SX1280/SX1281

Long Range, Low Power, 2.4 GHz Transceiver with Ranging Capability

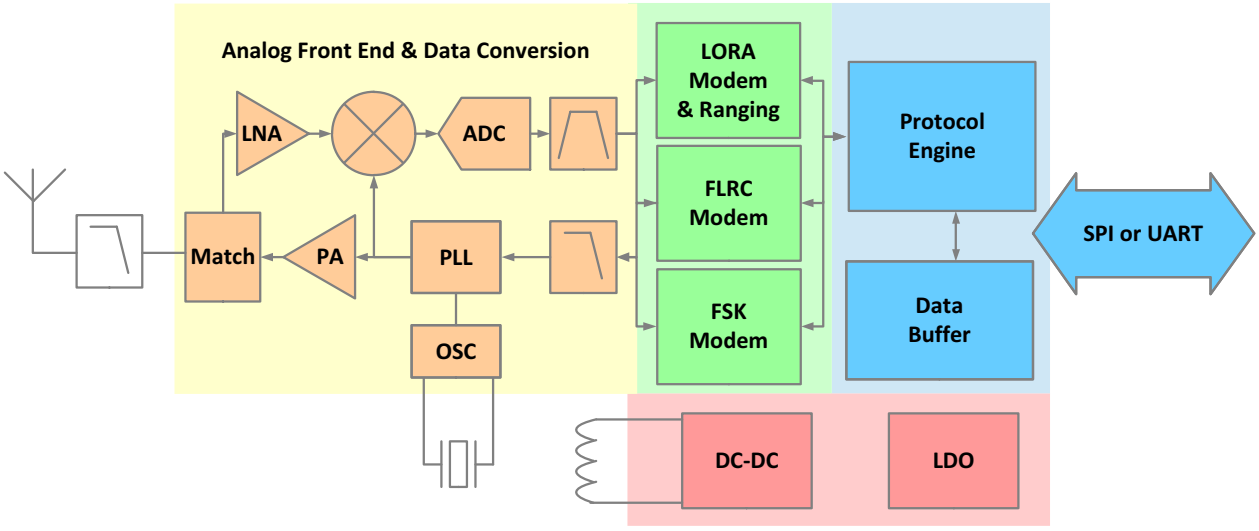


Figure A: Transceiver Block Diagram

General Description

The SX1280 and SX1281 transceivers provide ultra long range communication in the 2.4 GHz band with the linearity to withstand heavy interference. This makes them the ideal solution for robust and reliable wireless solutions. They are the first ISM band transceiver IC of their kind to integrate a time-of-flight functionality, opening up application solutions to track and localize people, pets, drones, or objects in a factory. These long range 2.4 GHz products include multiple physical layers and modulations to optimize long range communication at high data rate for video and security applications. Very small products for wearables can easily be designed thanks to the high level of integration and the ultra-low current consumption which allows the use of miniaturized batteries.

The radio is fully compliant with all worldwide 2.4 GHz radio regulations including EN 300 440, FCC CFR 47 Part 15 and the Japanese ARIB STD-T66.

The level of integration, low consumption and ranging function within the long range 2.4 GHz product line enable enhanced connectivity and provide additional functionality to a new generation of previously unconnected devices and applications.

Key Features

- Long Range 2.4 GHz transceiver
- High sensitivity, down to -132 dBm
- +12.5 dBm, high efficiency PA
- Low energy consumption, on-chip DC-DC
- LoRa, FLRC, (G)FSK supported modulations
- Programmable bit rate
- Excellent blocking immunity
- Ranging Engine, Time-of-flight function
- BLE PHY layer compatibility
- Low system cost

Applications

- Home automation & appliances
- Security systems
- Tracking applications
- Wearables & sports/fitness sensors
- Radio-controlled toys & drones
- Smart watches & beacons
- Healthcare

Ordering Information

Part Number	Delivery	Order Quantity
SX1280IMLTRT	Tape & Reel	3'000 pieces
SX1281IMLTRT	Tape & Reel	3'000 pieces

QFN 24 Package, with the temperature operating range from -40 to 85°C

Pb-free, Halogen free, RoHS/WEEE compliant product

Revision History

Version	ECO	Date	Changes and/or Modifications
Rev 1.0	035543	February 2017	First Release
Rev 1.1	037029	May 2017	Added table of effective data rates for the LoRa Modem Correction of the formulas for time-on-air in LoRa Correction of typos in the chapter Host Controller Interface Update of the application schematic with optional TCXO Update of the reference design BOM Deletion of redundant information in the chapter Thermal Impedance

Table of Contents

List of Figures

List of Tables

1. Introduction

The SX1280 and SX1281 are half-duplex transceivers capable of low power operation in the worldwide 2.4 GHz ISM band. The radio comprises 5 main parts, which are described in the following chapters.

1.1 Analog Front End

The radio features a high efficiency +12.5 dBm transmitter and a high linearity receive chain that are both accessed via a common antenna port pin. Frequency conversion between RF and baseband (low-IF) is governed by a digital PLL that is referenced to a 52 MHz crystal. Both transmit and receive chains are interfaced by data converters to the ensuing digital blocks. For more information see the Section 4. "Analog Front End" on page 24 .

1.2 Power Distribution

Two forms of voltage regulation are available, either a integrated Low-DropOut (LDO) or a high efficiency buck (step down) DC to DC converter. This allows the designer to choose between high energy efficiency or miniaturisation of the radio depending upon the design priorities of the application. For more information, please see the Section 5. "Power Distribution" on page 27 .

1.3 Modem

There are a range of modulation options available in the LoRa family's three modems, each of which has packet options that include many MAC layer functionalities. For a description of each modulation format and the performance benefits associated with that modulation, please see the corresponding section below:

- LoRa Modem and Packet: Section 6.2 "LoRa Modem" on page 30
- FLRC Modem and Packet: Section 6.3 "FLRC Modem" on page 32
- FSK Modem and Packet: Section 6.4 "FSK Modem" on page 35

The long range 2.4 GHz product line also features the Ranging Engine, a long distance ranging functionality that permits time-of-flight measurement between a pair of LoRa radios. The availability of each modem and the Ranging Engine, for each part number in the long range 2.4 GHz product line is shown below.

Table 1-1: Product Portfolio and Modem Functionality

Product Reference	SX1280	SX1281
LoRa	✓	✓
FLRC	✓	✓
GFSK	✓	✓
Ranging Engine	✓	

1.4 Packet Processing

The radio can operate in a fully automatic mode where the processing of packets for transmission or reception can be performed without the intervention of an external host microcontroller. For more details see Section 7. "Packet Engine" on page 38 .

In both transmit and receive modes the payload interface to the transceiver is the packet data buffer described in Section 8. "Data Buffer" on page 48 of this datasheet.

1.5 Digital Interface and Control

The specification and processing for all digital communication with the transceiver is described in Section 9. "Digital Interface and Control" on page 50 . This includes descriptions of the SPI and UART interfaces, that can be used to configure the transceiver together with the Digital Input / Output (DIO) that are used to send interrupts to an external host microcontroller.

- For the SPI interface see Section 9.3 "SPI Interface" on page 51
- For the UART interface see Section 9.4 "UART Interface" on page 54
- For the DIO see Section 9.6 "Multi-Purpose Digital Input/Output (DIO)" on page 54

2. Pin Connections

2.1 Transceiver Pinout

Table 2-1: Transceiver Pinout

Pin Number	Pin Name	Type (I = input O = Ouptut)	SPI description	UART description
0	GND	-	Exposed Ground pad	
1	VR_PA	-	Regulated supply for the PA	
2	VBAT_IN	I	Regulated supply input. Connect to Pin 12.	
3	NRESET	I	Reset signal, active low with internal pull-up at 50 kΩ	
4	XTA	-	Reference oscillator connection or TCXO input	
5	GND	-	Ground	
6	XTB	-	Reference oscillator connection	
7	BUSY	O	Transceiver busy indicator	
8	DIO1	I/O	Optional multi-purpose digital I/O	
9	DIO2	I/O	Optional multi-purpose digital I/O	
10	DIO3	I/O	Optional multi-purpose digital I/O	
11	VBAT_IO	I	Supply for the Digital IO interface (1.8 V to 3.7 V). Must be ≤ VBAT.	
12	DCC_FB	O	Regulated output voltage from the internal regulator	
13	GND	-	Ground	
14	DCC_SW	O	DC-DC Switcher Output	
15	VBAT	I	Supply for the RFIC (1.8 V to 3.7 V). Must be ≥ VBAT_IO.	
16	MISO_TX	O	SPI slave output	UART Transmit pin
17	MOSI_RX	I	SPI slave input	UART Receive pin
18	SCK_RTSN	I	SPI clock	UART Request To Send
19	NSS_CTS	I	SPI Slave Select	UART Clear To Send
20	GND	-	Ground	
21	GND	-	Ground	
22	RFIO	I/O	RF transmit output and receive input	
23	GND	-	Ground	
24	GND	-	Ground	

