



LR1110 Transceiver

User Manual

Disclaimers

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.

Semtech's products are designed to be used in connection with qualified Bluetooth® products and applications but are not certified or qualified Bluetooth® products.

Table of Contents

1. Introduction.....	15
1.1 Scope	15
1.2 Overview	15
2. System Processes.....	17
2.1 System Modes	17
2.1.1 Boot	18
2.1.2 Standby.....	18
2.1.3 Calibrations.....	19
2.1.4 Power Down.....	20
2.1.5 Sleep.....	21
2.1.6 Reset	22
2.1.7 GNSS Scanning Mode	23
2.1.8 Wi-Fi Passive Scanning Mode	23
2.1.9 DSP Mode.....	23
2.1.10 RX Mode	23
2.1.11 TX Mode	24
2.1.12 FS Mode.....	25
2.2 Startup Sequence	25
2.3 Firmware Upgrade	26
2.3.1 GetVersion	26
2.3.2 EraseFlash.....	26
2.3.3 WriteFlashEncrypted.....	27
2.4 Mode Transitions & Timings	27
2.5 Info Page	28
2.5.1 EraseInfoPage	28
2.5.2 WriteInfoPage	28
2.5.3 ReadInfoPage.....	29
3. Host-Controller Interface	30
3.1 Write Commands	30
3.2 Read Commands	30
3.3 Command Endianness	31
3.4 Status Registers	32
3.4.1 GetStatus	32
3.4.2 Stat1	32
3.4.3 Stat2	33
3.5 BUSY	34
3.6 Errors	35
3.6.1 GetErrors.....	35
3.6.2 ClearErrors.....	35
3.7 Memory Access	36
3.7.1 WriteRegMem32.....	36
3.7.2 ReadRegMem32.....	36
3.7.3 WriteRegMemMask32	37
3.7.4 WriteBuffer8	37

3.7.5 ReadBuffer8	38
3.7.6 ClearRxBuffer	38
3.7.7 GetRandomNumber	38
3.7.8 EnableSpiCrc	39
4. GPIOs	40
4.1 Interrupts	40
4.1.1 SetDioIrqParams	42
4.1.2 ClearIrq	42
4.2 RF Switch Control	43
4.2.1 SetDioAsRfSwitch	43
4.2.2 DriveDiosInSleepMode	44
4.3 Temperature Sensor	45
4.3.1 GetTemp	45
5. Power Distribution	46
5.1 Power Modes	46
5.1.1 SetRegMode	46
5.2 VBAT Measurement	47
5.2.1 GetVbat	47
5.3 Power-On-Reset and Brown-Out-Reset	48
5.4 Low Battery Detector	48
5.5 Over Current Protection	48
6. Clock Sources	49
6.1 RC Oscillators Clock References	49
6.2 High-Precision Clock References	49
6.2.1 32.768kHz Clock	49
6.2.2 32MHz Crystal	49
6.2.3 32MHz TCXO	50
6.3 Commands	51
6.3.1 ConfigLfClock	51
6.3.2 SetTxcoMode	52
7. Sub-GHz Radio	53
7.1 Overview	53
7.2 Commands	54
7.2.1 SetRfFrequency	54
7.2.2 SetRx	54
7.2.3 SetTx	55
7.2.4 AutoTxRx	56
7.2.5 SetRxTxFallbackMode	57
7.2.6 SetRxDutyCycle	58
7.2.7 StopTimeoutOnPreamble	60
7.2.8 GetRssiInst	60
7.2.9 GetStats	61
7.2.10 ResetStats	61
7.2.11 GetRxBufferStatus	62
7.2.12 SetRxBoosted	62
7.2.13 SetLoRaSyncWord	63

7.2.14 GetLoRaRxHeaderInfos	63
7.2.15 SetRssiCalibration	64
8. Modems	67
8.1 Modem Configuration	67
8.1.1 SetPacketType	68
8.1.2 GetPacketType	68
8.2 LoRa® Modem	69
8.2.1 LoRa Modulation Principle.....	69
8.2.2 LoRa Packet Format.....	70
8.2.3 Channel Activity Detection (CAD)	71
8.3 LoRa Commands	72
8.3.1 SetModulationParams.....	72
8.3.2 SetPacketParams	73
8.3.3 SetCad	73
8.3.4 SetCadParams.....	74
8.3.5 SetLoRaSynchTimeout	74
8.3.6 SetLoRaPublicNetwork.....	75
8.3.7 GetPacketStatus.....	75
8.4 (G)FSK Modem	76
8.4.1 (G)FSK Modulation Principle	76
8.4.2 (G)FSK Packet Engine	76
8.4.3 (G)FSK Packet Format.....	77
8.5 (G)FSK Commands	79
8.5.1 SetModulationParams	79
8.5.2 SetPacketParams	80
8.5.3 SetGfskSyncWord	81
8.5.4 SetPacketAdrs	81
8.5.5 SetGfskCrcParams	82
8.5.6 SetGfskWhitParams	82
8.5.7 GetPacketStatus.....	83
8.6 LR-FHSS Modulation	84
8.6.1 LR-FHSS Modulation Principle	84
8.7 LR-FHSS Commands	85
8.7.1 SetModulationParams	85
8.7.2 LrFhssBuildFrame	85
8.7.3 LrFhssSetSyncWord	87
8.7.4 Circuit Configuration for Long Range FHSS Transmission	87
8.8 Sigfox Modulation	88
8.8.1 SetModulationParams	88
8.8.2 SetPacketParams	88
8.9 Data Buffer	89
8.10 RSSI Functionality	90
9. Power Amplifiers	92
9.1 PA Supply Scheme	93
9.1.1 Low Power PA	94
9.1.2 High Power PA	95

9.2 PA Output Power	96
9.2.1 Low Power PA.....	96
9.2.2 High Power PA.....	97
9.3 PA Current Consumption	98
9.3.1 Low Power PA.....	98
9.3.2 High Power PA.....	99
9.3.3 High Frequency PA	101
9.4 Impedance Matching Networks	102
9.4.1 Multi-Band Operation.....	102
9.4.2 RF Switch Implementation.....	103
9.4.3 Direct-Tie Implementation	104
9.5 Commands	105
9.5.1 SetPaConfig	105
9.5.2 SetTxParams	106
10. Wi-Fi Passive Scanning	107
10.1 Principle of Operation	107
10.1.1 Repetition Schemes	108
10.1.2 Preamble Search Step	108
10.1.3 Signal Capture Step	109
10.1.4 Signal Demodulation Step	109
10.2 Wi-Fi Commands	110
10.2.1 List of Wi-Fi Commands.....	110
10.2.2 WifiScan	111
10.2.3 WifiScanTimeLimit.....	112
10.2.4 WifiCountryCode.....	113
10.2.5 WifiCountryCodeTimeLimit	114
10.2.6 WifiGetNbResults	115
10.2.7 WifiReadResults	116
10.2.8 WifiResetCumulTimings	117
10.2.9 WifiReadCumulTimings	117
10.2.10 WifiGetNbCountryCodeResults.....	118
10.2.11 WifiReadCountryCodeResults.....	118
10.2.12 WifiCfgTimestampAPphone.....	119
10.2.13 WifiReadVersion.....	119
10.3 Wi-Fi Results Formats	120
10.3.1 Wi-Fi Passive Scanning Result Formats.....	120
10.3.2 Basic MAC/Type/Channel Result Format.....	121
10.3.3 Full Result Format	122
10.3.4 Extended Complete Result Format	125
10.3.5 WifiCountryCode Result Format.....	127
11. GNSS Scanning.....	128
11.1 GNSS Geolocation System Overview	128
11.2 GNSS Principle of Operation	129
11.3 GNSS Commands	130
11.3.1 List of GNSS Commands.....	130
11.3.2 GnssSetConstellationToUse	130

11.3.3 GnssReadConstellationToUse	131
11.3.4 GnssReadSupportedConstellations	132
11.3.5 GnssSetMode	132
11.3.6 GnssAutonomous	133
11.3.7 GnssAssisted	134
11.3.8 GnssSetAssistancePosition.....	135
11.3.9 GnssReadAssistancePosition	135
11.3.10 GnssGetContextStatus	136
11.3.11 GnssReadVersion	137
11.3.12 GnssSetAlmanacUpdate	137
11.3.13 GnssReadAlmanacUpdate	138
11.3.14 GnssPushSolverMsg	138
11.3.15 GnssPushDmMsg.....	138
11.4 GNSS Scanning Results & Commands	139
11.4.1 GNSS Scan Result Message Description	139
11.4.2 GnssGetResultSize	140
11.4.3 GnssReadResults	140
11.4.4 GnssGetNbSvDetected	141
11.4.5 GnssGetSvDetected	141
11.4.6 GnssGetConsumption	142
11.4.7 GnssGetSvVisible	143
11.5 GNSS Almanac	144
11.5.1 GnssAlmanacFullUpdate.....	145
12. Sub-GHz Ranging	146
12.1 Sub-GHz Ranging Commands API Interface	146
12.2 SetRangingAddr	146
12.3 SetRangingReqAddr	147
12.4 GetRangingResult	147
12.5 SetRangingTxRxDelay	148
12.6 SetRangingParameter	148
13. Bluetooth® Low Energy Beacons Compatibility	149
13.1 ConfigBleBeacon	149
13.2 BleBeaconSend	149
14. Cryptographic Engine	150
14.1 Description	150
14.2 Cryptographic Keys Definition	150
14.3 Commands	152
14.3.1 CESTatus.....	152
14.3.2 CryptoSetKey.....	152
14.3.3 CryptoDeriveKey	153
14.3.4 CryptoProcessJoinAccept.....	154
14.3.5 CryptoComputeAesCmac	155
14.3.6 CryptoVerifyAesCmac	156
14.3.7 CryptoAesEncrypt01	157
14.3.8 CryptoAesEncrypt.....	158
14.3.9 CryptoAesDecrypt	159

14.3.10 CryptoStoreToFlash.....	160
14.3.11 CryptoRestoreFromFlash.....	160
14.3.12 CryptoSetParam.....	161
14.3.13 CryptoGetParam.....	161
14.3.14 CryptoCheckEncryptedFirmwareImage.....	162
14.3.15 CryptoCheckEncryptedFirmwareImageResult	163
15. LR1110 Provisioning.....	164
15.1 Description	164
15.2 Provisioning Commands	164
15.2.1 GetChipEui.....	164
15.2.2 GetSemtechJoinEui.....	165
15.2.3 DeriveRootKeysAndGetPin	166
15.3 Crypto Engine Use With LoRaWAN V1.1.x	168
15.4 Crypto Engine Use With LoRaWAN V1.0.x	169
16. Test Commands.....	170
16.1 Regulatory Overview	170
16.1.1 ETSI.....	170
16.1.2 FCC	170
16.2 Commands	171
16.2.1 SetTxCw.....	171
16.2.2 SetTxInfinitePreamble.....	171
17. Known Limitations.....	172
18. List Of Commands.....	173
18.1 Register / Memory Access Operations	173
18.2 System Configuration / Status Operations	174
18.3 Radio Configuration / Status Operations	176
18.4 Wi-Fi Configuration / Status Operations	178
18.5 GNSS Configuration / Status Operations	179
18.6 CryptoElement Configuration / Status Operations	180
18.7 Bootloader Commands	181
19. Revision History	182

List of Figures

Figure 1-1: LR1110 Block Diagram	15
Figure 2-1: LR1110 Modes and Transitions.....	17
Figure 2-2: Bootloader	18
Figure 3-1: Write Command Timing Diagram.....	30
Figure 3-2: Read Command Timing Diagram	30
Figure 3-3: GetVersion Write Capture	31
Figure 3-4: GetVersion Read Capture	31
Figure 3-5: BUSY Timing Diagram	34
Figure 5-1: LR1110 POR and BRN Functions.....	48
Figure 6-1: LR1110 Thermal Insulation on PCB Top Layer.....	50
Figure 6-2: TCXO Circuit Diagram	51
Figure 7-1: LR1110 Sub-GHz Radio Block Diagram.....	53
Figure 7-2: LR1110 Current Profile During RX Duty Cycle Operation.....	59
Figure 7-3: RX Duty Cycle Upon Preamble Detection.....	59
Figure 8-1: Modem Command Order.....	67
Figure 8-2: LoRa Signal Bandwidth.....	69
Figure 8-3: LoRa Packet Format	70
Figure 8-4: Fixed-Length Packet	77
Figure 8-5: Variable-Length Packet.....	77
Figure 8-6: LR-FHSS Spectral Plot Example	84
Figure 8-7: RSSI Propagation Time.....	90
Figure 9-1: LR1110 Power Amplifiers Block Diagram.....	92
Figure 9-2: PA Supply Scheme.....	93
Figure 9-3: Low Power PA VR_PA Voltage vs TxPower.....	94
Figure 9-4: High Power PA VR_PA Voltage vs. TxPower.....	95
Figure 9-5: Low Power PA Output Power vs. TxPower	96
Figure 9-6: HP PA Output Power vs. TxPower.....	97
Figure 9-7: IDDTX vs TxPower, Low Power PA, DC-DC Configuration	98
Figure 9-8: IDDTX vs TxPower, Low Power PA, LDO Configuration	99
Figure 9-9: IDDTX vs TxPower, High Power PA, DC-DC Configuration	100
Figure 9-10: IDDTX vs TxPower, High Power PA, LDO Configuration	100
Figure 9-11: IDDTX vs TxPower, High Frequency PA, DC-DC Configuration	101
Figure 9-12: IDDTX vs TxPower, High Frequency PA, LDO Configuration	101
Figure 9-13: RF Switch, Double PA Operation	103
Figure 9-14: RF Switch, Single PA Operation (High Power PA Example).....	103
Figure 9-15: Single Tie Implementation: Only One PA Used (High Power PA Example).....	104
Figure 9-16: Single Tie Implementation: Both PAs Used (High Power PA Example)	104
Figure 10-1: Wi-Fi Passive Scanning Sequence	107
Figure 11-1: GNSS System Overview	128
Figure 11-2: GNSS Dual Constellation Timing	131
Figure 11-3: GNSS Scan Result Message Format	139
Figure 15-1: Key Derivation Scheme for LoRaWAN 1.1.x	168
Figure 15-2: Key Derivation Scheme for LoRaWAN 1.0.x	169

List of Tables

Table 2-1: SetStandby Command.....	18
Table 2-2: CalibImage Command.....	19
Table 2-3: ISM Band Values	19
Table 2-4: Calibrate Command.....	20
Table 2-5: CalibParams Parameter	20
Table 2-6: SetSleep Command	21
Table 2-7: SleepConfig Parameter.....	21
Table 2-8: Sleep Mode Summary.....	21
Table 2-9: Reboot Command	22
Table 2-10: SetFsCommand.....	25
Table 2-11: GetVersion Command	26
Table 2-12: GetVersion Response	26
Table 2-13: EraseFlash Command	26
Table 2-14: WriteFlashEncrypted Command	27
Table 2-15: Mode Transitions and Timings.....	27
Table 2-16: EraseInfoPage Command.....	28
Table 2-17: WriteInfoPage Command	28
Table 2-18: ReadInfoPage Command	29
Table 2-19: ReadInfoPage Response	29
Table 3-1: GetStatus Command	32
Table 3-2: Stat1 Values	32
Table 3-3: Stat2 Values	33
Table 3-4: GetErrors Command	35
Table 3-5: GetErrors Response	35
Table 3-6: ClearErrors Command.....	35
Table 3-7: WriteRegMem32 Command.....	36
Table 3-8: ReadRegMem32 Command.....	36
Table 3-9: ReadRegMem32 Response	36
Table 3-10: WriteRegMemMask32 Command	37
Table 3-11: WriteBuffer8 Command	37
Table 3-12: ReadBuffer8 Command.....	38
Table 3-13: ReadBuffer8 Response	38
Table 3-14: ClearRxBuffer Command	38
Table 3-15: GetRandomNumber Command	38
Table 3-16: GetRandomNumber Response	38
Table 3-17: EnableSpiCrc Command.....	39
Table 4-1: Digital I/Os.....	40
Table 4-2: IrqToEnable Interruption Mapping	41
Table 4-3: SetDioIrqParams Command	42
Table 4-4: ClearIrq Command	42
Table 4-5: SetDioAsRfSwitch Command	43
Table 4-6: DriveDiosInSleepMode Command	44
Table 4-7: GetTemp Command	45
Table 4-8: GetTemp Response	45
Table 5-1: SetRegMode Command	46
Table 5-2: Power Regulation Options	46
Table 5-3: GetVbat Command	47
Table 5-4: GetVbat Response	47
Table 6-1: ConfigLfClock Command.....	51

Table 6-2: SetTcxoMode Command	52
Table 6-3: TCXO Supply Voltage Programming Values	52
Table 7-1: SetRfFrequency Command	54
Table 7-2: SetRx Command	54
Table 7-3: SetTx Command.....	55
Table 7-4: AutoTxRx Command	56
Table 7-5: SetRtTxFallbackMode Command.....	57
Table 7-6: SetRxDutyCycle Command.....	58
Table 7-7: StopTimeoutOnPreamble Command	60
Table 7-8: GetRssiInst Command.....	60
Table 7-9: GetRssiInst Response.....	60
Table 7-10: GetStats Command	61
Table 7-11: GetStats Response	61
Table 7-12: ResetStats Command	61
Table 7-13: GetRxBufferStatus Command.....	62
Table 7-14: GetRxBufferStatus Response.....	62
Table 7-15: SetRxBoosted Command	62
Table 7-16: SetLoRaSyncWord Command	63
Table 7-17: GetLoRaRxHeaderInfos Command	63
Table 7-18: GetLoRaRxHeaderInfos Response.....	63
Table 7-19: SetRssiCalibration Command	64
Table 7-20: Gain Tune Values.....	64
Table 7-21: Recommended Values for Reference EVK.....	64
Table 8-1: SetPacketType Command	68
Table 8-2: GetPacketType Command	68
Table 8-3: GetPacketType Response	68
Table 8-4: SetModulationParams Command	72
Table 8-5: SetPacketParams Command	73
Table 8-6: SetCad Command	73
Table 8-7: SetCadParams Command.....	74
Table 8-8: CadExitMode Parameter	74
Table 8-9: SetLoRaSynchTimeout Command	74
Table 8-10: SetLoRaPublicNetwork Command	75
Table 8-11: GetPacketStatus Command	75
Table 8-12: GetPacketStatus Response	75
Table 8-13: SetModulationParams Command.....	79
Table 8-14: Bandwidth Parameter.....	79
Table 8-15: SetPacketParams Command.....	80
Table 8-16: SetGfskSyncWord Command.....	81
Table 8-17: SetPacketAdrs Command	81
Table 8-18: SetGfskCrcParams Command.....	82
Table 8-19: SetGfskWhitParams Command	82
Table 8-20: GetPacketStatus Command	83
Table 8-21: GetPacketStatus Response	83
Table 8-22: SetModulationParams Command.....	85
Table 8-23: LrFhssBuildFrame Command	85
Table 8-24: Maximum user Payload Length (in Bytes).....	86
Table 8-25: LrFhssSetSyncWord Command	87
Table 8-26: SetModulationParams Command.....	88
Table 8-27: SetPacketParams Command.....	88
Table 8-28: RSSI Information Origin and Meaning.....	90
Table 8-29: Twait & Trssi Values for GFSK Bandwidths	91
Table 9-1: Optimized Settings for LP PA with Same Matching Network.....	102
Table 9-2: Optimized Settings for HP PA with Same Matching Network.....	102

Table 9-3: Optimized Settings for HF PA with the Same Matching Network	102
Table 9-4: SetPaConfig Command	105
Table 9-5: DutyCycle Parameter Allowed Ranges	105
Table 9-6: SetTxParams Command	106
Table 9-7: RampTime Values	106
Table 10-1: Summary Of Available Wi-Fi Commands	110
Table 10-2: WifiScan Command	111
Table 10-3: WifiScanTimeLimit Command	112
Table 10-4: WifiCountryCode Command	113
Table 10-5: WifiCountryCode Example	113
Table 10-6: WifiCountryCodeTimeLimit Command	114
Table 10-7: WiFiGetNbResults Command	115
Table 10-8: WiFiGetNbResults Response	115
Table 10-9: WiFiReadResults Command	116
Table 10-10: WiFiReadResults Response	116
Table 10-11: Example to Read Basic Results of Passive Scan	116
Table 10-12: WiFiResetCumulTimings Command	117
Table 10-13: Wi-Fi Cumulative Timings Description	117
Table 10-14: WiFiReadCumulTimings Command	117
Table 10-15: WiFiReadCumulTimings Response	117
Table 10-16: WiFiGetNbCountryCodeResults Command	118
Table 10-17: WiFiGetNbCountryCodeResults Response	118
Table 10-18: WiFiReadCountryCodeResults Command	118
Table 10-19: WiFiReadCountryCodeResults Response	118
Table 10-20: WiFiCfgTimestampAPhone Command	119
Table 10-21: WiFiReadVersion Command	119
Table 10-22: WiFiReadVersion Response	119
Table 10-23: Wi-Fi Result Formats and Wi-Fi Scan Mode Relationship	120
Table 10-24: Basic Results Format per MAC Address	121
Table 10-25: Basic Complete Results Format per MAC Address	122
Table 10-26: Wi-Fi DataRateID Field	123
Table 10-27: Wi-Fi ChannelID Field	123
Table 10-28: Wi-Fi MacOrigin Field	123
Table 10-30: FrameCtl SubType Values	124
Table 10-29: Wi-Fi FrameCtl Field	124
Table 10-31: Extended Basic Complete Results Format per MAC Address	125
Table 10-32: WiFiCountryCode Result Format Sent Over SPI (12 bytes)	127
Table 11-1: Summary of Available GNSS Commands	130
Table 11-2: GnssSetConstellationToUse Command	130
Table 11-3: GnssReadConstellationToUse Command	131
Table 11-4: GnssReadConstellationToUse Response	131
Table 11-5: GnssReadSupportedConstellations Command	132
Table 11-6: GnssReadSupportedConstellations Response	132
Table 11-7: GnssSetMode Command	132
Table 11-8: GnssAutonomous Command	133
Table 11-9: GnssAssisted Command	134
Table 11-10: GnssSetAssistancePosition Command	135
Table 11-11: GnssReadAssistancePosition Command	135
Table 11-12: GnssReadAssistancePosition Response	135
Table 11-13: GnssGetContextStatus Command	136
Table 11-14: GnssGetContextStatus Response	136
Table 11-15: GnssReadVersion Command	137
Table 11-16: GnssReadVersion Response	137
Table 11-17: GnssSetAlmanacUpdate Command	137

Table 11-18: GnssReadAlmanacUpdate Command	138
Table 11-19: GnssReadAlmanacUpdate Response	138
Table 11-20: GnssPushSolverMsg Command	138
Table 11-21: GnssPushDmMsg Command.....	138
Table 11-22: GnssGetResultSize Command.....	140
Table 11-23: GnssGetResultSize Response.....	140
Table 11-24: GnssReadResults Command.....	140
Table 11-25: GnssReadResults Response.....	140
Table 11-26: GnssGetNbSvDetected Command.....	141
Table 11-27: GnssGetNbSvDetected Response.....	141
Table 11-28: GnssGetSvDetected Command.....	141
Table 11-29: GnssGetSvDetected Response.....	141
Table 11-30: GnssGetConsumption Command	142
Table 11-31: GnssGetConsumption Response	142
Table 11-32: GnssGetSvVisible Command	143
Table 11-33: GnssGetSvVisible Response	143
Table 11-34: GnssAlmanacFullUpdate Command	145
Table 11-35: AlmanacFullUpdatePayload Parameter Format.....	145
Table 11-36: AlmanacHeader Parameter Format	145
Table 11-37: SVn Almanac Parameter Format	145
Table 12-1: Ranging Commands	146
Table 12-2: SetRangingAddr Command	146
Table 12-3: SetRangingReqAddr Command	147
Table 12-4: GetRangingResult Request.....	147
Table 12-5: GetRangingResult Response.....	147
Table 12-6: SetRangingTxRxDelay Command	148
Table 12-7: Typical Delay Values for Semtech EVKs.....	148
Table 12-8: SetRangingParameter Command	148
Table 13-1: ConfigBleBeacon Command.....	149
Table 13-2: BleBeaconSend Command.....	149
Table 14-1: Cryptographic Keys Usage and Derivation	150
Table 14-2: CryptoSetKey Command	152
Table 14-3: CryptoSetKey Response.....	152
Table 14-4: CryptoDeriveKey Command	153
Table 14-5: CryptoDeriveKey Response	153
Table 14-6: CryptoProcessJoinAccept Command	154
Table 14-7: CryptoProcessJoinAccept Response	154
Table 14-8: CryptoComputeAesCmac Command	155
Table 14-9: CryptoComputeAesCmac Response	155
Table 14-10: CryptoComputeAesCmac Command Example	155
Table 14-11: CryptoComputeAesCmac Response Example	155
Table 14-12: CryptoVerifyAesCmac Command	156
Table 14-13: CryptoVerifyAesCmac Response	156
Table 14-14: CryptoAesEncrypt01 Command	157
Table 14-15: CryptoAesEncrypt01 Response	157
Table 14-16: CryptoAesEncrypt Command	158
Table 14-17: CryptoAesEncrypt Response	158
Table 14-18: CryptoAesDecrypt Command.....	159
Table 14-19: CryptoAesDecrypt Response.....	159
Table 14-20: CryptoStoreToFlash Command	160
Table 14-21: CryptoAesDecrypt Response.....	160
Table 14-22: CryptoRestoreFromFlash Command	160
Table 14-23: CryptoRestoreFromFlash Response.....	160
Table 14-24: CryptoSetParam Command.....	161

Table 14-25: CryptoSetParam Response.....	161
Table 14-26: CryptoGetParam Command.....	161
Table 14-27: CryptoGetParam Response.....	161
Table 14-28: CryptoCheckEncryptedFirmwareImage Command.....	162
Table 14-29: CryptoCheckEncryptedFirmwareImageResult Command	163
Table 14-30: CryptoCheckEncryptedFirmwareImageResult Response	163
Table 15-1: GetChipEui Command.....	164
Table 15-2: GetChipEui Response	164
Table 15-3: GetSemtechJoinEui Command.....	165
Table 15-4: GetSemtechJoinEui Response.....	165
Table 15-5: DeriveRootKeysAndGetPin Command (Standard)	166
Table 15-6: DeriveRootKeysAndGetPin Response (Standard)	166
Table 15-7: DeriveRootKeysAndGetPin Command (Advanced)	167
Table 15-8: DeriveRootKeysAndGetPin Response (Advanced)	167
Table 15-9: LoRaWAN 1.0.x vs. 1.1.x Security Correspondence Table.....	169
Table 16-1: ETSI Test Signals	170
Table 16-2: SetTxCw Command.....	171
Table 16-3: SetTxInfinitePreamble Command.....	171
Table 18-1: Register / Memory Access Operations.....	173
Table 18-2: System Configuration / Status Operations.....	174
Table 18-3: Radio Configuration / Status Operations	176
Table 18-4: Wi-Fi Scanning Configuration / Status Operations	178
Table 18-5: GNSS Scanning Configuration / Status Operations	179
Table 18-6: CryptoElement Configuration / Status Operations.....	180
Table 18-7: Bootloader Commands	181
Table 19-1: Revision History	182

1. Introduction

1.1 Scope

This document provides complete information on how to use the LR1110 transceiver in an application. It covers both hardware and software aspects. For LR1110 functionalities and circuit specifications, refer to the LR1110 Datasheet.

