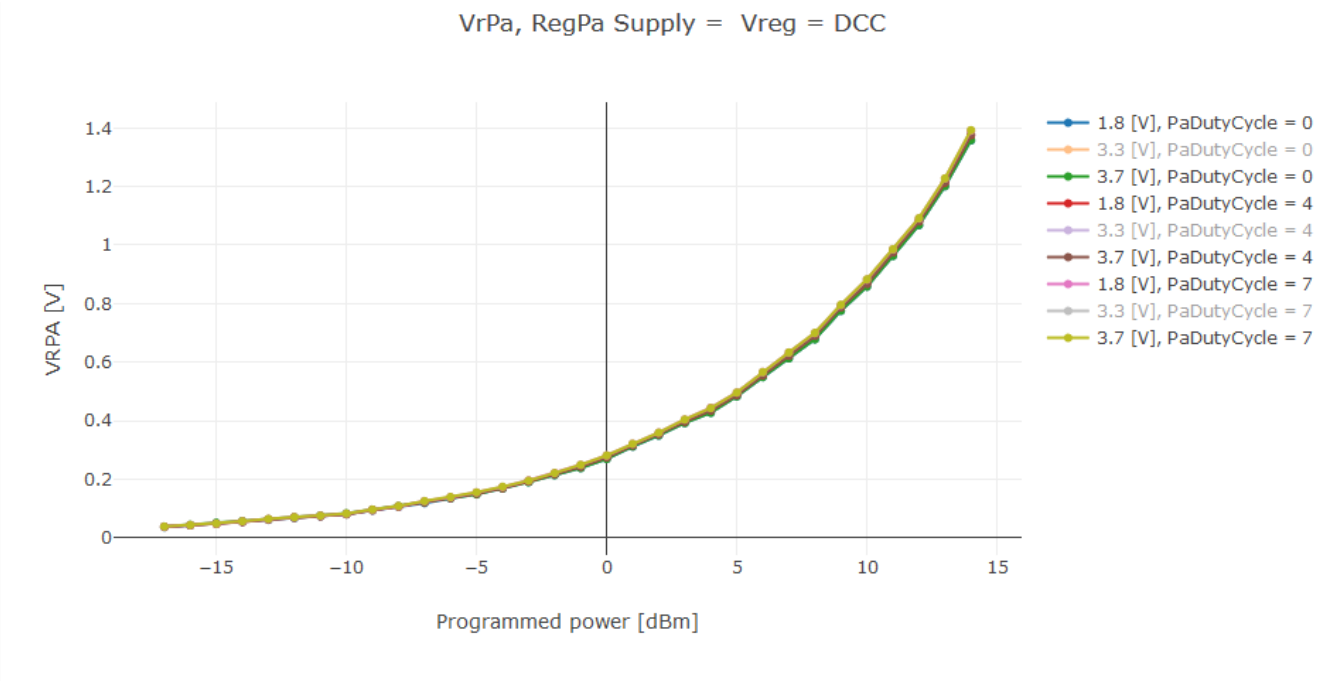

Figure 9-3: Low Power PA VR_PA Voltage vs. TxPower

9.1.2 High Power PA

For maximum efficiency, the high power PA should be operated with a maximum VR_PA near 3.1V.

To get VR_PA = 3.1V the user should set *TxPower* = 22. At this setting:

- The PA can deliver up to 22dBm output power. The output power in this case varies when the battery voltage drops below 3.3V
- The actual maximum output power can be set according to the *PaDutyCycle* and *PaHpSel*
- If needed, the output power can be decremented in steps of 1dB from maximum by using *TxPower* < 22

The VR_PA variation over the programmed power *TxPower* for different supply voltages and duty cycle (*PaDutyCycle*) conditions is depicted in Figure 9-4: High Power PA VR_PA Voltage vs. TxPower below.

The internal regulator for VR_PA has 200mV of drop-out, which means VBAT must be 200mV higher than the VR_PA voltage in order to attain the corresponding output power.

For example: For Pout = +20dBm,

- VR_PA = 2.5V is required (brown curve)
- The high power PA can maintain Pout = +20dBm on the 2.7V < VBAT < 3.7V voltage range (2.5V +200mV = 2.7V)
- Below 2.7V, the output power degrades as VBAT reduces

At 1.8V of supply voltage, the maximum VR_PA value is 1.6V (1.8V - 200mV), allowing therefore a +17dBm output power.

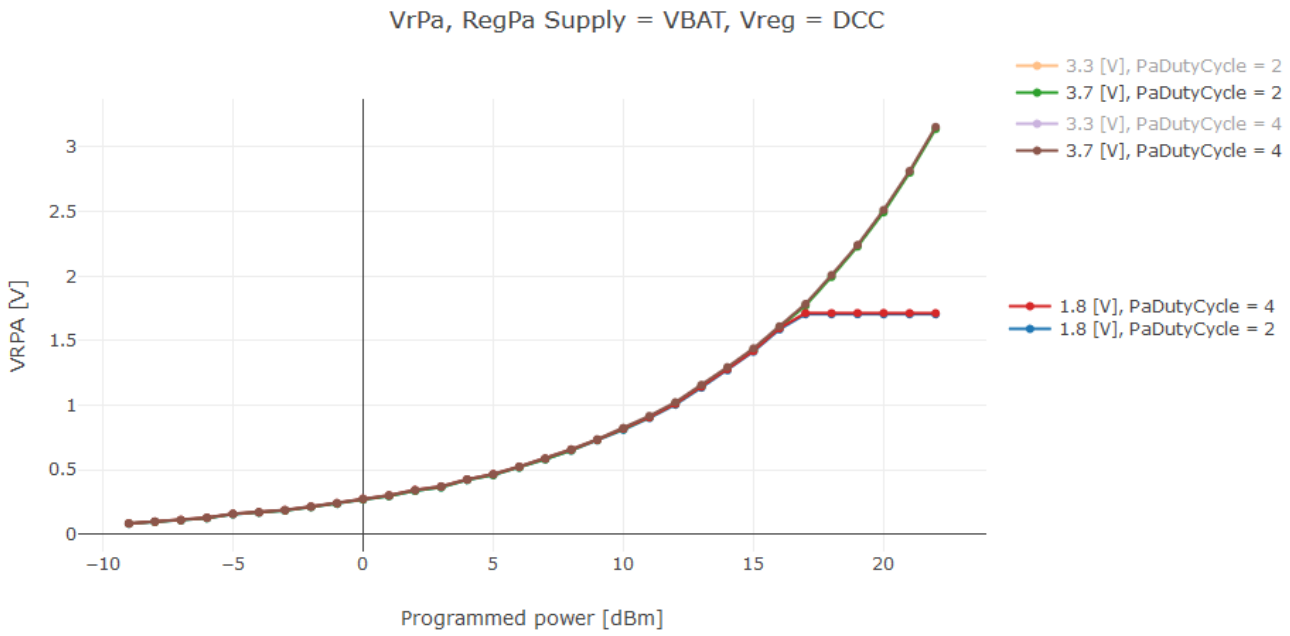

Figure 9-4: High Power PA VR_PA Voltage vs. TxPower

9.2 PA Output Power

As stated previously, two parameters do have an impact on the TX output power generated by each PA: the programmed power *TxPower* and the duty cycle *PaDutyCycle*. A third parameter, *PaHPSel*, controls the size of the high power PA, and therefore has a direct impact on the high power PA output power.
In order to reach +22dBm output power, *PaHPSel* must be set to 7. *PaHPSel* has no impact on either the low power PA or the high frequency PA.

9.2.1 Low Power PA

Figure 9-5: Low Power PA Output Power vs. TxPower shows the output power of the low power PA with *TxPower* for different *PaDutyCycle* settings and over the supply voltage.

The supply voltage has no impact on the output power, since the low power PA is internally regulated. Only the *PaDutyCycle* has an influence on the output power. Therefore the plots for 1.8V, 3.3V and 3.7V are superimposed, and only the plots for 3.7V are visible.

For example:

- *TxPower* = 14 and *PaDutyCycle* = 0 gives +10dBm whatever the supply voltage (1.8V, 3.3V and 3.7V)
- *TxPower* = 14 and *PaDutyCycle* = 4 gives +14dBm whatever the supply voltage (1.8V, 3.3V and 3.7V)
- *TxPower* = 14 and *PaDutyCycle* = 7 gives +15dBm whatever the supply voltage (1.8V, 3.3V and 3.7V)

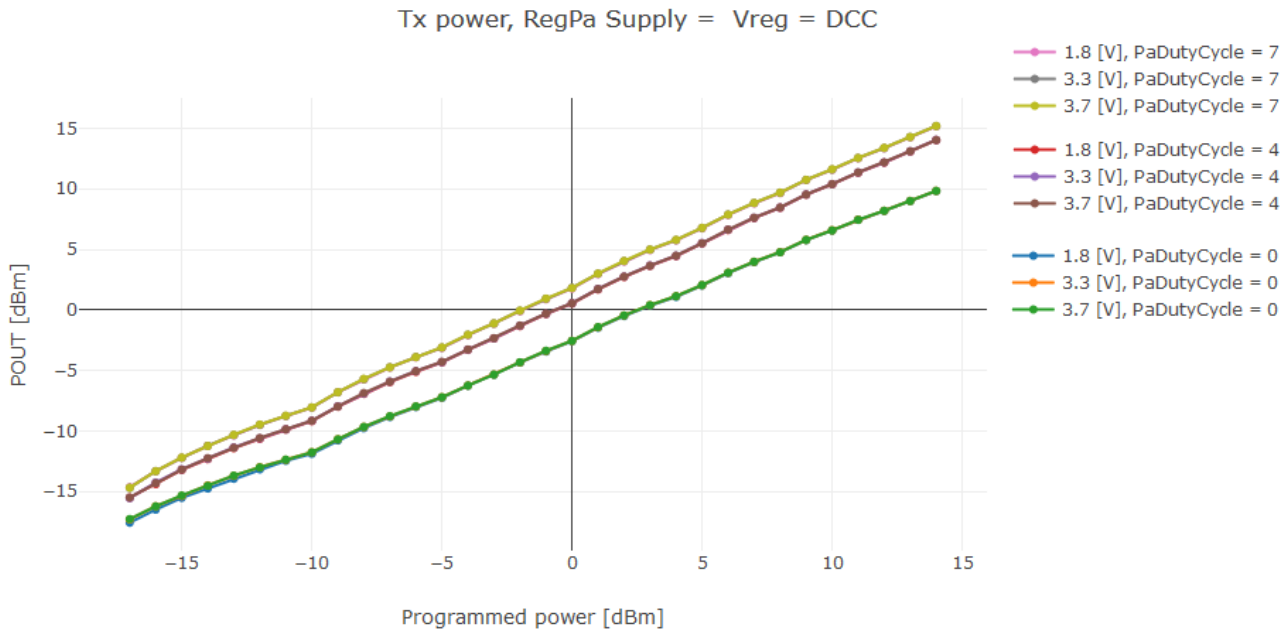

Figure 9-5: Low Power PA Output Power vs. TxPower

9.2.2 High Power PA

Figure 9-6: HP PA Output Power vs. TxPower shows the output power of the high power PA with *TxPower* for different *PaDutyCycle* settings and over the supply voltage.