2.2 Package view

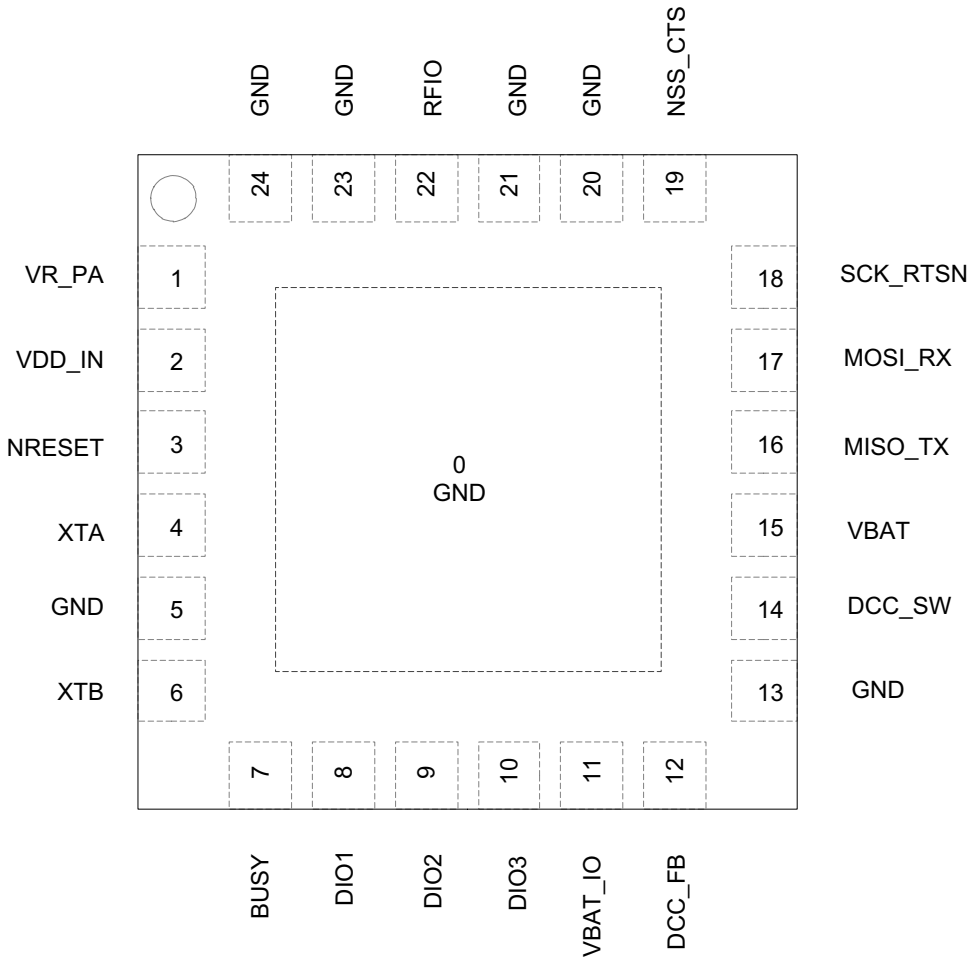


Figure 2-1: Transceiver Pin Locations

3. Specifications

The following specifications are given for the typical operating conditions of VBAT_IO = VBAT = 3.3 V, temperature = 25 °C, crystal oscillator frequency = 52 MHz, RF centre frequency = 2.4 GHz. All RF impedances are matched using the reference design, see Section 14.1 "Reference Design" on page 122. Blocking, ACR and co-channel rejection are given for a single tone interferer and referenced to sensitivity level +6 dB. The current supply is given as the sum of current on VBAT and VBAT_IO. The buck converter (DC-DC) is considered switched ON unless otherwise stated.

3.1 ESD Notice

The SX1280/SX1281 transceivers are high-performance radio frequency devices.

They all satisfy:

- Class 2 of the JEDEC standard JESD22-A114 (Human Body Model) on all pins
- Class III of the JEDEC standard JESD22-C101 (Charged Device Model) on all pins

3.2 Absolute Minimum and Maximum Ratings

Table 3-1: Minimum and Maximum Ratings

Symbol	Description	Minimum	Typical	Maximum	Unit
VBATmr	Supply voltage on VBAT and VBAT_IO	-0.5	-	3.9	V
Tmr	Temperature	-55	-	115	°C
Pmr	RF Input level	-	-	10	dBm

3.3 Operating Range

Table 3-2: Operating Range

Symbol	Description	Minimum	Typical	Maximum	Unit
VBATop	Supply voltage VBAT and VBAT_IO	1.8	-	3.7	V
Top	Temperature under bias	-40	-	85	°C
Clop	Load capacitance on digital ports	-	-	10	pF
ML	RF Input power	-	-	10	dBm

3.4 General Electrical Specifications

Table 3-3: General Electrical Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IDDSL	Supply current in Sleep mode with data buffer retained	-	0.215	1.0	µA
	Supply current in Sleep mode with register content retained (context saved) and data buffer not retained	-	0.25	1.0	µA
	Supply current in Sleep mode with instruction RAM retained	-	0.4	1.0	µA
	Supply current in Sleep mode with data buffer, instruction RAM, data RAM retained. RC running	-	1.2	1.8	µA
IDDSTDBYRC	Supply current in STDBY_RC mode	-	760	-	µA
IDDSTDBYXOSC	Supply current in STDBY_XOSC mode	-	1.2	-	mA
IDDFS	Supply current in FS mode	-	2.8	-	mA
FR	Synthesizer frequency range	2400	-	2500	MHz
FSTEP	Synthesizer frequency step (52 MHz reference)	-	198	-	Hz
PHN	Phase noise at 2.45 GHz				
	1 MHz offset	-	-117	-	dBc/Hz
	10 MHz offset	-	-133	-	dBc/Hz
FXOSC	Crystal oscillator frequency	-	52	-	MHz
TS_FS	Frequency synthesizer wake-up time with XOSC enabled	-	54	-	µs
TS_HOP	Frequency synthesizer hop time to within 10 kHz of target frequency				
	1 MHz	-	20	-	µs
	10 MHz	-	30	-	µs
	100 MHz	-	50	-	µs
TS_OS	Crystal oscillator wake-up time from STDBY_RC mode	-	100	-	µs

For the digital specifications, see Table 10-2: "Switching Time (TswMode) for all Possible Transitions" on page 58.

3.5 Receiver Electrical Specifications

All receiver sensitivity numbers are given for a Packer Error Rate (PER) of 1%, for packet with 10 bytes of payload.

Values are given for maximum AGC gain which is the highest low power gain.

A continuous wave (CW) interferer is used for all blocking and rejection measurements unless otherwise stated.

3.5.1 Receiver Specifications

Table 3-4: Receiver Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IIP3	3rd Order input intercept for maximum low power gain setting				
	In-band interferer <6 MHz	-	-25	-	dBm
	In-band interferer @ 6 MHz	-	-6	-	dBm
	In-band interferer @10 MHz	-	-4	-	dBm
	In-band interferer @ 20 MHz	-	-4	-	dBm
IMR	Image rejection (CW tone 1% PER)	-	30	-	dB

3.5.2 LoRa Modem

Table 3-5: LoRa Modem Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IDDRXLP_L	Supply current for low power mode				
	for BW = 203 kHz	-	5.5	-	mA
	for BW = 406 kHz	-	6.0	-	mA
	for BW = 812 kHz	-	7.0	-	mA
	for BW = 1625 kHz	-	7.5	-	mA
IDDRXHS_L	Supply current for high sensitivity mode				
	for BW = 203 kHz	-	6.2	-	mA
	for BW = 406 kHz	-	6.7	-	mA
	for BW = 812 kHz	-	7.7	-	mA
	for BW = 1625 kHz	-	8.2	-	mA
RB_L	LoRa bitrate programmable range with CR = 4/5				
	SF5, BW = 1625 kHz	-	202	-	kb/s
	SF6, BW = 1625 kHz	-	122	-	kb/s
	SF7, BW = 1625 kHz	-	71	-	kb/s
	SF12, BW = 203 kHz	-	0.476	-	kb/s
BW_L	LoRa bandwidth programmable range	203	-	1625	kHz
RFSLP_L	LoRa receiver sensitivity with CR = 4/5 and low power mode enabled [1]				
	SF7, BW = 1625 kHz,	-	-106	-	dBm
	SF12, BW = 203 kHz	-	-130	-	dBm
RFSHS_L	LoRa receiver sensitivity with CR = 4/5 and high sensitivity mode enabled 1				
	SF7, BW = 1625 kHz,	-	-109	-	dBm
	SF12, BW = 203 kHz	-	-132	-	dBm
CCR_L	Co-channel rejection LoRa				
	SF7	-	7.5	-	dB
	SF12	-	19.5	-	dB
BI_L	Blocking immunity SF12				
	+/- 1 MHz	-	60	-	dB
	+/- 2 MHz	-	63	-	dB
	+/- 10 MHz	-	81	-	dB

Table 3-5: LoRa Modem Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
ACR_L	Adjacent channel rejection at 1.5 BW of CW				
	SF = 12, BW = 203 kHz	-	37	-	dB
	SF = 7, BW = 1.6 MHz	-	37	-	dB

1. See Section 4.2.1 "Low Power Mode and High Sensitivity Mode" on page 26.

3.5.3 FLRC Modem

Table 3-6: FLRC Modem Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IDDRX_FL	Supply currents				
	BW = 300 kHz, BR = 260 kb/s	-	6.5	-	mA
	BW = 1200 kHz, BR = 1300 kb/s	-	8.6	-	mA
RB_FL	FLRC Modem programmable bitrate	260	-	1300	kb/s
BW_FL	Programmable channel bandwidth range	300	-	2400	kHz
RFS_FL	FLRC Receiver Sensitivity				
	260 kSymb/s, 130 kb/s BW = 300 kHz CR=1/2	-	-106	-	dBm
	2.6 MSymb/s, 1.3 Mb/s, BW = 2.4 MHz, CR=1/2	-	-97	-	dBm
CCR_FL	Co-channel rejection FLRC	-	-10	-	dB
BI_FL	Blocker level for Max low power gain setting				
	+/- 1 MHz	-	41	-	dB
	+/- 2 MHz	-	44	-	dB
	+/- 10 MHz	-	62	-	dB
	+/- 20 MHz	-	69	-	dB
ACR_FL	Adjacent channel rejection at 1.5 BW for CW				
	260 kb/s, BW = 300 kHz	-	44	-	dB
	1.3 Mb/s, BW = 2.4 MHz	-	49	-	dB

Notice: all data rates listed in the table above are in raw bits. All values are given with BT = 0.5.