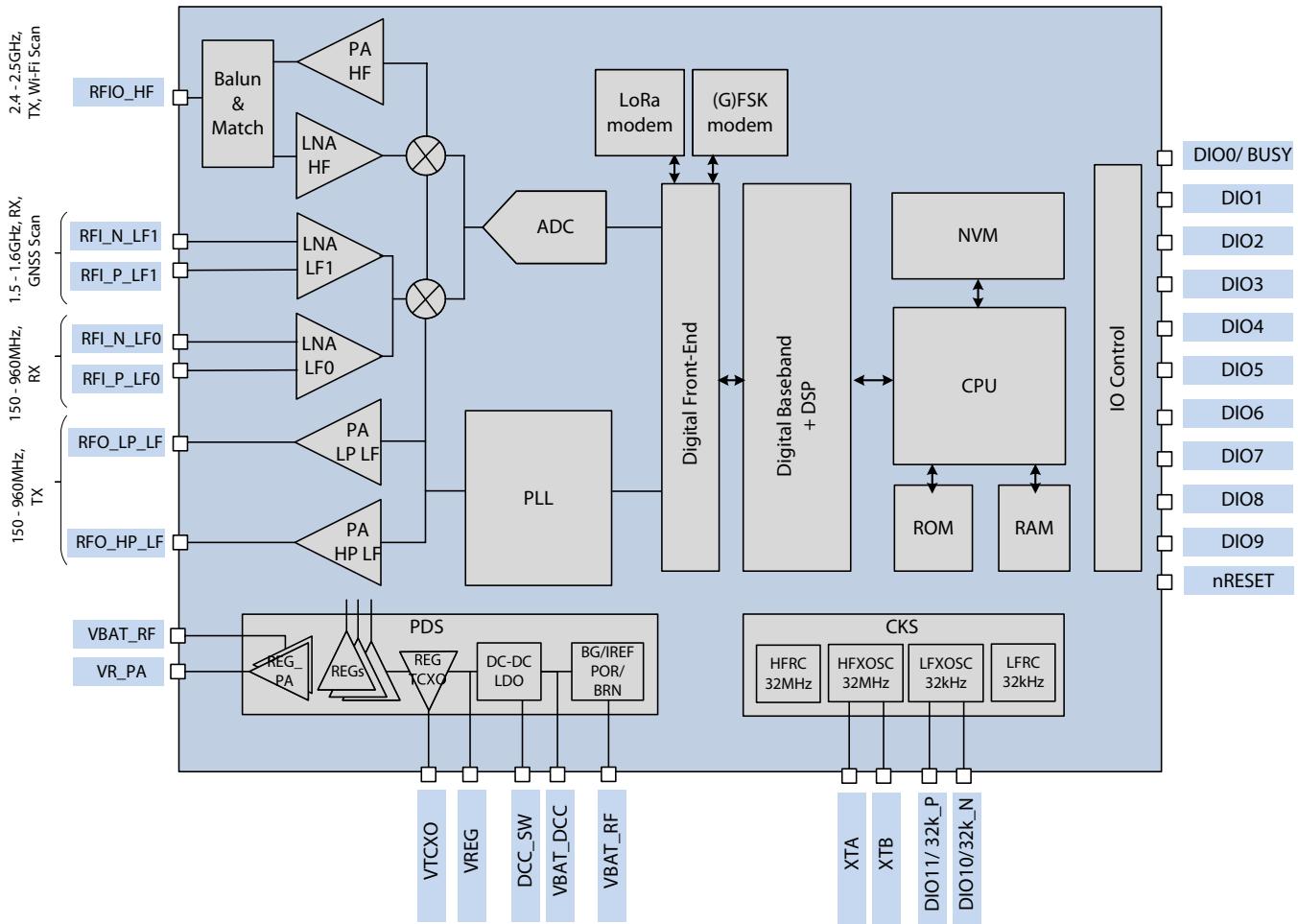


Figure 1-1: LR1110 Block Diagram

1.2 Overview

The LR1110 chip is a long range, ultra-low power transceiver that enhances LoRa®-enabled geolocation applications. For LPWAN use cases, the LR1110 supports LoRa, (G)FSK and Sigfox® modulation, as well as Long Range Frequency Hopping Spread Spectrum (LR-FHSS) on sub-GHz bands. The device is highly configurable over the 150-960MHz ISM bands to meet different application requirements utilizing the global LoRaWAN® Standard or proprietary protocols.

Besides the world-wide sub-GHz transceiver capabilities, the LR1110 features a very low power multi-band front-end that can acquire several signals of opportunity for geolocation purposes (802.11b/g/n Wi-Fi® AP MAC addresses, GNSS (GPS, BeiDou) satellite signals). The acquired information is transmitted using an LPWAN network to a geolocation server, which computes the position of the object.

The LR1110 is optimized for low power and long battery life applications requiring indoor and outdoor geolocation. Its efficient Wi-Fi and GNSS geolocation capabilities, coupled with highly optimized detection algorithms, allow to achieve a geolocation at a fraction of the power needed by existing solutions on the market.

The LR1110 geolocation capability is further increased by Bluetooth® Low Energy beaconing compatibility to allow indoor and outdoor geolocation use cases, as well as a powerful sub-GHz ranging engine for localization and recovery of lost assets in logistics applications.

2. System Processes

2.1 System Modes

The LR1110 operating modes are shown in [Figure 2-1: LR1110 Modes and Transitions](#):

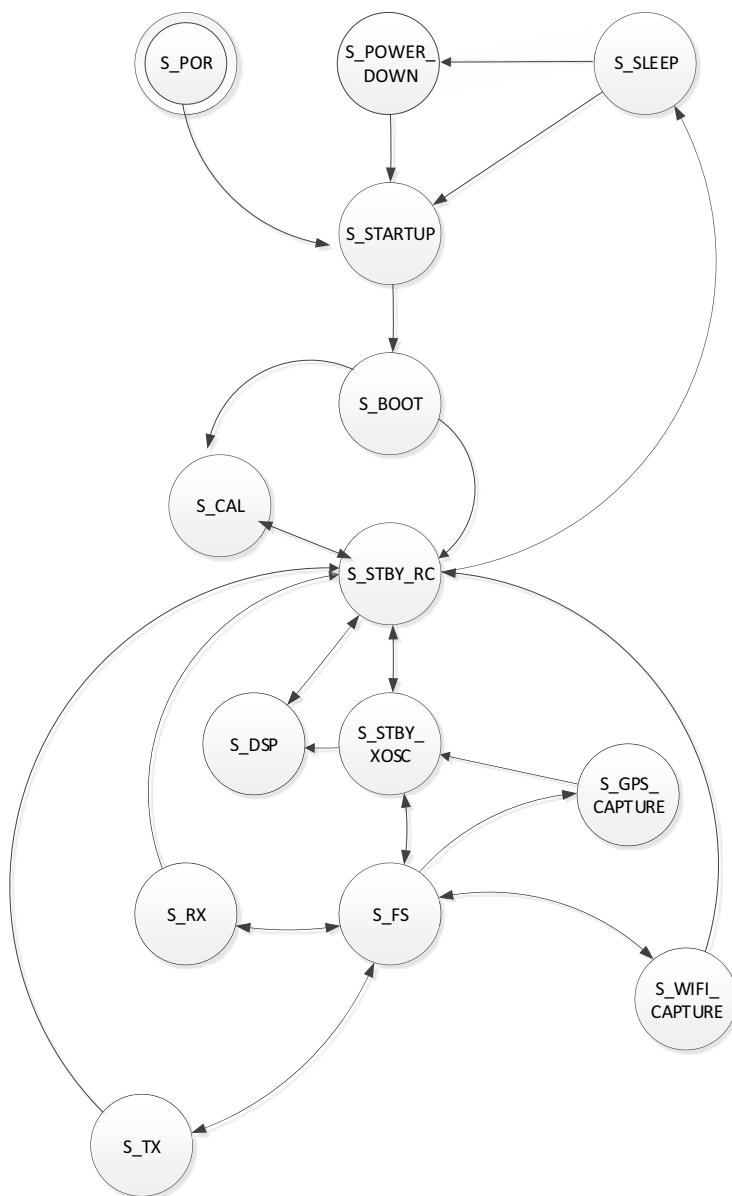


Figure 2-1: LR1110 Modes and Transitions

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.

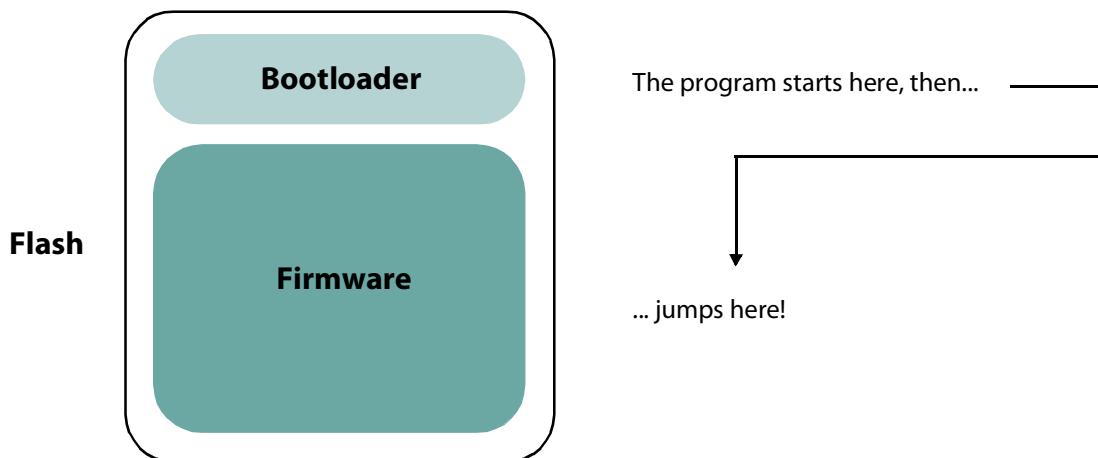


Figure 2-2: Bootloader

2.1.2 Standby

This is the default mode of the LR1110. 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)	0xE9 (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:

- $Freq1 = \text{floor}((f_{\min_mhz} - 1)/4)$
- $Freq2 = \text{ceil}((f_{\max_mhz} + 1)/4)$

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.

Note: For Bluetooth® Low Energy beaconing compatibility the LR1110 requires a PLL_TX calibration sequence to be performed prior to the first Bluetooth® Low Energy beaconing compatible transmission. Refer to [Section 13](#) for additional information.

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 LR1110 into a low power consumption mode between radio or geolocation operations, while retaining the configuration register values and storing the firmware data in RAM. When the LR1110 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	Sleep w/ 8kB retention, no RTC	IDDSL3	<1
1	1	Sleep w/ 8kB retention	IDDSL3A / IDDSL4A	<1

2.1.6 Reset

Four reset sources are available to trigger a LR1110 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 LR1110 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 LR1110. 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 LR1110 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 GNSS Scanning Mode

GNSS scanning mode detects GPS and BeiDou signals on the RFI_N_LF1 and RFI_P_LF1 pins for outdoor geolocation. Satellite signals are digitized and processed by the integrated DSP. At the end of the satellite signal processing, the BUSY signal returns to low, indicating to the host controller that the GNSS scanning data is available. The result can then be transmitted to a geolocation server using an LPWAN network to compute the device position.

Different GNSS scanning sub-modes are available, depending on the availability of almanac data and assistance information. Refer to the section [11. GNSS Scanning](#) for more details.

2.1.8 Wi-Fi Passive Scanning Mode

Wi-Fi Passive Scanning mode detects and demodulates Wi-Fi signals (802.11b, g, or n) from access points in the proximity of the device on the RFIO_HF pin. The Wi-Fi signal is processed by the integrated DSP, and the available MAC addresses are extracted. At the end of the Wi-Fi signal processing, the BUSY signal returns to low, indicating that the MAC addresses are available to the host controller and ready to be sent to a geolocation server using an LPWAN network to compute the device position.

2.1.9 DSP Mode

The LR1110 geolocation functions need to process the Wi-Fi or GNSS environment captures. In this mode, only the DSP and the associated regulators are kept active in order to minimize the power consumption. The BUSY signal is high.

This mode is activated automatically by the LR1110 during the GNSS and Wi-Fi scanning processes (IDDRXGPS2 and IDDRXWIFI3 respectively), and is not actionable by the user.

2.1.10 RX Mode

RX mode receives incoming RF packets on the RFI_N_LF0/RFI_P_LF0 pins in the sub-GHz band (150-960MHz) 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 LR1110 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.

2.1.11 TX Mode

TX mode uses the supported modems to transmit 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).

After ramping-up the PA, the LR1110 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 LR1110 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.

2.1.12 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.

2.1.12.1 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 LR1110 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 LR1110.

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 LR1110 hardware
- *Use Case* specifies the usage of the device:
 - 0x01: LR1110
 - 0x02: LR1120
 - 0x03: LR1121
 - 0xDF: Bootloader mode
- *FW Major + FW Minor* is the version of the LR1110 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 LR1110. More details are supplied in the AN1200.57 Application Note. Command [CryptoCheckEncryptedFirmwareImage](#) can be used prior to writing the image.

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 LR1110. Please refer to [Figure 2-1: LR1110 Modes and Transitions](#) for a representation of the LR1110 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	37e3 (FW 03.04+03.05), 40e3 (≥FW 03.06)
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
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 LR1110 exposes an API which allows the Host controller to communicate with the LR1110 through a set of SPI commands / responses. The BUSY signal is used as a handshake to indicate if the LR1110 is ready to accept a command. Therefore, it is necessary to check the status of BUSY prior to sending a command.

Note: The SPI protocol differs between the Transceiver/Bootloader and the Modem. See LoRa® Basics™ Modem-E Reference Manual for modem usage.

3.1 Write Commands

During write commands, the LR1110 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 LR1110 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 LR1110, such as internal status or geolocation 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 LR1110 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.

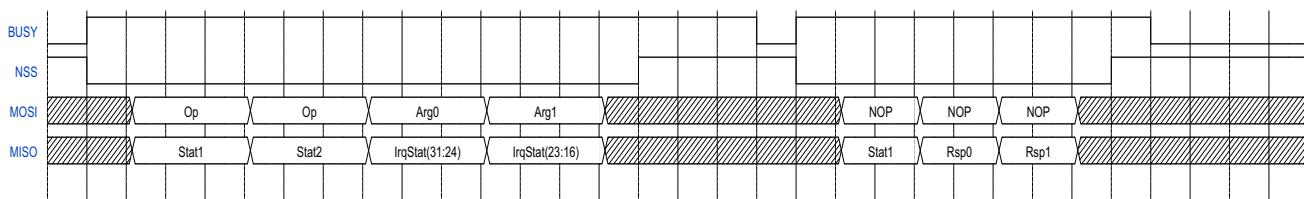


Figure 3-2: Read Command Timing Diagram

3.3 Command Endianness

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

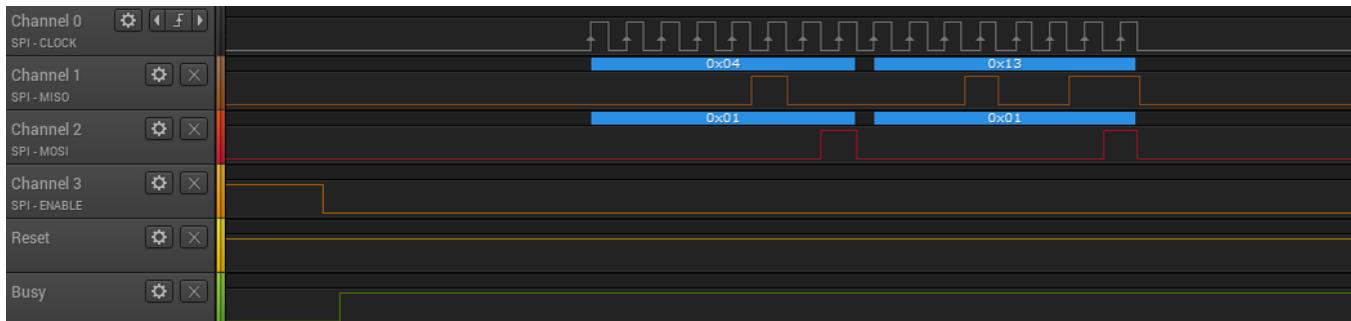


Figure 3-3: GetVersion Write Capture

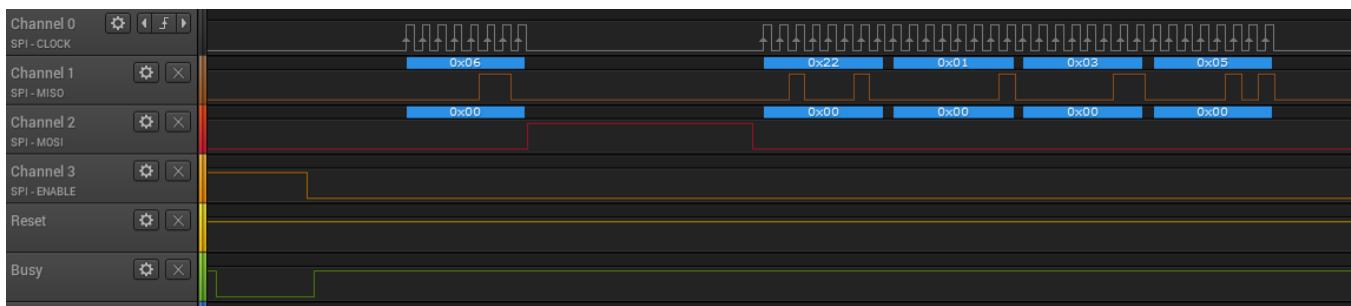


Figure 3-4: GetVersion Read Capture

3.4 Status Registers

The LR1110 features 2 status variables *Stat1* and *Stat2*, which determine the status of the LR1110 (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 LR1110.

3.4.1 GetStatus

Command GetStatus(. . .) returns the LR1110 status flags *stat1* and *stat2*, and the LR1110 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 LR1110 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
 - 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 LR1110 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 LR1110 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 LR1110:
 - ◆ 0: Sleep
 - ◆ 1: Standby with RC Oscillator
 - ◆ 2: Standby with external Oscillator
 - ◆ 3: FS
 - ◆ 4: RX
 - ◆ 5: TX
 - ◆ 6: Wi-Fi or GNSS geolocation
 - ◆ 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:

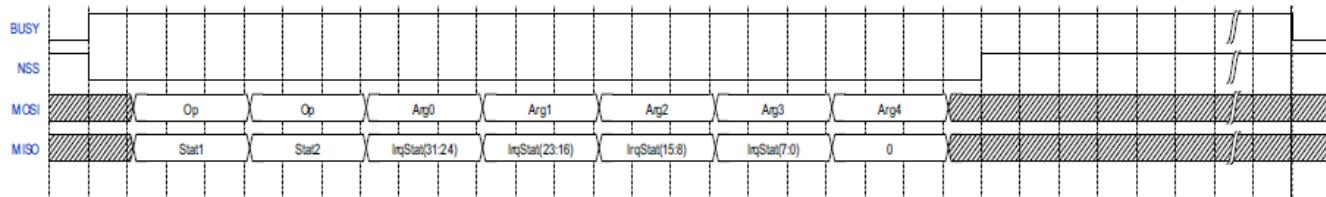


Figure 3-5: BUSY Timing Diagram

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 27](#).

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 LR1110 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 Wi-Fi, GNSS, and sub-GHz RF paths.
DIO8	Can control external RF switches or LNAs on the Wi-Fi, GNSS, 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 LR1110 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 Wi-Fi, GNSS, 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 LR1110, 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 LR1110 features numerous interrupt sources, allowing the host to react to a large variety of events in the LR1110 system without the need to poll registers, therefore allowing power-optimized applications.

The LR1110 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 LR1110 interrupts can be read using command `GetStatus(...)`, but they are also returned by the LR1110 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 LR1110 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	GNSSDone	GNSS Scan finished
20	WifiDone	Wi-Fi Scan finished
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	-	RFU
27	LoRaRxTimestamp	Last LoRa symbol received. To be used for time-stamping the received packet. The device is still in RX mode.
28-31	-	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, GNSS, Wi-Fi 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	RFU	RfSw GnssCfg	RfSw WifiCfg
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
- *RfSwGnssCfg*: Each bit indicates the state of the relevant RFSW DIO when in GNSS scanning mode
- *RfSwWifiCfg*: Each bit indicates the state of the relevant RFSW DIO when in Wi-Fi scanning mode or high frequency RX mode
- Byte 7 is 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 LR1110 DIOs, when the LR1110 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 LR1110 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:

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

NOTE: GetTemp() uses XOSC mode to get the temperature, so if a TCXO is connected to the LR1110, 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)

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

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

The LR1110 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: LR1110 POR and BRN Functions](#) for an illustration of the POR and BRN functions.

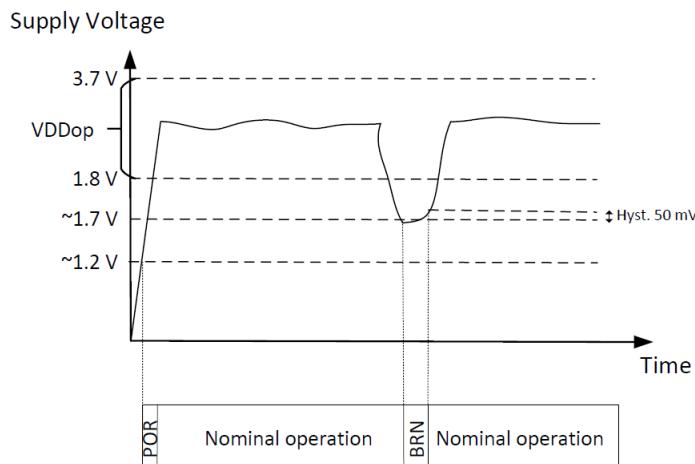


Figure 5-1: LR1110 POR and BRN Functions

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 through command `Set Dio IRQ Params()`.

5.5 Over Current Protection

The LR1110 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)

6. Clock Sources

The LR1110 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:

- 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 Clock

A 32.768kHz clock source can be used instead of the (default) 32.768kHz RC oscillator as a low frequency clock source using command `ConfigLfClock(...)`. This clock source can be provided either by a 32.768kHz crystal attached to the LR1110 chip, or by an external 32.768kHz signal fed to the DIO11 pin. The amplitude of the external 32.768kHz signal fed to the DIO11 pin should be a full swing rail-to-rail clock signal. Smaller or much larger amplitudes could cause extra consumption.

A 32.768kHz clock source is mandatory for GNSS Dual Constellation and for GNSS Advanced scan mode. Refer to sections [11.3.2 GnssSetConstellationToUse](#) and [11.3.5 GnssSetMode](#) for details.

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 LR1110. 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 LR1110 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 LR1110 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 LR1110 and the 32MHz crystal, as shown in Figure 6-1: LR1110 Thermal Insulation on PCB Top Layer.

Refer to the LR1110 PCB reference design on www.semtech.com for an implementation example in PCB design.

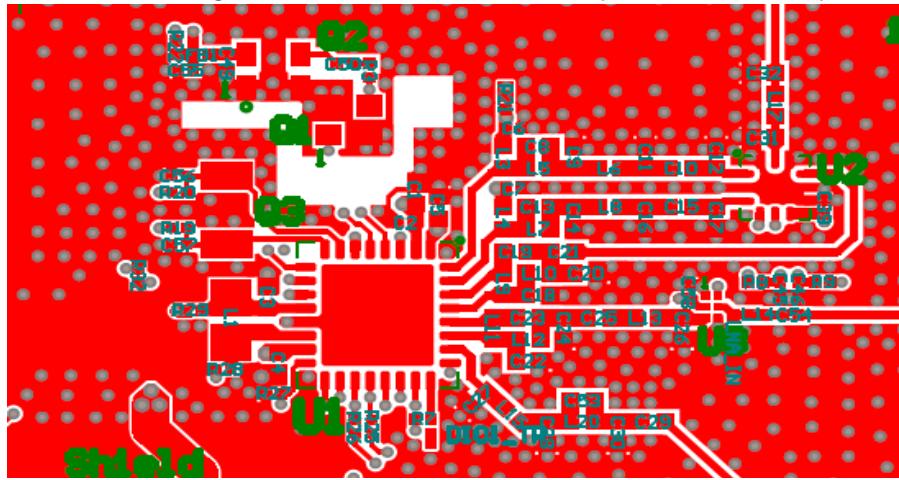


Figure 6-1: LR1110 Thermal Insulation on PCB Top Layer

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. A TCXO is required by the LR1110 GNSS features in order to minimize the power consumption required to perform an outdoor geolocation. When a TCXO is used:

- The TCXO should be connected to pin XTA, through a 220Ω resistor and a 10pF 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 `SetTxoMode()`.
 - 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.

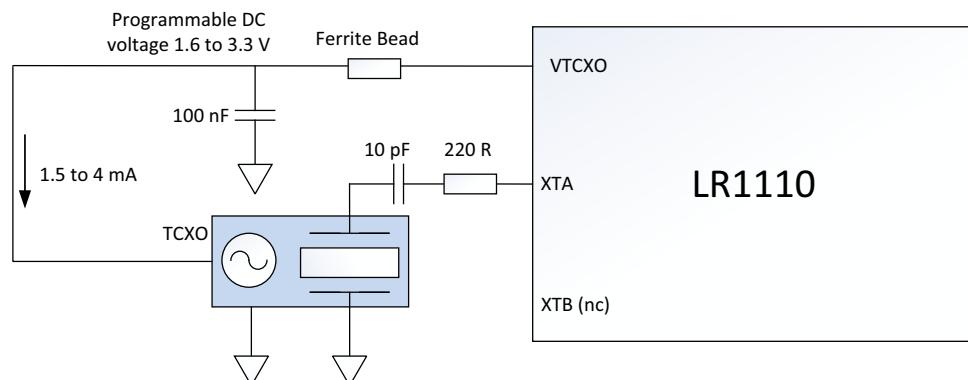


Figure 6-2: TCXO Circuit Diagram

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](#). The TCXO supply voltage precision varies with *RegTcxoTune* setting. It is typically +/-50mV for VTCXO = 1.8V setting.

Table 6-3: TCXO Supply Voltage Programming Values

RegTcxoTune	TCXO Supply Voltage (typical)
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. Sub-GHz Radio

7.1 Overview

The LR1110 is a half-duplex RF transceiver capable of handling constant envelope modulation schemes such as LoRa, (G)FSK, Sigfox, and LR-FHSS. The LoRa, Sigfox and (G)FSK modulations are fully compatible with the SX1261/62/68 family for sub-GHz.