For a given *PaDutyCycle*, the output power of the high power PA is maintained on a certain voltage range, and then decreases with VBAT. For example:

- For the +22dBm power setting, a VR_PA of ~3.1V is required (refer to Figure 9-4). Therefore, given the 200mV drop-out of the PA regulator, the +22dBm output power can only be obtained from a 3.3V to 3.7V supply voltage range.
- For +17dBm, VR_PA around 2V is required. Therefore the LR1121 output power will drop to +17dBm for the minimum supply voltage 1.8V.

Therefore, the 3.3V and 3.7V plots are then superimposed for a given *PaDutyCycle*, and only the plots for 3.7V are visible.

For a given supply voltage, increasing the *PaDutyCycle* increases the output power.

For example:

- For the +22dBm power setting at 3.3V, *PaDutyCycle* = 4 allows the high power PA to deliver +22dBm
- For the +22dBm power setting at 3.3V, *PaDutyCycle* = 2 allows the high power PA to deliver +20dBm

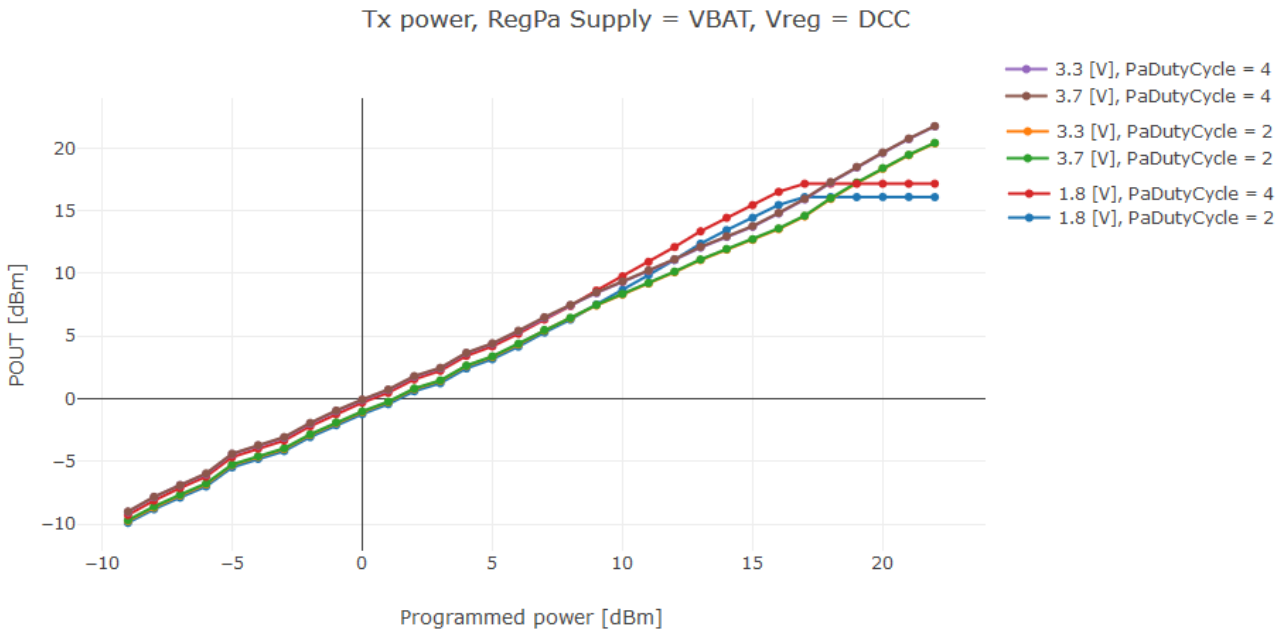

Figure 9-6: HP PA Output Power vs. TxPower

9.2.3 High Frequency PA

Figure 9-7: High Frequency PA Output Power vs. TxPower shows the output power of the high frequency PA with TxPower for different supply voltage values.

- The supply voltage has no impact on the output power, since the HF PA is internally regulated. Therefore the plots for 1.8V, 3.3V and 3.7V are superimposed, and only the plots for 3.7V are visible.

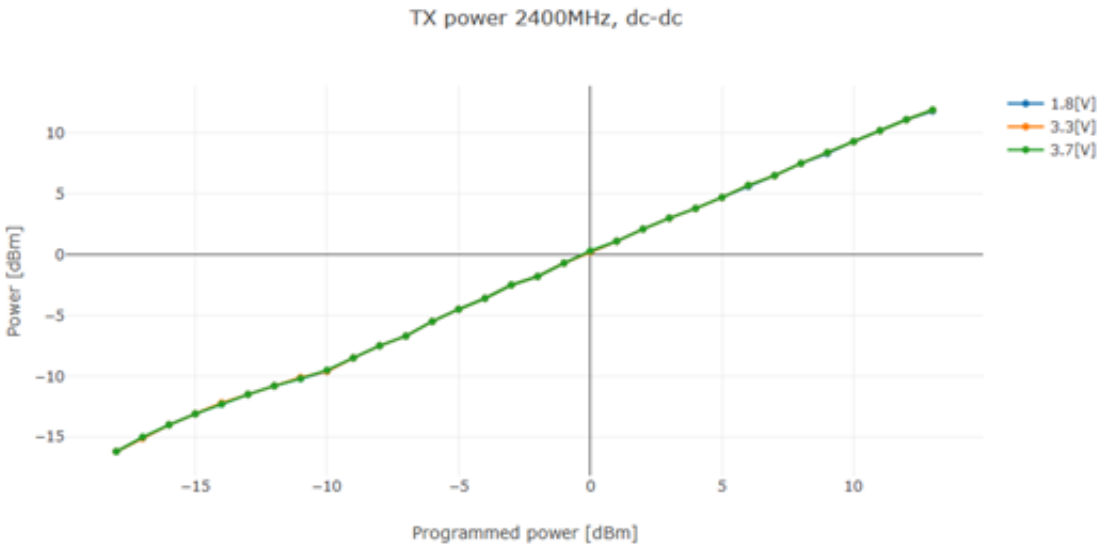

Figure 9-7: High Frequency PA Output Power vs. TxPower

9.3 PA Current Consumption

9.3.1 Low Power PA

Figure 9-8: IDDTX vs TxPower, Low Power PA, DC-DC Configuration shows the impact of the supply voltage for three *PaDutyCycle* settings (0, 4 and 7) in DC-DC configuration.

At a given supply voltage, a higher *PaDutyCycle* setting increases the device current consumption. At a given *PaDutyCycle* setting, the current consumption is optimum for a supply voltage equal or greater to 3.3V, therefore the plots for 3.3V and 3.7V are superimposed. A power supply of 1.8V is not be as power efficient as 3.3V or more, resulting in a higher current consumption.

For example:

- For 3.7V, *PaDutyCycle* = 0 the current consumption is approx. 28mA, for *PaDutyCycle* = 4 approx. 47mA and for *PaDutyCycle* = 7 approx. 62mA
- For *PaDutyCycle* = 4, the current consumption is approx. 47mA for 3.3V and 3.7V, and approx. 49mA for 1.8V