3.5.4 FSK Modem

Table 3-7: FSK Modem Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IDDRX_FSK_250_LP	Supply currents for low power mode, demodulation running[1]				
	BW = 300 kHz, BR = 250 kb/s	-	4.8	-	mA
IDDRX_FSK_1000_LP	BW = 1200 kHz, BR = 1000 kb/s	-	5.3	-	mA
IDDRX_FSK_2000_LP	BW = 2400 kHz, BR = 2000 kb/s	-	5.7	-	mA
IDDRX_FSK_250_HS	Supply currents for high sensitivity mode, demodulation running[1]				
	BW = 300 kHz, BR = 250 kb/s	-	5.5	-	mA
IDDRX_FSK_1000_HS	BW = 1200 kHz, BR = 1000 kb/s	-	6.0	-	mA
IDDRX_FSK_2000_HS	BW = 2400 kHz, BR = 2000 kb/s	-	6.4	-	mA
BR_FSK	FSK Modem programmable bitrate	125	-	2000	kb/s
BW_FSK	Programmable channel bandwidth range DSB	300	-	2400	kHz
RFS_FSK1 low power mode	FSK Receiver Sensitivity BER 0.1%				
	250 kb/s, β = 0.5, BW = 300 kHz	-	-100	-	dBm
	1 Mb/s, β = 0.5, BW = 1200 kHz	-	-94	-	dBm
RFS_FSK1_HS high sensitivity mode	FSK Receiver Sensitivity BER 0.1%				
	250 kb/s, β = 0.5, BW = 300 kHz	-	-102	-	dBm
	1 Mb/s, β = 0.5, BW = 1200 kHz	-	-96	-	dBm
RFS_FSK2 low power mode	FSK Receiver Sensitivity PER 1%				
	250 kb/s, β = 0.5, BW = 300 kHz	-	-93	-	dBm
	1 Mb/s, β = 0.5, BW = 1200 kHz	-	-88	-	dBm
RFS_FSK2_HS high sensitivity mode	FSK Receiver Sensitivity PER 1%				
	250 kb/s, β = 0.5, BW = 300 kHz	-	-94	-	dBm
	1 Mb/s, β = 0.5, BW = 1200 kHz	-	-90	-	dBm
CCR_FSK	Co-Channel Rejection	-	-10	-	dB
BI_FSK	Blocker level for max low power gain setting, BR = 250 kb/s, BW = 300 kHz				
	+/- 1 MHz	-	41	-	dB
	+/- 2 MHz	-	44	-	dB
	+/- 10 MHz	-	62	-	dB
	+/- 20 MHz	-	69	-	dB

Table 3-7: FSK Modem Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
ACR_FSK	Adjacent channel rejection at 1.5 BW for CW				
	BW = 300 kHz	-	34	-	dB
	BW = 1200 kHz	-	34	-	dB

1. See Section 4.2.1 "Low Power Mode and High Sensitivity Mode" on page 26.

Notice: all values listed in the table above are given with the modulation index β = 0.5.

3.6 Transmitter Electrical Specifications

Table 3-8: Transmitter Electrical Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
IDD_T13	12.5 dBm	-	24	-	mA
IDD_T10	10 dBm	-	18	-	mA
IDD_T0	0 dBm	-	10	-	mA
RFOPMIN	Minimum RF output power	-	-18	-	dBm
RFOPMAX	Maximum RF output power	-	12.5	-	dBm
FDA	Programmable FSK frequency deviation	62.5	-	1000	kHz

3.7 Crystal Oscillator Specifications

Table 3-9: Crystal Oscillator Specifications

Symbol	Description	Minimum	Typical	Maximum	Unit
FXOSC	Crystal oscillator frequency	-	52	-	MHz
CLOAD	Crystal loading capacitance	-	10	-	pF
C0XTAL	Crystal shunt capacitance	-	2	5	pF
RSXTAL	Crystal series resistance	-	10	50[1]	Ω
CMXTAL	Crystal motional capacitance	3	3.5[2]	4	fF

1. An RSXTAL of up to 90 Ω may be used if C0XTAL is restricted to < 3 pF.
2. Other CMXTAL values may be used, noting that smaller values reduce start up time whilst larger values will degrade frequency accuracy and phase noise.

3.8 Digital Pin Levels

Table 3-10: Digital Levels and Timings

Symbol	Description	Minimum	Typical	Maximum	Unit	Conditions
V_{IH}	Digital input level high	0.8	-	-	VBAT_IO	-
V_{IL}	Digital input level low	-	-	0.2	VBAT_IO	-
V_{OH}	Digital output level high	0.9	-	-	VBAT_IO	I_{max} = 2.5 mA
V_{OL}	Digital output level low	-	-	0.1	VBAT_IO	I_{max} = -2.5 mA
I_{Leak}	Digital input leakage current (NSS, MOSI, SCK)	-1	-	1	µA	-

4. Analog Front End

The analog front end features a single antenna port connection to an integrated matching circuit that permits half- duplex operation of the radio without external RF switching.

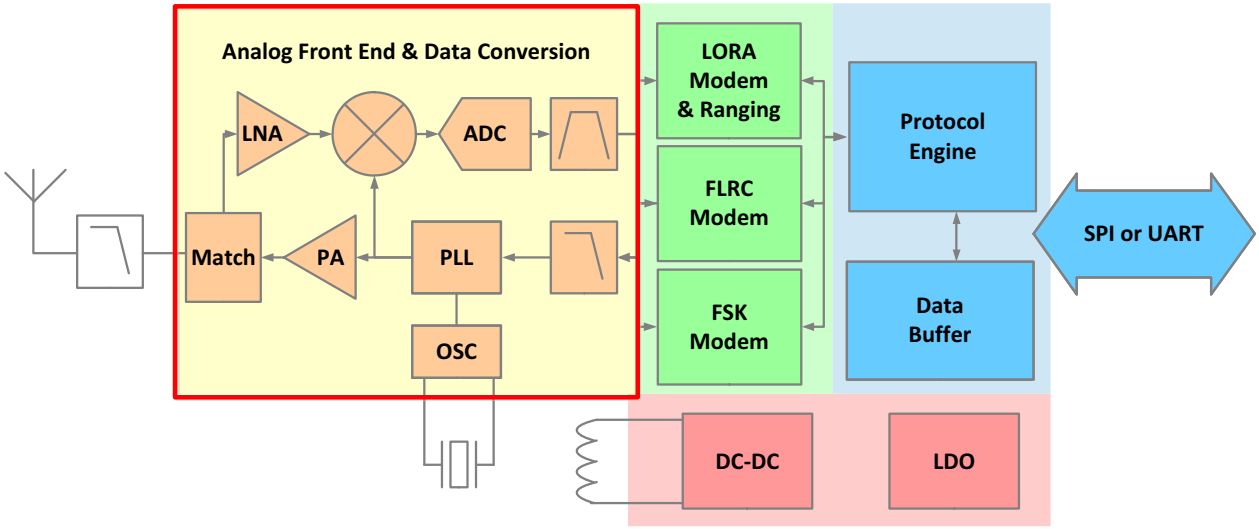


Figure 4-1: Transceiver Block Diagram, Analog Front End Highlighted

4.1 Transmitter

The transmit chain comprises the modulated output from the modem bank which directly modulates the fractional-N PLL. An optional pre-filtering of the bit stream can be enabled to reduce the power in the adjacent channels, also dependent upon the selected modulation type.

The transmitter is enabled by using the *SetTx(periodBase, periodBaseCount)* command. Upon issuing this command, the transmitter sends the packet stored in the data buffer. The transmitter then returns to STDBY_RC mode, either upon completion of the packet transmission, or after a predefined time-out period defined by the time base of the interrupt timer, *periodBase*, and the preset number of clock ticks *periodBaseCount* as in Section 12. "List of Commands" on page 83.

The RF output power of the transmitter is controllable in 1 dB increments in the range -18 dBm to +12 dBm, the final power step is then a 0.5 dB increment to the maximum transmitter output power of 12.5 dBm. The RF output power (PRF) and the ramp time are determined by the command *SetTxParam(power, rampTime)*. The output power is set using the formula:

$$P_{RF[dBm]} = -18 + power$$

Where the maximum output power, $P_{RF[dBm]}$, is 12.5 dBm.

This corresponds to the RF output power at the antenna feedpoint of the reference design (see Section 14.1.1 "Application Design Schematic" on page 122). Abrupt switching of an RF power amplifier can cause undesirable spurious spectral emissions. A precision DAC is therefore used as a reference for the transceiver PA supply allowing smooth transition to transmit mode. The time over which the PA is ramped, prior to packet transmission, *rampTime* can be varied from 2 to 20 µs accordingly. In some applications and for regulatory testing purposes it can be useful to generate a continuous wave (CW) tone in transmit mode or enable a continuously modulated output. These two functionalities are accessible through

the *SetTxContinuousWave()* and *SetTxContinuousPreamble()* functions. The latter provides a stream of alternating logical '1' and '0' modulated data using whatever configured modulation settings.

4.2 Receiver

LoRa, FLRC or FSK systems operate as a half-duplex low-IF/zero-IF transceiver. The received RF signal is first amplified by the LNA via the on-chip impedance matching network. The single-ended to differential conversion is performed afterwards to improve the second order linearity of the receiver. The signal is then down-converted to baseband or an intermediate frequency by quadrature mixers to obtain the I and Q signals. These signals are then low-pass filtered and finally digitized.