The sub-GHz radio system is shown in [Figure 7-1: LR1110 Sub-GHz Radio Block Diagram](#) below. It is composed of the frequency synthesizer (also referred as PLL), two TX paths (High Power and Low Power), and an RX path, followed by a high-bandwidth ADC. Both the ADC and the PLL are tied to the digital subsystem, the supported modems, and the LR-FHSS modulator.

The PAs, supported modems, and LR-FHSS modulation are described in dedicated sections.

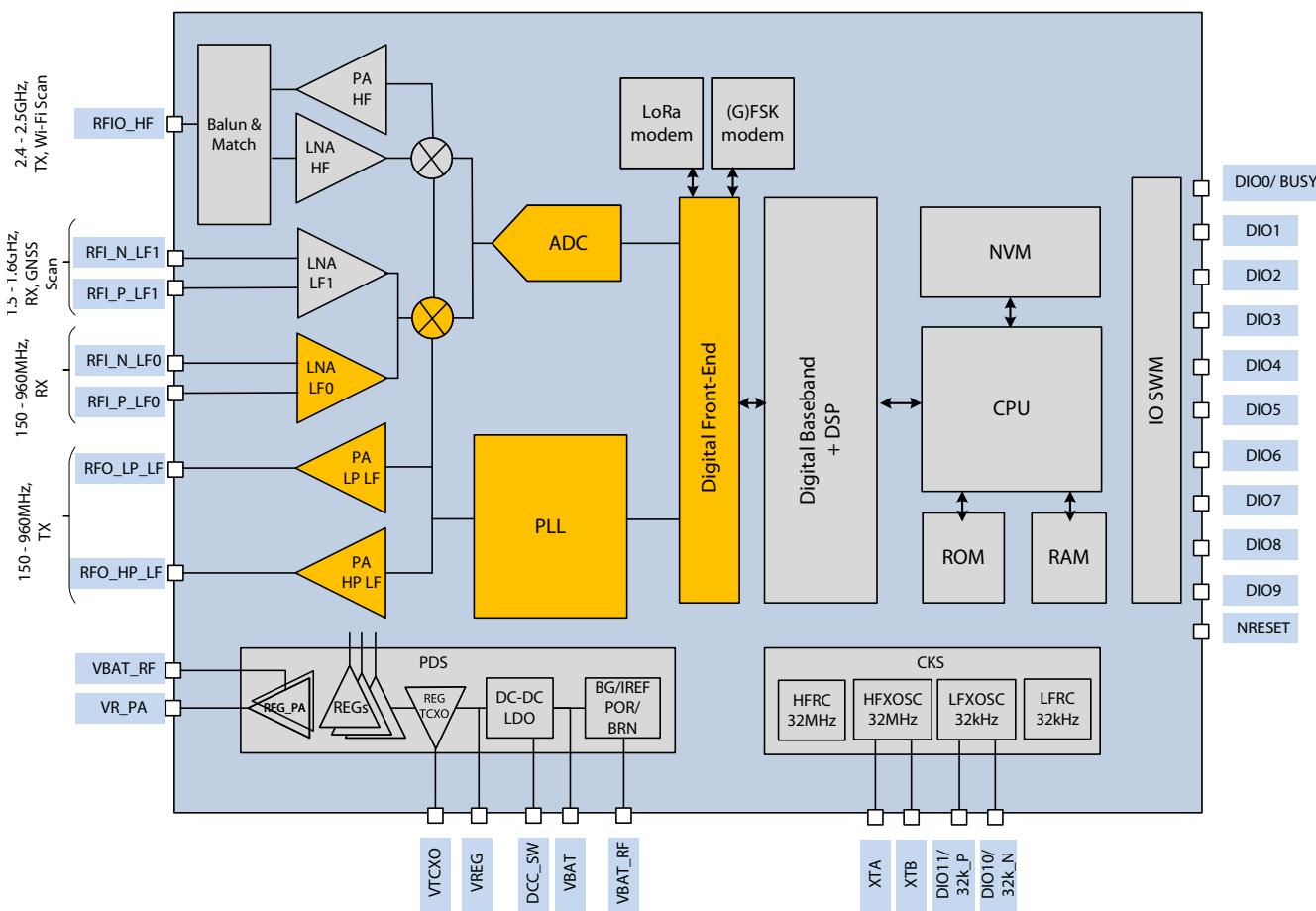


Figure 7-1: LR1110 Sub-GHz Radio Block Diagram

The LR1110 frequency synthesizer allows a continuous operation in the 150MHz-2500MHz frequency range. It is shared between the GNSS and Wi-Fi scanning engines, and the sub-GHz radio , therefore no simultaneous GNSS scanning, Wi-Fi scanning, or sub-GHz radio operation is possible.

The LR1110 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 49](#) for details.

7.2 Commands

7.2.1 SetRfFrequency

Command SetRfFrequency(. . .) sets the RF (PLL) frequency of the sub-GHz 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 sub-GHz radio in Hz. All frequency dependent parameters are automatically recomputed by the LR1110 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. 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 0xFFFFFFF disable the timeout function.
 - 0x000000 sets the device in RX mode until a reception occurs. After packet reception, the device returns to Standby mode.
 - 0xFFFFFFF 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 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.
- 0xFFFFFFF: 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: LR1110 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 57](#)).
- 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.

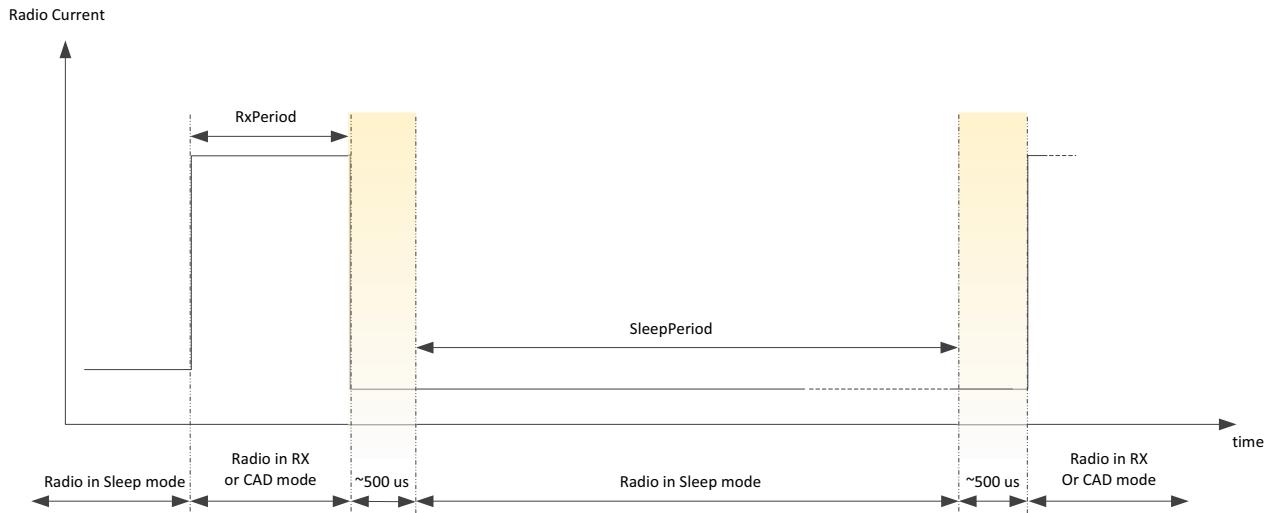


Figure 7-2: LR1110 Current Profile During RX Duty Cycle Operation

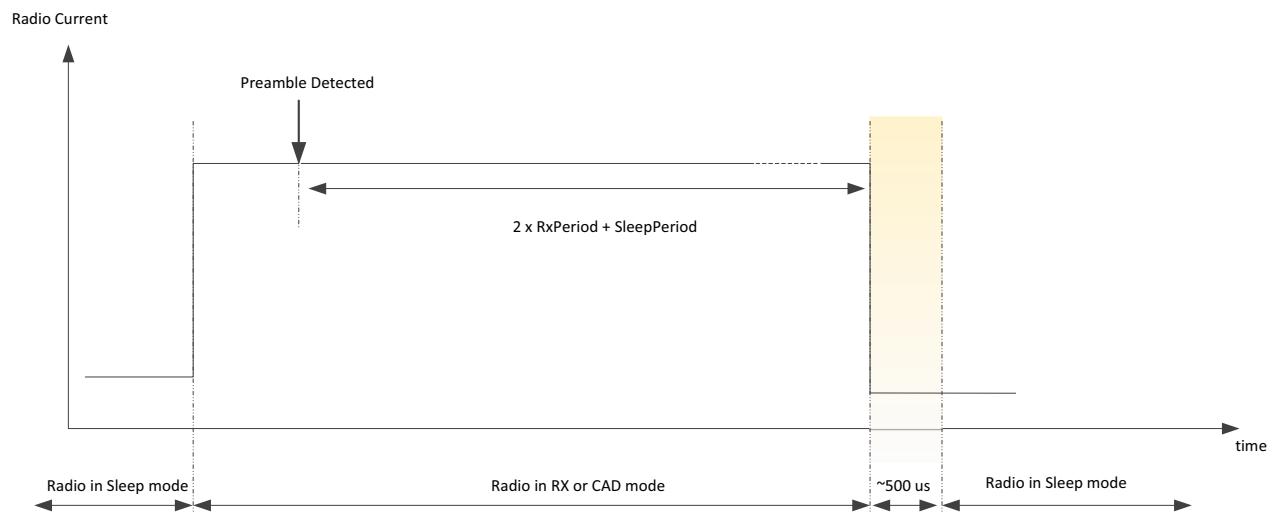


Figure 7-3: RX Duty Cycle Upon Preamble Detection

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 LR1110 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 LR1110 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	8	8	9	9	

Gain Tune and Gain Offset values can be determined using an RF generator with the LR1110 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 (rss_i_inst).
 - c. Calculate the rss_i_error = rss_i_inst - gen_pwr.
 - d. Log the rss_i_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, 0x71110000 );
```

```
lr11xx_radio_set_rssi_calibration( context, rssि_cal_table ); // All parrameters of  
rssि_cal_table set to 0  
    lr11xx_radio_set_rx_with_timeout_in_rtc_step( context, 0xFFFFFFF );  
  
    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, rssि_in_dbm );  
  