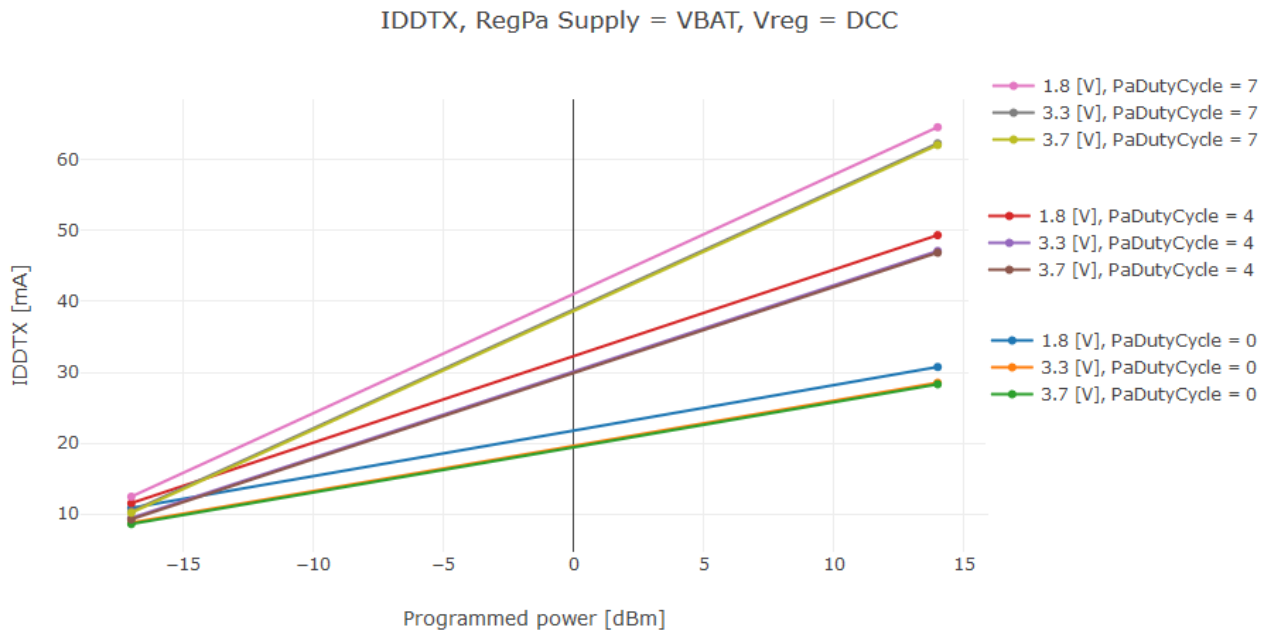

Figure 9-8: IDDTX vs TxPower, Low Power PA, DC-DC Configuration

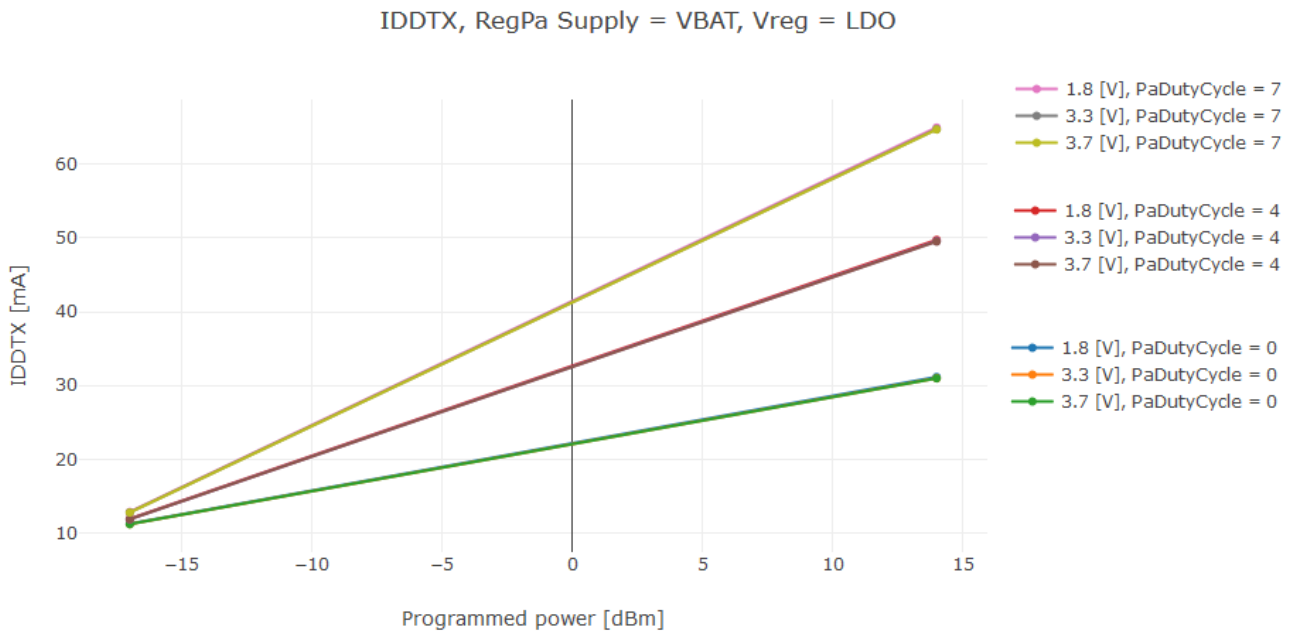

Figure 9-9: IDDTX vs TxPower, Low Power PA, LDO Configuration

Figure 9-9: IDDTX vs TxPower, Low Power PA, LDO Configuration shows the impact of the supply voltage for three *PaDutyCycle* settings (0, 4 and 7) in LDO configuration.

Similarly to the DC-DC configuration, we can see that, at a given supply voltage, a higher *PaDutyCycle* setting increases the device current consumption. However, the supply voltage has no influence on the current consumption at a given *PaDutyCycle* setting, which means that the plots for 1.8V, 3.3V, and 3.7V are superimposed.

Figure 9-8 and Figure 9-9 show that the power efficiency of the low power PA is maximized when the internal DC-DC regulator is used at, or above, 3.3V.

9.3.2 High Power PA

Figure 9-10: IDDTX vs TxPower, High Power PA, DC-DC Configuration and Figure 9-11: IDDTX vs TxPower, High Power PA, LDO Configuration shows the impact of the supply voltage for two *PaDutyCycle* settings (2 and 4) in both DC-DC and LDO configurations.

Similarly to the low power PA, at a given supply voltage a higher *PaDutyCycle* setting increases the device current consumption. However, at a given *PaDutyCycle* setting, the current consumption is stable with respect to the supply voltage, providing this latter is high enough to allow the generation of the VR_PA voltage required for the programmed power value *TxPower*.

For example:

- For 3.3V, the current consumption is approx. 98mA for *PaDutyCycle* = 2, and approx. 118mA for *PaDutyCycle* = 4
- For 1.8V, the current consumption is approx. 69mA for *PaDutyCycle* = 2, and approx. 81mA for *PaDutyCycle* = 4. This is due to the fact that at 1.8V supply voltage, the maximum VR_PA voltage is 1.6V, therefore a maximum output power of +17dBm

During the high power PA operation, the DC-DC supplies the analog and digital core of the devices, whereas the PA itself -the largest power consumption contributor- is supplied directly from VBAT. Therefore, there is no significant current consumption difference between the DC-DC or the LDO modes during the high power PA operation.

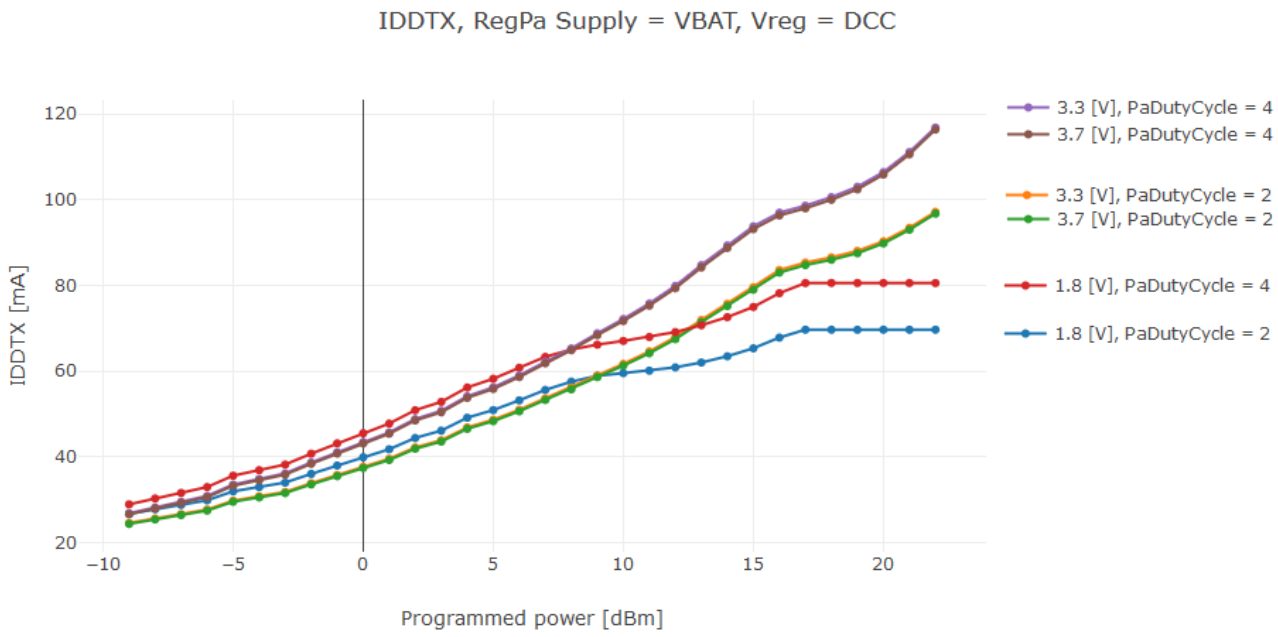

Figure 9-10: IDDTX vs TxPower, High Power PA, DC-DC Configuration

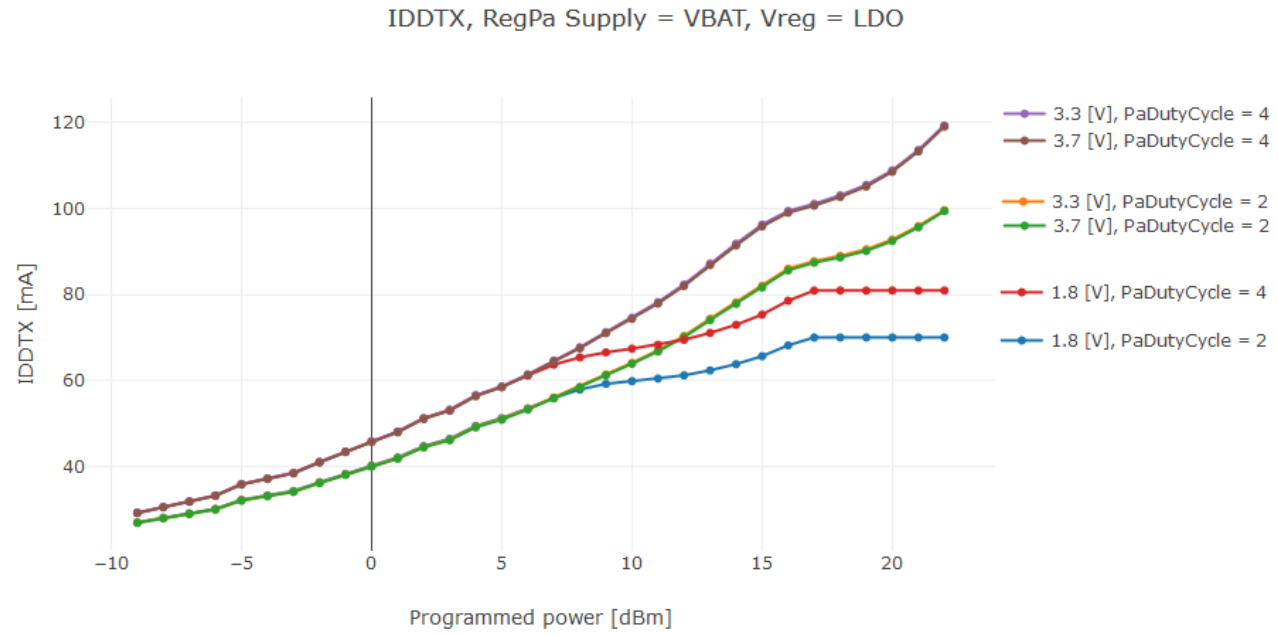

Figure 9-11: IDDTX vs TxPower, High Power PA, LDO Configuration

9.3.3 High Frequency PA

Figure 9-12 shows the impact of the TxPower setting for three supply voltages in a DC-DC configuration.

The current consumption is optimum for a supply voltage equal or greater to 3.3V, therefore the plots for 3.3V and 3.7V are superimposed. A power supply of 1.8V is not as power efficient as 3.3V or more, resulting in a higher current consumption.

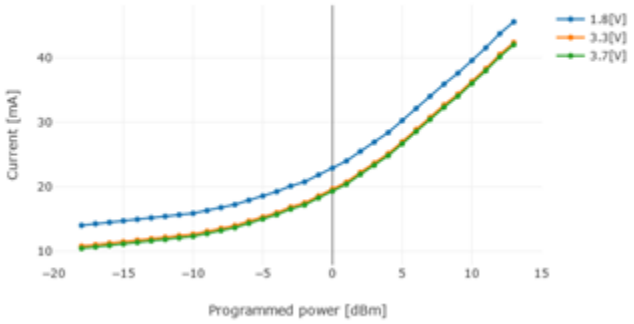

Figure 9-12: IDDTX vs TxPower, High Frequency PA, DC-DC Configuration

Figure 9-13: shows the impact of the TxPower setting for three supply voltages in LDO configuration.

The supply voltage has no influence on the current consumption, which means that the plots for 1.8V, 3.3V, and 3.7V are superimposed.

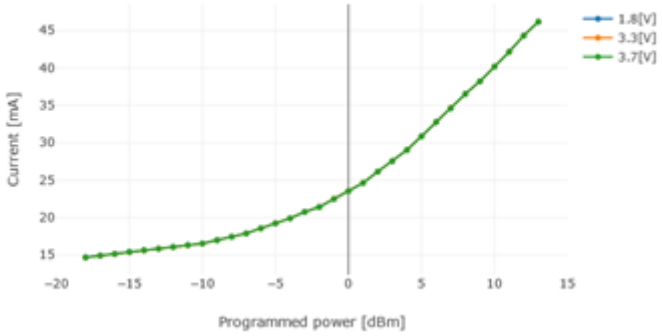

Figure 9-13: IDDTX vs TxPower, High Frequency PA, LDO Configuration

Figure 9-12 and Figure 9-13 show that the power efficiency of the High Frequency PA is maximized when the internal DC-DC regulator is used at, or above, 3.3V.