The receive chain employs an Automatic Gain Control (AGC) that is enabled by default and is used to ensure that the optimal front end gain is selected for reception of a given detected signal power. This can be disabled and the gain of the RF front end set manually. To do this the following registers must be configured:

Table 4-1: Procedure for Receiver Gain Manual Setting

Register	Bit	Value	Comments
0x89F	bit 7	1	Enable Manual Gain Control
0x895	bit 0	0	Enable Manual Gain Control
0x89E	bit 0:3	1 to 13	Manual Gain Setting (see following table)

The gain can then be set according to the settings indicated in the table below:

Table 4-2: Receiver Gain Manual Setting

Setting	Gain [dB]
13	Max
12	Max -2
11	Max -4
10	Max -6
9	Max -8
8	Max -12
7	Max -18
6	Max -24
5	Max -30
4	Max -36
3	Max -42
2	Max -48
1	Max -54

The procedure for reading from and writing to a control register is described in Section 12. "List of Commands" on page 83.

The transition to receive mode is made by issuing the *SetRx(periodBase, periodBaseCount)* command with the *periodBase* oscillator timebase and *periodBaseCount* number of clock ticks specifying the time-out upon which receive mode (see Section 10.5 "Receive (Rx) Mode" on page 56) will be exited to STDBY_RC mode. The process of periodic reception can be fully automated in the transceiver. This process and the processing specific to each modulation format are described in Section 13.1.3 "Rx Setting and Operations" on page 92.

When a signal or packet is received the transceiver reports a signal strength using the Received Signal Strength Indicator (RSSI). This information is returned with a *GetPacketStatus()* request as in Section 12. "List of Commands" on page 83.

4.2.1 Low Power Mode and High Sensitivity Mode

In receive mode, the SX1280 can operate in one of two distinct regimes of operation. Low power mode allows maximium efficiency of the SX1280 to be attained, optimizing the performance of the device for receiver current consumption. This is enabled by default and limits the gain of the receiver LNA from accessing the highest three steps of LNA gain.

Conversely, high sensitivity mode can be used to unlock the very lowest noise, highest sensitivity gain steps for a slight increase in receiver current consumption. High sensitivity mode is enabled by setting bits 7:6 at address *0x891* to *0x3*. Once enabled the noise figure of the receiver is improved by up to 3 dB for 500 µA of additional current consumption.

4.3 PLL

A fractional-N third order sigma-delta PLL acts as the frequency synthesizer for the LO (Local Oscillator) for both receiver and transmitter chains. The PLL is capable of fast auto-calibration with a low switching time. Modulation is performed automatically either within or outside the PLL bandwidth depending upon the selected modulation type.

The PLL frequency is derived from the crystal oscillator circuit which uses an external 52 MHz crystal reference. The PLL and reference frequency determine the RF centre frequency of the radio. With the default crystal reference frequency, FXosc, and FRF values this is set to 2.4 GHz. All other reference oscillator and PLL settings are automatically optimized for the selected modem settings.

To set the RF centre frequency of transceiver the *SetRFFrequency()* command is used. The frequency is passed as a 24-bit operand, *rfFrequency*, as shown below:

$$F_{RF} = \frac{FXosc}{2^{18}} rfFrequency$$

The PLL can be enabled individually by using the *SetFS()* command, which tunes the PLL to the transmit frequency. This is an intermediate mode that is automatically enabled on the transition from sleep or standby to transmit or receive modes.

4.4 RC Oscillators

Two RC oscillators are available: 64 kHz and 13 MHz RC oscillators. The 64 kHz RC oscillator is optionally used by the transceiver in Sleep mode to wake the transceiver to perform periodic or duty cycled operations. The 13 MHz RC oscillator is enabled for all SPI or UART communication to permit configuration of the device without starting the crystal oscillator.

The presence of the two oscillators allows ultra low consumption in Sleep mode with only the 64 kHz oscillator running, whereas once communication is initiated, the faster higher consumption 1.3 MHz oscillator is started to allow efficient high-speed communication with an external host processor. Optionally the crystal oscillator can be used instead of the RC oscillator in all modes other than sleep mode, as described in Section 10. "Operational Modes" on page 55.

5. Power Distribution

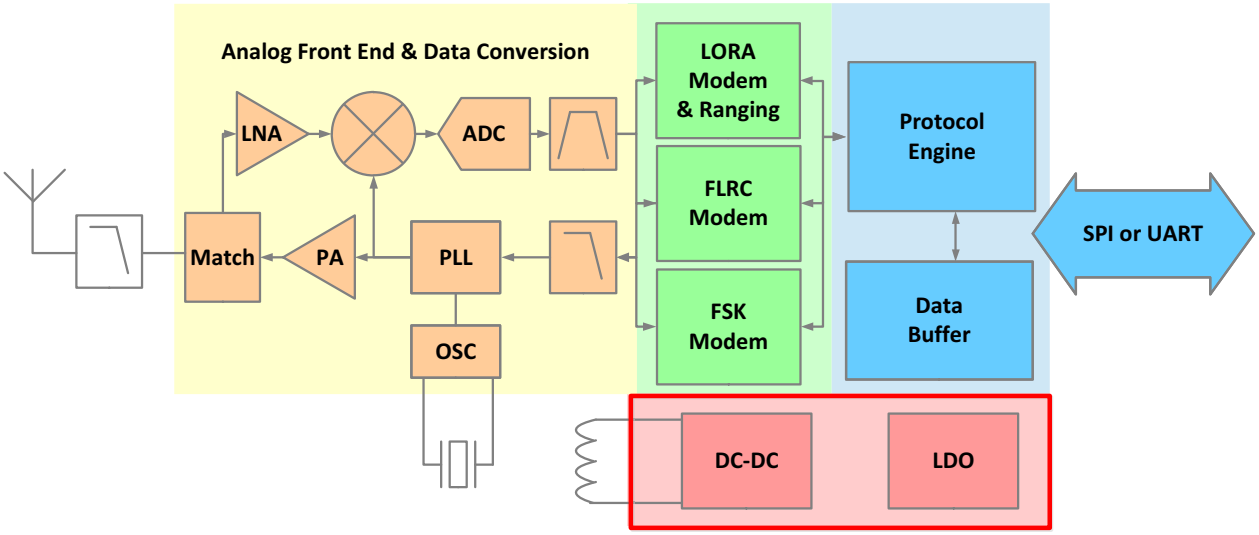


Figure 5-1: Transceiver Block Diagram, Power Distribution Highlighted

5.1 Selecting DC-DC Converter or LDO Regulation

Two forms of voltage regulation (DC-DC buck converter or linear regulator) are available depending upon the design priorities of the application. By default the linear LDO regulator is used in all modes. Alternatively a high efficiency DC to DC buck converter (DC-DC) can be enabled in FS, Rx and Tx modes.

All specifications of the transceiver are given with the DC-DC regulator enabled. For applications where cost and size are constrained, LDO-only operation is possible which negates the need for the 15 µH inductor between pins 12 and 14, conferring the following benefits:

- Reduced Bill Of Materials
- Reduced board space

Conversely, the energy consumption of the radio will be increased. The following table illustrates the power regulation options for different modes and user settings.

Table 5-1: Regulation Type versus Circuit Mode

Circuit Mode	Sleep	STDBY_RC	STDBY_XOSC	FS	Rx	Tx
Regulator Type = 0	-	LDO	LDO	LDO	LDO	LDO
Regulator Type = 1	-	LDO	DC-DC	DC-DC	DC-DC	DC-DC

The user can specify the use of DC-DC by using the command *SetRegulatorType(regulatorType)*. This operation must be carried out in STDBY_RC mode only.

5.2 Flexible DIO Supply

The transceiver has two power supply pins, one for the core of the transceiver called VBAT and one for the host controller interface (SPI/UART, DIOs, BUSY) called VBAT_IO. Both power supplies can be connected together in application. In case a low voltage micro-controller (typically with IO pads at 1.8 V) is used to control the transceiver, the user can:

- use VBAT at 3.3V for optimal RF performance
- directly connect VBAT_IO to the same supply used for the micro-controller
- connect the digital IOs including SPI or UART directly to the micro-controller DIOs.

At any time, VBAT_SX1280_DIO must be lower than or equal to VBAT.

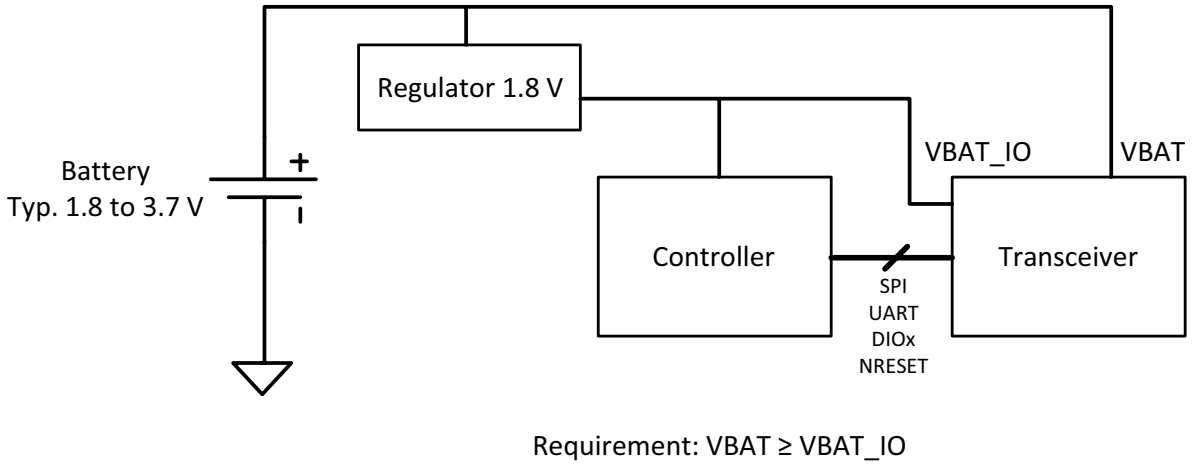


Figure 5-2: Separate DIO Supply

6. Digital Baseband

6.1 Overview

The transceiver features three modems that are all implemented in the digital baseband portion of the circuit. Corresponding to each modem are independent packet format and packet configuration options.