/* Add a way to log (gain[i], power[i], rssि_in_dbm) triplet to be able to compute offset and  
tunes for the RSSI calibration */  
}
```

8. Modems

8.1 Modem Configuration

The LR1110 contains different modems capable of handling different constant envelope modulations. For LoRa®/(G)FSK modems, LR-FHSS and Sigfox modulations, Bluetooth® Low Energy beaconing compatibility, and Sub-GHz ranging, 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).

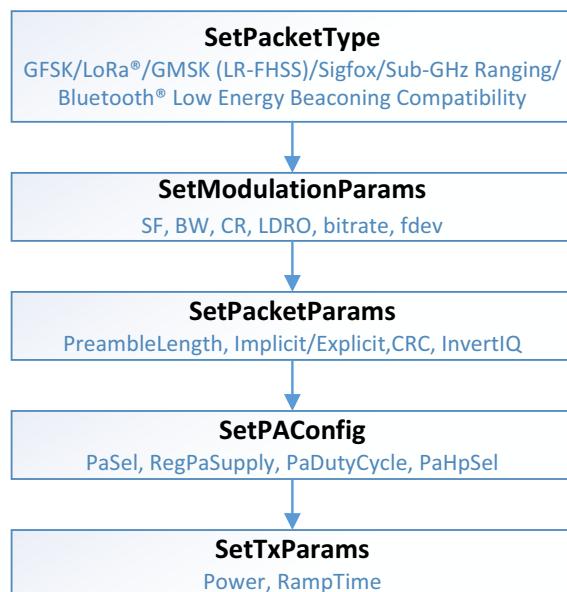


Figure 8-1: Modem Command Order

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
 - ◆ 0x03: Sigfox Uplink¹
 - ◆ 0x04: GMSK (LR-FHSS)
 - ◆ 0x05: Ranging
 - ◆ 0x06: Bluetooth Low Energy beaconing compatible
 - ◆ 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. It 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
 - ◆ 3: Sigfox Uplink.¹
 - ◆ 4: LR-FHSS
 - ◆ 5: Ranging

1. Sigfox downlink is performed using *PacketType* 0x01. See AN1200.73 for details.

- 6: Bluetooth Low Energy beaconing compatible
- 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](#).

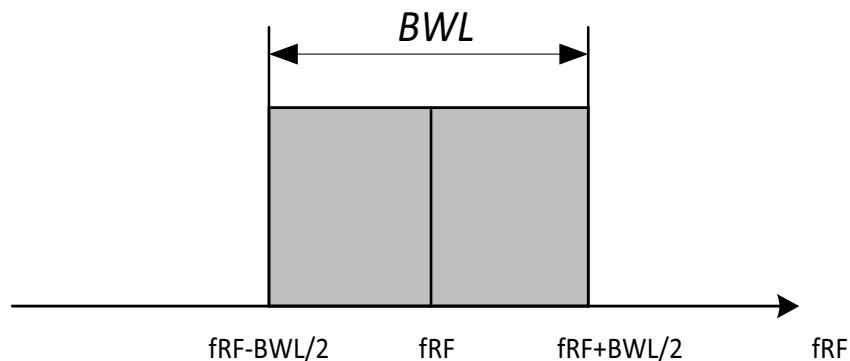


Figure 8-2: LoRa Signal Bandwidth

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:

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

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.

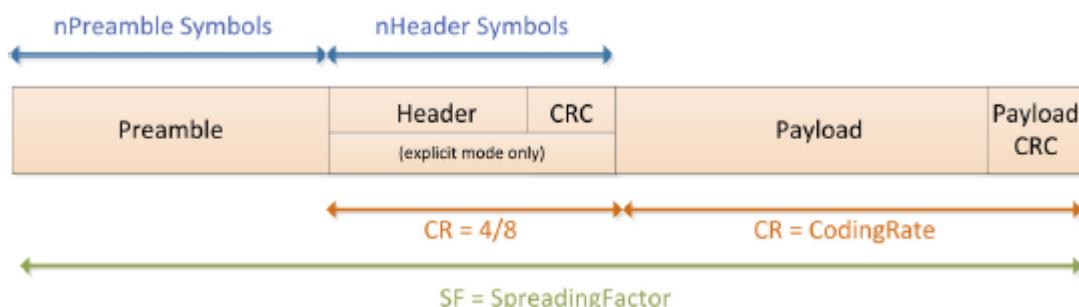


Figure 8-3: LoRa Packet Format

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 LR11xx 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 setting¹.
 - ◆ 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
- *CR* configures the Coding Rate. Long interleaver increases robustness against interferers. (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 = 4/5² Overhead Ratio 1.25
 - ◆ 0x06: Long Interleaver CR = 4/6² Overhead Ratio 1.5
 - ◆ 0x07: Long Interleaver CR = 4/8² 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 <i>Timeout</i> *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 <i>Timeout</i> 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 $SnrPkt/4$. The SNR value is an unbiased scaled value.
If the division by 4 is performed in floating-point, standard floating-point rounding provides the correct value.
If the division by 4 is performed in fixed point, it is recommended to round towards the nearest integer. In our drivers coded in C, this can be done by first adding 2, and then performing a fixed-point truncated division by 4.
$$pkt_status->snr_pkt_in_db = (((int8_t) rbuffer[1]) + 2) >> 2;$$
- *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 8.10 "RSSI Functionality" on page 90](#).

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:

$$(2 \times Fdev + BR) < BWF$$

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

The bandwidth must be chosen so that:

$$\text{Bandwidth [DSB]} \geq BR + 2 \times \text{frequency deviation} + \text{frequency error}$$

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

8.4.2 (G)FSK Packet Engine

The LR1110 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 LR1110 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

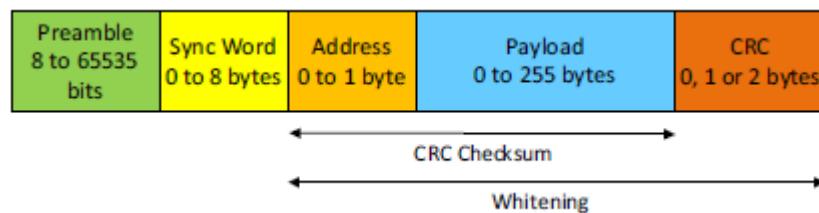


Figure 8-4: Fixed-Length Packet

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.

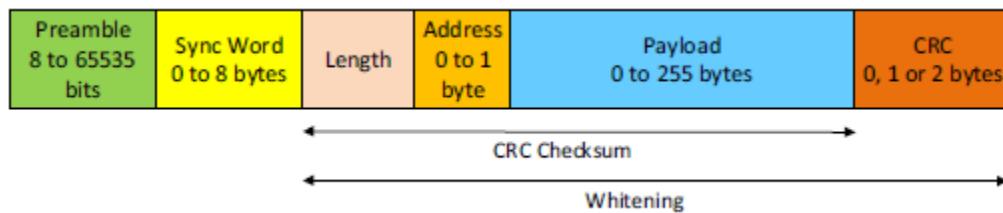


Figure 8-5: Variable-Length Packet

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 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 LR1110 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.

8.5 (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*: (G)FSK bit rate interpretation depends on the *BitRate*(31). It is either bits/s or the 8 LSBs are a fractional part:
 - *BitRate*(31) = 0: GFSK Bitrate in bits/s is *BitRate*(30:0)
 - *BitRate*(31) = 1: GFSK Bitrate in bits/s is *BitRate*(30:0)/256
- *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
 - 0x16: Raise Cosine BPSK BT 0.7
- *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 (234.3kHz 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

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 LR1110 is transmit only. The LR1110 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](#).

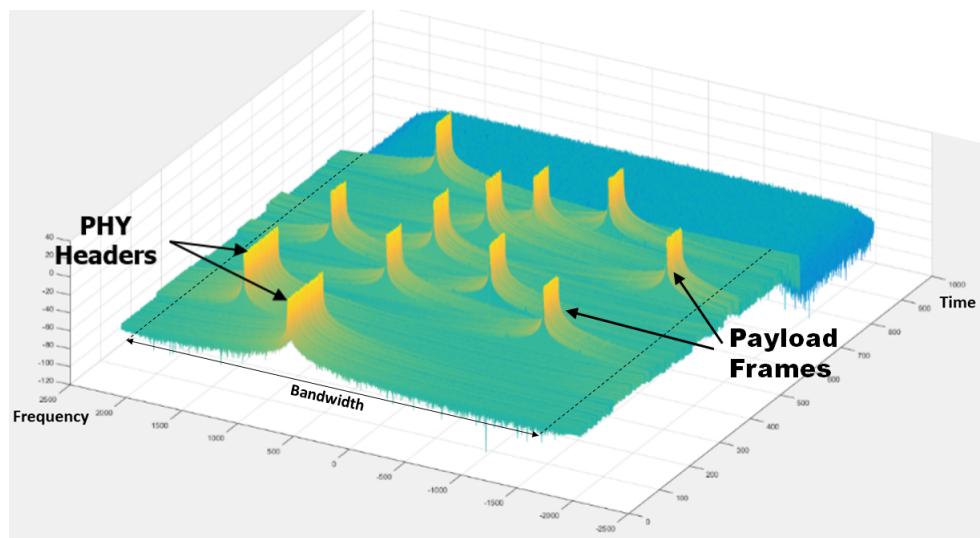


Figure 8-6: LR-FHSS Spectral Plot Example

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 LR1110 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 Sigfox Modulation

The LR1110 can transmit Sigfox packets at both 100 and 600bps. This allows support of all the Sigfox Radio Configurations from RC1 to RC7.

The LR1110 is also able to receive Sigfox downlinks at 600bps in the selected Sigfox Radio Configuration.
Please contact a Semtech representative for additional information.

No Sigfox protocol stack is embedded inside the LR1110. The user must configure and transmit the Sigfox packets according to the command order depicted in [Figure 8-1: Modem Command Order](#).

Since commands SetModulationParams(. . .) and SetPacketParams(. . .) are modem specific, the description of those commands for Sigfox support is described below.

8.8.1 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 Sigfox modulation.

Table 8-26: 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*: see [Section 8.5.1](#). Only supports 100bps (RC1/RC3/RC5/RC6 and RC7) and 600bps (RC2 and RC4).
- *PulseShape*: refer to the description in [Section 8.5.1](#). Only BT 0.7 supported (0x16).

8.8.2 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 Sigfox modulation..

Table 8-27: SetPacketParams Command

Byte	0	1	2	3	4	5	6	5	5
Data from Host	0x02	0x10	Payloadlen	RampUp Delay (15:8)	RampUp Delay (7:0)	RampDown Delay (15:8)	RampDown Delay (7:0)	BitNum (15:8)	BitNum (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0	0

- *PayloadLen*: in bytes. To configure the packet controller.
- *RampUpDelay*: in us. Delay to fine tune ramp up time.
- *RampDownDelay*: in us. Delay to fine tune ramp down time.
- *BitNum*: in bits. Used to ramp down the PA before the end of payload length, in case actual length is not a multiple of 8.

8.9 Data Buffer

The LR1110 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 LR1110 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 LR1110 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](#))

8.10 RSSI Functionality

The RSSI information of the LR1110 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 8-28: RSSI Information Origin and Meaning](#) below:

Table 8-28: RSSI Information Origin and Meaning

Command	Modem	Name	Description
GetRssiInst(...)	All	RssiInst	Instantaneous RSSI
	(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.
GetPacketStatus(...)	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.

The RSSI information has a propagation time in the digital chain of the receiver. As depicted in [Figure 8-7](#) hereafter, a longer time is typically necessary for the first RSSI value to be available (Twait) than for the following RSSI samples (Trssi).

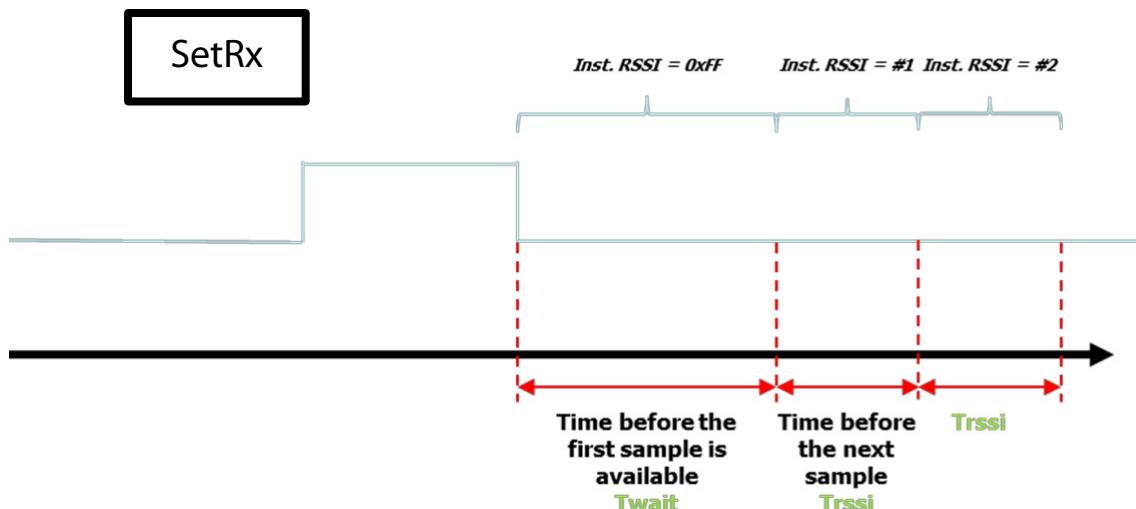


Figure 8-7: RSSI Propagation Time

For GFSK, Twait and Trssi depend on the selected bandwidth, as shown in [Table 8-29](#) below:

Table 8-29: Twait & Trssi Values for GFSK Bandwidths

BWF kHz	Twait us	Trssi us
467	30.68	2.00
373.6	37.78	2.50
312	44.89	3.00
234.3	54.61	4.00
187.2	67.69	5.00
156.2	80.76	6.00
117.3	102.30	8.00
93.8	127.28	10.00
78.2	152.26	12.00
58.6	197.61	16.00
46.9	246.41	20.00
39	295.20	24.00
29.3	388.18	32.00
23.4	484.59	40.00
19.5	581.01	48.00
14.6	769.24	64.00
11.7	960.91	80.00
9.7	1152.58	96.00
7.3	1531.30	128.00
5.8	1913.47	160.00
4.8	2295.64	192.00

For LoRa modulation, Twait and Rssi are identical, and equal to the LoRa Symbol time.

9. Power Amplifiers

The LR1110 features 2 power amplifiers for sub-GHz 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

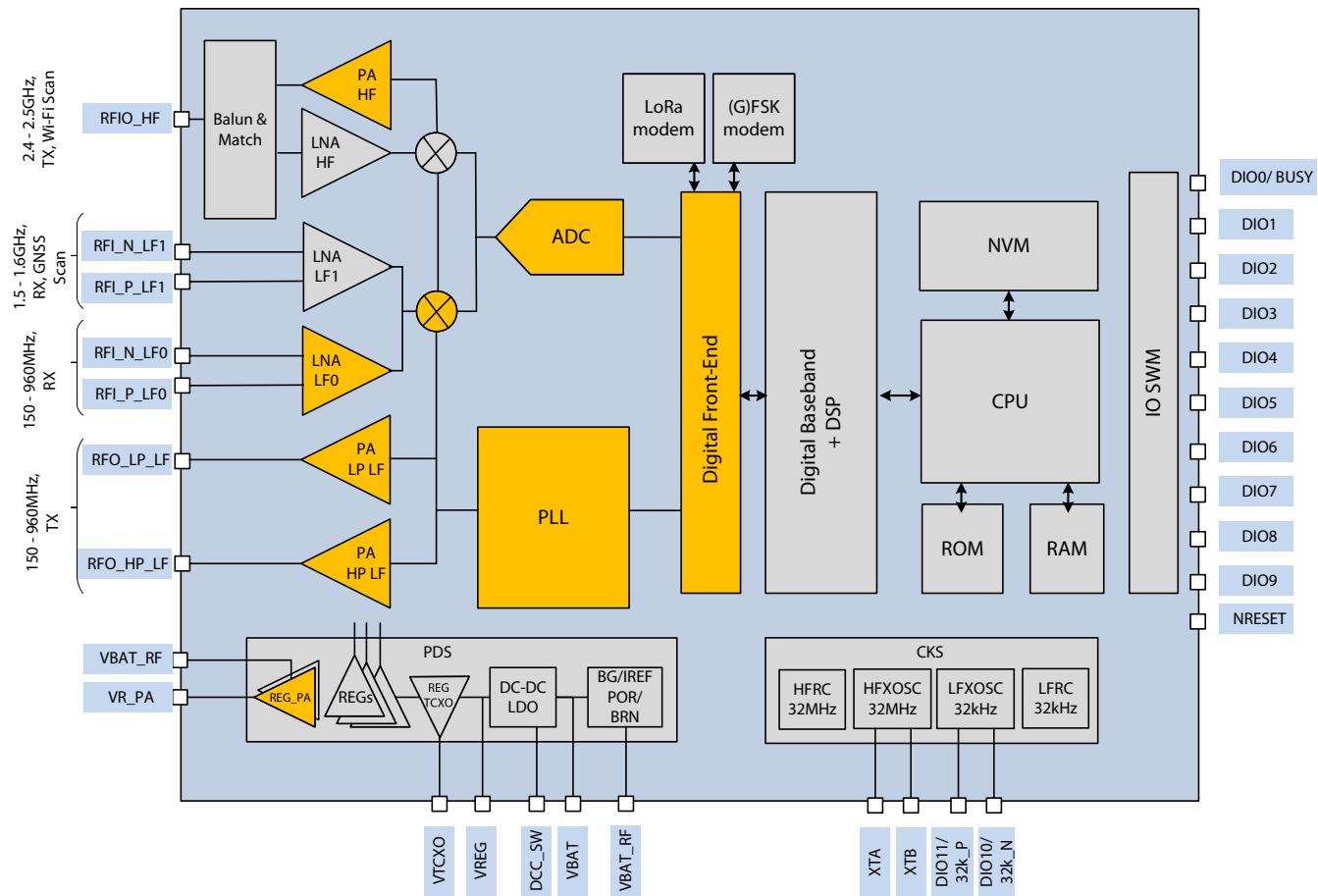


Figure 9-1: LR1110 Power Amplifiers Block Diagram

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

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

- Select the PA to be used (HP or LP)
- 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 Supply Scheme](#).

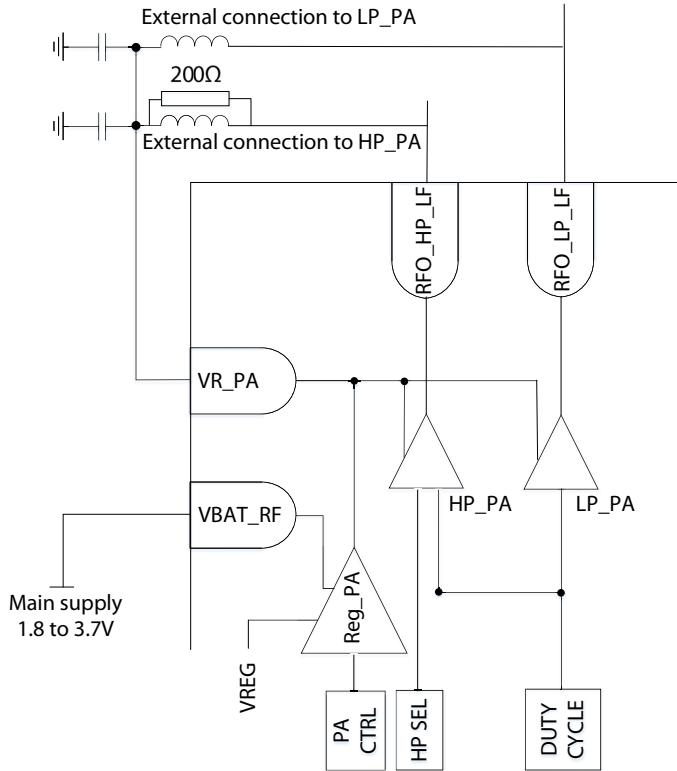


Figure 9-2: PA Supply Scheme

The PA regulator (Reg_PA) supplies the PAs through the VR_PA pin. It is best practice to place a 200 Ohm resistor in parallel to the HP PA choke inductor. Each amplifier requires a high-Q choke inductor connected externally between their respective outputs, and VR_PA to provide the bias and control the output power.

The PA regulator is internally connected to the DC-DC /LDO output for the low power PA, allowing a +15 dBm operation in both DCDC or LDO configurations. It is also connected to the VBAT_RF pin for the high power PA, 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 LR1110 incorporates a precise duty cycle trimmer shared between the two 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.

VrPa, RegPa Supply = Vreg = DCC

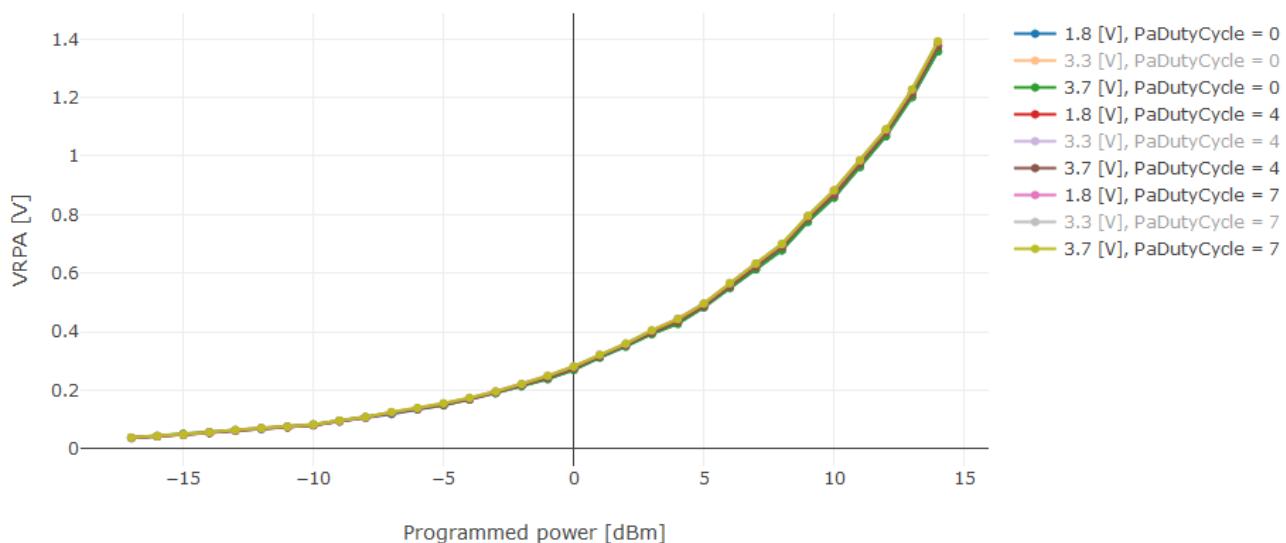


Figure 9-3: Low Power PA VR_PA Voltage vs TxPower

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.

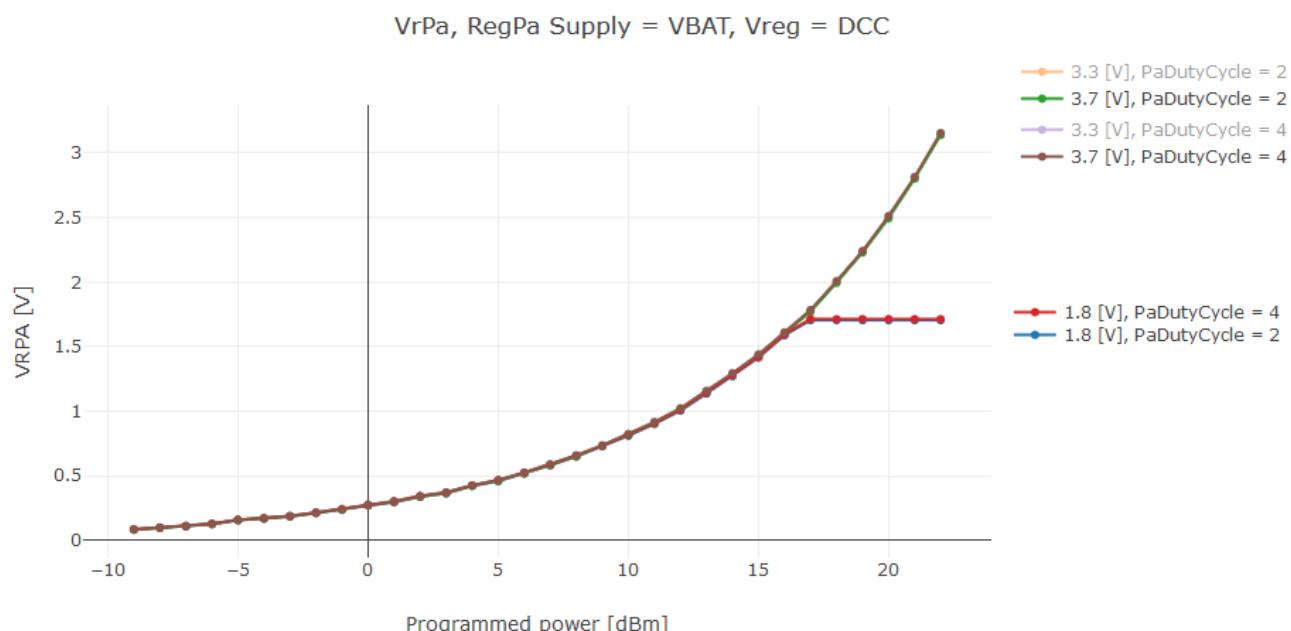


Figure 9-4: High Power PA VR_PA Voltage vs. TxPower

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 the low power 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)

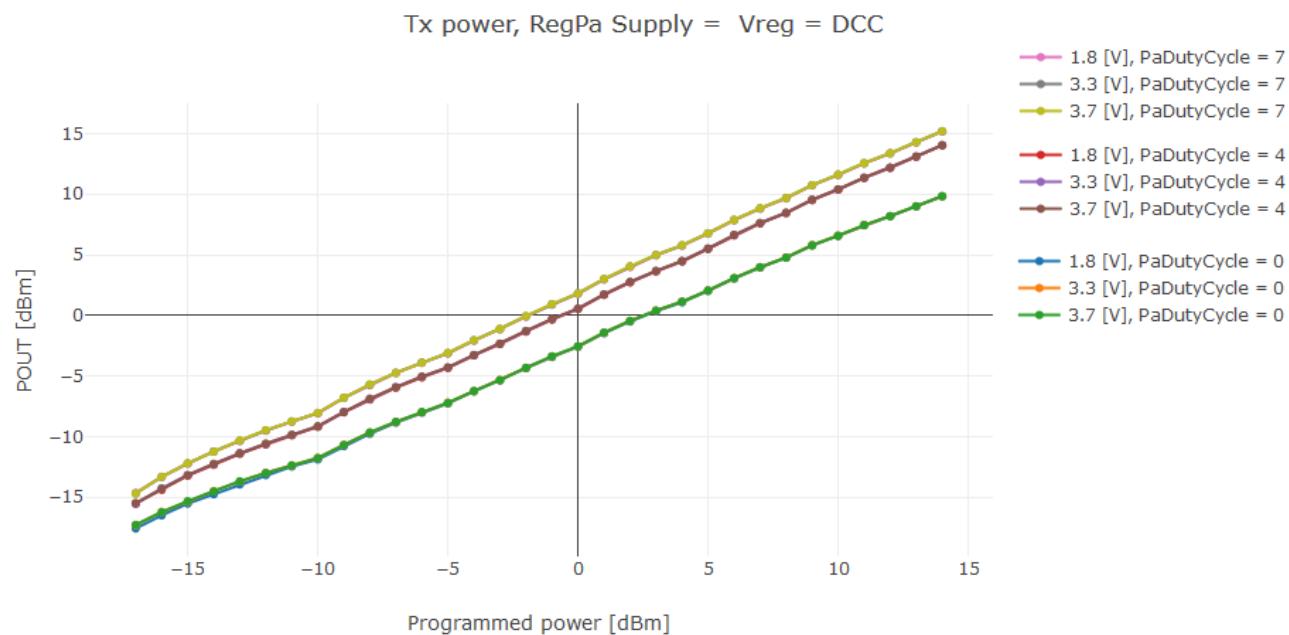


Figure 9-5: Low Power PA Output Power vs. *TxPower*

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 LR1110 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

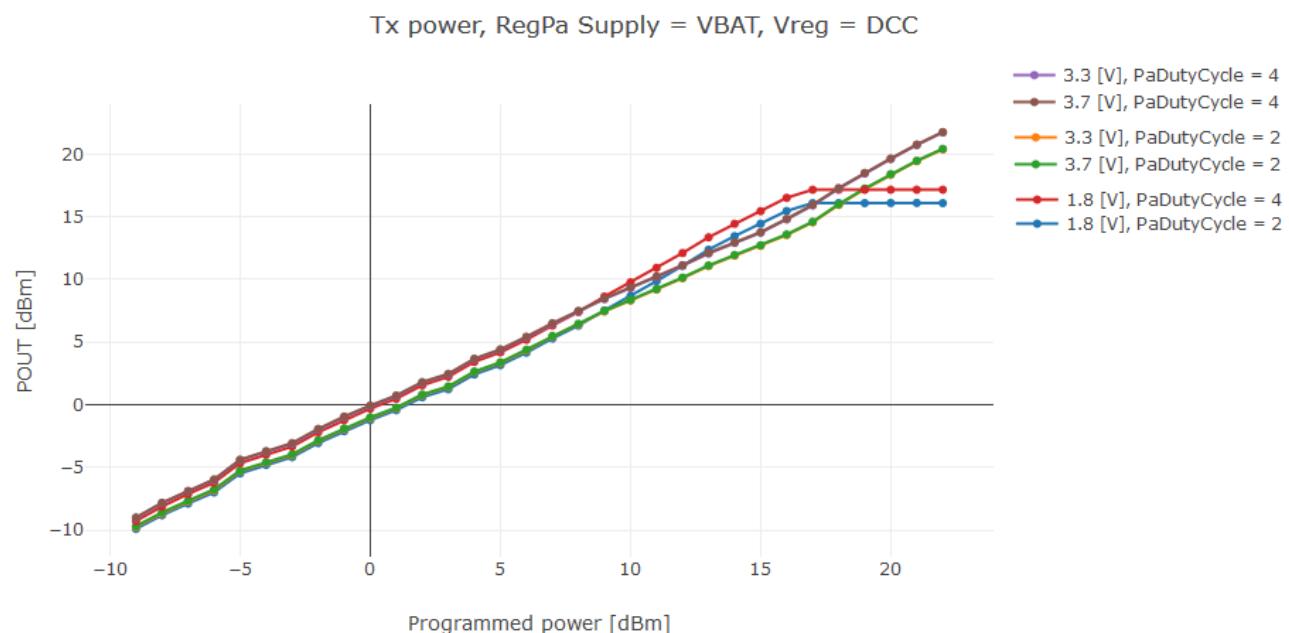


Figure 9-6: HP PA Output Power vs. TxPower

9.3 PA Current Consumption

9.3.1 Low Power PA

Figure 9-7: 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

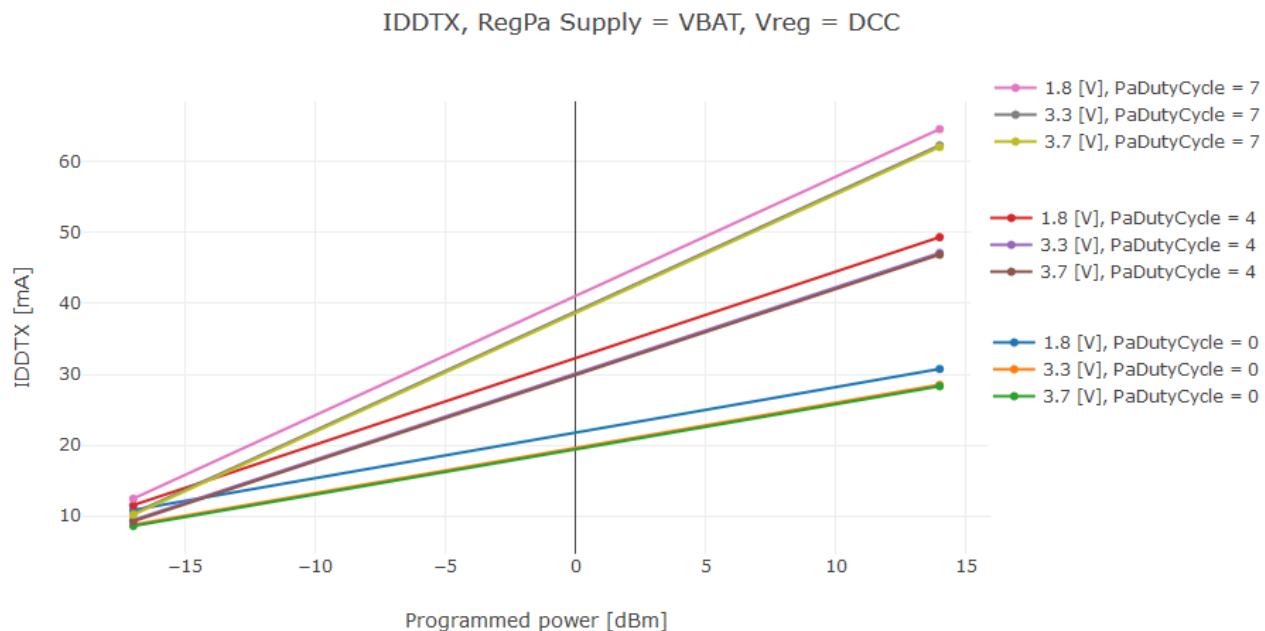


Figure 9-7: IDDTX vs TxPower, Low Power PA, DC-DC Configuration

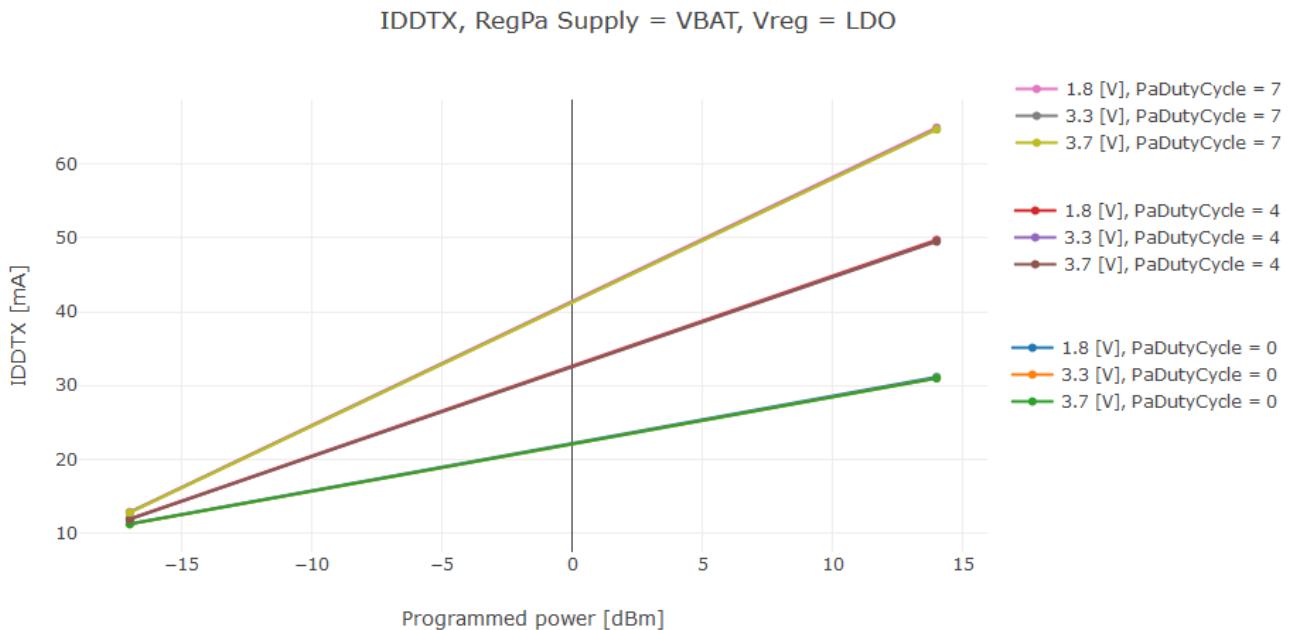


Figure 9-8: IDDTX vs TxPower, Low Power PA, LDO Configuration

[Figure 9-8: 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 9-7](#) and [Figure 9-8](#) 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 9-9: IDDTX vs TxPower, High Power PA, DC-DC Configuration](#) and [Figure 9-10: 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.

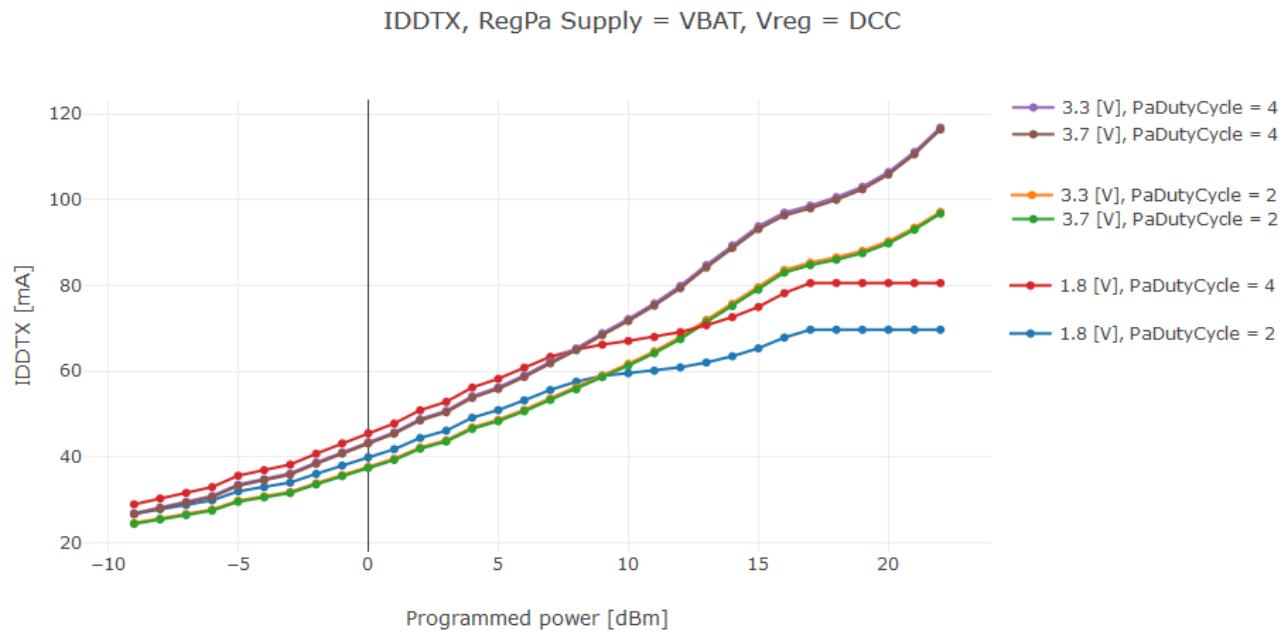


Figure 9-9: IDDTX vs TxPower, High Power PA, DC-DC Configuration

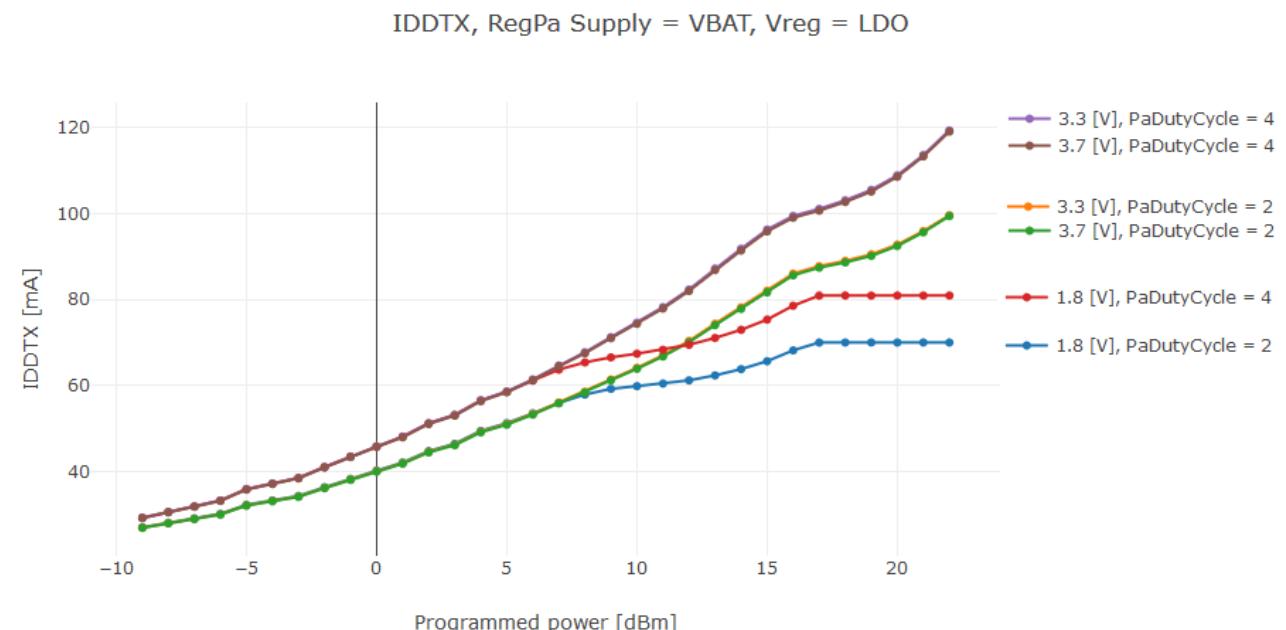


Figure 9-10: IDDTX vs TxPower, High Power PA, LDO Configuration

9.3.3 High Frequency PA

Figure 9-11 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.

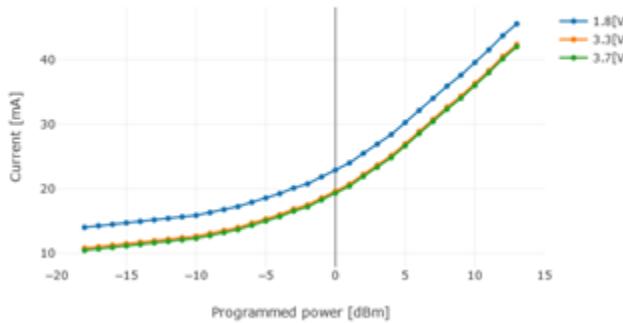


Figure 9-11: IDDTX vs TxPower, High Frequency PA, DC-DC Configuration

Figure 9-12: 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.

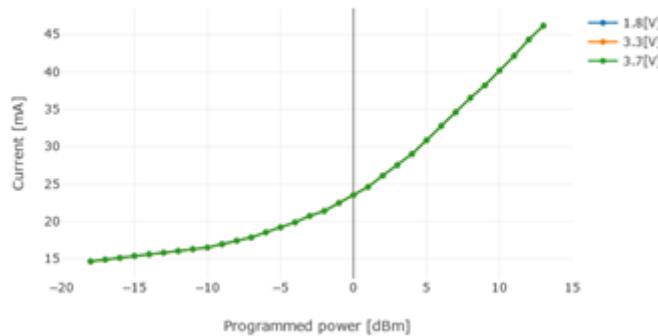


Figure 9-12: IDDTX vs TxPower, High Frequency PA, LDO Configuration

Figure 9-11 and Figure 9-12 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.

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](#) and [Table 9-2](#) 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 [Table 9-1](#) and [Table 9-2](#) 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 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 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 LR1110 itself (pins DIO5, DIO6, DIO7, DIO8 and DIO10), using the SetDioAsRfSwitch(. . .) command.

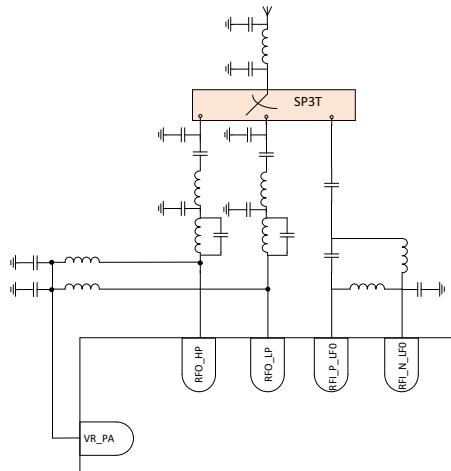


Figure 9-13: RF Switch, Double PA Operation

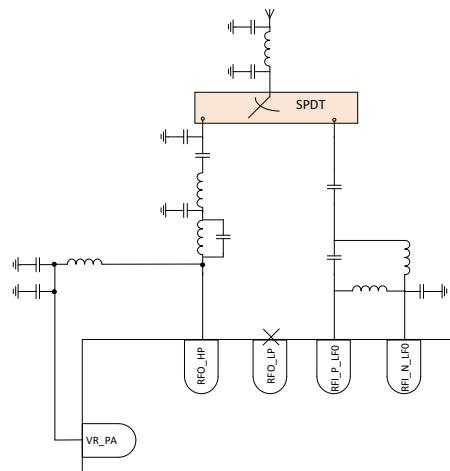


Figure 9-14: RF Switch, Single PA Operation (High Power PA Example)

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 LR1110 due to current flow in the RX stage.

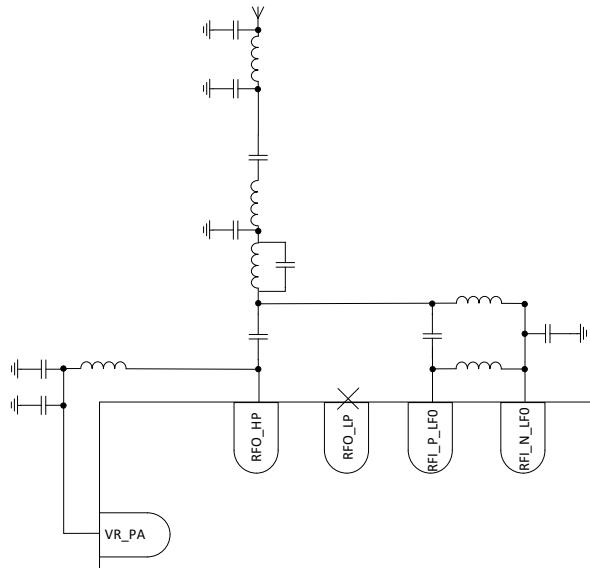


Figure 9-15: Single Tie Implementation: Only One PA Used (High Power PA Example)

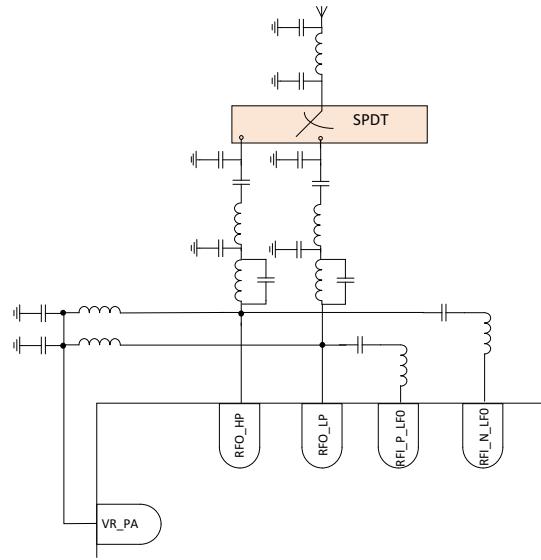


Figure 9-16: Single Tie Implementation: Both PAs Used (High Power PA Example)

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: DutyCycle Parameter Allowed Ranges

Low Power PA	High Power PA
For frequency \geq 400MHz, $0 \leq \text{PaDutyCycle} \leq 0x07$	For PaHpSel $\geq 0x05$, $0 \leq \text{PaDutyCycle} \leq 0x04$
For frequency $<$ 400MHz, $0 \leq \text{PaDutyCycle} \leq 0x04$	For PaHpSel $\leq 0x04$, $0 \leq \text{PaDutyCycle} \leq 0x07$

- *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. Wi-Fi Passive Scanning

The device provides device geolocation by scanning and processing of 802.11b/g/n Wi-Fi signals of opportunity in an energy efficient manner using three main concepts:

- Access Points MAC address extraction.
- Wi-Fi network SSID extraction.
- Country Code extraction.

The Wi-Fi types and Wi-Fi channels to scan are configurable:

- Wi-Fi types B and/or G/N:
 - When configuring the Wi-Fi signal to G/N, both 802.11g and 802.11n signals can be detected by the LR1110.
 - The device does not allow to scan for only 802.11g signals or only 802.11n signals.
- Wi-Fi channels:
 - From Channel ID 1 to Channel ID 14.

The maximal number of results to return is also configurable:

- From 1 to 32.
- This parameter can stop Wi-Fi passive scanning as soon as the required number of results is reached.

MAC addresses obtained during Wi-Fi scan can be resolved to a location using LoRa Cloud services or any other Wi-Fi solver provider.

10.1 Principle of Operation

The Wi-Fi passive scan entails the repetition of three steps:

- Preamble search
- Signal capture
- Signal demodulation

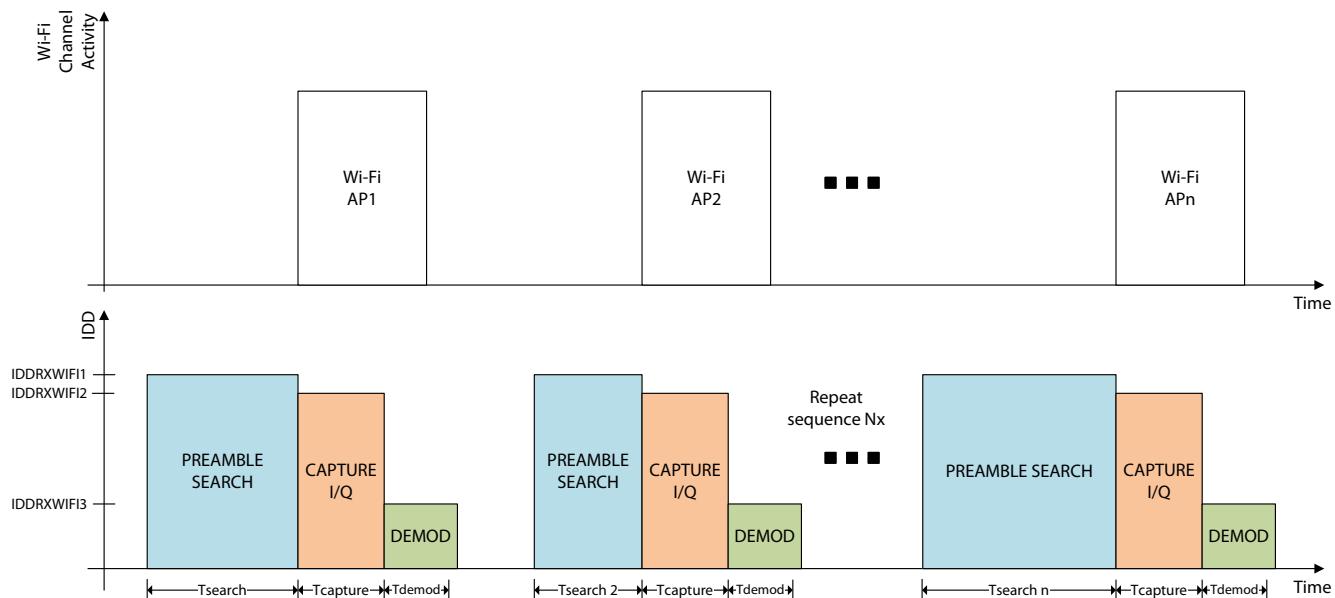


Figure 10-1: Wi-Fi Passive Scanning Sequence

10.1.1 Repetition Schemes

The device has two possible repetition schemes, which differ in the way they balance scan duration and power consumption:

- A maximal number of repetitions, used by `WifiScan` and `WifiCountryCode` commands:
 - Controls the maximal power consumed by the scan operation.
 - Configured by setting the maximal number of repetitions to execute per channel, per Wi-Fi type (`NbScanPerChan`).
- A maximal passive scan duration, used by `WifiScanTimeLimit` and `WifiCountryCodeTimeLimit` commands:
 - Controls the availability of the device for other tasks. Typically useful in the context of time slotted radio communications where a limited amount of time is available for Wi-Fi passive scanning before the next time slot.
 - Configured by setting the maximal duration of scan to spend per channel, per Wi-Fi type (`ScanTimePerChannel`).

10.1.2 Preamble Search Step

The duration of the preamble search step is configurable by the user. During this time, the radio is in reception mode configured for a Wi-Fi channel and a Wi-Fi signal type, waiting for a Wi-Fi preamble.

Note: When the LR1110 is configured to use a TCXO, the very first preamble search steps include the starting time of the TCXO. This happens only for the first preamble search as the TCXO is kept in running state until the end of the scanning.

If no Wi-Fi preamble has been detected by the radio during the configured duration, one of the following actions is automatically executed by the LR1110:

- Start another preamble search on the same Wi-Fi channel and same Wi-Fi type if:
 - Maximal number of repetitions has not been reached (for maximal number repetition) `NbScanPerChan`, or
 - Maximal duration has not been reached (for maximal duration repetition).
- Start another preamble search on next Wi-Fi channel, same Wi-Fi type if:
 - Maximal number of repetitions has been reached (for maximal number repetition), or
 - Maximal duration has been reached (for maximal duration repetition).
- Start preamble search on next Wi-Fi type, first channel, if:
 - All configured channels have been scanned for the current Wi-Fi type.
- Stop Wi-Fi scan if all channels have been scanned for all Wi-Fi types `NbMaxRes`.

When a Wi-Fi preamble is detected, the signal capture step is executed.

10.1.3 Signal Capture Step

This step is executed as soon as the preamble search steps detected a Wi-Fi preamble. During this step, part of the Wi-Fi signal is captured and stored, only part of the Wi-Fi signal is stored, in order to reduce the power consumption. There are two types of capture to define the length of the captured signal:

- Short capture:
 - Executed when the following acquisition modes are configured:
 - ◆ Beacon search mode (0x01)
 - ◆ Beacon and Packet search mode (0x02)
 - Reaches the lowest power consumption.
 - Capture duration is:
 - ◆ 500 microseconds in Wi-Fi type B
 - ◆ 100 microseconds in Wi-Fi type G/N
- Long capture:
 - Executed when:
 - ◆ Acquisition mode Full beacon search mode (0x04) is selected, or
 - ◆ WifiCountryCode or WifiCountryCodeTimeLimit is used.
 - Only available for Wi-Fi type B.
 - Extracts more information from the Wi-Fi signal, but increases power consumption
 - Capture time is 3 milliseconds

At the end of the signal capture step, the LR1110 automatically starts the signal demodulation step.

10.1.4 Signal Demodulation Step

During this step the device demodulates the captured Wi-Fi signal and extracts the result fields from it. The fields extracted depend on the acquisition mode.

For every captured signal, the extracted fields are stored in retention RAM memory.

The duration of this step depends on the capture type (short or long) and on the data rate of the signal received.

When this step terminates:

- If the configured number of results is not reached, the device automatically starts a new sequence of preamble search steps.
- If the configured number of results is reached, the device stops scanning.

10.2 Wi-Fi Commands

10.2.1 List of Wi-Fi Commands

Table 10-1: Summary Of Available Wi-Fi Commands

Command Name	Details
WifiScan	10.2.2 WifiScan
WifiScanTimeLimit	10.2.3 WifiScanTimeLimit
WifiCountryCode	10.2.4 WifiCountryCode
WifiCountryCodeTimeLimit	10.2.5 WifiCountryCodeTimeLimit
WifiGetNbResults	10.2.6 WifiGetNbResults
WifiReadResults	10.2.7 WifiReadResults
WifiResetCumulTimings	10.2.8 WifiResetCumulTimings
WifiReadCumulTimings	10.2.9 WifiReadCumulTimings
WifiGetNbCountryCodeResults	10.2.10 WifiGetNbCountryCodeResults
WifiReadCountryCodeResults	10.2.11 WifiReadCountryCodeResults
WifiCfgTimestampAPphone	10.2.12 WifiCfgTimestampAPphone
WifiReadVersion	10.2.13 WifiReadVersion

10.2.2 WifiScan

Captures Wi-Fi packets on the RFIO_HF pin. During Wi-Fi passive scanning, the BUSY signal is set high, indicating that the LR1110 is not ready to accept a command from the host. This can take a few hundred milliseconds, depending on the Wi-Fi passive scanning parameters. The BUSY signal returns to low when the Wi-Fi passive scanning procedure is complete. If the *WifiScanDone* interrupt has been enabled, the IRQ signal goes high at the end of the Wi-Fi passive scanning process on the given channel mask for the given signal type.

Table 10-2: WifiScan Command

Byte	0	1	2	3	4	5	6	7	8	9	10
Data from Host	0x03	0x00	Signal Type	ChanMask (15:8)	ChanMask (7:0)	Acq Mode	NbMax Res	NbScan PerChan	Timeout (15:8)	Timeout (7:0)	AbortOn Timeout
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0	0	0	0

- *SignalType* defines the type of the 802.11 signal to be scanned:
 - ◆ 0x01: Wi-Fi 802.11b type
 - ◆ 0x02: Wi-Fi 802.11g type
 - ◆ 0x03: Wi-Fi 802.11n type
 - ◆ 0x04: All signals: Wi-Fi b, then Wi-Fi g/n on the same channel
 - ◆ 0x05-0xFF: RFU
- *ChanMask* defines which Wi-Fi channels to be scanned:
 - ◆ [0 0 Ch14 Ch13 Ch12 Ch11 Ch10 Ch9 Ch8 Ch7 Ch6 Ch5 Ch4 Ch3 Ch2 Ch1]
 - ◆ A channel bit set to 1 indicates that this channel must be scanned
- *AcqMode* indicates the *WifiScan* acquisition mode:
 - ◆ 0x01: Beacon search mode. Use only the Wi-Fi beacons to extract the MAC addresses.
 - ◆ 0x02: Beacon and Packet search mode. Use both the Wi-Fi beacons and Wi-Fi data packets to extract the MAC addresses.
 - ◆ 0x03: Full traffic mode. Recover all type/subtype packets. Same capture as AcquisitionMode 0x01 and 0x02.
 - ◆ 0x04: Full beacon mode. Scan for Beacons and Probe responses until Frame Check Sequence (FCS) field.
 - ◆ 0x05: SSID Beacon search mode. Capture and demodulate until the end of maximum possible SSIDs of a beacon or probe response. Fields until SSID are extracted and written in full results. Supported for Wi-Fi 802.11 b/g types only.
 - ◆ Other values are RFU.
- *NbMaxRes*: Maximum number of different MAC addresses allowed as a result for all scans on various channels and Wi-Fi types (must be inferior or equal to 32). If this number is reached, passive scanning is stopped. If a MAC address already present in the result structure is detected a second time with a different RSSI value, then the new result is ignored.
- *NbScanPerChan*: Number of Wi-Fi passive scans to be executed per channel (range: 1 to 255). If *NbMaxRes* results are found, scan is stopped.
- *Timeout*: 16-bit timeout of the Preamble Search mode in ms. Range: 1: 65535ms. For example, for a beacon period of 102.4ms, a 105ms timeout value can be set to ensure the *WifiScan* covers the whole beacon period.
- *AbortOnTimeout*: If set to 1, when a timeout preamble detect occurs, the passive scanning on this channel is aborted and the device jumps to the other channel to scan.

For example, the configuration (Scan Wi-Fi b / Channels 1, 6, 11 / Beacon and Packet search mode / 10 Maximum Results / 6 scans per channel / 70 ms timeout for Preamble Search mode / No abort on timeout) is coded as:

Opcode	SignalType	ChanMask	AcqMode	NbMaxRes	NbScanPerChan	Timeout	AbortOnTimeout