9.4 Impedance Matching Networks

The high power PA and low power PAs are available on the RFO_HP_LF and RFO_LP_LF pins respectively. They are connected to the antenna through a dedicated impedance matching network, which aims at presenting the optimized load at the output pins when loaded with 50Ohms at the antenna level.

The High Frequency PA is available on the RFIO-HF pin. It is internally matched at 50Ohms, reducing the need of external components for HF operation. Some components present on the RFIO_HF pin in the Semtech reference design are aimed at providing filtering to comply with the RF regulations.

9.4.1 Multi-Band Operation

It is possible to implement a multi-band configuration using a single impedance matching network, allowing the same set of SMD components to cover multiple sub-GHz ISM bands. Table 9-1, Table 9-2 and Table 9-3 show the optimal settings for the PA when using the Semtech matching network obtained on Semtech' s reference design. The user can fine-tune the *PaDutyCycle* and *PaHpSel* according to their requirements of matching network, efficiency, output power, and harmonic emission. Please note that the effect of *PaDutyCycle* and *PaHpSel* on the efficiency, output power, and harmonic emission are PCB-design dependent, therefore the PA settings shown in must be optimized on the user's application.

The matching network implementation proposed by Semtech is optimized for +22dBm and +15dBm for the higher ISM bands, i.e. a 868-928MHz operation.

Table 9-1: Optimized Settings for LP PA with the Same Matching Network

Target Power	*TxPower*	*PaSel*	*RegPASupply*	*PaDutyCycle*	*PaHPSel*
+15dBm	14	0	0	7	-
+14dBm	14	0	0	5	-
+10dBm	14	0	0	1	-

Table 9-2: Optimized Settings for HP PA with the Same Matching Network

Target Power	*TxPower*	*PaSel*	*RegPASupply*	*PaDutyCycle*	*PaHPSel*
+22dBm	22	1	1	4	7
+20dBm	22	1	1	3	7
+17dBm	22	1	1	1	5
+14dBm	22	1	1	1	3

Table 9-3: Optimized Settings for HF PA with the Same Matching Network

Target Power	*TxPower*	*PaSel*	*RegPASupply*	*PaDutyCycle*	*PaHPSel*
+13dBm	13	2	0	0	0
+11dBm	12	2	0	3	0
+10dBm	11	2	0	4	0

9.4.2 RF Switch Implementation

The implementation examples hereafter show a combined high power PA and high efficiency PA operation, with the use of a 3-port RF switch SP3T. A single-band operation is also possible, the unused PA pin being left unconnected. In that case a 2-port RF switch SPDT would be necessary.

The RF switch implementation optimizes the impedance presented to the PA and the impedance presented to the LNA separately. Therefore one can optimize TX efficiency without compromising RX sensitivity.

The RF switch can be controlled either by the host controller, or by the LR1121 itself (pins DIO5, DIO6, DIO7, DIO8 and DIO10), using the `SetDioAsRfSwitch(...)` command.

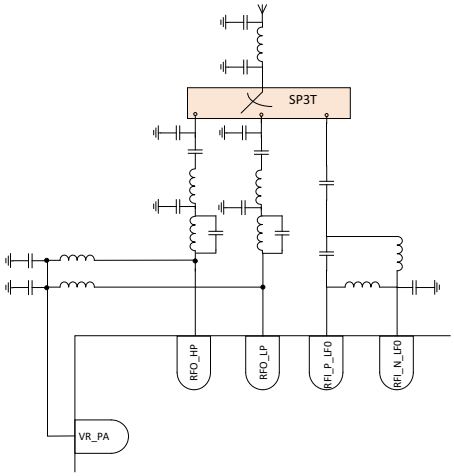

Figure 9-14: RF Switch, Double PA Operation

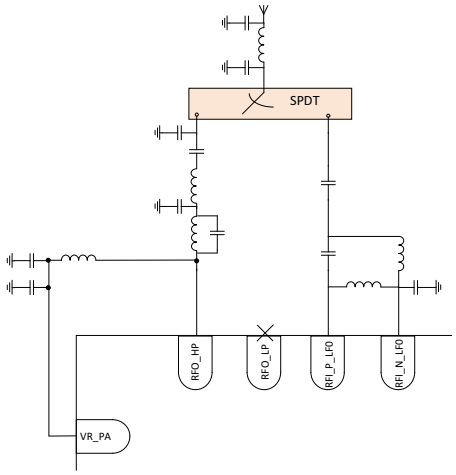

Figure 9-15: RF Switch, Single PA Operation (High Power PA Example)

9.4.3 Direct-Tie Implementation

In case of a cost-sensitive application, it is possible to get rid of the RF switch, and implement a so-called direct-tie implementation.

In such a configuration, the PA and the RX differential stages are connected as depicted in the figure hereafter. Please note that series capacitances are required between the PA and the RX stage in order to avoid damaging the LR1121 due to current flow in the RX stage.

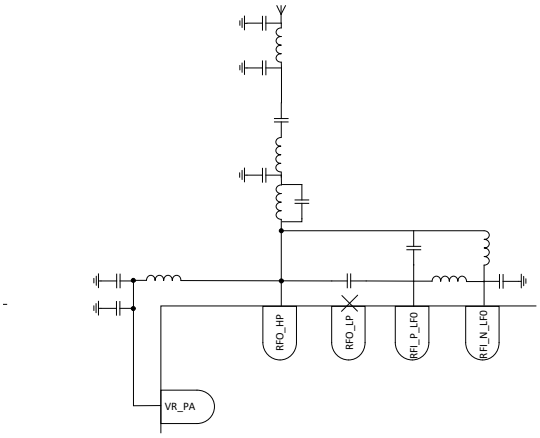

Figure 9-16: Single Tie implementation: Only one PA Used (High Power PA Example)

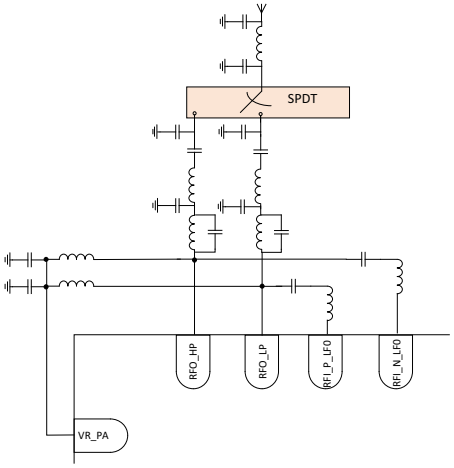

Figure 9-17: Single Tie implementation: Both PAs Used (High Power PA Example)

Compared to the switched implementation, the direct-tie suffers a trade-off between TX efficiency and RX sensitivity. This is unavoidable because the transmitter and receiver require different optimal impedances, which may not be simultaneously feasible. In the case of a direct-tie, the user should expect a degradation of 2 ~ 3dB in RX sensitivity.

9.5 Commands

9.5.1 SetPaConfig

Command `SetPaConfig(...)` selects which PA to use and configures the supply of this PA.

Table 9-4: SetPaConfig Command

Byte	0	1	2	3	4	5
Data from Host	0x02	0x15	PaSel	RegPaSupply	PaDutyCycle	PaHPSel
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *PaSel* selects the PA:
  - 0x00: Selects the low power PA
  - 0x01: Selects the high power PA
  - 0x02: Selects the high frequency PA
- *RegPaSupply* selects the PA power source:
  - 0x00: Powers the PA from the internal regulator
  - 0x01: Powers the PA from VBAT. The user must use *RegPaSupply* = 0x01 whenever *TxPower* > 14
- *PaDutyCycle* controls the duty cycle of each PA. Its default value is 0x04.

Table 9-5: PaDutyCycle Parameter Allowed Ranges

Low Power PA	High Power PA	High Frequency PA
For frequency ≥ 400MHz, 0 ≤ *PaDutyCycle* ≤ 0x07 For frequency < 400MHz, 0 ≤ *PaDutyCycle* ≤ 0x04	At PaHpSel ≥ 0x05, 0 ≤ *PaDutyCycle* ≤ 0x04 At PaHpSel ≤ 0x04, 0 ≤ *PaDutyCycle* ≤ 0x07	0 ≤ *PaDutyCycle* ≤ 0x04

- *PaHPSel* controls the size of the high power PA.

9.5.2 SetTxParams

Command `SetTxParams(...)` sets the Tx Power and Ramp Time of the selected PA. `SetPaConfig(...)` must be sent prior to this command.

Table 9-6: SetTxParams Command

Byte	0	1	2	3
Data from Host	0x02	0x11	TxPower	RampTime
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)

- *TxPower* defines the output power in dBm in a range of:
  - - 17dBm (0xEF) to +14dBm (0x0E) by steps of 1dB if the low power PA is selected
  - - 9dBm (0xF7) to +22dBm (0x16) by steps of 1dB if the high power PA is selected
  - -18dBm (0xEE) to +13dBm (0x0F) by steps of 1dB if the high frequency PA is selected
  - If *TxPower* > +14dBm, the user must select the VBAT supply for the PA using the `SetPaConfig` command
- *RampTime* defines the PA power ramping time, which can be from 16 to 304µs according to the following table:

Table 9-7: RampTime Values

RampTime	Value	Ramp Time in µs
SET_RAMP_16U	0x00	16
SET_RAMP_32U	0x01	32
SET_RAMP_48U	**0x02**	**48**
SET_RAMP_64U	0x03	64
SET_RAMP_80U	0x04	80
SET_RAMP_96U	0x05	96
SET_RAMP_112U	0x06	112
SET_RAMP_128U	0x07	128
SET_RAMP_144U	0x08	144
SET_RAMP_160U	0x09	160
SET_RAMP_176U	0x0A	176
SET_RAMP_192U	0x0B	192
SET_RAMP_208U	0x0C	208
SET_RAMP_240U	0x0D	240
SET_RAMP_272U	0x0E	272
SET_RAMP_304U	0x0F	304

A *RampTime* of 48µs allows the best trade-off between a fast RF power establishment and minimum RF spurious emissions, therefore complying with radio standards.

10. Cryptographic Engine

10.1 Description

The Cryptographic Engine provides a dedicated hardware accelerator for AES-128 encryption based algorithms and dedicated flash and RAM memory to handle device parameters such as encryption keys, with no read access possible.

The Cryptographic Engine improves the power efficiency of cryptographic operations and reduces the code size of the software stack. Verifying the integrity of data such as the payload of downlink frames is important to guarantee a secure communication. The message integration check (MIC) uses the AES-CMAC algorithm to calculate a hash. Implementing the MIC calculation in software would jeopardize the confidentiality of the used key. The cryptographic engine provides a hardware implementation of the AES-CMAC to internally calculate and check the MIC.

The status of cryptographic operations can be checked by either polling the internal status register or using an interrupt service routine.