All modems use a digital Automatic Frequency Correction (AFC). This process is fully automated and transparent to the user. The frequency tolerance of each modem is detailed in its corresponding Section. The interfaces controlling the modem configuration and the memory in which the packets are stored are also common to all modems providing a simple unified interface to both the modulated and demodulated data.

The available modems and the corresponding packet types for each modem are shown in the highlighted block below:

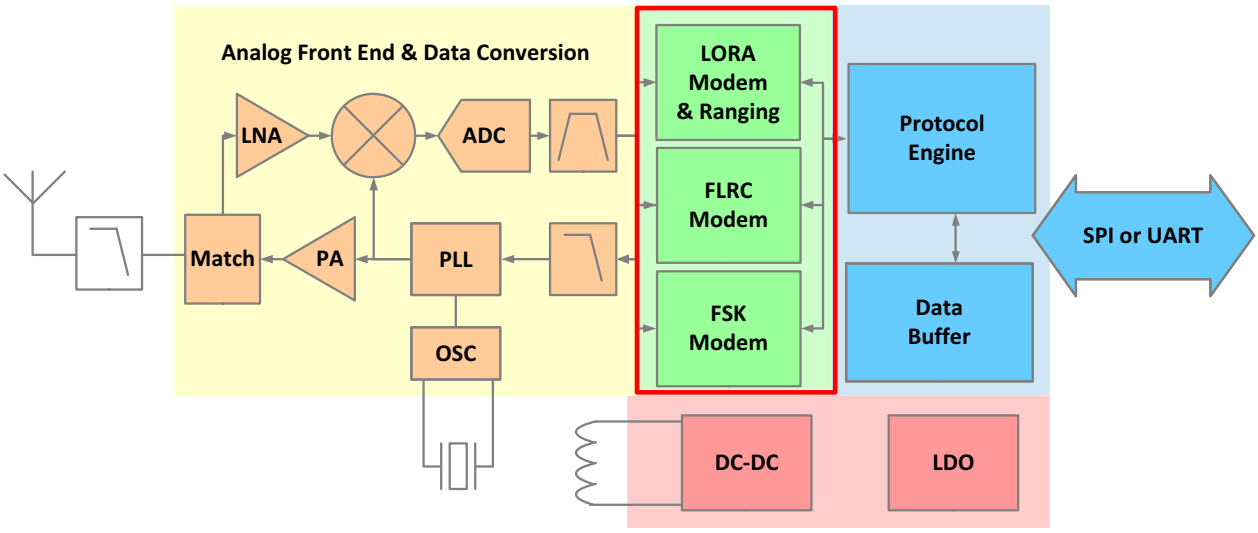


Figure 6-1: Transceiver Block Diagram, Modems Highlighted

Associated with each physical layer modulation available, there is also a range of corresponding packet formats.

Note: care must therefore be taken to ensure that modulation parameters are set using the command *SetModulationParam()* only after defining the packet type *SetPacketType()* to be used.

6.2 LoRa Modem

The LoRa modem provides both long range communication based upon LoRa spread spectrum modulation and incorporates a ranging engine which provides the facility to measure the time-of-flight, thus offering the possibility to deduce the range between a pair of transceivers.

6.2.1 LoRa Modulation

The LoRa modem uses spread spectrum modulation and Forward Error Correction (FEC) techniques to increase the range and robustness of radio communication links compared to traditional FSK or OOK based modulations.

An important aspect of the LoRa modem is its superior immunity to interference. It is capable of co-channel rejection up to 19.5 dB. This immunity to interference allows the coexistence of LoRa modulated systems either in bands of heavy spectral usage or in hybrid communication networks that use LoRa to extend range and robustness when legacy modulation schemes fail.

When used for communication the LoRa packet is compatible with this modem. Full details on this format and its use can be found in Section 13.4 "LoRa Packet" on page 111.

6.2.2 Spreading Factor

The LoRa modem uses a chirp spread spectrum based modulation. As for any spread spectrum device, the LoRa modulation represents each symbol of payload information by multiple chips of information. The Spreading Factor (SF) determines the ratio between the symbol rate (*Rs*) and chip rate (*Rc*):

$$R_c = 2^{SF} R_s$$

Note that 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.

The following table shows the receiver sensivities when using the LoRa modem. The receiver sensitivities are given with:

- Packer Error Rate (PER) of 1%,
- Packet with 10 bytes of payload
- 25°C, 3.3. V, CR = 4/5

Table 6-1: Receiver Sensitivity when using LoRa

Bandwidth (kHz)	Receiver Sensitivity (dBm)							
	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12
203	-109	-111	-115	-118	-121	-124	-127	-130
406	-107	-110	-113	-116	-119	-122	-125	-128
812	-105	-108	-112	-115	-117	-120	-123	-126
1625	-99	-103	-106	-109	-111	-114	-117	-120

The following table shows the effective data rates that can be obtained when using the LoRa modem:

Table 6-2: Effective Data Rates when using LoRa

Bandwidth (kHz)	Effective Data Rates (kbps)							
	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12
203	31.72	19.03	11.1	6.34	3.57	1.98	1.09	0.595
406	63.44	38.06	22.2	12.69	7.14	3.96	2.18	1.19
812	126.88	76.13	44.41	25.38	14.27	7.93	4.36	2.38
1625	253.91	152.34	88.87	50.78	28.56	15.87	8.73	4.76

6.2.3 Bandwidth

In an LoRa system the bandwidth setting sets the double sided modulation bandwidth, which is equivalent to the chip rate. An increase in signal bandwidth is equivalent to a higher effective data rate. This means that the symbol period is given by:

$$T_s = \frac{2^{SF}}{BW}$$

The symbol period is an important parameter in calculating the time on air of the LoRa packet as shown in Section 7.4 "LoRa Packet" on page 43. The trade-off between sensitivity and time on air of the signal is defined by setting Spreading Factor and bandwidth of LoRa modulation. We define a raw data rate, Rb, for the LoRa modem equivalent to:

$$R_b = \frac{SF}{T_s}$$

6.2.4 Forward Error Correction Coding Rate

To further improve the robustness of the link, the LoRa modem employs cyclic error coding to perform forward error detection and correction. Although Forward Error Correction (FEC) will not substantially improve the sensitivity of the modem in the presence of burst interference, it is particularly efficient in improving the reliability of the link in presence of interference. The coding rate, and therefore robustness to interference, can be changed in response to channel conditions. The coding rate can optionally be included in the packet header for use by the receiver. The increased overhead in time on air is proportional to the error correcting capability of the FEC. The resulting *effective* bit rate, including the influence of the FEC, can be calculated according to:

$$R_{beff} = R_b \times \frac{4}{(4 + CR)}$$

where CR is the programmed coding rate. The settings permissible for the LoRa modem give data rates in the range from 71 kb/s to 202 kb/s., with a BW of 1625 kHz, down to 476 bps for BW = 200 kHz.

6.2.5 Ranging Engine

The ranging engine uses the LoRa modem to perform a time-of-flight measurement between a pair of transceiver radios. Full details of operation of the ranging functionality are given in Section 13.5 "Settings for Ranging" on page 116. Time-of-flight requires using the ranging engine packet format as in Section 7.5 "LoRa Ranging Engine Packet" on page 46.

6.3 FLRC Modem

The Fast Long Range Communication (FLRC) modem is based upon a coherent demodulation of GMSK combined with forward error correction and interleaving techniques to improve receiver sensitivity. These parameters are accessible to the user, allowing high speed communication with an 8 to 10 dB improvement in link budget when compared with FSK modulation at the same data rate.

The available packet type to be used with the FLRC modem is the FLRC packet described in Section 13.3 "FLRC Packet" on page 102

6.3.1 Modem Bandwidth and Data Rates

These higher data rates cover the range from 260 kb/s to 1.3 Mb/s. To support these raw data rates, modulation bandwidths from 0.3 MHz to 2.4 MHz are supported. Note that not all combinations of bandwidth and data rate are supported. For this reason, the raw data rate is programmed using the *SetModulationParam()* command, the first parameter selects one of the valid combinations of raw data rate and double side band modulation bandwidth.

Table 6-3: Valid FLRC Data Rate and Bandwidth Combinations

Symbol	Raw Bit Rate Rb [Mb/s]	Bandwidth BW [MHz DSB]
FLRC_BR_1_300_BW_1_2	1.3	1.2
FLRC_BR_1_040_BW_1_2	1.04	1.2
FLRC_BR_0_650_BW_0_6	0.65	0.6
FLRC_BR_0_520_BW_0_6	0.52	0.6
FLRC_BR_0_325_BW_0_3	0.325	0.3
FLRC_BR_0_260_BW_0_3	0.26	0.3

6.3.2 FEC Coding Rate

The FLRC modem can optionally use forward error correction controlled by parameter *codingRate* (CR). The convolutional coding applied to the packet requires the addition of redundant information used in the process of error correction. Error correction makes the packet payload information robust to bursts of interference from other radio services in the same band or channel. The overhead is expressed below as a table of raw bit rate and effective bit rate that takes into consideration the influence of the FEC overhead.