0x0300	0x01	0x0421	0x02	0x0A	0x06	0x0046	0x00

10.2.3 WifiScanTimeLimit

This API searches for Wi-Fi MAC addresses during a configurable maximal amount of time. The API is as follows:

Table 10-3: WifiScanTimeLimit Command

Byte	0	1	2	3	4	5	6	7	8	9	10
Data from Host	0x03	0x01	Signal Type	ChanMask (15:8)	ChanMask (7:0)	Acq Mode	NbMax Res	ScanTime PerChannel (15:8)	ScanTime PerChannel (7:0)	Timeout PerScan (15:8)	Timeout PerScan (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00	0x00	0x00

- *SignalType*: See definition in [10.2.2 WifiScan](#)
- *ChanMask*: See definition in [10.2.2 WifiScan](#)
- *AcqMode*: See definition in [10.2.2 WifiScan](#)
- *NbMaxRes*: See definition in [10.2.2 WifiScan](#)
- *ScanTimePerChannel*: Maximal duration of a scan on a single channel. Expressed in ms. Possible values go from 1ms to 65535ms included.
- *TimeoutPerScan*: Maximal duration of the wait for preamble. Expressed in ms. Possible values go from 0ms to 65535ms. Each time a preamble is detected and a capture is done, the counter of this timeout is restarted for the next scan. If set to 0ms, the LR1110 stays in preamble search for a maximal duration defined by *ScanTimePerChannel*.

The duration of a scan is (number of channels to scan) x (configured *ScanTimePerChannel*). However, this duration may be exceeded depending on the crystal drift of the clock source and on the instant the last Wi-Fi signal is detected by the device. Therefore the maximal duration of a Wi-Fi scan with this API is provided by the following equation:

$T_{max} = N_{channel} \times ((1 + Xtal_precision) \text{ Timeout per Channel} + T_{offset})$, where:

- *Xtal_precision* depends on the clock source crystal. If clock source is 32kHz internal RC, then *Xtal_precision* = 1/100
- *T_offset* depends on the *SignalType* and the *AcqMode* selected for Wi-Fi B:
 - ♦ Wi-Fi b:
 - ◆ if Acquisition Mode != LR1110_WIFI_SCAN_MODE_FULL_BEACON: 2.31ms
 - ◆ if Acquisition Mode == LR1110_WIFI_SCAN_MODE_FULL_BEACON: 9.59ms
 - ♦ Wi-Fi g: 52.55ms

10.2.4 WifiCountryCode

This is the main function to extract Country code from Beacon or Probe Response. Only Wi-Fi b signals are searched.

Country code results are filtered to not have duplicates by comparing MAC addresses associated with the country code.

The number of parameters is tested and the range of each parameter is tested. If a range is not respected or a parameter is missing, a CMD_PERR status is returned. If CMD_FAIL is returned, it's due to radio configuration errors.

Table 10-4: WifiCountryCode Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x03	0x02	ChanMask (15:8)	ChanMask (7:0)	NbMaxRes	NbScan PerChannel	Timeout (15:8)	Timeout (7:0)	AbortOn Timeout
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0	0

- *ChanMask*: Mask of channels to scan. 14 bits from LSB:
 - [0 0 ch14 ch13 ch12 ch11 ch10 ch9 ch8 ch7 ch6 ch5 ch4 ch3 ch2 ch1]
 - Channel bit at 1 indicates that this channel must be scanned
- *NbMaxRes*: See definition in [10.2.2 WifiScan](#)
- *NbScanPerChannel*: See definition in [10.2.2 WifiScan](#)
- *Timeout*: See definition in [10.2.2 WifiScan](#)
- *AbortOnTimeout*: See definition in [10.2.2 WifiScan](#)

Example: [Table 10-5: WifiCountryCode Example](#) shows a search for:

- Country code channels 1, 6 and 11
- For a maximum of 20 results
- 15 scans per channel
- 105ms timeout for preamble detection phase
- Don't abort if timeout on preamble detection is triggered

Table 10-5: WifiCountryCode Example

Opcode	ChanMask	NbMaxRes	NbScanPerChannel	TimeoutPerScan	AbortIfTimeout
0x03 0x02	0x04 0x21	0x14	0x0F	0x00 0x46	0x00

10.2.5 WifiCountryCodeTimeLimit

This API searches for Wi-Fi MAC addresses during a configurable maximal amount of time. The API is as follows:

Table 10-6: WifiCountryCodeTimeLimit Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x03	0x03	ChanMask (15:8)	ChanMask (7:0)	NbMaxRes (7:0)	ScanTime PerChannel (15:8)	ScanTime PerChannel (7:0)	Timeout PerScan (15:0)	Timeout PerScan (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

- *ChanMask*: See definition in [10.2.2 WifiScan](#)
- *NbMaxRes*: See definition in [10.2.2 WifiScan](#)
- *ScanTimePerChannel*: See definition in [10.2.3 WifiScanTimeLimit](#)
- *TimeoutPerScan*: Maximal duration of the wait for preamble. Expressed in ms. Possible values go from 0ms to $2^{16} - 1$ ms (or 65535). Each time a preamble is detected and a capture is done, the counter of this timeout is restarted for the next scan. If set to 0ms, the LR1110 stays in preamble search for a maximal duration defined by *ScanTimePerChannel*.

The maximal duration of a scan is determined by the number of channels to scan, times the *ScanTimePerChannel* configured. However, this duration may be exceeded depending on the crystal drift of the clock source and on the instant the last Wi-Fi signal is detected by the device. Therefore the maximal duration of a Wi-Fi scan with this API is provided by the following equation:

$T_{max} = N_{channel} \times ((1 + Xtal_precision) \text{ Timeout per Channel} + T_{offset})$, where:

- *Xtal_precision* depends on the clock source crystal. If clock source is 32kHz internal RC, then *Xtal_precision* = 1/100.
- *T_offset* is always 9.59ms.

10.2.6 WifiGetNbResults

Gives the number of Wi-Fi Scanning results.

The number of results is returned on 8 bits and can be read at the next SPI transaction.

Table 10-7: WifiGetNbResults Command

Byte	0	1
Data from Host	0x03	0x05
Data to Host	Stat1	Stat2

Table 10-8: WifiGetNbResults Response

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

10.2.7 WifiReadResults

Reads the byte stream containing a defined number of Wi-Fi Passive Scanning results from a given index, in the requested format. It is necessary to issue the command `WifiGetNbResults(...)` before this command. NOP bytes (0x00) must be issued to read back the results.

Table 10-9: WifiReadResults Command

Byte	0	1	2	3	4
Data from Host	0x03	0x06	Index	NbResults	Format
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)

Table 10-10: WifiReadResults Response

Byte	0	1	2	...	N+1
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	ResultsByte0	ResultsByte1	...	ResultsByteN

- *Index*: Index of Wi-Fi Passive Scanning results to read, from 0 to 31.
- *NbResults*: Number of Wi-Fi AP MAC Addresses to read, from 1 to 32.
- *Format*: Format of the Wi-Fi Passive Scanning results to read, depending on the previous acquisition mode (see [Table 10-23](#)):
 - 1: Basic Complete results format
 - 4: Basic MAC/Type/Channel Results format
 - Other values are RFU

For example, in order to read the results of a Wi-Fi Passive Scanning that returned 6 MAC Addresses in Basic MAC/Type/Channel Results format, the user shall send:

Table 10-11: Example to Read Basic Results of Passive Scan

Opcode	Index	NbResults	Format
0x0306	0x00	0x06	0x04

The result data is sent in a stream of $6 \times 9 = 54$ bytes.

The maximum number of bytes that can be read from one `WifiReadResults(...)` command is 1020 bytes. Therefore if the size to read is greater than 1020 bytes, the read operation must be separated into two requests.

10.2.8 WifiResetCumulTimings

Resets the Wi-Fi Passive Scanning cumulative timings (refer to [10.2.9 WifiReadCumulTimings](#)).

This command must be called prior to the executing the Wi-Fi Passive Scanning, in order to initialize the Wi-Fi Passive Scanning cumulative timings if those are to be read.

Table 10-12: WifiResetCumulTimings Command

Byte	0	1
Data from Host	0x03	0x07
Data to Host	Stat1	Stat2

10.2.9 WifiReadCumulTimings

Reads the Wi-Fi Passive Scanning cumulative timings, coded on 16 bytes, coded as in [Table 10-13: Wi-Fi Cumulative Timings Description](#). The Cumulative Timing represents the total time in the various modes during a `WifiScan(...)` command, therefore summed up for all Wi-Fi acquisitions, over the different `WifiScan(...)` parameters (Wi-Fi Types, Wi-Fi channels and so on). These timings are expressed in microseconds.

Table 10-13: Wi-Fi Cumulative Timings Description

Byte	0:3	4:7	8:11	12:15
Meaning	RFU	Total duration in preamble detection mode	Total duration in capture mode	Total duration in demodulation mode

This cumulative timing can be read regularly to compute the energy consumption of the device for Wi-Fi Passive Scanning operations. All 16 bytes shall be read. Cumulative timing must be reset by the host.

Table 10-14: WifiReadCumulTimings Command

Byte	0	1
Data from Host	0x03	0x08
Data to Host	Stat1	Stat2

Table 10-15: WifiReadCumulTimings Response

Byte	0	1	2	...	17
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	Byte 0	Byte 1	...	Last Byte

10.2.10 WifiGetNbCountryCodeResults

Returns the number of results after Country Code scanning execution by WifiCountryCode or WifiCountryCodeTimeLimit commands.

Table 10-16: WifiGetNbCountryCodeResults Command

Byte	0	1
Data from Host	0x03	0x09
Data to Host	Stat1	Stat2

Table 10-17: WifiGetNbCountryCodeResults Response

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

10.2.11 WifiReadCountryCodeResults

Reads the byte stream containing a defined number of Wi-Fi Passive Scanning Country Code results from a given index. It is necessary to issue the command WifiGetNbCountryCodeResults (...) before this command.

NOP bytes (0x00) must be issued to read back the results. The size of one Country Code result is 10 bytes.

Table 10-18: WifiReadCountryCodeResults Command

Byte	0	1	2	3
Data from Host	0x03	0xA	Index	NbResults
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

Table 10-19: WifiReadCountryCodeResults Response

Byte	0	1	2	---	N+1
Data from Host	0x00	0x00	0x00		0x00
Data to Host	Stat1	Byte0Res	Byte1Res		LastByteRes

- Index:* Index of the Wi-Fi Passive Scanning Country Code result to start reading from, from 0 to 31.
- NbResults:* Number of Wi-Fi Passive Scanning Country Code results to read, from 1 to 32.

Refer to [10.3.5 WifiCountryCode Result Format](#) for details concerning interpretation of Wi-Fi Passive Scanning Country Code results.

10.2.12 WifiCfgTimestampAPhone

Configures a timestamp threshold used to discriminate mobile access point from gateways.

Table 10-20: WifiCfgTimestampAPPhone Command

Byte	0	1	2	3	4	5
Data from Host	0x03	0x0B	TimeStamp (31:24)	TimeStamp (23:16)	TimeStamp (15:8)	TimeStamp (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *TimeStamp*: Threshold timestamp value expressed in seconds. Default is 1 day. If the timestamp extracted from a beacon or probe response is greater than this limit, the field MAC validation from Channel information field of results indicates that the frame is probably from a gateway and not from a mobile device access point.

10.2.13 WifiReadVersion

Returns the internal Wi-Fi firmware version major and minor numbers.

Table 10-21: WifiReadVersion Command

Byte	0	1
Data from Host	0x03	0x20
Data to Host	Stat1	Stat2

Table 10-22: WifiReadVersion Response

Byte	0	1	2
Data from Host	0x00	0x00	0x00
Data to Host	Stat1	VersionMajor	VersionMinor

10.3 Wi-Fi Results Formats

10.3.1 Wi-Fi Passive Scanning Result Formats

Various formats of Wi-Fi Passive Scanning Results are implemented, allowing the user to retrieve either:

- Basic MAC/Type/Channel results: The minimum set of geolocation information in order to optimize the application power consumption (Basic MAC/Type/Channel results format),
- Basic Complete results: The maximum amount of information available for the `WifiScan(. . .)` operation (Basic Complete results format).
- Extended Complete results: Basic Complete results with additional Information Elements.

Table 10-23: Wi-Fi Result Formats and Wi-Fi Scan Mode Relationship

Acquisition Mode value	Result Format value	Result format to read
0x01	0x01	Full Result (22 bytes per MAC address detected)
0x01	0x04	Basic MAC/Type/Channel Result (9 bytes per MAC address detected)
0x02	0x01	Full Result (22 bytes per MAC address detected)
0x02	0x04	Basic MAC/Type/Channel Result (9 bytes per MAC address detected)
0x04	0x01	Extended Complete Result (79 bytes per MAC address detected)

Other values are RFU.

10.3.2 Basic MAC/Type/Channel Result Format

The Basic MAC/Type/Channel Result Format is the result format returned by the LR1110 when reading results with format 0x04 after executing a Wi-Fi scan with acquisition mode 0x01 or 0x02.

This result structure is organized as a continuous series of MAC Address Basic MAC/Type/Channel Results, each coded on 9 bytes, defined in [Table 10-24: Basic Results Format per MAC Address](#) below. The maximum number of MAC Addresses reported is 32.

Table 10-24: Basic Results Format per MAC Address

Byte	0	1	2	3	4	5	6	7	8
Content	WifiType	ChannelInfo	RSSI	MAC6	MAC5	MAC4	MAC3	MAC2	MAC1

- *WifiType*: Coded on 8 bits:
 - ◆ Bits 0-1: Wi-Fi signal type:
 - ◆ 1: Wi-Fi b
 - ◆ 2: Wi-Fi g
 - ◆ 3: Wi-Fi n
 - ◆ Bits 2-7: *DatarateID*, coded as indicated in [Table 10-26: Wi-Fi DatarateID Field](#).
- *ChannelInfo*: coded on 8 bits:
 - ◆ Bits 0-3: *ChannelID*, coded as indicated in [Table 10-27: Wi-Fi ChannelID Field](#). *ChannelID* indicates the Wi-Fi channels configured for the scan.
 - ◆ Bits 4-5: *MacOrigin*, coded as indicated in [Table 10-28: Wi-Fi MacOrigin Field](#). *MacOrigin* indicates if the MAC address belongs to a gateway, to a phone, or if it is undetermined because MAC address is extracted from a packet.
 - ◆ Bit 6: *RssiValidation*: indicates if the signal comes from an access point (therefore reliable for localization) or from an end device:
 - ◆ 0: Access point.
 - ◆ 1: End device.
 - ◆ Bit 7: Reserved.
- *RSSI*: RSSI value of the signal captured, coded on 8 bits.
- *MAC*: MAC address of the Access Point, coded on 6 bytes:
 - ◆ MAC6:MAC5:MAC4:MAC3:MAC2:MAC1, from MSB to LSB.

10.3.3 Full Result Format

The Full Result Format is the result format returned by the LR1110 when reading results with format 0x01 after executing a Wi-Fi scan with acquisition mode 0x01 or 0x02.

The full result structure is organized as continuous series of MAC Address Basic Complete results, each MAC Address Full Result being coded on 22 bytes, defined in [Table 10-25: Basic Complete Results Format per MAC Address](#) below. The maximum number of MAC Addresses reported is 32.

Table 10-25: Basic Complete Results Format per MAC Address

Byte	0	1	2	3	4	5	6	7
Content	WifiType	Channel Info	RSSI	FrameCtrl	MAC6	MAC5	MAC4	MAC3
Byte	8	9	10	11	12	13	14	15
Content	MAC2	MAC1	PhiOffset (15:8)	PhiOffset (7:0)	Timestamp (63:56)	Timestamp (55:48)	Timestamp (47:40)	Timestamp (39:32)
Byte	16	17	18	19	20	21		
Content	Timestamp (31:24)	Timestamp (23:16)	Timestamp (15:8)	Timestamp (7:0)	PeriodBeacon (15:8)	PeriodBeacon (8:0)		

- *WifiType*: Coded on 8 bits:
 - ◆ Bits 0-1: Wi-Fi signal type:
 - ◆ 1: Wi-Fi b
 - ◆ 2: Wi-Fi g
 - ◆ 3: Wi-Fi n
 - ◆ Bits 2-7: *DatarateID*, coded as indicated in [Table 10-26: Wi-Fi DatarateID Field](#).
- *ChannelInfo*: coded on 8 bits:
 - ◆ Bits 0-3: *ChannelID*, coded as indicated in [Table 10-27: Wi-Fi ChannelID Field](#).
 - ◆ Bits 4-5: *MacOrigin*, coded as indicated in [Table 10-28: Wi-Fi MacOrigin Field](#).
 - ◆ Bit 6: *RssiValidation*, indicates if the signal comes from an access point (therefore reliable for localization) or from an end device:
 - ◆ 0: Access point
 - ◆ 1: End device
 - ◆ Bit 7: Reserved.
- *RSSI*: RSSI value of the signal captured, coded on 8 bits.
- *FrameCtrl*: 16-bit frame control, coded as indicated in [Table 10-29: Wi-Fi FrameCtl Field](#).
- *MAC*: MAC address of the Access Point, coded on 6 bytes:
 - ◆ MAC6:MAC5:MAC4:MAC3:MAC2:MAC1, from MSB to LSB.
- *PhiOffset*: Coded on 2 bytes. Used to compute frequency offset of the signal.
- *Timestamp*: Indicates the number of microseconds the AP is active, coded on 64 bits.
- *PeriodBeacon*: Beacon period expressed in Time Units. Must be multiplied for the time in ms.

Table 10-26: Wi-Fi DatarateID Field

DatarateID	Signal type	Modulation	Coding rate	Datarate (Mbps)
1		DBPSK		1
2	Wi-Fi b	DQPSK		2
3		BPSK	1/2	6
4		BPSK	3/4	9
5		QPSK	1/2	12
6	Wi-Fi g	QPSK	3/4	18
7		16-QAM	1/2	24
8		16-QAM	3/4	36
11		BPSK	1/2	6.5
12		QPSK	1/2	13
13		QPSK	3/4	19.5
14		16-QAM	1/2	26
15	Wi-Fi n	16-QAM	3/4	39
19	mixed mode	BPSK	1/2	7.2
20		QPSK	1/2	14.4
21		QPSK	3/4	21.7
22		16-QAM	1/2	28.9
23		16-QAM	3/4	43.3

Table 10-27: Wi-Fi ChannelID Field

Channel ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Center Freq (MHz)	2412	2417	2422	2427	2432	2437	2442	2447	2452	2457	2462	2467	2472	2484

Table 10-28: Wi-Fi MacOrigin Field

MacOrigin Value	Meaning
1	MAC Address from a gateway
2	MAC Address from a phone
3	Undetermined

Table 10-29: Wi-Fi FrameCtl Field

Bit	(0:1)	(2:5)	6	7
Fields	Type	SubType	ToDS	FromDS

The *FrameCtl* field is transmitted LSB first, and therefore represented accordingly.

- *Type*: 00: Management Frame; 01: Control Frame; 10: Data Frame; 11: Reserved.
- *SubType*: Coded as indicated in [Table 10-30: FrameCtl SubType Values](#).
- *ToDS*: 1 indicates that the data frame is going from the client station (STA) to the Distribution System (DS).
- *FromDS*: 1 indicates that the data frame is going from the Distribution System (DS) to the client Station (STA).

Table 10-30: FrameCtl SubType Values

Type	Subtype Value	SubType	Type	Subtype Value	SubType	Type	Subtype Value	SubType
00	0000	Assoc req	01	0000	Reserved	10	0000	Data
	0001	Assoc res		0001	Reserved		0001	Data +CF-ACK
	0010	Reassoc req		0010	Reserved		0010	Data +CF-Poll
	0011	Reassoc res		0011	Reserved		0011	DATA+CF-ACK/Poll
	0100	Probe Req		0100	Reserved		0100	Null
	0101	Probe res		0101	Reserved		0101	CF-ACK
	0110	Reserved		0110	Reserved		0110	CF-Poll
	0111	Reserved		0111	Reserved		0111	CF-ACK /Poll
	1000	Beacon		1000	Block ACK Req		1000	Qos Data
	1001	Announcement		1001	Block Acq		1001	Qos + CF-ACK
	1010	Diassoc		1010	PS-Poll		1010	Qos + CF-Poll
	1011	Auth		1011	RTS		1011	Qos + CF-ACL/Poll
	1100	Deauth		1100	CTS		1100	Qos Null
	1101	Action		1101	ACK		1101	Reserved
	1110	Reserved		1110	CF-End		1110	Qos + CF-Poll
	1111	Reserved		1111	CF-END +CF-ACK		1111	Qos + CF-ACK

10.3.4 Extended Complete Result Format

The Extended Complete Result Format is the result format returned by the LR1110 when reading results with format 0x01 after executing a Wi-Fi scan with acquisition mode 0x04.

Table 10-31: Extended Basic Complete Results Format per MAC Address

Byte	0	1	2	3	4	5	6	7
Content	WifiType	Channel Info	RSSI	Rate	Service (15:8)	Service (7:0)	Length (15:8)	Length (7:0)
Byte	8	9	10	11	12	13	14	15
Content	FrameCtrl (15:8)	FrameCtrl (7:0)	MAC_5_0	MAC_4_0	MAC_3_0	MAC_2_0	MAC_1_0	MAC_0_0
Byte	16	17	18	19	20	21	22	23
Content	MAC_5_1	MAC_4_1	MAC_3_1	MAC_2_1	MAC_1_1	MAC_0_1	MAC_5_1	MAC_4_1
Byte	24	25	26	27	28	29	30	31
Content	MAC_3_1	MAC_2_1	MAC_1_1	MAC_0_1	Timestamp (63:56)	Timestamp (55:48)	Timestamp (47:40)	Timestamp (39:32)
Byte	32	33	34	35	36	37	38	39
Content	Timestamp (31:24)	Timestamp (23:16)	Timestamp (15:8)	Timestamp (7:0)	Period Beacon (15:8)	Period Beacon (7:0)	SeqCtrl [15:8]	SeqCtrl [7:0]
Byte	40	41	42	43	44	70
Content	SSID[0]	SSID[1]	SSID[2]	SSID[3]	SSID[4]	SSID[30]
Byte	71	72	73	74	75	76	77	78
Content	SSID[31]	Current Channel	Country Code[0]	Country Code[1]	IOReg	FCS CheckOk	PhiOffset (15:8)	PhiOffset (7:0)

- *WifiType*: (8 bits):
 - ◆ Bits 0-1: Wi-Fi signal type:
 - ◆ 1: Wi-Fi b
 - ◆ 2: Wi-Fi g
 - ◆ 3: Wi-Fi n
 - ◆ Bits 2-7: *DatarateID*, coded as indicated in [Table 10-26: Wi-Fi DatarateID Field](#)
- *ChannelInfo*: (8 bits):
 - ◆ Bits 0-3: *ChannelID*, coded as indicated in [Table 10-27: Wi-Fi ChannelID Field](#)
 - ◆ Bits 4-5: *MacOrigin*, coded as indicated in [Table 10-28: Wi-Fi MacOrigin Field](#)
 - ◆ Bit 6: *RssiValidation*, indicates if the signal comes from an access point (therefore reliable for localization) or from an end device:
 - ◆ 0: Access point
 - ◆ 1: End device
 - ◆ Bit 7: Reserved
- *RSSI*: (8 bits) RSSI value of the signal captured.
- *Rate*: (16 bits) the meaning depends on Wi-Fi type:
 - ◆ Wi-Fi b (data rate)
 - ◆ 0x0A: 1Mb/s
 - ◆ 0x14: 2Mb/s
 - ◆ Wi-Fi g (data rate)
 - ◆ 0x0D: 6Mb/s
 - ◆ 0x0F: 9Mb/s
 - ◆ 0x05: 12Mb/s
 - ◆ 0x07: 18Mb/s
 - ◆ 0x09: 24Mb/s
 - ◆ 0x0B: 36Mb/s
 - ◆ 0x01: 48Mb/s
 - ◆ 0x03: 54Mb/s
 - ◆ Wi-Fi n: Modulation and Coding Scheme index (from 0 to 7)
- *Service*: Refer to IEEE Std 802.11, 2016, Part 11: Wireless LAN MAC and PHY Spec.
- *Length*: Refer to IEEE Std 802.11, 2016, Part 11: Wireless LAN MAC and PHY Spec.
- *FrameCtl*: (16 bits) Frame control, coded as indicated in [Table 10-29: Wi-Fi FrameCtl Field](#).
- *MACx_Y*: The x byte of the Y MAC Address.
- *Timestamp*: Indicates the number of microseconds the AP is active, coded on 64 bits.
- *PeriodBeacon* (2 bytes): The beacon period expressed in Time Unit (TU). 1 TU is 1024 microseconds.
- *SqCntr*: Refer to IEEE Std 802.11, 2016, Part 11: Wireless LAN MAC and PHY Spec.
- *SSID* 0 - 31 (32 bytes): Service Set IDentifier
- *CurrentChannel*: (1 byte) Current channel of the Wi-Fi frame as reported by the frame, in decimal.
- *CountryCode* (2 bytes).
- *IOReg*: Refer to IEEE Std 802.11, 2016, Part 11: Wireless LAN MAC and PHY Spec.
- *FCSCheckOk*: Indicates if the FCS of the frame has been checked and the check is OK.
- *PhiOffset*: Coded on 2 bytes. Used to compute frequency offset of the signal.

10.3.5 WifiCountryCode Result Format

Table 10-32: WifiCountryCode Result Format Sent Over SPI (12 bytes)