10.2 Cryptographic Keys Definition

The cryptographic keys are arranged into several groups, according to the function they serve, as shown in Table 10-1: Cryptographic Keys Usage and Derivation. The table summarizes the allowed uses of the keys and if some of the keys can be derived from other keys.

Table 10-1: Cryptographic Keys Usage and Derivation

Group Name	Key Source/ Dest. Index	Key Name	Usage	Derivation From

Network	2	NwkKey	`CryptoProcessJoinAccept()`
			`CryptoComputeAesCmac()`
			`CryptoDeriveKey()`
			`CryptoSetKey(...)`
				DKEY[1]

Application	3	AppKey	`CryptoDeriveKey()` `CryptoSetKey(...)`	DKEY[1]

LifeTimeEnc	4	JSEncKey	`CryptoProcessJoinAccept()`
			`(Decryption)`
			`CryptoSetKey(...)`
				From Network & Application

LifeTimeInt	5	JSIntKey	`CryptoProcessJoinAccept()`
			`(MIC Computation)`
			`CryptoComputeAesCmac()`
			`CryptoSetKey(...)`
				From Network & Application

GpTransport	6	GpKEKey0	`CryptoDeriveKey(...)`
			`CryptoSetKey(...)`
			`Any multicast Key`
				From any other Gp Transport key or from Application Key

	7	GpKEKey1		
	8	GpKEKey2		
	9	GpKEKey3		
	10	GpKEKey4		
	11	GpKEKey5		

Table 10-1: Cryptographic Keys Usage and Derivation (Continued)

Group Name	Key Source/ Dest. Index	Key Name	Usage	Derivation From

Unicast	12	AppSKey	`CryptoAesEncrypt01(...)`
			`CryptoComputeAesCmac()`
			`CryptoSetKey(...)`
				From Network & Application

	13	FNwkSIntKey		
	14	SNwkSIntKey		
	15	NwkSEncKey		
	16	RFU0		
	17	RFU1		

Multicast	18	McAppSKey0	`CryptoAesEncrypt01(...)`
			`CryptoVerifyAesCmac(...)`
			`CryptoSetKey(...)`
				Only from GpTransport Key

	19	McAppSKey1		
	20	McAppSKey2		
	12	McAppSKey3		
	22	McNwkSKey0		
	23	McNwkSKey1		
	24	McNwkSKey2		
	25	McNwkSKey3		

General Purpose	26	GP0	`CryptoAesEncrypt(...)`
			`CryptoAesDecrypt(...)`
			`CryptoSetKey(...)`
				Not Allowed

	27	GP1		

1. Built-in the device, derived upon `DeriveRootKeysAndGetPin()` command.

10.3 Commands

10.3.1 CEStatus

The Crypto Status byte *CEStatus* indicates the Crypto Engine state. It is returned after each command invoking the Crypto Engine.

CEStatus:

- 0: CRYP_API_SUCCESS. The previous command was successful
- 1: CRYP_API_FAIL_CMAC. MIC (first 4 bytes of the CMAC) comparison failed
- 2: RFU
- 3: CRYP_API_INV_KEY_ID. Key/Param Source or Destination ID error
- 4: RFU
- 5: CRYP_API_BUF_SIZE. Data buffer size is invalid. For `CryptoAesEncrypt(...)` the buffer size must be multiple of 16 bytes
- 6: CRYP_API_ERROR. Any other error

10.3.2 CryptoSetKey

Command `CryptoSetKey(...)` sets a specific *Key* identified by *KeyID* into the Crypto Engine:

Table 10-2: CryptoSetKey Command

Byte	0	1	2	3	4	5	...	18
Data from Host	0x05	0x02	KeyID (7:0)	Key1	Key2	Key3	…	Key16
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	…	0x00

Table 10-3: CryptoSetKey Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *KeyID* goes from 2 to 27, as defined in Table 10-1: Cryptographic Keys Usage and Derivation. Other values are reserved.
- *Key* is an array of bytes as defined in the FIPS-197. With the key K (2b7e1516 28aed2a6abf71588 09cf4f3c) provided in test vectors of the rfc4493, we then have *Key1* = 0x2b, *Key2* = 0x7e, *Key3* = 0x15, *Key4* = 0x16 thru to *Key16* = 0x3c.
- *CEStatus* is defined in section CEStatus on page 102

10.3.3 CryptoDeriveKey

Command `CryptoDeriveKey(...)` derives (encrypts) into a specific Key identified by *DstKeyID*, the input (including the LoRaWAN DevNonce) value provided, using a source key identified by *SrcKeyID*.

Table 10-4: CryptoDeriveKey Command

Byte	0	1	2	3	4	5	6	...	19
Data from Host	0x05	0x03	SrcKeyID (7:0)	DstKeyID (7:0)			Input[1:16]		
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	…	0x00

Table 10-5: CryptoDeriveKey Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *DstKeyID* and *SrcKeyID* for this function are defined in Table 10-1: Cryptographic Keys Usage and Derivation
  - *DstKeyID*: Destination Key ID. Valid values are from 4 to 25
  - *SrcKeyID:* Source Key IDs 2-3 and 6-11 are possible for this function
- *Input[1:16]* is an array of bytes. An example of its construction is given in Section 11.3 and Section 11.4
- *CEStatus* is defined in section CEStatus on page 102

Note: At the end of the `CryptoDeriveKey()` process, the generated key is located in the dedicated Crypto Engine RAM, and can be stored in the flash memory using the `CryptoStoreToFlash()` command.

10.3.4 CryptoProcessJoinAccept

Command `CryptoProcessJoinAccept(...)` decrypts the join accept message (using AES-ECB encrypt as per LoRaWAN specification) on the Data and Header, and then verifies the MIC of the decrypted message.

The decrypted data is then provided, if the MIC verification is successful.

Table 10-6: CryptoProcessJoinAccept Command

Byte	0	1	2	3	4	5	…	N+4	N+5	...	N+4+M
Data from Host	0x05	0x04	DecKeyID (7:0)	VerKeyID (7:0)	LoRa WANVer (7:0)	Header1	…	HeaderN	Data1	...	DataM
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	…	0x00	0x00	...	0x00

Table 10-7: CryptoProcessJoinAccept Response

Byte	0	1	2	...	M+1
Data from Host	0x00	0x00	0x00	...	0x00
Data to Host	Stat1	CEStatus	Data1	...	DataM

- *DecKeyID* and *VerKeyID* are defined in Table 10-1: Cryptographic Keys Usage and Derivation*:*
  - *DecKeyID*: Key used for decryption of the message
  - *VerKeyID*: Key used for MIC verification
- *LoRaWANVer*: Determines the expected *Header* size N: 1 byte (v1.0) or 12 bytes (v1.1)
  - *LoRaWANVer* = 0: LoRaWAN version 1.0.x, only MHDR (1 byte)
  - *LoRaWANVer* = 1: LoRaWAN version 1.1.x, SIntKey, JoinReqType | JoinEUI | DevNonce | MHDR
- *Header1 to HeaderN*: Header. N depends on *LoRaWANVer*
- *Data1 to DataM*: Data. Data size M is either 16 or 32 bytes. Data must include the encrypted MIC
- *CEStatus* is defined in section CEStatus on page 102

10.3.5 CryptoComputeAesCmac

Command `CryptoComputeAesCmac(...)` computes the AES CMAC of the provided data using the specified Key and returns the MIC.

Table 10-8: CryptoComputeAesCmac Command

Byte	0	1	2	3	4	5	...	N+2
Data from Host	0x05	0x05	KeyID (7:0)	Data1	Data2	Data3	…	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	…	0x00

Table 10-9: CryptoComputeAesCmac Response

Byte	0	1	2	3	4	5
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	CEStatus	MIC1	MIC2	MIC3	MIC4

- *KeyID*: Specified Key ID, as defined in Table 10-1: Cryptographic Keys Usage and Derivation; 2, 5, 12-17 are supported
- *Data1, Data2, ..., DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section CEStatus on page 102
- *MIC*: Message Integrity Check (first 4 bytes of the CMAC)

For example, when using the test vectors of the RFC4493 example 2, we would have:

- Message: 6BC1BEE2 2E409F96 E93D7E11 7393172A (N = 16)
- MIC: 070A16B4

Table 10-10: CryptoComputeAesCmac Command Example

Byte	0	1	2	3	4	5	...	18
Data from Host	0x05	0x05	KeyID (7:0)	0x6b	0xc1	0xbe	...	0x2a
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	…	0x00

Table 10-11: CryptoComputeAesCmac Response Example

Byte	0	1	2	3	4	5
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	CEStatus	0x07	0x0a	0x16	0xb4

10.3.6 CryptoVerifyAesCmac

Command `CryptoVerifyAesCmac(...)` computes the AES CMAC of the provided data using the specified Key, and compares the provided MIC with the actual calculated MIC (first 4 bytes of the CMAC).

Table 10-12: CryptoVerifyAesCmac Command

Byte	0	1	2	3	4	5	6	7	...	N+6
Data from Host	0x05	0x06	KeyID (7:0)	Expected MIC1	Expected MIC2	Expected MIC3	Expected MIC4	Data1	…	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	…	0x00

Table 10-13: CryptoVerifyAesCmac Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *KeyID*: Specified Key ID, as defined in Table 10-1: Cryptographic Keys Usage and Derivation. 2, 5; 12-25 are possible
- *ExpectedMIC*: Provided MIC (first 4 bytes of the CMAC)
- *Data1, Data2 to DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section CEStatus on page 102

If the 2 MICs are identical, the command returns CRYP_API_SUCCESS, otherwise, CRYP_API_FAIL_CMAC.

10.3.7 CryptoAesEncrypt01

Command `CryptoAesEncrypt01(...)` encrypts the provided data using the specified Key and returns the encrypted data. It can't be used on key indexes 2-11, in order to prevent re-calculating the session keys.

Table 10-14: CryptoAesEncrypt01 Command

Byte	0	1	2	3	4	...	N+2
Data from Host	0x05	0x07	KeyID (7:0)	0x01	Data2	…	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	…	0x00

Table 10-15: CryptoAesEncrypt01 Response

Byte	0	1	2	...	N+1
Data from Host	0x00	0x00	0x00	…	0x00
Data to Host	Stat1	CEStatus	Encrypted Data1	…	Encrypted DataN

- *KeyID*: Specified Key ID, as defined in Table 10-1: Cryptographic Keys Usage and Derivation. Valid values are from12 to 25**.**
- *Data1, Data2 to DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section CEStatus on page 102
- *EncryptedData1, EncryptedData2,... , EncryptedDataN*: Encrypted data, considered as byte buffers

10.3.8 CryptoAesEncrypt

Command `CryptoAesEncrypt(...)` encrypts the provided data using the specified Key and returns it. It is to be used for generic, non-LoRaWAN cryptographic operations, where the Crypto Engine is used as a hardware accelerator, on key indexes 26 and 27 only (General Purpose keys).