Table 6-4: Effective FLRC Data Rates Based upon FEC Usage with Resulting Sensitivities

Symbol	Raw Programmed Data Rate Rb [Mb/s]	Programmed Coding Rate CR	Effective Data Rate Rbeff [Mb/s]	Sensitivity [dBm] at PER 1%
FLRC_BR_1_300_BW_1_2	1.3	1	1.3	-96
	1.3	3/4	0.975	-100
	1.3	1/2	0.65	-99
FLRC_BR_1_040_BW_1_2	1.04	1	1.04	-97
	1.04	3/4	0.78	-100
	1.04	1/2	0.52	-101
FLRC_BR_0_650_BW_0_6	0.65	1	0.65	-99
	0.65	3/4	0.488	-103
	0.65	1/2	0.325	-104
FLRC_BR_0_520_BW_0_6	0.52	1	0.52	-100
	0.52	3/4	0.39	-104
	0.52	1/2	0.26	-104
FLRC_BR_0_325_BW_0_3	0.325	1	0.325	-101
	0.325	3/4	0.195	-106
	0.325	1/2	0.130	-106
FLRC_BR_0_260_BW_0_3	0.26	1	0.26	-103
	0.26	3/4	0.195	-105
	0.26	1/2	0.130	-106

6.3.3 Gaussian Filtering

In transmit mode an optional Gaussian filter controlled by parameter BT is also available. This filtering function is used to reduce the sidelobe emissions of the transmitted FLRC signal. Valid values of filtering parameter BT in order of reducing filtering effort are: 0.5, 1or OFF. Filter BT is also configured by the *SetModulationParam()* command.

6.3.3.1 FLRC Frequency Tolerance.

The modem is configured with the data rate parameters set through the *SetPacketParam()* and *SetModulationParam()* commands described in Section 12. "List of Commands" on page 83. In reception there are three phases in the reception process. The first relies on a bank of correlators all looking for a valid incoming preamble. The number of correlators running is a function of the bandwidth and data rate to ensure that, for data rates of 1.3 Mb/s and 1.04 Mb/s, +/- 30 ppm of frequency drift can be accommodated. For lower data rates this drops to +/- 10 ppm of frequency misalignment between transmitter and receiver.

Once a valid preamble is detected, the modem goes on to check the synchronisation word to ensure that the received packet is intended for that radio. The final phase of the demodulation process is demodulation of the packet data itself.

The acceptable frequency tolerance is shown in the following table.

Table 6-5: Receiver Performance of the FLRC Modem

Data Rate [Mb/s]	Bandwidth [MHz]	Frequency Tolerance [kHz]
1.3	1.2	+/- 150
1.04	1.2	+/- 150
0.65	0.6	+/- 150
0.52	0.6	+/- 150
0.325	0.3	+/- 75
0.260	0.3	+/- 75

6.4 FSK Modem

The FSK modem is a conventional FSK modem that also features optional Gaussian filtering. With this modem, FSK, GFSK, MSK and GMSK modulation formats are supported.

This modulator is also used to provide physical layer compatibility with Bluetooth Low Energy, thus two frame types are compatible with the FSK modem: BLE frame and GFSK frame, for more information on these frame formats and their use please see Section 13.2 "BLE Packet" on page 94 and Section 13.1 "GFSK Packet" on page 85 respectively.

6.4.1 Modem Bandwidth and Data Rates

The FSK modem is capable of 2-FSK modulation over a range of data rates from 125 kb/s to 2 Mb/s. The data rate is controlled by the *SetModulationParams()* command of Section 11.6.7 "SetModulationParams" on page 74. The FSK double side band (DSB) occupied bandwidth is defined, together with other modulation parameters, in the image below:

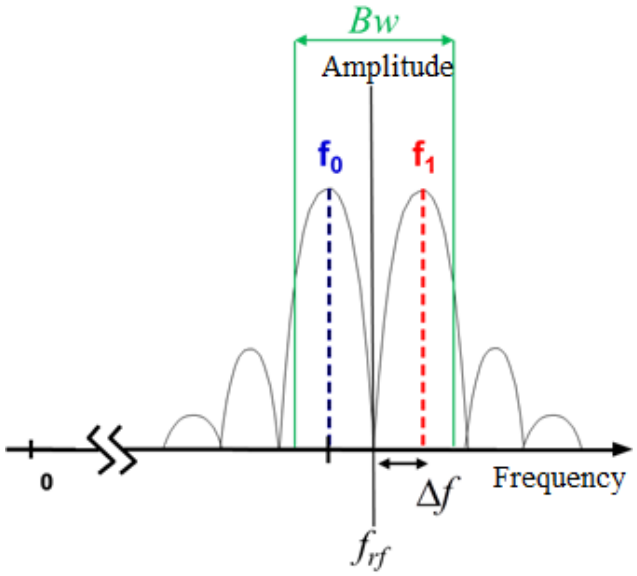


Figure 6-2: FSK Modulation Parameters

Where Δf is the frequency deviation, and f_{rf} is the RF centre frequency.

In receive mode the bandwidth is configured to the lowest receiver bandwidth that can accommodate the signal bandwidth of the FSK signal is defined as:

$$B_{-20\text{dB}} = 2\Delta f + R_b$$

Programmable bandwidths in the range 0.3 MHz to 2.4 MHz are available, however, not all combinations of raw data rate and bandwidth are valid. The range of valid combinations are shown in the following table.

Table 6-6: Valid FSK Data Rate and Bandwidth Combinations with Resulting Sensitivities

Symbol	Raw Bitrate Rb [kb/s]	Bandwidth BW [kHz DSB]	Sensitivity [dBm]
FSK_BR_2_000_BW_2_4	2.0	2.4	-83
FSK_BR_1_600_BW_2_4	1.6	2.4	-84
FSK_BR_1_000_BW_2_4	1.0	2.4	-87
FSK_BR_1_000_BW_1_2	1.0	1.2	-88
FSK_BR_0_800_BW_2_4	0.8	2.4	-87
FSK_BR_0_800_BW_1_2	0.8	1.2	-89
FSK_BR_0_500_BW_1_2	0.5	1.2	-90
FSK_BR_0_500_BW_0_6	0.5	0.6	-89
FSK_BR_0_400_BW_1_2	0.4	1.2	-91
FSK_BR_0_400_BW_0_6	0.4	0.6	-90
FSK_BR_0_250_BW_0_6	0.25	0.6	-92
FSK_BR_0_250_BW_0_3	0.25	0.3	-93
FSK_BR_0_125_BW_0_3	0.125	0.3	-95

Note that due to the absence of an error correcting code in the FSK modem, there is no notion of effective data rate.

6.4.2 Modem Modulation Index

In addition to the raw bit rate and bandwidth, the designer also has the flexibility to change the modulation index over the range 0.35 to 2. The modulation index, β, is a figure of merit that describes the proximity of '1' and '0' frequencies for a given data rate. This influences the ease with which each logical level can be discriminated by the demodulator and is given by:

$$\beta = \frac{2\Delta f}{R_b}$$

where Δf is the frequency deviation and R_b is the programmed data rate.

6.5 Guidance on Modem Selection

The relative receive performance of the three modems in the transceiver is shown in the figure below. The blue line represents the Shannon limit for error-free communication at settings equivalent to those used to measure the performance of the modems. Here we see that the conventional FSK modem, as used for legacy and Bluetooth communication yields conventional sensitivity figures for 2.4 GHz operation.

In contrast to this, the LoRa modulation gives access to lower effective data rates thanks to the use of spread spectrum techniques. This significantly improves the sensitivity, bringing it within 10 to 11 dB of the theoretical limit.

The FLRC modem based upon a coherent MSK demodulator provides access to higher effective data rates - maintaining the same improvement in sensitivity relative to the Shannon bound. Therefore, for links looking to get longer range without being penalized by longer time-on-air, the FLRC modem provides the required design flexibility.