Byte	0	1	2	3	4	5	6	7	8	9
Content	Country Code[0]	Country Code[1]	IOReg	Channel Info	MAC6	MAC5	MAC4	MAC3	MAC2	MAC1

- *CountryCode*: 2 char encoded in ASCII that represents the country code:
 - ◆ Ex: 'FR' -> France, 'US' -> United States
- *IOReg*: One character encoded in ASCII that represents if the AP is indoor or outdoor or anywhere:
 - ◆ Ex: 'I' -> Indoor, 'O' -> Outdoor, 'Space (ASCII)' -> Any
- *ChannelInfo*: coded on 8 bits:
 - ◆ Bits 0-3: *ChannelID*, coded as indicated in [Table 10-27: Wi-Fi ChannelID Field](#). *ChannelID* indicates the Wi-Fi channels configured for the scan.
 - ◆ Bits 4-5: *MacOrigin*, coded as indicated in [Table 10-28: Wi-Fi MacOrigin Field](#). *MacOrigin* indicates if the MAC address belongs to a gateway, to a phone, or if it is undetermined because MAC address is extracted from a packet.
 - ◆ Bit 6: *RssiValidation*: indicates if the signal comes from an access point (therefore reliable for localization) or from an end device:
 - ◆ 0: Access point.
 - ◆ 1: End device.
 - ◆ Bit 7: Reserved.
- *MAC6-1*: MAC AP associated results.

11. GNSS Scanning

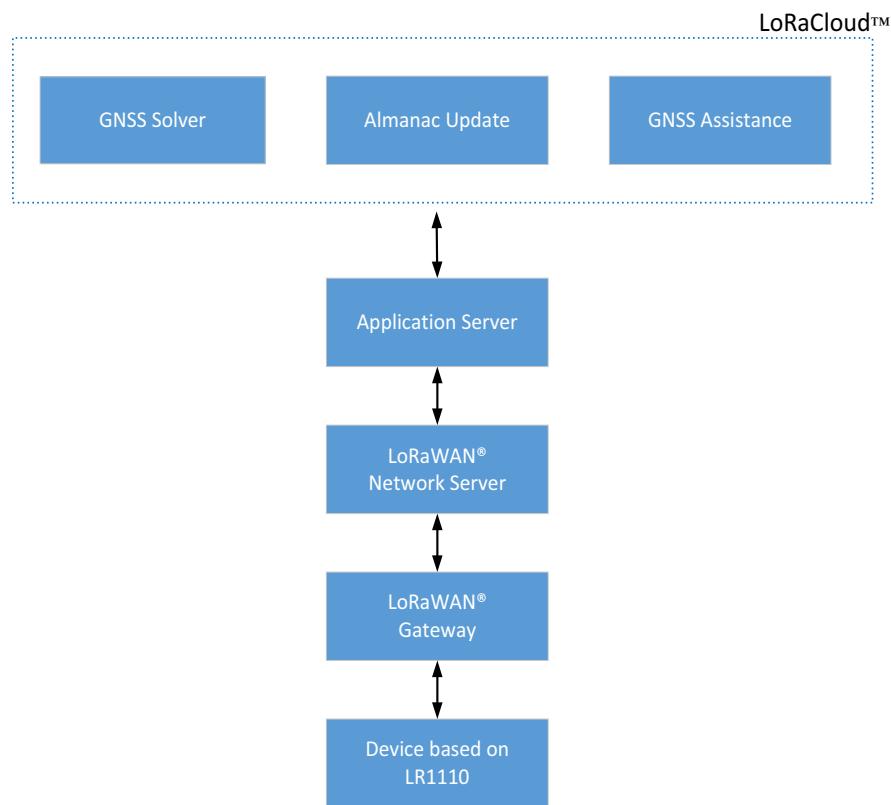


Figure 11-1: GNSS System Overview

11.1 GNSS Geolocation System Overview

The LR1110 features a GNSS receiver that allows a fast and energy efficient outdoor geolocation. The LR1110 GNSS Geolocation System achieves low energy geolocation by offloading time- and compute-intensive operations to back-end system components. In particular, the following three back-end system components are needed to operate the LR1110's GNSS Geolocation System:

- **GNSS Position Solving Component:** The LR1110 does not resolve the full position on-device. Instead, the measurements from GNSS signals are combined into a binary message (the NAV message) and expected to be sent via any communication channel to the GNSS Position Solver back-end component for final position calculation. This GNSS position solving component is required in all operation modes.
- **GNSS Almanac Update Component (required in assisted mode):** The LR1110 reduces the GNSS scanning time by taking into account coarse orbital parameters for different GNSS constellations (the Almanac parameters). In conjunction with a coarse time and position estimate, the LR1110 uses this information to optimize the search and acquisition of GNSS signals. Over time, the true satellite positions diverge from the fixed Almanac parameters, which requires them to be updated. This can be achieved by a back-end component which estimates the quality of the almanac image on device and issues updates when needed.

- GNSS Assistance Component (required in assisted mode): In order to operate the GNSS Geolocation System in assisted mode, coarse estimates of time and position must be provided to the LR1110. This information can be obtained in a variety of ways including application-level knowledge. In LoRaWAN® the Application Layer Clock Synchronization protocol can retrieve assistance time information. The assistance position information can generally be derived from past position solutions.

LoRa Cloud™ offers these components in a single, easy to use, managed service as part of the Modem & Geolocation Services (MGS). Visit www.Loracloud.com for more information.

[Figure 11-1: GNSS System Overview](#) shows the system components for a LoRaWAN® -enabled integration. In GNSS mode, the LR1110 searches for available GNSS signals and extracts the minimum set of satellite information needed for a position calculation. The GNSS satellite signal data (also referred as NAV message) is then transmitted via the LPWAN communication stack to a GNSS solver for the geolocation position calculation. If an update to the almanac parameters is needed, the Almanac Update Component schedules appropriate downlink messages.

The LoRa Cloud solver allows both 2D and 3D solving. For a successful 3D solve, a minimum of 5 satellites is required for a Single constellation scan, and 6 satellites for a Double constellation scan. For a 2D solve, the minimum is 4 satellites in Single constellation, or 5 satellites in Double constellation. Moreover, a Double constellation scan requires at least two satellites in each constellation (for example, solving a scan of 5 GPS + 1 BeiDou SV will fail).

Note that in Double constellation mode, if the LR1110 only detects one satellite in one of the two constellations, it generates a NAV message containing only one constellation.

11.2 GNSS Principle of Operation

Two GNSS modes are implemented:

- GNSS autonomous mode does not require any assistance location or almanac data, and aims to detect strong satellite signals. Therefore it is suitable for outdoor conditions with good sky visibility.
- GNSS assisted mode allows the most efficient GNSS geolocation. Assistance information can build a list of the satellites in view at the current time and location, in order to reduce the GNSS satellites search space, and therefore optimize the time and energy spent geolocating. The assistance information is tailored to an LPWAN network, limiting the data sent, especially the downlink size and frequency. It consists of:
 - LR1110 approximate position
 - Current time
 - Up-to-date reduced size Almanac information (less than 3 months old)

The LR1110 supports both GPS L1 and BeiDou B1 signals. It can perform either a single GNSS in any (or both) GPS and BeiDou constellations, or a dual GNSS BeiDou in any (or both) GPS and BeiDou constellations.

During the GNSS, the BUSY signal is set high, indicating that LR1110 is not ready to accept SPI transactions. BUSY returns to low when the procedure is complete. If the *GNSSDone* interrupt has been enabled, the IRQ signal goes high at the end of the GNSS process. To achieve the best GNSS sensitivity performance, an external LNA is necessary on the GNSS path. If the external GNSS LNA is controlled from the LR1110, the GNSS scan power consumption can be optimized.

A 32MHz TCXO is mandatory for any GNSS operation.

A 32.768kHz clock source is mandatory for GNSS Dual Constellation and for GNSS Advanced scan mode.

Refer to sections [11.3.2 GnssSetConstellationToUse](#) and [11.3.5 GnssSetMode](#) for details.

The clock requirements associated with the GNSS feature of LoRa Edge devices are summarized in AN1200.74 LoRa Edge Clock Requirements.

11.3 GNSS Commands

11.3.1 List of GNSS Commands

Table 11-1: Summary of Available GNSS Commands

Command Name	Details
GnssSetConstellationToUse	11.3.2 GnssSetConstellationToUse
GnssReadConstellationToUse	11.3.3 GnssReadConstellationToUse
GnssReadSupportedConstellations	11.3.4 GnssReadSupportedConstellations
GnssSetMode	11.3.5 GnssSetMode
GnssAutonomous	11.3.6 GnssAutonomous
GnssAssisted	11.3.7 GnssAssisted
GnssSetAssistancePosition	11.3.8 GnssSetAssistancePosition
GnssReadAssistancePosition	11.3.9 GnssReadAssistancePosition
GnssGetContextStatus	11.3.10 GnssGetContextStatus
GnssReadVersion	11.3.11 GnssReadVersion
GnssSetAlmanacUpdate	11.3.12 GnssSetAlmanacUpdate
GnssReadAlmanacUpdate	11.3.13 GnssReadAlmanacUpdate
GnssPushSolverMsg	11.3.14 GnssPushSolverMsg
GnssPushDmMsg	11.3.15 GnssPushDmMsg

11.3.2 GnssSetConstellationToUse

Command SetGnssConstellationToUse(. . .) configures the GNSS scanning for the selected constellation (GPS /BeiDou).

Table 11-2: GnssSetConstellationToUse Command

Byte	0	1	2
Data from Host	0x04	0x00	ConstellationBitMask (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *ConstellationBitMask*: Selection between GPS, or BeiDou, or both GPS and BeiDou.
 - ◆ bit 0 = 1: GPS selected
 - ◆ bit 1 = 1: BeiDou selected
 - ◆ Other values are RFU

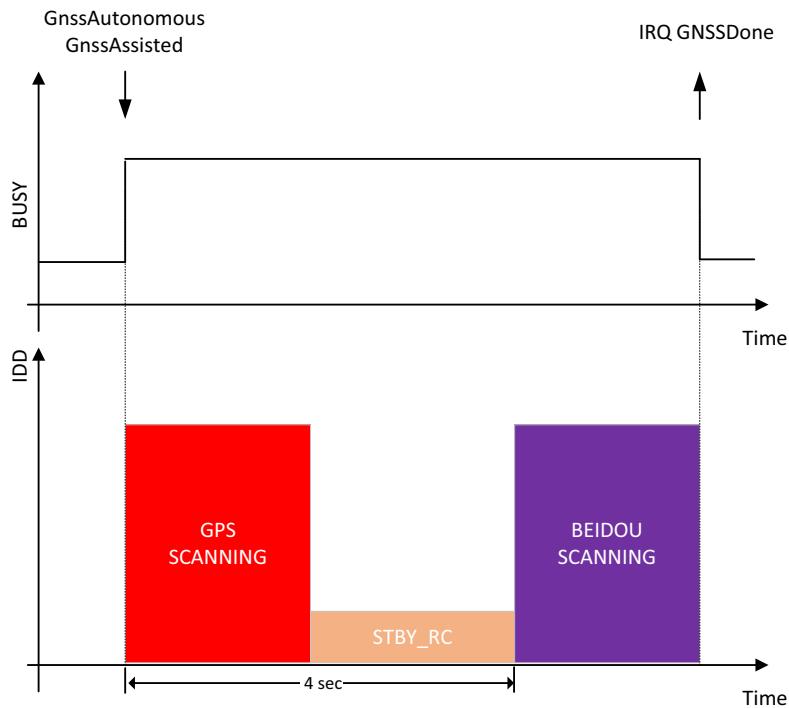


Figure 11-2: GNSS Dual Constellation Timing

If both GPS and BeiDou are selected, the GPS scanning is triggered immediately after the `GnssAutonomous()` or `GnssAssisted()` command is issued by the Host MCU, followed by the BeiDou scanning. The BeiDou scanning is autonomously triggered by the LR1110 exactly 4 seconds (with +/- 1ms accuracy) after the start of the GPS scanning. In order to respect this timing, the LR1110 requires a 32.768kHz clock source.

Between both scans, the LR1110 automatically returns in STBY_RC mode for minimum power consumption. The BUSY signal is set high until the end of the BeiDou scanning -including during the STBY_RC phase. At the end of the GNSS capture, the `GNSSDone` interruption is set high.

11.3.3 GnssReadConstellationToUse

Command `GnssReadConstellationToUse(...)` reads the selected constellation (GPS /BeiDou). See [11.3.2 GnssSetConstellationToUse](#) for parameter descriptions.

Table 11-3: GnssReadConstellationToUse Command

Byte	0	1
Data from Host	0x04	0x01
Data to Host	Stat1	Stat2

Table 11-4: GnssReadConstellationToUse Response

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

11.3.4 GnssReadSupportedConstellations

Command GnssReadSupportedConstellations(. . .) reads the supported constellations.
See [11.3.2 GnssSetConstellationToUse](#) for parameter descriptions.

See AN1200.74 LoRa Edge Clock Requirements for 32MHz TCXO and 32.768kHz clock considerations

Table 11-5: GnssReadSupportedConstellations Command

Byte	0	1
Data from Host	0x04	0x07
Data to Host	Stat1	Stat2

Table 11-6: GnssReadSupportedConstellations Response

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

11.3.5 GnssSetMode

Command GnssSetMode(. . .) configures the GNSS for a single or multiple scanning of the selected constellation (GPS and/or BeiDou).

Table 11-7: GnssSetMode Command

Byte	0	1	2
Data from Host	0x04	0x08	GnssMode
Data to Host	Stat1	Stat2	IrqStatus (31:24)

- *GnssMode*: Selection between single or multiple GNSS scanning.
 - 0x00: Single (Legacy) scanning
 - 0x03: Multiple (Advanced) scanning
 - Other values are RFU

During multiple scanning, the LR1110 performs and averages multiple GNSS captures to increase the precision of the GNSS geolocation. Therefore, a multiple scan will last longer and consume more energy than a single scan, but the GNSS precision will be increased.

11.3.6 GnssAutonomous

Command GnssAutonomous(. . .) captures GNSS signals in autonomous mode, for example in case no assistance information is available, or for fast indoor/outdoor detection.

Table 11-8: GnssAutonomous Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x04	0x09	Time (31:24)	Time (23:16)	Time (15:8)	Time (7:0)	Effort Mode	Result Mask	NbSvMax
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

- *Time*: GPS time (GPST), in number of seconds elapsed since 6 January 1980 00:00:00. Hint: When converting from UTC to GPST, the UTC-GPST corresponding leap second offset must be taken into account.
- *EffortMode* = 0x00. Other values are RFU.
- *ResultMask*: bit mask indicating which information is added in the NAV message. This mask has a different meaning depending on the value of *GnssMode* (refer to [11.3.5 GnssSetMode](#)).
 - ◆ Bit 0:
 - ◆ GnssMode = 0: If set, includes pseudo-ranges in the NAV message: 19 bits per satellite. Warning: if this bit is not set, then the solver cannot compute the location.
 - ◆ GnssMode = 1: if set, include Doppler information in the NAV message.
 - ◆ Bit 1:
 - ◆ GnssMode = 0: If set, include the Doppler information in the NAV message: 15 bits per satellite. Note however that at least 5 satellites will have their Doppler reported in the NAV message independently of this configuration. The Doppler information is used by the solver to compute a coarse location of the device. It is advised to set this bit.
 - ◆ GnssMode = 1: If set, include the Doppler information in the NAV message, maximum of 14 satellites Doppler information per constellation used. Note however that at least 7 satellites will have their Doppler reported in the NAV message independently of this configuration.
 - ◆ Bit 2: If set, includes bit changes in the NAV message: 1 byte per satellite. This bit is currently not used by the solver so setting it does not bring any improvement.
- *NbSvMax* defines the maximum number of satellites wanted as a result of the GnssAutonomous(. . .). If more satellites are detected during the scanning than *NbSvMax*, then the satellites with the highest C/N0 are returned.
 - ◆ If *NbSvMax* = 0, then all the detected satellites are returned.
 - ◆ Values *NbSvMax* = 1 and *NbSvMax* = 2 are not accepted.

Please note that calling this command resets the previous GNSS results, if any.

11.3.7 GnssAssisted

Command `GnssAssisted(. . .)` captures GNSS signals using assistance data (current time, approximate position, and Almanac information).

Table 11-9: GnssAssisted Command

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x04	0x0A	Time (31:24)	Time (23:16)	Time (15:8)	Time (7:0)	Effort Mode	ResultMask	NbSvMax
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0x00	0x00	0x00

- *Time*: GPS time (GPST), in number of seconds elapsed since 6 January 1980 00:00:00. Hint: When converting from UTC to GPST, the UTC-GPST corresponding leap second offset must be taken into account.
 - *EffortMode*
 - ◆ 0x00: Low Power mode. GNSS assisted scanning stops detection if no strong satellite is detected. Behaves as explained below:
 1. Performs a strong satellite search on all visible SV. (A strong satellite is typically > 30dBHz).
 2. If no SV are found during step 1, then stops.
 3. If SV are found but the frequency error is too high, provides detected SV and raises "Doppler error" flag. The search window is reduced by the Doppler estimation done with SW found in step 1.
 4. Performs a search on all visible SV (up to 12, + EGNOS/WAAS for GPS).
 - ◆ 0x01: Best Effort mode. GNSS assisted scanning continues detection even if no strong satellite is detected.
 1. Performs a strong satellite search on all visible SV.
 2. If SV are found in step 1, computes the right search window. Else, uses the default search window.
 3. Performs a search (using the search window set in step 2) on all visible SV (up to 12, + EGNOS/WAAS for GPS - limited to 11 when double constellation).
 - *ResultMask*: Refer to [11.3.6 GnssAutonomous](#) for description.
 - *NbSvMax* defines the maximum number of satellites to detect.
 - ◆ If *NbSvMax* = 0, all the detected satellites are returned. Otherwise, only the *NbSvMax* satellites with higher C/N0 are returned.
 - ◆ Values *NbSvMax* = 1 and *NbSvMax* = 2 are not accepted
- Please note that calling this command resets the previous GNSS results, if any.

11.3.8 GnssSetAssistancePosition

Command GnssSetAssistancePosition(...) configures the approximate position for GNSS assisted mode.

Table 11-10: GnssSetAssistancePosition Command

Byte	0	1	2	3	4	5
Data from Host	0x04	0x10	Latitude (15-8)	Latitude (7-0)	Longitude (15-8)	Longitude (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *Latitude*: Latitude, coded on 12 bits (resolution of 0.044°)
Latitude = latitude in degrees (decimal value)* 2048/90
For example, for 47.006° latitude: $47.006 * 2048 / 90 = 1070$ (rounded) = 0x042E
- *Longitude*: Longitude, coded on 12 bits (resolution of 0.088°)
Longitude = longitude in degrees (decimal value)* 2048/180
For example, for 6.966°longitude: $6.966 * 2048 / 180 = 79$ (rounded) = 0x004F

11.3.9 GnssReadAssistancePosition

Command GnssReadAssistancePosition(...) reads the assistance position.

Table 11-11: GnssReadAssistancePosition Command

Byte	0	1
Data from Host	0x04	0x11
Data to Host	Stat1	Stat2

Table 11-12: GnssReadAssistancePosition Response

Byte	0	1	2	3	4
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	Latitude (15-8)	Latitude (7-0)	Longitude (15-8)	Longitude (7-0)

See 11.3.8 GnssSetAssistancePosition for parameter description.

11.3.10 GnssGetContextStatus

Command GnssGetContextStatus(. . .) reads the GNSS context status.

Table 11-13: GnssGetContextStatus Command

Byte	0	1
Data from Host	0x04	0x16
Data to Host	Stat1	Stat2

Table 11-14: GnssGetContextStatus Response

Byte	0	1	2	3	4-7	8	9
Data from Host	0x00	0x02	0x18	0x00	0x00	0x00	0x00
Data to Host	Stat1	DestinationId	Opcode	GNSSFw Version	Global Almanac CRC	Error Code (7:4) Almanac Update BitMask (3:1)	Freq Search Space (0) Freq Search Space (7) RFU (6:0)

- *GNSSFwVersion*: Firmware version.
- *GlobalAlmanacCRC* is the 32-bit CRC computed on all the flash memory content, on all 128 satellites. Per SV 21 bytes:
 - Byte 0: SvId
 - Bytes 1-2: AlmanacDate
 - Bytes 3-17: Almanac
 - Bytes 18-19: CaCodeGenerator
 - Byte 20: ModulationBitMask
 - Byte 21: ConstellationId
- *ErrorCode*
 - 0: No error
 - 1: Almanac too old
 - 2: Last Almanac update CRC mismatch
 - 3: Flash memory integrity error
 - 4: Almanac update not allowed (GPS and Beidou satellite can't be updated in a same request)
 - 5-15: RFU
- *AlmanacUpdateBitMask*:
 - Bit 0: GPS
 - Bit 1: Beidou
 - Bit 2: RFU
- *FreqSearchSpace*: (2-bit field)
 - 0: 250Hz
 - 1: 500Hz
 - 2: 1kHz
 - 3: 2kHz
- Other bits are RFU.

11.3.11 GnssReadVersion

Command GnssReadVersion(. . .) returns the internal GNSS firmware version and almanac version..

Table 11-15: GnssReadVersion Command

Byte	0	1
Data from Host	0x04	0x06
Data to Host	Stat1	Stat2

Table 11-16: GnssReadVersion Response

Byte	0	1	2
Data from Host	0x00	0x00	0x00
Data to Host	Stat1	Firmware Version	Almanac Version

11.3.12 GnssSetAlmanacUpdate

Command GnssSetAlmanacUpdate(. . .) configures the constellation almanac information to be updated..

Table 11-17: GnssSetAlmanacUpdate Command

Byte	0	1	2
Data from Host	0x04	0x02	AlmanacUpdateBitMask
Data to Host	Stat1	Stat2	IrqStatus(31-24)

AlmanacUpdateBitMask: activation of GPS or BeiDou. By default, both constellations are activated.

- Bit 0: GPS
- Bit 1: BeiDou
- Bit 2: RFU

11.3.13 GnssReadAlmanacUpdate

Command GnssReadAlmanacUpdate(. . .) reads the almanac update information (refer to 11.3.12 GnssSetAlmanacUpdate)..

Table 11-18: GnssReadAlmanacUpdate Command

Byte	0	1
Data from Host	0x04	0x03
Data to Host	Stat1	Stat2

Table 11-19: GnssReadAlmanacUpdate Response

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

11.3.14 GnssPushSolverMsg

Command GnssPushSolverMsg(. . .) pushes the messages from the GNSS solver to the LR1110 (assistance position update, and so on).

Table 11-20: GnssPushSolverMsg Command

Byte	0	1	2	n
Data from Host	0x04	0x14	Payload	...
Data to Host	Stat1	Stat2	IrqStatus(31-24)	...

Payload: Byte stream to be forwarded by the Host MCU to the LR1110.

11.3.15 GnssPushDmMsg

Command GnssPushDmMsg(. . .) pushes the messages from the LoRaWAN network to the LR1110.

Table 11-21: GnssPushDmMsg Command

Byte	0	1	2	n
Data from Host	0x04	0x15	Payload	...
Data to Host	Stat1	Stat2	IrqStatus(31-24)	...

Payload: Byte stream to be forwarded by the Host MCU to the LR1110.

11.4 GNSS Scanning Results & Commands

GNSS scanning results are formatted in messages, of variable length depending on the number of satellites detected and on the *GnssMode*. The message destination can be either:

- Host (for status information)
- GNSS Position Solving component (for geolocation cloud calculation)
- GNSS Almanac Update component of Semtech LoRa Cloud

To read the GNSS scanning results, the result stream size must be determined first using command `GnssGetResultSize(...)`. Afterwards, the results can be read using command `GnssReadResult(...)`.

11.4.1 GNSS Scan Result Message Description

The message format is shown below. It is composed of a *DestinationID* field, followed by a *Payload* of variable length



Figure 11-3: GNSS Scan Result Message Format

- *DestinationID* = 0x00: Scan result status message to the Host (see [Section 11.4.1.1](#))
- *DestinationID* = 0x01: NAV message to the GNSS Solver
- *DestinationID* = 0x02: Almanac update message to the device management service

11.4.1.1 Scan Result Status Message

The scan result status messages to the host (*DestinationID* = 0x00) have a single byte *Payload*, coded as below:

- 0x00: OK
- 0x01: Command unexpected
- 0x02: Command not implemented
- 0x03: Command parameters invalid
- 0x04: Message Sanity check error
- 0x05: Scanning failed
- 0x06: No time
- 0x07: No satellite detected
- 0x08: Almanac too old
- 0x09: Almanac update fails due to CRC errors
- 0x0A: Almanac update fails due to flash integrity error
- 0x0B: Almanac update fails due to almanac date too old
- 0x0C: Almanac update not allowed (GPS and Beidou satellite can't be updated in a same request)
- 0x0D: Global Almanac CRC error
- 0x0E: Almanac version not supported
- All other values are RFU

These messages must not be transmitted to the GNSS solver.

11.4.2 GnssGetResultSize

Command GnssGetResultSize(...) reads the size in bytes of the byte stream containing the available GNSS results.

Table 11-22: GnssGetResultSize Command

Byte	0	1
Data from Host	0x04	0x0C
Data to Host	Stat1	Stat2

Table 11-23: GnssGetResultSize Response

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

11.4.3 GnssReadResults

Command GnssReadResults(...) retrieves the last GNSS results.

Table 11-24: GnssReadResults Command

Byte	0	1
Data from Host	0x04	0x0D
Data to Host	Stat1	Stat2

Table 11-25: GnssReadResults Response

Byte	0	1	2	3	...
Data from Host	0x00	0x00	0x00	0x00	0x00
Data to Host	Stat1	ResultsByte1	ResultsByte2	ResultsByte3	ResultsByteN

11.4.4 GnssGetNbSvDetected

Command GnssGetNbSvDetected(. . .) retrieves the number of Satellite Vehicles detected during the last GNSS Scanning.

Table 11-26: GnssGetNbSvDetected Command