Table 10-16: CryptoAesEncrypt Command

Byte	0	1	2	3	...	N+2
Data from Host	0x05	0x08	KeyID (7:0)	Data1	…	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	…	0x00

Table 10-17: CryptoAesEncrypt Response

Byte	0	1	2	...	N+1
Data from Host	0x00	0x00	0x00	…	0x00
Data to Host	Stat1	CEStatus	Encrypted Data1	…	Encrypted DataN

- *KeyID*: Specified Key ID, as defined in Table 10-1: Cryptographic Keys Usage and Derivation. Valid values are from 26 to 27.
- *Data1, Data2 to DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section CEStatus on page 102
- *EncryptedData1, EncryptedData2 to EncryptedDataN*: Encrypted data, considered as byte buffers

10.3.9 CryptoAesDecrypt

Command `CryptoAesDecrypt(...)` decrypts the provided data using the specified Key and returns it, and can only be used on keys indexes 26-27, when the Crypto Engine is used as a stand-alone hardware accelerator for non-LoRaWAN security tasks.

Table 10-18: CryptoAesDecrypt Command

Byte	0	1	2	3	...	N+2
Data from Host	0x05	0x09	KeyID (7:0)	Data1	…	DataN
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	…	0x00

Table 10-19: CryptoAesDecrypt Response

Byte	0	1	2	...	N+1
Data from Host	0x00	0x00	0x00	…	0x00
Data to Host	Stat1	CEStatus	Decrypted Data1	…	Decrypted DataN

- *KeyID*: Specified Key ID, as defined in Table 10-1: Cryptographic Keys Usage and Derivation. Valid values are from 0 to 27.
- *Data1, Data2 to DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section CEStatus on page 102
- *DecryptedData1, DecryptedData2 to DecryptedDataN*: Decrypted data, considered as byte buffers

10.3.10 CryptoStoreToFlash

Command `CryptoStoreToFlash(...)` makes the Crypto Engine store the data (Keys and Parameters) from RAM into flash memory.

Table 10-20: CryptoStoreToFlash Command

Byte	0	1
Data from Host	0x05	0x0A
Data to Host	Stat1	Stat2

Table 10-21: CryptoAesDecrypt Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *CEStatus*: Defined in section CEStatus on page 102

10.3.11 CryptoRestoreFromFlash

Command `CryptoRestoreFromFlash(...)` makes the Crypto Engine restore the data (Keys and Parameters) from flash memory into RAM.

Table 10-22: CryptoRestoreFromFlash Command

Byte	0	1
Data from Host	0x05	0x0B
Data to Host	Stat1	Stat2

Table 10-23: CryptoRestoreFromFlash Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *CEStatus*: Defined in section CEStatus on page 102

10.3.12 CryptoSetParam

Command `CryptoSetParam()` sets a specific Parameter into the Crypto Engine RAM.

Table 10-24: CryptoSetParam Command

Byte	0	1	2	3	4	5	6
Data from Host	0x05	0x0D	ParamID(7:0)	Data(31:24)	Data(23:16)	Data(15:8)	Data(7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00

Table 10-25: CryptoSetParam Response

Byte	0	1
Data from Host	0x00	0x00
Data to Host	Stat1	CEStatus

- *ParamID*: Parameter ID, 0 to 119
- *Data*: Parameter Data
- *CEStatus*: Defined in CEStatus on page 102

10.3.13 CryptoGetParam

Command `CryptoGetParam()` gets a specific Parameter from the Crypto Engine RAM.

Table 10-26: CryptoGetParam Command

Byte	0	1	2
Data from Host	0x05	0x0E	ParamID(7:0)
Data to Host	Stat1	Stat2	0x00

Table 10-27: CryptoGetParam Response

Byte	0	1	2	3	4	
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	CEStatus	Data(31:24)	Data(23:16)	Data(15:8)	Data(7:0)

- *ParamID*: Parameter ID, values 0 to 119
- *Data*: Parameter Data
- *CEStatus*: Defined in CEStatus on page 102

11. LR1121 Provisioning

11.1 Description

The LR1121 is pre-provisioned during Semtech's production test flow with two identifiers, which can be used to identify devices on LoRaWAN® networks. For more information, please refer to the LoRa Alliance® website: https://LoRa-alliance.org/

- The ChipEui number is globally unique, and identifies the device.
- The SemtechJoinEui is re-used on a set of Semtech devices.
- A unique Device Key, DKEY, is also pre-provisioned.
- With these numbers, a Device PIN is calculated upon execution of the `DeriveRootKeysAndGetPin(...)` command. This PIN is necessary to claim the device on the LoRa Cloud™ JoinServer services. For more information, please refer to the LoRa Cloud website: https://www.LoRacloud.com/portal/join_service.

All these unique identifiers are stored in the device's persistent memory. They are pre-configured by Semtech to ease the LoRaWAN implementation and access LoRa Cloud Join Server services, but can also be ignored by the user.

11.2 Provisioning Commands

11.2.1 GetChipEui

Command `GetChipEui(...)` reads the LR1121's pre-provisioned *ChipEui*. It is a globally-unique number, assigned by Semtech in production, using one of Semtech's IEEE assigned EUIs. In the standard use-case, the *ChipEui* is used to derive the two LoRaWAN root keys (*AppKey*, *NwkKey*), and should also be used as *DevEui* (in the LoRaWAN definition) to generate the Join Request (which itself transports the DevEui).

Table 11-1: GetChipEui Command

Byte	0	1
Data from Host	0x01	0x25
Data to Host	Stat1	Stat2

Table 11-2: GetChipEui Response

Byte	0	1	...	8
Data from Host	0x00	0x00	...	0x00
Data to Host	Stat1	ChipEui(63:56)	...	ChipEui(7:0)

- *ChipEui* is coded on 8 bytes, in big endian format.

11.2.2 GetSemtechJoinEui

Command `GetSemtechJoinEui(...)` reads LR1121's pre-programmed JoinEui, installed in production by Semtech.

The two LoRaWAN root keys (*AppKey*, *NwkKey*) in the device are derived from this JoinEui, amongst other numbers.

On top, in the standard use-case, the user should use SemtechJoinEui as the LoRaWAN JoinEui field to build the Join Request frame (which itself transports the JoinEui).

Table 11-3: GetSemtechJoinEui Command

Byte	0	1
Data from Host	0x01	0x26
Data to Host	Stat1	Stat2

Table 11-4: GetSemtechJoinEui Response

Byte	0	1	...	8
Data from Host	0x00	0x00	...	0x00
Data to Host	Stat1	SemtechJoinEui(63:56)	...	SemtechJoinEui(7:0)

*SemtechJoinE*ui is coded on 8 bytes, in big endian format.

11.2.3 DeriveRootKeysAndGetPin

Command `DeriveRootKeysAndGetPin(...)` derives the *AppKey* and *NwkKey* root keys, and calculates the corresponding PIN, required to claim a device on the Semtech Join Server. It is a very versatile function with a standard use and a more advanced use, as described in the coming sections:

Three use-cases are possible:

- Standard: Uses the pre-provisioned ChipEui, DevEui and use the Semtech Join Server.
- Advanced: The DevEui and/or the JoinEui can be personalized, whilst still using the Join Server.
- Alternate: AppKey and NwkKey are forced by the user, and the Join Server can't be used.

11.2.3.1 Standard Use

Table 11-5: DeriveRootKeysAndGetPin Command (Standard)

Byte	0	1
Data from Host	0x01	0x27
Data to Host	Stat1	Stat2

Table 11-6: DeriveRootKeysAndGetPin Response

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	PIN(31:24)	PIN(23:16)	PIN(15:8)	PIN(7:0)

PIN: Coded on 4 bytes, in big endian format

In the standard use-case, the ChipEui is used as LoRaWAN DevEui, SemtechJoinEui is used as LoRaWAN JoinEui, and therefore no specific action should be taken. The host should:

1. Call the `DeriveRootKeysAndGetPin()` command with no argument.
2. Read the SemtechJoinEui (`GetSemtechJoinEui()` command) and assign it to a JoinEui variable.
3. Read the ChipEui (`GetChipEui()` command) and assign it to a DevEui variable.
4. Execute the Join procedure using the elements that have just been read.
5. At the reception of a valid Join Answer, all lifetime and session keys can be derived according to the required LoRaWAN Standard.

The device must also be claimed on the Semtech Join Server for the Join Request to be accepted.

11.2.3.2 Advanced Use

The LR1121 supports a user-defined DevEui and / or a dedicated JoinEui, both different from the ChipEui and SemtechJoinEui onboard the LR1121. If the derivation scheme defined by the Semtech Join Server service is re-used, the function call to `DeriveRootKeysAndGetPin()` should be done as follows.

Table 11-7: DeriveRootKeysAndGetPin Command (advanced)

Byte	0	1	2	3	4	5	6:9	10:17	18
Data from Host	0x01	0x27	DevEui (63:0)					JoinEui	RFU (0x00)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

Table 11-8: DeriveRootKeysAndGetPin Response (advanced)

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	PIN(31:24)	PIN(23:16)	PIN(15:8)	PIN(7:0)

PIN: Coded on 4 Bytes, in big endian format.

In the advanced use-case, a user-specific DevEui and / or JoinEui can be used. The host should:

1. Call the `DeriveRootKeysAndGetPin()` command with your own DevEui and/or JoinEui, get back the PIN.
2. Execute the Join procedure using DevEui and JoinEui (i.e., not ChipEui and SemtechJoinEui).
3. At the reception of a valid Join Answer, all lifetime and session keys can be derived according to the required LoRaWAN Standard.

The device must also be claimed for the Semtech Join Server for the Join Request to be accepted.

11.2.3.3 Alternate Use

The LR1121 can be used outside of the Join Server service and its built-in root key derivation schemes. In this scenario, the PIN concept becomes irrelevant, and the user can provision his own *NwkKey* and *AppKey*, but still leverage the Crypto Engine for any of the authentication, signature and encryption tasks, for improved security. Note that, in this scenario, the `DeriveRootKeysAndGetPin(...)` command MUST NOT be used, as it would overwrite the root keys personalized in the device by the user with the `CryptoSetKey(...)` command. The host should:

1. Use `CryptoSetKey(...)` to personalize NwkKey and AppKey (according to the LoRaWAN version of interest).
2. Execute the Join procedure using DevEui and JoinEui of the user's choosing (ChipEui and SemtechJoinEui may be re-used for that matter).
3. Get the *DevAddr* and derive the session keys with the acceptance of the Join Response.