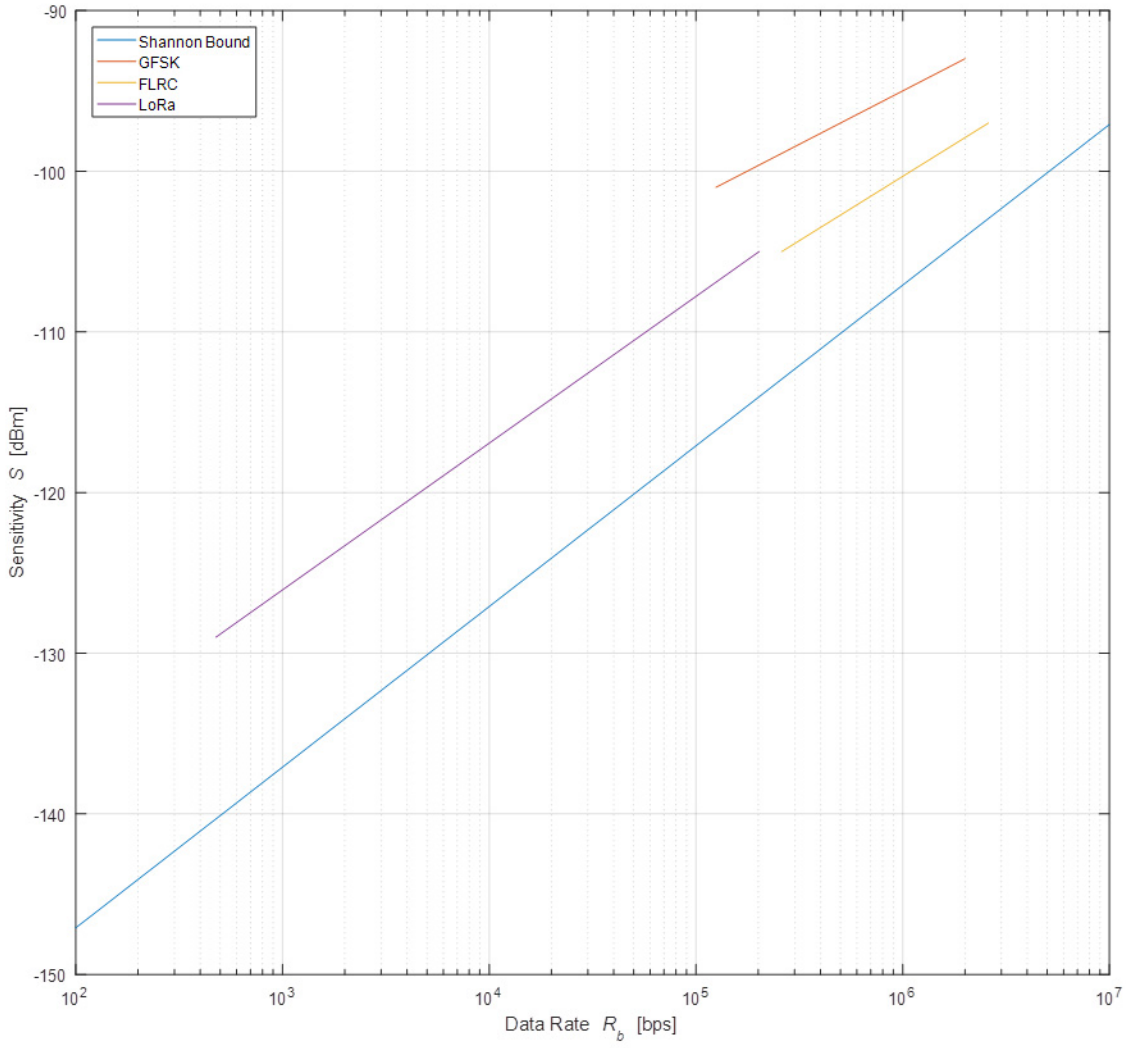


Figure 6-3: Sensitivity Performance of the Transceiver Modems

7. Packet Engine

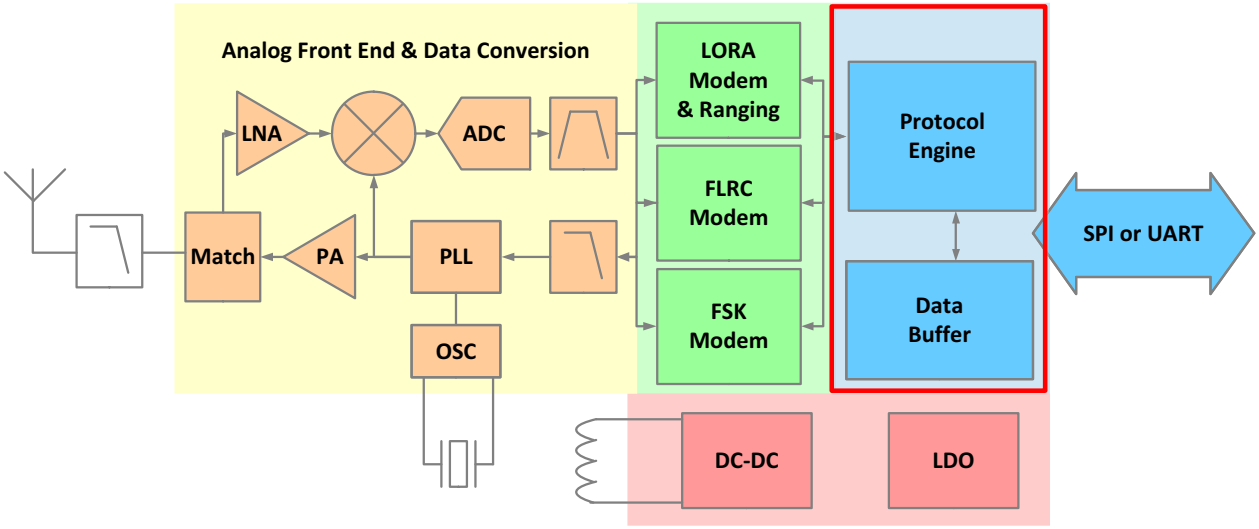


Figure 7-1: Transceiver Block Diagram, Packet Engine Highlighted

The transceiver is designed for packet-based operation. The packet controller works in half-duplex mode i.e. either in transmit or receive at a time. The packet controller is configured using the command *SetPacketParam()* outlined in Section 12. "List of Commands" on page 83.

Given that operation of the packet engine depends upon the selected packet type, the packet type must be selected using the *SetPacketType()* command prior to configuration of the packet parameters.

In receive mode the packet controller block is responsible for assembly and recovery of the data bit-stream and its storage in the data buffer. The data buffer is described in more detail in Section 8. "Data Buffer" on page 48. It also performs the bit-stream decoding operations such as de-whitening and CRC-checks on a received bit-stream.

In transmit mode the packet handler constructs the packet and sends it to the modulator for transmission. It can also perform all coding and decoding required specific to the selected packet type including data whitening, CRC-checksum, interleaving, convolutional coding and FEC.

The packet controller block supports five different packet frames, namely a GFSK frame, Bluetooth Low Energy (BLE), Fast Long Range Communication (FLRC) and LoRa packets (which in turn comprises both communication and ranging modes packet types).

CAUTION!

The transceiver implements aspects of the Bluetooth Low Energy Physical and Link layers necessary to implement the Bluetooth low energy communication frame. In this Section, details of the packet format, data transmission and reception are given. In this mode of operation bit rates different from 1 Mb/s are also available to address other applications using the same packet format. It is important to note that the header, payload, packet type and packet length are not interpreted by the transceiver and are only stored in the data buffer.

7.1 GFSK Packet

The GFSK packet format provides a conventional packet format for application in proprietary Non-Return-to-Zero (NRZ) coded, long range, low energy communication links. The packet format has built-in facilities for CRC checking of the payload and dynamic payload size. Optionally a whitening-transformation based upon Pseudo-Random Number Generation (PRNG) can be enabled. The GFSK packet is used with FSK modulation.

Two main packet formats are available in the GFSK frame: fixed-length and variable-length packets.

7.1.1 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 *packetLength* parameter which determines the packet length in bytes (0 to 255).

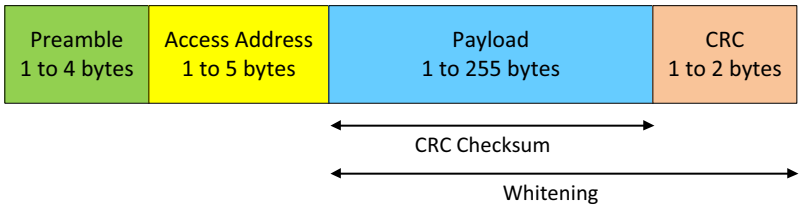


Figure 7-2: Fixed-length Packet Format

The preamble length is set from 0.5 to 4 bytes in nibble increments using the *PreambleLen* parameter. For 1 Mb/s communication, at least 1 byte of preamble is recommended. For all other data rates, at least 2 bytes are required. The CRC operation, packet length and preamble length are defined using the *SetPacketParam()* command as defined in Section 12. "List of Commands" on page 83.

7.1.2 Variable-length Packet

Where the packet is of uncertain or variable size, then information about the packet length must be transmitted within the packet. The format of the variable length packet is shown below.

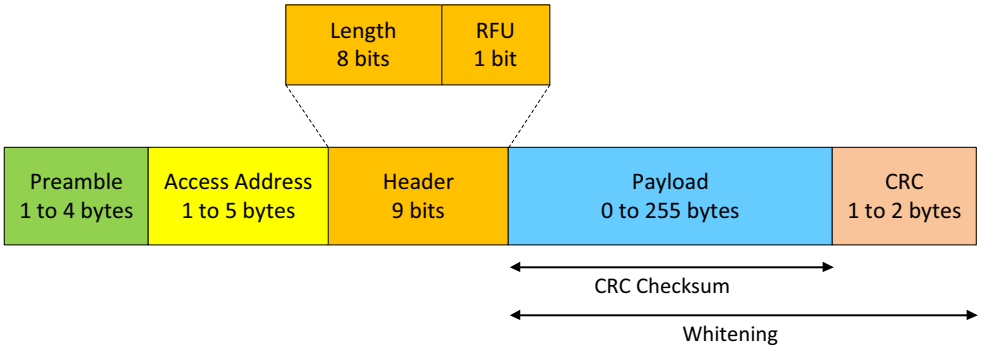


Figure 7-3: Variable-length Packet Format

7.2 BLE Packet Format

The BLE packet format, common to all communication modes, is shown in the diagram below. It comprises a single byte of preamble followed by 4 bytes of access codes which correlate with the RF channel number used, a Protocol Data Unit (PDU) and 3 CRC bytes.

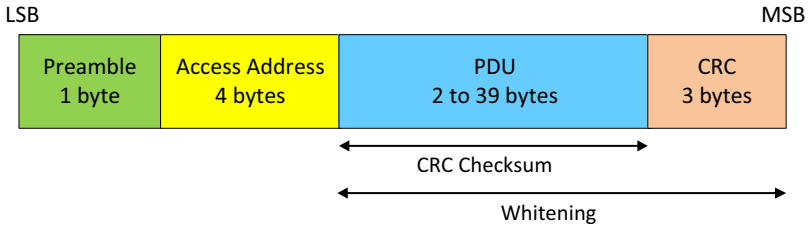


Figure 7-4: BLE Packet Format

The PDU has two formats: the advertising channel PDU and the data channel PDU. In both cases, the PDU consists of a 2-byte header and payload data, 0 to 31 bytes of advertising payload and 0 to 37 bytes of data channel payload. The advertising PDU format is used to periodically broadcast and initiate a connection request to any listening (initiator) devices on one of three pairing channels. Once the communication is established, the initiator becomes the master device and the advertiser the slave device.