Byte	0	1
Data from Host	0x04	0x17
Data to Host	Stat1	Stat2

Table 11-27: GnssGetNbSvDetected Response

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

11.4.5 GnssGetSvDetected

Command GnssGetSvDetected(. . .) retrieves the ID, the SNR and the Doppler of the Satellite Vehicles detected during the last GNSS Scanning.

Table 11-28: GnssGetSvDetected Command

Byte	0	1
Data from Host	0x04	0x18
Data to Host	Stat1	Stat2

Table 11-29: GnssGetSvDetected Response

Byte	0	1	2	3	4	5	...
Data from Host	0x00	0x00	0x00	0x00	0x00	0x00	...
Data to Host	Stat1	SvId1	SNR1	Doppler1 (15-8)	Doppler1 (7-0)	SvId2	...

- *SvId*: Satellite Vehicle ID
- *SNR*: Signal to Noise of the detected Satellite Vehicle, expressed in dB. To convert into C/N0, add 31dB
- *Doppler*: Signed 16-bit value

11.4.6 GnssGetConsumption

Command GnssGetConsumption(. . .) reads the duration of the Radio capture and the CPU processing phases of the GNSS Scanning capture. These timings are expressed in microseconds.

This can be used to determine the GNSS Scanning power consumption.

Table 11-30: GnssGetConsumption Command

Byte	0	1
Data from Host	0x04	0x19
Data to Host	Stat1	Stat2

Table 11-31: GnssGetConsumption 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	CPUTime (31:24)	CPUTime (23:16)	CPUTime (15:8)	CPUTime (7:0)	Radio Time (31:24)	Radio Time (23:16)	Radio Time (15:8)	Radio Time (7:0)

11.4.7 GnssGetSvVisible

Command GnssGetSvVisible(. . .) returns the number of visible Space Vehicles in the sky, based on given assistance data (assistance position and time).

This command can be used to schedule a GNSS scan to target the maximum SV visibility.

Table 11-32: GnssGetSvVisible Command

Byte	0	1	2	3	4	5	6	7	8	9	10
Data from Host	0x04	0x1F	Time (31-24)	Time (23-16)	Time (15-8)	Time (7-0)	Latitude (15-8)	Latitude (7-0)	Longitude (15-8)	Longitude (7-0)	Constel lation
Data to Host	Stat1	Svld1	SNR1	Doppler 1 (15-8)	Doppler 1 (7-0)	Svld2					

Constellation: selected constellation

- 0 = GPS
- 1 = BeiDou

Refer to sections [11.3.7 GnssAssisted](#) and [11.3.8 GnssSetAssistancePosition](#) for the other parameters descriptions.

Table 11-33: GnssGetSvVisible Response

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

11.5 GNSS Almanac

The GNSS Almanac consists of information about the state of the entire GNSS satellite constellation and coarse data on every satellite's orbit. There is a specific almanac for each constellation. The almanac data is valid for up to 90 days.

The use of the almanac significantly optimizes the GNSS scanning duration: the LR1110 searches only for visible satellites within the user location and time, and therefore reduces the energy required for a GNSS scanning.

The Almanac is used by the LR1110 in GNSS assisted mode.

The LR1110 is pre-programmed with the latest Almanac data at the date of the production test. Even if the Almanac data is valid for 90 days, it is advised to use the latest Almanac data for power optimization. The up-to-date Almanac is available from the Full Almanac image Download component of LoRa Cloud.

The whole Almanac data can be updated (see [11.5.1 GnssAlmanacFullUpdate](#)). The Almanac is entirely stored in flash memory, therefore kept after power off or Sleep mode without retention.

11.5.1 GnssAlmanacFullUpdate

The Almanac data for all the satellites can be updated using command `GnssAlmanacFullUpdate(...)`. Each constellation must be updated separately.

Table 11-34: GnssAlmanacFullUpdate Command

Byte	0	1	2: See Table 11-35 below
Data from Host	0x04	0x0E	AlmanacFullUpdatePayload
Data to Host	Stat1	Stat2	IrqStatus(31:24)

- *AlmanacFullUpdatePayload*: defined as in below:

Table 11-35: AlmanacFullUpdatePayload Parameter Format

Byte	(0:19)	(20:39)	(40:56)	...	(2560:2579) ¹
Field	AlmanacHeader	SV1 Almanac	SV2 Almanac	...	SV128 Almanac

1. In transmission, the frame cannot be longer than 512 bytes. In reception, the frame cannot be longer than 1020 bytes. See below for more detail.

- *AlmanacHeader*: defined as below:

Table 11-36: AlmanacHeader Parameter Format

Byte	0	(1:2)	(3:6)	(7:19)
Field	128	AlmanacDate	Global CRC	RFU

- Each *SVn Almanac* is a 20-byte structure, defined as below:

Table 11-37: SVn Almanac Parameter Format

Byte	0	(1:15)	(16:17)	18	19
Field	SV id	Almanac Content	CA code	Modulation bit mask	Constellation Id

The user must ensure that the list of almanacs and the list of satellites ids are coherent. The Almanac data must be provided in the same order as satellite ids.

The *AlmanacFullUpdatePayload* takes 20 bytes (Header) +128 (number of SV) * 20 bytes = 2580 bytes.

The payload must be made up of complete Blocks, where a Block is 20 data bytes.

The maximum number of bytes that can be sent from the host MCU to the LR1110 is 512 bytes. Therefore, the Almanac Full Update must be handled in multiple SPI transactions. For example, the two following approaches are possible:

- Minimum memory overhead: The *AlmanacFullUpdatePayload* can be sent in 129 successive SPI transactions of blocks (20 bytes each).
- Minimum number of SPI transactions: The *AlmanacFullUpdatePayload* can be sent in 5 SPI transactions of 25 blocks (500 bytes), and a sixth SPI transaction of 4 blocks (80 bytes).

12. Sub-GHz Ranging

The LR1110 features a sub-GHz ranging engine that can estimate the distance between two LR1110 devices. It is based on time of flight measurements of dedicated LoRa modulated communication exchanges between the two devices.

Additional information on [ranging](#) using other LoRa products can be found in the following [references](#):

- AN1200.50: SX1280 Ranging Protocol
- AN1200.29 Introduction to Ranging with the SX1280
- AN1200.31 How to Perform Ranging Tests with the SX1280 Development Kit

12.1 Sub-GHz Ranging Commands API Interface

Starting from transceiver firmware version 0x0308, five additional commands are supported to enable the use of the LR1110 Sub-GHz ranging functionality. The user must configure and transmit the ranging packets according to the command order depicted in [Figure 8-1](#). The additional commands are as follows:

Table 12-1: Ranging Commands

Command	Opcode	Parameter	Description
SetRangingAddr	0x021C	<i>Addr, Checklength</i>	Sets the ranging id of the device and the number of bytes to be checked on a request
SetRangingReqAddr	0x021D	<i>ReqAddr</i>	Sets the ranging id for the request
GetRangingResult	0x021E	<i>Type</i>	Gets the ranging results (distance, RSSI)
SetRangingTxRxDelay	0x021F	<i>Delay</i>	Sets the TxRx delay for the ranging
SetRangingParameter	0x0228	<i>SymbNb</i>	Sets the number of synchronization symbols

12.2 SetRangingAddr

Command `SetRangingAddr(...)` sets the ranging ID for this slave.

Table 12-2: SetRangingAddr Command

Byte	0	1	2	3	4	5	6
Data from host	0x02	0x1C	Addr (31:24)	Addr (23:16)	Addr (15:8)	Addr (7:0)	CheckLength
Data to host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0

- *Addr*: 32 bits, default = 0x00000019
- *CheckLength*: Defines how many of the 4 address bytes are checked against the requested address sent by the Master. Default=4. Checked bytes are the LSB if CheckLength<4.

12.3 SetRangingReqAddr

Command SetRangingReqAddr(...) sets the address that is requested by the Master in the ranging request. This must match the receiving Slave's ranging address.

Table 12-3: SetRangingReqAddr Command

Byte	0	1	2	3	4	5
Data from host	0x02	0x1D	ReqAddr (31:24)	ReqAddr (23:16)	ReqAddr (15:8)	ReqAddr (7:0)
Data to host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)

- *ReqAddr*: 32 bits, default 0x00000019

12.4 GetRangingResult

Command GetRangingResult(...) reads the results in the Master, based on the *Type*:

Table 12-4: GetRangingResult Request

Byte	0	1	2
Data from host	0x02	0x1E	Type
Data to host	Stat1	Stat2	IrqStatus (31:24)

- *Type*: Defines the type of result to be returned:
 - 0: last ranging distance result: distance measurement
 - 1: last ranging RSSI result: signal strength indication

Table 12-5: GetRangingResult Response

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

- *Res*: Ranging results information depending upon the *Type*:
 - Ranging Distance *Res(31:0)*: Distance (signed 24-bit), which is converted into meters as follows:
Round Trip Distance [m] = $Res * 3e8 / (2^{12} * BW)$, where BW is the LoRa bandwidth in Hz
 - Ranging RSSI *Res(7:0)*: Received signal strength, linear, in dB units calculated as follows:
RSSI [dB] = $Res / 2$

12.5 SetRangingTxRxDelay

Command SetRangingTxRxDelay (...) sets the TxRx delay for the ranging calibration. The processing of the ranging response and request incurs a deterministic fixed delay which must be compensated for in order to provide an accurate range estimation. This fixed timing (distance) offset is removed by subtracting the TxRx delay offset calibration value. Note that the same value must be written in both Master and Slave.

Table 12-6: SetRangingTxRxDelay Command

Byte	0	1	2	3	4	5
Data from host	0x02	0x1F	Delay (31:24)	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)

- *Delay*: The value depends on the LoRa BW/SF used (the following are typical values for Semtech EVKs and may differ on other PCB designs):

Table 12-7: Typical Delay Values for Semtech EVKs

BW	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12
125	19035	19040	19040	19039	19036	19038	19036	19024
250	20235	20239	20238	20237	20236	20235	20236	20232
500	20149	20227	20258	20277	20286	20292	20295	20298

12.6 SetRangingParameter

Command SetRangingParameter (...) defines how many symbols are used during synchronization.

Table 12-8: SetRangingParameter Command

Byte	0	1	2	3
Data from host	0x02	0x28	Reserved	SymbNb
Data to host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)

- *Reserved*: 0
- *SymbNb*: The recommended value of 15 gives a good compromise between the accuracy of the measurement result and the time on air and energy consumption). Increasing the number of symbols can help increase the accuracy of the result at the expense of longer time on air.

13. Bluetooth® Low Energy Beacons Compatibility

The LR1110 embeds a transmitter that is compatible with Bluetooth® technology, to address indoor/outdoor localization use cases based on Bluetooth Low Energy beaconing.

- The Bluetooth Low Energy beaconing compatible data is detected by Bluetooth Low Energy beaconing receivers to determine the device's location.
- The Bluetooth Low Energy beaconing compatible parameters are fully configurable to permit transmission of any type of Bluetooth Low Energy beacon.

Note: There is no Bluetooth Low Energy beaconing compatible stack or state machine implemented inside the LR1110, therefore the user must handle the transmission of the Bluetooth Low Energy compatible beacons in the 3 advertising channels at the correct advertising interval.

13.1 ConfigBleBeacon

Command ConfigBleBeacon(. . .) configures the radio to be Bluetooth Low Energy beaconing compatible on the given channel: syncword 0x8E89BED6, 3 byte CRC (polynomial 0x100065B, seed 0x555555).

Note: This command does not send the packet. It must be sent with the normal SetTx() command.

Table 13-1: ConfigBleBeacon Command

Byte	0	1	2	3	4	5	N+3
Data from host	0x02	0x2E	Channel(7:0)	Payload[0]	Payload[1]	...	Payload[N]
Data to host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	...	0

- *Channel*: Allowed channel values are 37, 38, or 39. The channel determines the frequency and whitening seed settings.
- *Payload*: (optional). To send the same beacon on different channels, this command can be issued with only the *channel* as parameter.

13.2 BleBeaconSend

Command BleBeaconSend(. . .) configures the radio to be Bluetooth Low Energy beaconing compatible on the given channel and sends the packet: syncword 0x8E89BED6, 3 byte CRC (polynomial 0x100065B, seed 0x555555).

Table 13-2: BleBeaconSend Command

Byte	0	1	2	3	4	5	N+3
Data from host	0x02	0x31	Channel(7:0)	Payload[0]	Payload[1]	...	Payload[N]
Data to host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	...	0

See [13.1 ConfigBleBeacon](#) for parameter descriptions.

14. Cryptographic Engine

14.1 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.

14.2 Cryptographic Keys Definition

The cryptographic keys are arranged into several groups, according to the function they serve, as shown in [Table 14-1: 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 14-1: 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		
	7	GpKEKey1		
	8	GpKEKey2	CryptoDeriveKey(...)	From any other Gp Transport key or from Application Key
	9	GpKEKey3	CryptoSetKey(...)	
	10	GpKEKey4	Any multicast Key	
	11	GpKEKey5		

Table 14-1: Cryptographic Keys Usage and Derivation (Continued)

Group Name	Key Source/ Dest. Index	Key Name	Usage	Derivation From
Unicast	12	AppSKey		
	13	FNwkSIntKey		
	14	SNwkSIntKey	CryptoAesEncrypt01(...) CryptoComputeAesCmac() CryptoSetKey(...)	From Network & Application
	15	NwkSEncKey		
	16	RFU0		
	17	RFU1		
Multicast	18	McAppSKey0		
	19	McAppSKey1		
	20	McAppSKey2		
	12	McAppSKey3	CryptoAesEncrypt01(...)	Only from GpTransport
	22	McNwkSKey0	CryptoVerifyAesCmac(...) CryptoSetKey(...)	Key
	23	McNwkSKey1		
General Purpose	24	McNwkSKey2		
	25	McNwkSKey3		
General Purpose	26	GP0	CryptoAesEncrypt(...)	Not Allowed
	27	GP1	CryptoAesDecrypt(...) CryptoSetKey(...)	

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

14.3 Commands

14.3.1 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

14.3.2 CryptoSetKey

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

Table 14-2: 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 14-3: 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 14-1: 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 152](#)

14.3.3 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 14-4: 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 14-5: 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 14-1: 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 15.3](#) and [Section 15.4](#)
- `CEStatus` is defined in section [CEStatus on page 152](#)

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.

14.3.4 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 14-6: CryptoProcessJoinAccept Command

Byte	0	1	2	3	4	5	...	N+4	N+5	...	N+4+M
Data from Host	0x05	0x04	DeckKeyID (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 14-7: CryptoProcessJoinAccept Response

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

- *DeckKeyID* and *VerKeyID* are defined in [Table 14-1: Cryptographic Keys Usage and Derivation](#):
 - *DeckKeyID*: 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 152](#)

14.3.5 CryptoComputeAesCmac

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

Table 14-8: 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 14-9: 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 14-1: 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 152](#)
- *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 14-10: 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 14-11: 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	0xa0	0x16	0xb4

14.3.6 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 14-12: 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 14-13: CryptoVerifyAesCmac Response

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

- *KeyID*: Specified Key ID, as defined in [Table 14-1: 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 152](#)

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

14.3.7 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 14-14: 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 14-15: 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 14-1: Cryptographic Keys Usage and Derivation](#). Valid values are from 12 to 25.
- *Data1, Data2 to DataN*: Provided data, considered as byte buffers
- *CEStatus*: Defined in section [CEStatus on page 152](#)
- *EncryptedData1, EncryptedData2,..., EncryptedDataN*: Encrypted data, considered as byte buffers

14.3.8 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 14-16: 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 14-17: 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 14-1: 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 152](#)
- *EncryptedData1, EncryptedData2 to EncryptedDataN*: Encrypted data, considered as byte buffers

14.3.9 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 14-18: 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 14-19: 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 14-1: 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 152](#)
- *DecryptedData1, DecryptedData2 to DecryptedDataN*: Decrypted data, considered as byte buffers

14.3.10 CryptoStoreToFlash

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

Table 14-20: CryptoStoreToFlash Command

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

Table 14-21: CryptoAesDecrypt Response

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

- *CEStatus*: Defined in section [CEStatus on page 152](#)

14.3.11 CryptoRestoreFromFlash

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

Table 14-22: CryptoRestoreFromFlash Command

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

Table 14-23: CryptoRestoreFromFlash Response

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

- *CEStatus*: Defined in section [CEStatus on page 152](#)

14.3.12 CryptoSetParam

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

Table 14-24: 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 14-25: 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 152](#)

14.3.13 CryptoGetParam

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

Table 14-26: CryptoGetParam Command

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

Table 14-27: CryptoGetParam Response

Byte	0	1	2	3	4	5
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 152](#)

14.3.14 CryptoCheckEncryptedFirmwareImage

Command `CryptoCheckEncryptedFirmwareImage()` adds a chunk to the encrypted FW image to be checked.

It must be called several times until the complete image is sent to the chip:

- It only takes 64 chunks of 32-bit data (overall 256 bytes + command and opcode) in one go, except for the last segment which may be shorter
- The first offset is 0
- The BUSY pin is released by the LR11xx when it is ready to accept another chunk of data

Once the firmware image has been sent to the chip, you can call the `CryptoCheckFirmwareImageResult()` command to check the result. These commands are useful to check the validity of a firmware before flashing it.

Table 14-28: CryptoCheckEncryptedFirmwareImage Command

Byte	0	1	2	3	4	5	6	7	8	9
Data from host	0x05	0x0F	Offset (31:24)	Offset (23:16)	Offset (15:8)	Offset (7:0)	Data1 (31:24)	Data1 (23:16)	Data1 (15:8)	Data1 (7:0)
Data to host	Stat1	Stat2	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

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

Offset: Must be a multiple of 4 (bytes)

Data: Length must be a multiple of 4. Max length = 256 bytes

Note: N = 64 for all chunks of data, except the last one which may be shorter

14.3.15 CryptoCheckEncryptedFirmwareImageResult

Command CryptoCheckEncryptedFirmwareImageResult() gets the result of the check.

Table 14-29: CryptoCheckEncryptedFirmwareImageResult Command

Byte	0	1
Data from Host	0x05	0x10
Data to Host	Stat1	Stat2

Table 14-30: CryptoCheckEncryptedFirmwareImageResult Response

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

- *Result:*
 - ◆ 1: firmware verification success
 - ◆ 0: firmware verification error

15. LR1110 Provisioning

15.1 Description

The LR1110 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.

15.2 Provisioning Commands

15.2.1 GetChipEui

Command `GetChipEui(...)` reads the LR1110'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 15-1: GetChipEui Command

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

Table 15-2: 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.

15.2.2 GetSemtechJoinEui

Command GetSemtechJoinEui(. . .) reads LR1110'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 15-3: GetSemtechJoinEui Command

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

Table 15-4: GetSemtechJoinEui Response

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

SemtechJoinEui is coded on 8 bytes, in big endian format.

15.2.3 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.

15.2.3.1 Standard Use

Table 15-5: DeriveRootKeysAndGetPin Command (Standard)

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

Table 15-6: DeriveRootKeysAndGetPin Response (Standard)

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.

15.2.3.2 Advanced Use

The LR1110 supports a user-defined DevEui and / or a dedicated JoinEui, both different from the ChipEui and SemtechJoinEui onboard the LR1110. 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 15-7: 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 15-8: 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.

15.2.3.3 Alternate Use

The LR1110 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.

15.3 Crypto Engine Use With LoRaWAN V1.1.x

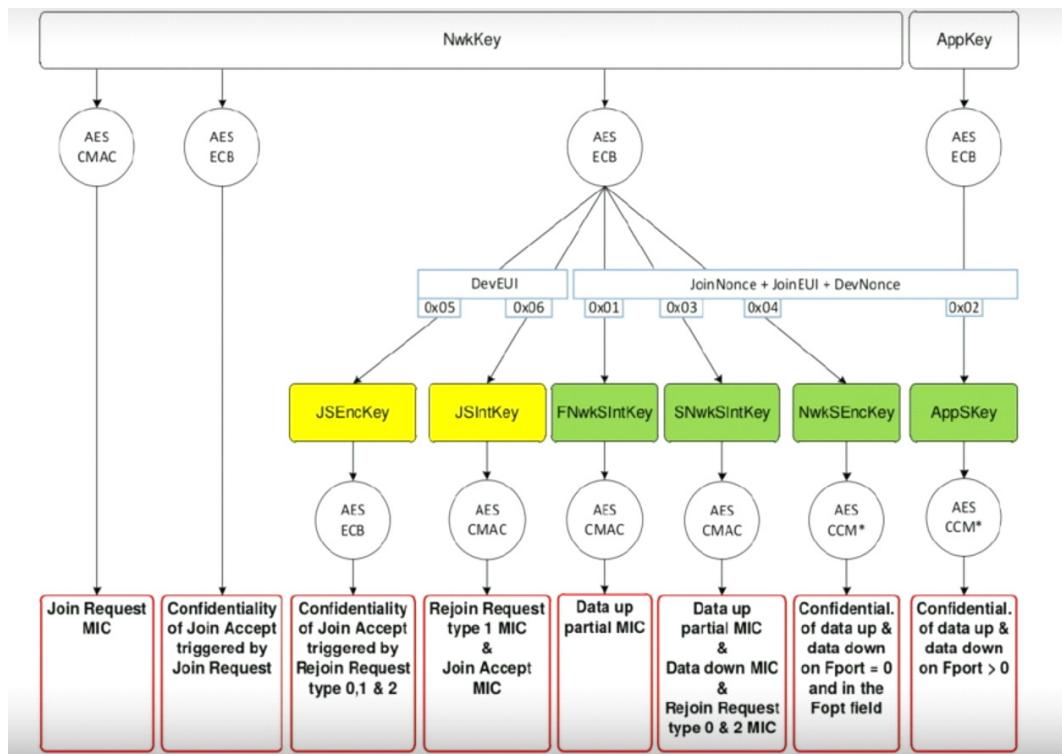


Figure 15-1: Key Derivation Scheme for LoRaWAN 1.1.x

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 15-1](#). 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 | pad16)
```