11.3 Crypto Engine Use With LoRaWAN V1.1.x

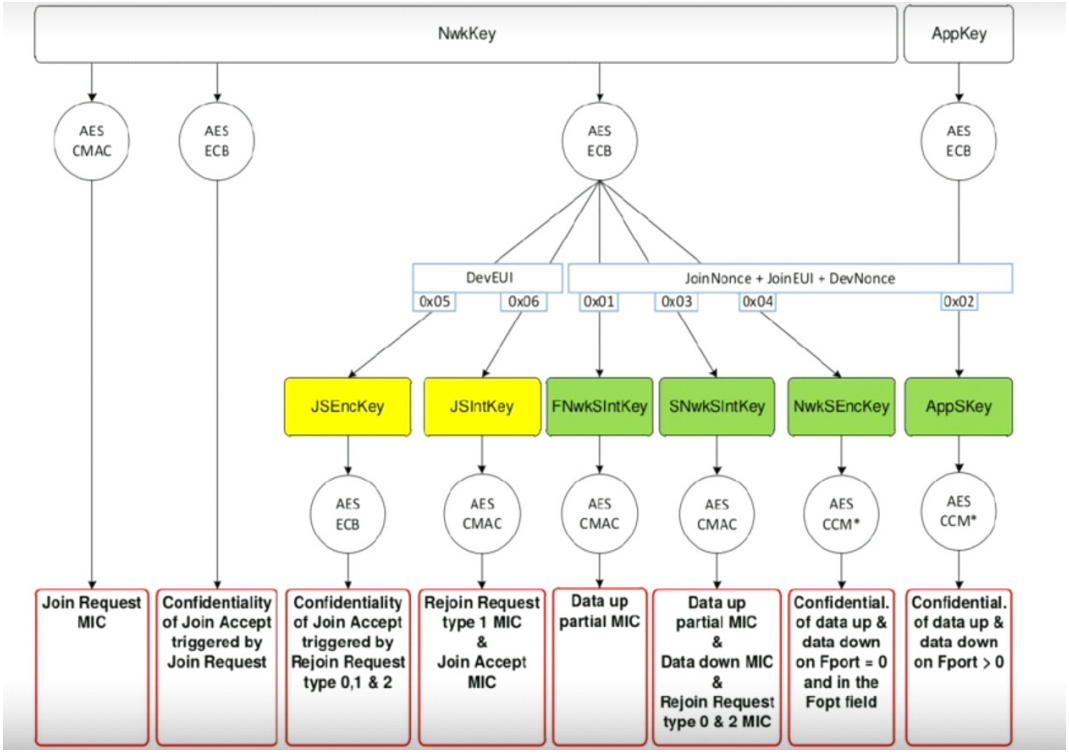

Figure 11-1: Key Derivation Scheme For LoRaWAN 1.1.x

All the keys required to sign, authenticate, or encrypt Join traffic, as well as the session keys, can be derived from the provided root keys (*AppKey*, *NwkKey*), following the derivation scheme described in Figure 11-1. The arguments of the `CryptoDeriveKey(...)` function should be computed by the host controller, and supplied to the function as its *Input[1:16]* argument, with appropriate padding bytes when applicable. For instance, when computing FNwkSIntKey, the command should be used as follows:

`FNwkSIntKey = CryptoDeriveKey(NwkKey, 0x01 | JoinNonce | NetID | DevNonce | pad`$_{16}$`)`

Where pad_{16} is the required number of 0x00 bytes to expand to a 16-byte number.

11.4 Crypto Engine Use with LoRaWAN V1.0.x

The key derivation scheme is available from the LoRaWAN specification:

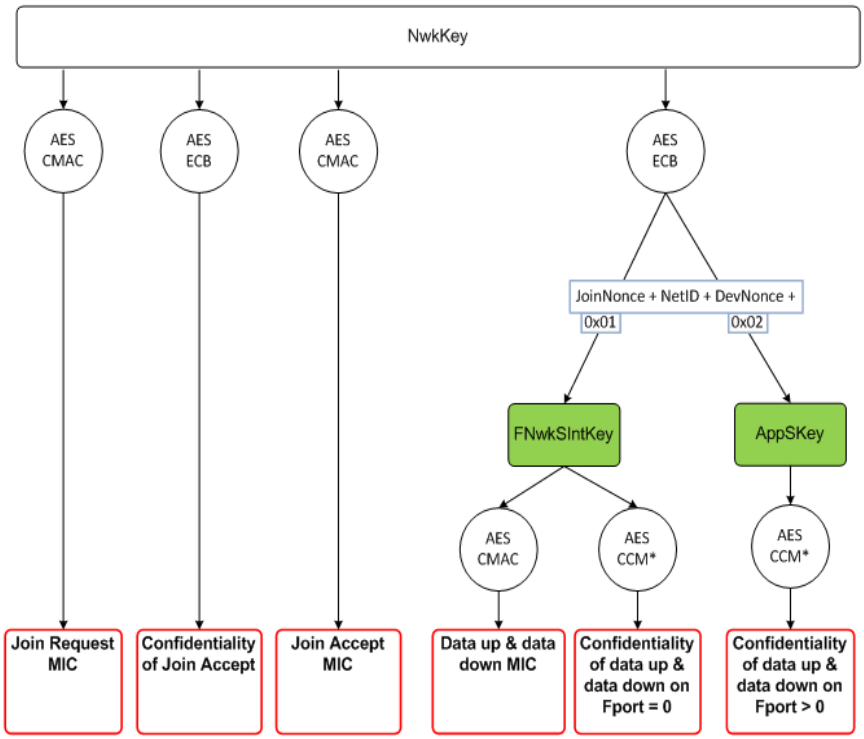

Figure 11-2: Key Derivation Scheme for LoRaWAN 1.0.x

Although the crypto engine of the LR1121 follows the key naming convention of the LoRaWAN 1.1.x Standard, it can be used for LoRaWAN 1.0.x. The correspondence table is delivered below:

Table 11-9: LoRaWAN 1.0.x vs. 1.1.x Security Correspondence Table

1.0.x	1.1.x
LoRaWAN_DEVICE_EUI	LoRaWAN_DEVICE_EUI
LoRaWAN_APP_EUI	LoRaWAN_JOIN_EUI
LoRaWAN_GEN_APP_KEY[1]	LoRaWAN_APP_KEY[1]
LoRaWAN_APP_KEY	LoRaWAN_NWK_KEY
LoRaWAN_NWK_S_KEY	LoRaWAN_F_NWK_S_INT_KEY
LoRaWAN_NWK_S_KEY	LoRaWAN_S_NWK_S_INT_KEY
LoRaWAN_NWK_S_KEY	LoRaWAN_NWK_S_ENC_KEY
LoRaWAN_APP_S_KEY	LoRaWAN_APP_S_KEY

1. The *GenAppKey* can be used as mother key to derive multicast-specific key material in LoRaWAN 1.0.x, whereas *AppKey* is used in LoRaWAN 1.1.x

12. Test Commands

Several LR1121 test commands allow an easy configuration of the device for regulatory ETSI or FCC compliance.

12.1 Regulatory Overview

This section only describes the RF modes necessary for ETSI and FCC regulatory testing. Please refer to the ETSI and FCC documents for a detailed test description and for the test limits indication.

12.1.1 ETSI

The EN 300 220 standards describe 4 test signals which the EUT (Equipment Under Test) must be able to transmit for CE certification. These test signals are listed in the table hereafter, with the operating mode correspondence for the LR1121.

Table 12-1: ETSI Test Signals

Test Signal	Description	LR1121 Operation
D-M1	Unmodulated carrier	TX CW mode (`SetTxCw(...)` command)
D-M2	Continuously modulated signal with the greatest occupied RF bandwidth	Continuous modulation (`SetTxInfinitePreamble(...)` command)
D-M2a	Same as D-M2 signal, but not continuous	RF packet transmission: LoRa SF12, BW500, 50% duty cycle
D-M3	Normal operating mode of the EUT in the application	Operation as in the application

The user should be able to modify the operating frequency, output power, and modulation parameters for the ETSI tests. The user should also be able to receive the incoming RF packets for any configuration (frequency, modulation parameters), and to determine a PER (Packet Error Rate) indication of the receive quality.

All this can be done using the regular LR1121 radio commands.

12.1.2 FCC

The FCC part 15.247 is applicable to frequency hopping and digitally modulated systems. For those tests, only an unmodulated carrier (TX CW) and a regular packet transmission are required.

The user should be able to modify the operating frequency, output power, and modulation parameters for the FCC tests. This can be done using the regular LR1121 radio commands.

12.2 Commands

12.2.1 SetTxCw

Command `SetTxCw (...)` sets the device in TX continuous wave mode (unmodulated carrier).

Table 12-2: SetTxCw Command

Byte	0	1
Data from Host	0x02	0x19
Data to Host	Stat1	Stat2

This command immediately sets the device in TX CW mode. Therefore, the operating frequency and the PA configuration commands (including the RF output power) have to be called prior to this command.

12.2.2 SetTxInfinitePreamble

Command `SetTxInfinitePreamble(...)` transmits an infinite preamble sequence.

Table 12-3: SetTxInfinitePreamble Command

Byte	0	1
Data from Host	0x02	0x1A
Data to Host	Stat1	Stat2

This command immediately starts transmission of the infinite preamble sequence. Therefore, the operating frequency, the PA configuration commands (including the RF output power) and the packet type have to be called prior to this command.

13. List Of Commands

For a general description of SPI Read and Write commands processing, see sections 3.2 Read Commands and 3.1 Write Commands.

13.1 Register / Memory Access Operations

Table 13-1: Register / Memory Access Operations

Command	Opcode	Parameters	Description
WriteRegMem32	0x0105	Addr(4), Data(256)	Writes data at given register/memory address. Address must be 32-bit aligned and data length must be a multiple of 4
ReadRegMem32	0x0106	Addr(4), Len(1)	Reads data at given register/memory address. Address must be 32-bit aligned and data length < 65
WriteBuffer8	0x0109	Data(255)	Writes data to radio TX buffer (up to 255 bytes)
ReadBuffer8	0x010A	Offset(1), Len(1)	Reads data from radio RX buffer
ClearRxBuffer	0x010B	---	Clears all data from radio RX buffer
WriteRegMemMask32	0x010C	Addr(4), Mask(4), Data(4)	Reads-Modifies-Writes data at given register/memory address. Address must be 32-bit aligned.

13.2 System Configuration / Status Operations

Table 13-2: System Configuration / Status Operations (Sheet 1 of 2)

Command	Opcode	Parameters	Description
GetStatus	0x0100	---	Returns status of device
GetVersion	0x0101	---	Returns version of firmware
GetErrors	0x010D	---	Returns error status of device
ClearErrors	0x010E	---	Clears error bits in error status
Calibrate	0x010F	CalibParams(1)	Calibrates requested blocks according to parameter
SetRegMode	0x0110	RegMode(1)	Sets if DC-DC may be enabled for XOSC, FS, RX or TX mode, 0: LDO, 1: DC-DC
CalibImage	0x0111	Freq1(1) Freq2(1)	Launches an image calibration: Freq1, Freq2, in 4MHz steps
SetDioAsRfSwitch	0x0112	RfswEnable(1), RfswStbyCfg(1), RfswRxCfg(1), RfswTxCfg(1), RfswTxHPCfg(1), RfSwTxHfCfg(1), RFU(2)	Sets up RFSWx output configurations for each radio mode
SetDioIrqParams	0x0113	Irq1ToEn(4), Irq2ToEn2(4)	Configures IRQs to output on IRQ pin(s)
ClearIrq	0x0114	IrqToClear(4)	Clears pending IRQs
ConfigLfClock	0x0116	LfClockSetup(1)	Configures the used LF clock
SetTcxoMode	0x0117	RegTcxoTune(1) Delay(3)	Configures device for a connected TCXO
Reboot	0x0118	StayInBootloader(1)	Reboots (SW reset) the device
GetVbat	0x0119	---	Gets VBAT voltage
GetTemp	0x011A	---	Gets temperature
SetSleep	0x011B	SleepConfig(1), SleepTime(4)	Sets chip into SLEEP mode
SetStandby	0x011C	StdbyConfig(1)	0: RC, 1: XOSC Sets chip into RC or XOSC mode
SetFs	0x011D	---	Sets chip into FS mode
GetRandomNumber	0x0120	---	Gets a 32-bit random number
EraseInfoPage	0x0121	InfoPage(1)	Erases the Info Page data
WriteInfoPage	0x0122	InfoPage(1), IndoWordAddr(2), dataN(4)	Writes a 32-bit block of the Info Page
ReadInfoPage	0x0123	dataN(1)	Reads a 32-bit block of the Info Page
GetChipEui	0x0125	---	Returns the 8-byte factory DeviceEui
GetSemtechJoinEui	0x0126	---	Returns the 8-byte factory JoinEui
DeriveRootKeysAndGetPin	0x0127	---	Returns the 4-byte PIN which can be used to register the device with LoRa Cloud Services

Table 13-2: System Configuration / Status Operations (Sheet 2 of 2)

Command	Opcode	Parameters	Description
EnableSpiCrc	0x0128	Enable(1), CRC(1)	Enables / disables 8-bit CRC on SPI interface
DriveDiosInSleepMode	0x012A	Enable(1)	Enables / disables pull-up/down on configured RF switch and IRQ DIOs for sleep modes

13.3 Radio Configuration / Status Operations

Table 13-3: Radio Configuration / Status Operations (Sheet 1 of 2)

Command	Opcode	Parameters	Description
ResetStats	0x0200	---	Resets RX statistics
GetStats	0x0201	---	Gets RX statistics
GetPacketType	0x0202	---	Gets current radio protocol
GetRxBufferStatus	0x0203	---	Gets RS buffer status
GetPacketStatus	0x0204	---	Gets RX packet status
GetRssiInst	0x0205	---	Gets instantaneous RSSI
SetGfskSyncWord	0x0206	Syncword(8)	Set the 64-bit (G)FSK syncword
SetLoRaPublicNetwork	0x0208	PublicNetwork(1)	Changes LoRa sync word for private or public network
SetRx	0x0209	RxTimeout(3)	Set chip into RX mode
SetTx	0x020A	TxTimeout(3)	Set chip into TX mode
SetRfFrequency	0x020B	RfFreq(4)	Set PLL frequency
AutoTxRx	0x020C	Delay(3) IntermediaryMode(1) Timeout2(3)	Activate or deactivates the auto TX auto RX mode
SetCadParams	0x020D	SymbolNum(1) DetPeak(1) DetMin(1) CadExitMode(1) Timeout(3)	Configure LoRa CAD mode
SetPacketType	0x020E	PacketType(1)	Define radio protocol ((G)FSK, LoRa)
SetModulationParams	0x020F	Params(1)	Configure modulation parameters[1]
SetPacketParams	0x0210	Params(1)	Configure packet parameters[1]
SetTxParams	0x0211	TxPower(1) RampTime(1)	Set TX power and ramp time

Table 13-3: Radio Configuration / Status Operations (Sheet 2 of 2)

Command	Opcode	Parameters	Description
SetPacketAdrs	0x0212	NodeAddr(1) BroadcastAddr(1)	Set the Node address and the broadcast address for the (G)FSK packets
SetRxTxFallbackMode	0x0213	Fall-back mode(1)	Defines into which mode the chip goes after a TX / RX done
SetRxDutyCycle	0x0214	RxPeriod(3) SleepPeriod(3) Mode(1)	Start RX Duty Cycle mode
SetPaConfig	0x0215	PaSel(1) RegPaSupply(1) PaDutyCycle(1) PaHpSel(1)	Configures PA settings
StopTimeoutOnPreamble	0x0217	StopOnPreamble(1)	Stops RX time-out on 0: Syncword/Header (default) or 1: preamble detection
SetCad	0x0218	---	Sets chip into RX CAD mode (LoRa)
SetTxCw	0x0219	---	Sets chip into TX mode with infinite carrier wave
SetTxInfinitePreamble	0x021A	---	Sets chip into TX mode with infinite preamble
SetLoRaSynchTimeout	0x021B	SymbolNum(1)	Configures LoRa modem to issue a time-out after exactly SymbolNum symbols
SetGfskCrcParams	0x0224	InitValue(4), Poly(4)	Sets parameters for CRC polynomial
SetGfskWhitParams	0x0225	Seed(2)	Sets parameters for whitening
SetRxBoosted	0x0227	RxBoosted(1)	Sets RX to boosted mode
SetRssiCalibration	0x0229	TuneGxx(4), TuneG13hpx(3), TuneGxx(1), TuneG13hp7(1), GainOffset(2)	Sets gain offset for on-chip power estimation
SetLoRaSyncWord	0x022B	Syncword(1)	Sets LoRaSyncWord
LrFhssBuildFrame	0x022C	HeaderCount(1), CR(1), ModType(1), Grid(1), Hopping(1), BW(1), HopSequence(2), DeviceOffset(1), Payload(n)	Encodes LR-FHSS frame, sets up hopping table
LrFhssSetSyncWord	0x022D	Syncword(4)	Sets LR-FHSS Syncword
GetLoRaRxHeaderInfos	0x0230		Gets last packet header info

1. Parameters are packet dependant.

13.4 CryptoElement Configuration / Status Operations

Table 13-4: CryptoElement Configuration / Status Operations

Command	Opcode	Parameters	Description
CryptoSetKey	0x0502	KeyID(1) Key(2)	
CryptoDeriveKey	0x0503	SrcKeyID(1) DstKeyID(1) Input(2)	Derives and stores a key
CryptoProcessJoinAccept	0x0504	DecKeyID(1), VerKeyID(1) LoRaWANVer(1), Header(1 or 12) Data(16 or 32)	Processes a join accept message: decrypts full message (data+header) verifies MIC on the message, and if OK, provides decrypted message.
CryptoComputeAesCmac	0x0505	KeyID(1) Data(256)	Computes CMAC, returns MIC using specified Key.
CryptoVerifyAesCmac	0x0506	KeyID(1) ExpectedMIC(4) Data(256)	Verifies computed CMAC (compare calculated MIC with expected MIC)
CryptoAesEncrypt01	0x0507	KeyID(1) Data(256)	Encrypts data using specified Key
CryptoAesEncrypt	0x0508	KeyID(1) Data(256)	Encrypts data using specified Key
CryptoAesDecrypt	0x0509	KeyID(1) Data(256)	Decrypts data using specified Key
CryptoStoreToFlash	0x050A	---	Stores all Keys (and Parameters) to flash
CryptoRestoreFromFlash	0x050B	---	Restores all Keys (and Parameters) from flash
CryptoSetParam	0x050D	ParamID(1), Data(4)	Sets a parameter in RAM
CryptoGetParam	0x050E	ParamID(1)	Gets a parameter from RAM

13.5 Bootloader Commands

Table 13-5: Bootloader Commands[1]

Command	Opcode	Parameters	Description
EraseFlash	0x8000	---	Erases content of program memory
WriteFlashEncrypted	0x8003	Offset(4), Data(128)	Writes a new encrypted firmware image
Reboot	0x8005	StayInBootloader(1)	Reboots (SW reset) device from bootloader mode
GetPIN	0x800B	-	Reads the 4-byte PIN of the device (little endian)
GetChipEui	0x800C	-	Reads the 8-byte factory ChipEui of the device (big endian)
GetJoinEui	0x800D	-	Reads the 8-byte factory JoinEui of the device (big endian)

1. See AN1200.57 LR1110 Program Memory Update for usage.

14. Revision History

The following table details the versions of the User Manual documentation issued, and the corresponding LR1121 versions supported (Use Case and FW Major.FW Minor), as returned by command `GetVersion(...)`.

Table 14-1: Revision History

User Manual Version	ECO	Date	Applicable to	Changes
1.0	062818	Jul 2022	Use Case: 03 FW Version: 01.00 or later	First Release
1.1	064173	Mar 2023	Use Case: 03 FW Version: 01.00 or later	Added S-band information Added references to AN1200.74 LoRa Edge Clock Requirements and AN1200.59 Modified tables 9-1,9-2,9-3 & 9-5 Removed sections 11.3.4.1&2 (LoRa Edge Clock Requirements) Table 2.15 Mode Transitions and Timings
1.2	066719	May 2023	Use Case: 03 FW Version: 01.00 or later	Modified tables 9-1,9-2,9-3 & 9-5 Added return code CMD_PERR for ReadInfoPage command Note in List of Bootloader commands table applied to all table Mode Transitions and Timings value changed for SLEEP to STBY_RC (no data retention) if XOSC = XTAL.

Glossary

List of Acronyms and their Meaning (Sheet 1 of 2)

Acronym	Meaning
ADC	Analog-to-Digital Converter
API	Application Programming Interface
β	Modulation Index
BR	Bit Rate
BT	Bandwidth-Time bit period product
BW	BandWidth
BWF	BandWidth of the (G)FSK modem
BWL	BandWidth of the LoRa® Modem
CAD	Channel Activity Detection
CPHA	Clock Phase
CR	Coding Rate
CRC	Cyclical Redundancy Check
CW	Continuous Wave
DC-DC	Direct Current to Direct Current Converter
DIO	Digital Input / Output
DS	Distribution System
DSB	Double Side Band
DSP	Digital Signal Processing
ECO	Engineering Change Order
Fdev	Frequency Deviation
FIFO	First In First Out
FS	Frequency Synthesis
FSK	Frequency Shift Keying
IF	Intermediate Frequencies
IRQ	Interrupt Request
LDO	Low-Dropout
LDRO	Low Data Rate Optimization
LFSR	Linear-Feedback Shift Register
LNA	Low-Noise Amplifier

List of Acronyms and their Meaning (Sheet 2 of 2)

Acronym	Meaning
LoRa®	Long Range Communication the LoRa® Mark is a registered trademark of the Semtech Corporation
LR-FHSS	Long Range Frequency Hopping Spread Spectrum
LSB	Least Significant Bit
MIC	Message Integrity Code
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
MSB	Most Significant Bit
MSK	Minimum-Shift Keying
NOP	No Operation (0x00)
NRZ	Non-Return-to-Zero
NSS	Slave Select active low
OCP	Over Current Protection
PA	Power Amplifier
PER	Packet Error Rate
PHY	Physical Layer
PID	Product Identification
PLL	Phase-Locked Loop
POR	Power On Reset
RFO	Radio Frequency Output
RFU	Reserved for Future Use
RTC	Real-Time Clock
SCK	Serial Clock
SF	Spreading Factor
SN	Sequence Number
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
STDBY	Standby
TCXO	Temperature-Compensated Crystal Oscillator
XOSC	Crystal Oscillator

Contact Information

Semtech Corporation
Wireless, Sensing & Timing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone: (805) 498-2111, Fax: (805) 498-3804
www.semtech.com