Where *pad₁₆* is the required number of 0x00 bytes to expand to a 16-byte number.

15.4 Crypto Engine Use With LoRaWAN V1.0.x

The key derivation scheme is available from the LoRaWAN specification:

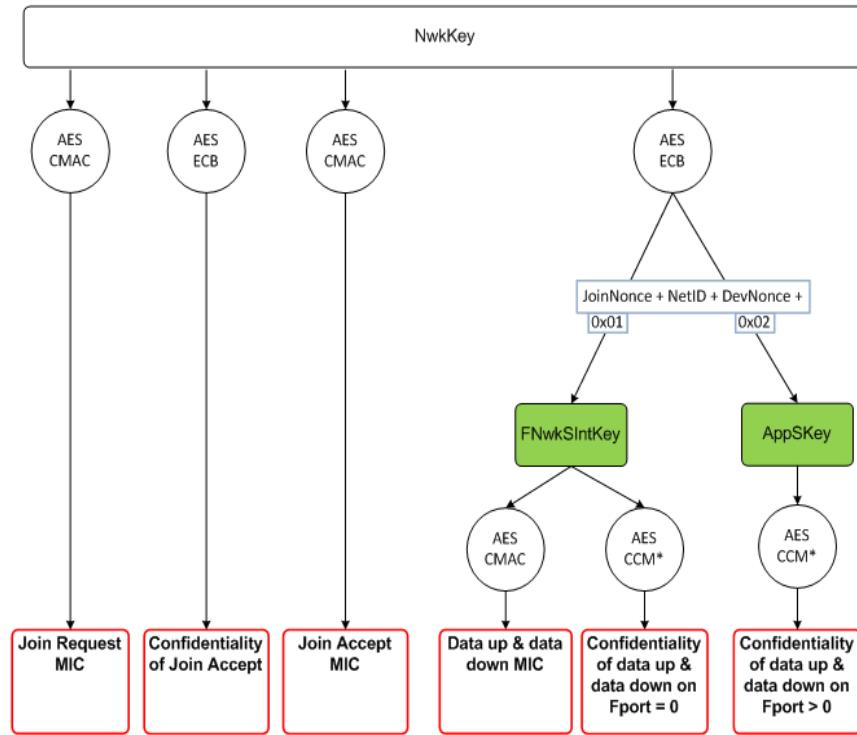


Figure 15-2: Key Derivation Scheme for LoRaWAN 1.0.x

Although the crypto engine of the LR1110 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 15-9: 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

16. Test Commands

Several LR1110 test commands allow an easy configuration of the device for regulatory ETSI or FCC compliance.

16.1 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.

16.1.1 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 LR1110.

Table 16-1: ETSI Test Signals

Test Signal	Description	LR1110 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 LR1110 radio commands.

16.1.2 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 LR1110 radio commands.

16.2 Commands

16.2.1 SetTxCw

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

Table 16-2: 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.

16.2.2 SetTxInfinitePreamble

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

Table 16-3: 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.

17. Known Limitations

Unexpectedly High Adjacent Channel Power

When the chip wakes up from sleep mode with retention, a parameter is not reconfigured properly. This misconfiguration can lead to an unexpectedly high adjacent channel power in all subsequent transmissions impacting the device regulatory compliance.

The issue appears only in LoRa modulation, for all bandwidths except 500kHz.

Affects all LR1110 firmware versions prior to and including 0x0307.

Workaround

The workaround is to reset bit 30 in the register 0x00F30054 when the chip wakes up from sleep mode with retention.

This workaround does not work when LR11XX_RADIO_MODE_SLEEP is configured with lr11xx_radio_auto_tx_rx and the chip is set to Rx mode, because the workaround cannot be applied before the subsequent transmission (which is automatically launched by the chip after waking up from sleep mode with retention).

18. 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](#).

18.1 Register / Memory Access Operations

Table 18-1: 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.

18.2 System Configuration / Status Operations

Table 18-2: 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) RFU(1) RfswGnssCfg(1) RfswWifiCfg(1)	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

Table 18-2: System Configuration / Status Operations (Sheet 2 of 2)

Command	Opcode	Parameters	Description
<code>WriteInfoPage</code>	0x0122	InfoPage(1) IndoWordAddr(2) dataN(4)	Writes a 32-bit block of the Info Page
<code>ReadInfoPage</code>	0x0123	dataN(1)	Reads a 32-bit block of the Info Page
<code>GetChipEui</code>	0x0125	---	Returns the 8-byte factory DeviceEui
<code>GetSemtechJoinEui</code>	0x0126	---	Returns the 8-byte factory JoinEui
<code>DeriveRootKeysAndGetPin</code>	0x0127	---	Returns the 4-byte PIN which can be used to register the device with LoRa Cloud Services
<code>EnableSpiCrc</code>	0x0128	Enable(1) CRC(1)	Enables / disables 8-bit CRC on SPI interface
<code>DriveDiosInSleepMode</code>	0x012A	Enable(1)	Enables / disables pull-up/down on configured RF switch and IRQ DIOs for sleep modes

18.3 Radio Configuration / Status Operations

Table 18-3: 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)	Sets the 64-bit (G)FSK syncword
SetLoRaPublicNetwork	0x0208	PublicNetwork(1)	Changes LoRa sync word for private or public network
SetRx	0x0209	RxTimeout(3)	Sets chip into RX mode
SetTx	0x020A	TxTimeout(3)	Sets chip into TX mode
SetRfFrequency	0x020B	RfFreq(4)	Sets PLL frequency
AutoTxRx	0x020C	Delay(3) IntermediaryMode(1) Timeout2(3)	Activates or deactivates the auto TX auto RX mode
SetCadParams	0x020D	SymbolNum(1) DetPeak(1) DetMin(1) CadExitMode(1) Timeout(3)	Configures LoRa CAD mode
SetPacketType	0x020E	PacketType(1)	Defines radio protocol
SetModulationParams	0x020F	Params ¹	Configures modulation parameters ¹
SetPacketParams	0x0210	Params ¹	Configures packet parameters ¹
SetTxParams	0x0211	TxPower(1) RampTime(1)	Sets TX power and ramp time
SetPacketAdrs	0x0212	NodeAddr(1) BroadcastAddr(1)	Sets 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)	Starts RX Duty Cycle mode

Table 18-3: Radio Configuration / Status Operations (Sheet 2 of 2)

Command	Opcode	Parameters	Description
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
SetRangingAddr	0x021C	Addr(4), Checklength(1)	Sets the ranging id of the device and the number of bytes to be checked on a request
SetRangingReqAddr	0x021D	ReqAddr(4)	Sets the ranging id for the request
GetRangingResult	0x021E	Type(1)	Gets the ranging results (distance, RSSI)
SetRangingTxRxDelay	0x021F	Delay(4)	Sets the TxRx delay for the ranging
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
SetRangingParameter	0x0228	SymbNb(1)	Sets the number of synchronization symbols
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
ConfigBleBeacon	0x022E	Channel(1), Payload(N+3)	Configures radio to be Bluetooth Low Energy beaconing compatible
GetLoRaRxHeaderInfos	0x0230	---	Gets last packet header info
BleBeaconSend	0x0231	Channel(1), Payload(N+3)	Configures radio to be Bluetooth Low Energy beaconing compatible and sends packet

1. Parameters are modem/packet dependant.

18.4 Wi-Fi Configuration / Status Operations

Table 18-4: Wi-Fi Scanning Configuration / Status Operations

Command	Opcode	Parameters	Description
<code>WifiScan</code>	0x0300	SignalType(1) ChanMask(2) AcqMode(1) NbMaxRes(1) NbScanPerChan(1) Timeout(2) AbortOnTimeout(1)	Launches a Wi-Fi passive scan
<code>WifiScanTimeLimit</code>	0x0301	SignalType(1) ChanMask(2) AcqMode(1) NbMaxRes(1) ScanTimePerChannel(2) TimeoutPerScan(2)	Searches for Wi-Fi MAC address
<code>WifiCountryCode</code>	0x0302	ChanMask(2) NbMaxRes(1) NbScanPerChan(1) Timeout(2) AbortOnTimeout(1)	Extracts Country code from Beacon or Probe Response
<code>WifiCountryCodeTimeLimit</code>	0x0303	ChanMask(2) NbMaxRes(1) ScanTimePerChannel(2) TimeoutPerScan(2)	Searches for Wi-Fi MAC addresses during a configurable maximal amount of time
<code>WifiGetNbResults</code>	0x0305	---	Gets the number of passive scanning results
<code>WifiReadResults</code>	0x0306	Index(1), NbResults(1) Format(1)	Returns Wi-Fi results
<code>WifiResetCumulTimings</code>	0x0307	---	Initializes cumulative times per phases for power consumption measurements
<code>WifiReadCumulTimings</code>	0x0308	---	Returns cumulative time per phase for power consumption measurements
<code>WifiGetNbCountryCodeResults</code>	0x0309	---	Returns number of results after Country Code scanning execution
<code>WifiReadCountryCodeResults</code>	0x030A	Index(1) NbResults(1)	Reads byte stream containing defined number of Wi-Fi Passive Scanning Country Code results from a given index
<code>WifiCfgTimestampAPphone</code>	0x030B	Timestamp(31:0)	Configures time-stamp threshold used to discriminate mobile access point from gateways
<code>WifiReadVersion</code>	0x0320	---	Returns internal Wi-Fi firmware version major and minor numbers

18.5 GNSS Configuration / Status Operations

Table 18-5: GNSS Scanning Configuration / Status Operations

Command	Opcode	Parameters	Description
GnssSetConstellationToUse	0x0400	ConstellationBitMask(1)	Sets GNSS constellation to use for GNSS scan
GnssReadConstellationToUse	0x0401	-	Reads GNSS constellation to use for GNSS scan
GnssSetAlmanacUpdate	0x0402	AlmanacUpdateBitMask(1)	Configures constellation almanac information to be updated
GnssReadAlmanacUpdate	0x0403		Reads almanac update information
GnssReadVersion	0x0406		Reads internal GNSS firmware and almanac versions.
GnssReadSupportedConstellations	0x0407	-	Reads supported GNSS constellations
GnssSetMode	0x0408	GnssMode(1)	Configures GNSS scan as single or dual capture
GnssAutonomous	0x0409	Time(4), EffortMode(1) ResultMask(1), NbSvMax(1)	Triggers GNSS autonomous scanning
GnssAssisted	0x040A	Time(4), EffortMode(1) ResultMask(1), NbSvMax(1)	Triggers GNSS assisted scanning
GnssSetAssistancePosition	0x0410	Latitude(2), Longitude(2)	Configures approx position for GNSS assisted mode
GnssReadAssistancePosition	0x0411	-	Reads the assistance position
GnssPushSolverMsg	0x0414	Payload(n)	Pushes messages from GNSS solver to LR1110
GnssPushDmMsg	0x0415	Payload(n)	Pushes messages from LoRaWAN network to LR1110
GnssGetContextStatus	0x0416	-	Reads the context status
GnssGetNbSvDetected	0x0417	-	Returns number of SV detected during last GNSS scan
GnssGetSvDetected	0x0418	-	Returns list of SV detected during the last GNSS scan, with their C/N0
GnssGetConsumption	0x0419	-	Returns radio capture and CPI processing duration of the last GNSS scan
GnssGetResultSize	0x040C	-	Returns the results payload size
GnssReadResults	0x040D	-	Returns the results payload byte stream
GnssAlmanacFullUpdate	0x040E	AlmanacFullUpdatePayload (2580)	Updates all the Almanac Data
GnssGetSvVisible	0x041F	Time(4), Latitude(2) Longitude(2) Constellation(1)	Returns number of visible SVs

18.6 CryptoElement Configuration / Status Operations

Table 18-6: 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
CryptoCheckEncryptedFirmwareImage	0x050F	Offset(31:0) Data[1 to N]	Adds a chunk to the encrypted firmware image to be checked
CryptoCheckEncryptedFirmwareImage Result	0x0510	-	Gets the result of the check

18.7 Bootloader Commands

Table 18-7: Bootloader Commands¹

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.

19. Revision History

The following table details the versions of the User Manual documentation issued, and the corresponding LR1110 versions supported (Use Case and FW Major.FW Minor), as returned by command GetVersion(...).

Table 19-1: Revision History

User Manual Version	ECO	Date	Applicable to	Changes
1.0	050946	March 2020	Use Case: 01 FW Version: 03.02 or later	First Release Clarified GNSS NAV message to be sent to Geolocation solver. Revised et corrected Section 14. and Section 15. Corrected Table 6-2 , Table 18-2 , Table 18-5 . Use Case: 01 FW Version: 03.03 to 03.05
1.1	053430	Sept. 2020	Use Case: 01 FW Version: 03.03 to 03.05	Added Section 6.2.2.1 Modified Section 11.3.2 Added Section 11.3.8 Replaced DMC by GNSS Almanac Update. Clarified <i>ResultMask</i> field of commands <i>GNSSAutonomous(...)</i> and <i>GNSSAssisted(...)</i> in Section 11.3
1.2	057439	July 2021	Use Case: 01 FW Version: 03.06 or later	Added EnableSpiCrc Added SetLoRaSyncWord Corrected bits in WifiScanTimeLimit. Added WifiCountryCodeTimeLimit, WifiCountryCode, WifiScanTimeLimit, WiFiGetNbCountryCodeResults, WiFiReadCountryCodeResults, WiFiCfgTimestampAPphone, WiFiReadVersion Added GnssReadConstellationToUse, GnssReadSupportedConstellations, GnssScanContinuous, GnssReadAssistance, GnssGetContextStatusPosition Added bootloader commands Added AcqMode 0x05: SSID Beacon search mode Added DriveDiosInSleepMode Some reformatting

Table 19-1: Revision History (Continued)

User Manual Version	ECO	Date	Applicable to	Changes
1.3	059149	Oct 2021	Use Case: 01 FW Version: 03.07 or later	Added GetLoRaRxHeaderInfos(...), GnssReadVersion(...), GnssSetAlmanacUpdate(...), GnssReadAlmanacUpdate(...), GnssGetSvVisible(...) Modified SetPacketParams.PayloadLen field Modified GnssSetMode(...), GnssAutonomous(...), GnssAssisted(...), GnssGetSvDetected(...) Modified Section 11.4 GNSS Scanning Results & Commands Removed GnssScanContinuous(...)
1.4	059758	Dec 2021	Use Case: 01 FW Version: 03.07 or later	Add relevant information for support of LR-FHSS modulation
1.5	061375	Apr 2022	Use Case: 01 FW Version: 03.07 or later	Changed: <ul style="list-style-type: none">• Device & Application Service (DAS) to Modem & Geolocation service (MGS)• LoRaWAN® protocol to LoRaWAN® Standard Added Known Limitations chapter
1.6	066719	May 2023	Use Case: 01 FW Version: 03.07 or later	Added: <ul style="list-style-type: none">• GnssPushDmMsg• GnssPushSolverMsg• EraseInfoPage, ReadInfoPage, WriteInfoPage• Section 2.5 Info Page• Section 2.1.5 Sleep: “all DIOs are in Hi-Z mode”• GNSS information to Section 11.1 & Section 11.2 about GNSS Dual Constellation and GNSS Advanced scan mode• References to AN1200.74 LoRa Edge Clock Requirements and AN1200.59• Section 11.2 info about external LNA• Return code CMD_PERR for ReadInfoPage command Modified: <ul style="list-style-type: none">• GnssReadAlmanacUpdate description• Section 6.2.1 32.768kHz Clock• SetRssiCalibration parameter units to dB• Note 1 in Section 18.7 Bootloader Commands• SetModulationParams in Table 18-3 with Note 1• Section 8.5.3 SetGfskSyncWord: explanation for syncword not bit-aligned• Section 9.1 and Section 9.4.1 improved• Table 9-1, Table 9-2, Table 9-5• Mode Transitions and Timings

Table 19-1: Revision History (Continued)

User Manual Version	ECO	Date	Applicable to	Changes
1.7	067386	June 2023	Use Case: 01 FW Version: 03.08 or later	<p>Added</p> <ul style="list-style-type: none">Section 12. Sub-GHz RangingSection 13. Bluetooth® Low Energy Beacons CompatibilitySection 8.8 Sigfox ModulationSection 6.3.2 TCXO supply voltage precisionCryptoCheckEncryptedFirmwareImage()CryptoCheckEncryptedFirmwareImageResult()Section 9.3.3 High frequency PA (HF PA)Information to Section 8.10 RSSI FunctionalityBit 27 to Table 4-2: IrqToEnable Interruption Mapping <p>Modified</p> <ul style="list-style-type: none">Section 7.1 phrase about SX126x compatibilityGFSK SetModulationParams BitRate & RC definitionsLoRa SetModulationParams long interleaver descriptionGetPacketStatus SnrPkt definitionTable 2-8: Sleep Mode SummaryFigure 9-13: RF Switch, Double PA OperationSection 7.1 first phraseSection 11.3.10 ErrorCode definitionSection 8.1.1Section 8.1.2 definitions

Glossary

List of Acronyms and their Meaning (Sheet 1 of 3)

Acronym	Meaning
ADC	Analog-to-Digital Converter
AP	Wi-Fi Access Point
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 3)

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
MAC	Wi-Fi Media Access Control
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
PLCP	Wi-Fi Physical Layer Conformance Procedure
PLL	Phase-Locked Loop
POR	Power On Reset
PSDU	Wi-Fi PLCP Service Data Unit
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
STA	Wi-Fi Client Station
STDBY	Standby

List of Acronyms and their Meaning (Sheet 3 of 3)

Acronym	Meaning
TCXO	Temperature-Compensated Crystal Oscillator
XOSC	Crystal Oscillator



IMPORTANT NOTICE

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Semtech assumes no liability for any errors in this document, or for the application or design described herein. Semtech reserves the right to make changes to the product or this document at any time without notice. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Semtech warrants performance of its products to the specifications applicable at the time of sale, and all sales are made in accordance with Semtech's standard terms and conditions of sale.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS, OR IN NUCLEAR APPLICATIONS IN WHICH THE FAILURE COULD BE REASONABLY EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the customer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

The Semtech name and logo are registered trademarks of the Semtech Corporation. All other trademarks and trade names mentioned may be marks and names of Semtech or their respective companies. Semtech reserves the right to make changes to, or discontinue any products described in this document without further notice. Semtech makes no warranty, representation or guarantee, express or implied, regarding the suitability of its products for any particular purpose. All rights reserved.

© Semtech 2023

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