The **advertising header** contains:

- 4 bits to indicate one of 7 types of PDU packet
- 2 bits as Reserved for Future Use (RFU)
- TxAdd(ress) and RxAdd(ress) bits to indicate the advertising channel packet type
- a 6-bit length field to indicate the length of the payload
- 2 reserved bits

The **data channel header** contains:

- LLID to indicate if the packet is control or data type
- NESN is the Next Expected Sequence Number
- SN is the current Sequence Number
- MD stands for More Data
- Length is the payload + Message Integrity Check (MIC) length in bytes

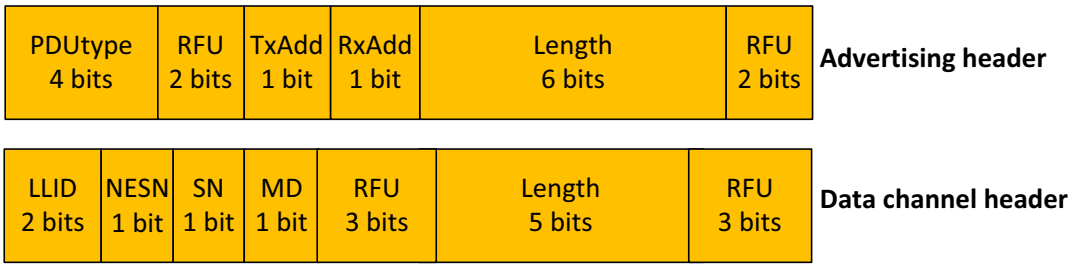


Figure 7-5: PDU Header Format

Note that the headers are not generated by the transceiver and must be calculated externally and passed as part of the payload to the data buffer.

7.3 FLRC Packet

FLRC is a coherent MSK modem that allows higher data rate communication than the LoRa modem. However the FLRC modem has higher sensitivity and better link budget than conventional FSK-based modulation. Convolutional coding and decoding is also employed to further enhance link budget and immunity to interference.

7.3.1 FLRC Packet Format

Although proprietary, the FLRC packet is conventional in its construction. It features a header, Sync Word and CRC structure. Similar to the GFSK mode, two packet formats are available for fixed and variable length packets.

7.3.2 Fixed-Length Packet Format

The fixed packet length format is shown in the diagram below. The packet contains the following elements:

- a variable-length AGC preamble - for 1.3 Mb/s data rates this can be reduced to 1 byte, for all other data rates 2 bytes are required
- a 21-bit timing recovery preamble
- a 4-byte Sync Word
- the fixed length payload which can be from 6 to 127 bytes long
- a CRC field of 2, 3 or 4 bytes in length
- finally the packet is terminated by a short 6-bit sequence of trailing zeros

The Sync Word size, payload length and the CRC length are configured by the command *SetPacketParam()* as described in Section 12. "List of Commands" on page 83. The CRC is performed on the whole preceding packet except the preamble.

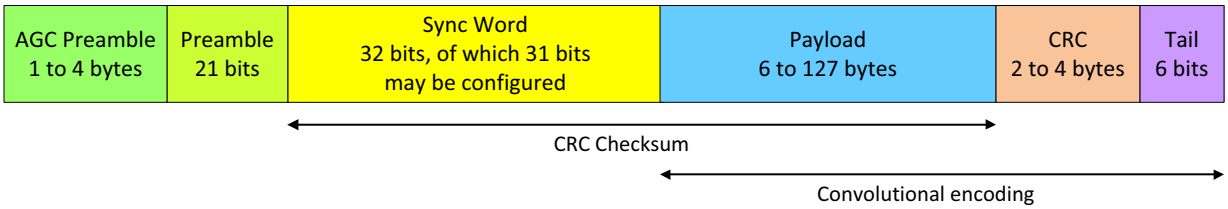


Figure 7-6: FLRC Fixed-length Packet Format

7.3.3 Variable-length Packet Format

The variable format packet is identical in form and function to the fixed packet length format but with the addition of a header to which the CRC and convolutional coding are applied. The header structure is fixed, featuring a 2-bit-type declaration, see the mapping in the figure below. It is followed by the payload length.

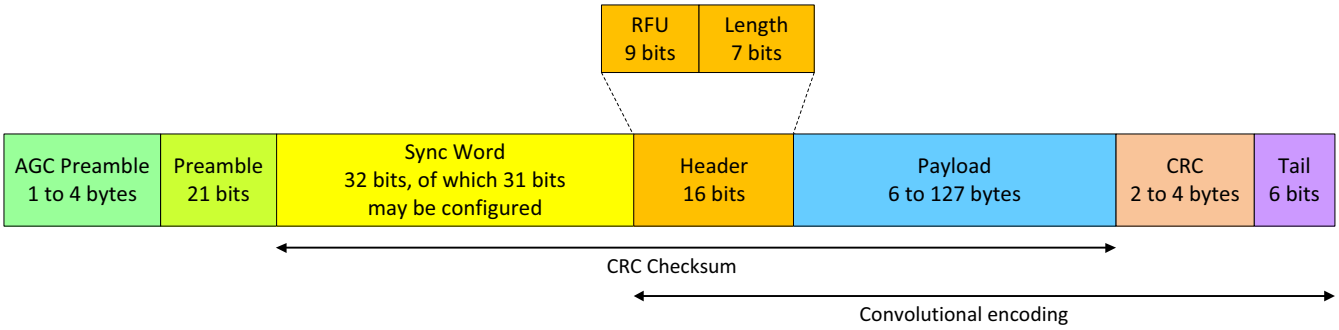


Figure 7-7: FLRC Variable-length Packet Format

7.3.4 FLRC Time-on-Air

The total number of bits transmitted in an FLRC packet is as defined in Figure 7-6: FLRC Fixed-length Packet Format. The calculation of the total time-on-air is therefore the combination of the number of payload bits (compensated for the influence of convolutional coding) and the number of header bits. Denoting the number of bits in each field of the packet, n, the number of header bits is:

$$n_{uncoded} = n_{AGCPreamble} + n_{Preamble} + n_{SyncWord} + n_{header} \times VL$$

where n_{header} is 16 bits and VL = 1 if variable length, = 0 otherwise.

and the effective number of coded bits by:

$$n_{coded} = ceil\left[(n_{Payload} + n_{CRC} + n_{Tail}) \times \left(\frac{1}{CR}\right)\right]$$

where CR is the value programmed as the coding rate, see Table 13-31: Modulation Parameters in FLRC Mode: Coding Rate.

n_{tail} depends on the CR: n_{tail} = 6 bit if CR = 1/2 or 3/4 ; n_{tail} = 0 in other cases.

The bit period for a given FLRC data rate is simply:

$$t_{bit} = \frac{1}{R_b}$$

Values of R_b in FLRC can be found at Table 6-3: Valid FLRC Data Rate and Bandwidth Combinations.

and the total packet time-on-air is given by:

$$ToA_{FLRC} = t_{bit} \times (n_{uncoded} + n_{coded})$$

7.4 LoRa Packet

The LoRa modem employs two types of packet format, explicit and implicit. The explicit packet includes a short header that contains information about the number of bytes, coding rate and whether a CRC is appended to the packet.

7.4.1 LoRa Packet Format

The LoRa packet starts with a preamble sequence which is used to synchronize the receiver with the incoming signal. By default the packet is configured with a 12-symbol long sequence. This preamble length is programmable and can be extended; for example to reduce the receiver duty cycle in receive intensive applications. The programmable preamble length is configurable from 8 to 65535 symbols. The LoRa modem automatically add 4.25 symbols making the range of real preamble length from 12.25 to 65539.25 symbols. This allows the transmission of near arbitrarily long preamble sequences.

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

An optional header may be included in the LoRa packet. Explicit (variable-length) and implicit (fixed-length) header modes respectively indicate the inclusion or exclusion of a packet header.

7.4.2 Explicit (Variable-length) Header Mode

This is the default mode of operation and includes a header that features the following:

- The payload length in bytes
- The forward error correction code rate
- The presence of an optional 16-bit CRC for the payload

The 8-symbols long header is always encoded with the strongest error correction code allowing reception of the packet for any value of FEC applied to the payload. The header also features an individual CRC, allowing the receiver to discard invalid headers.

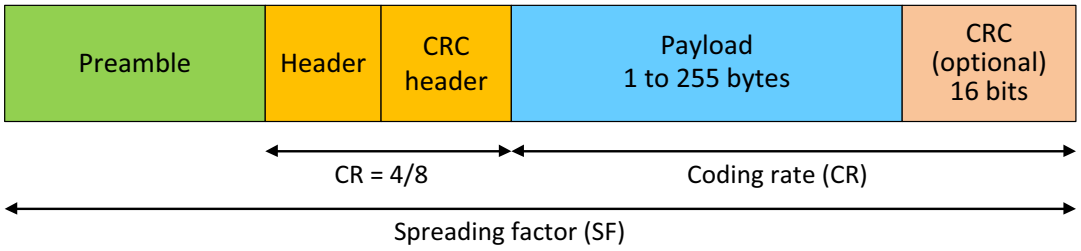


Figure 7-8: LoRa Variable-length Packet Format

7.4.3 Implicit (Fixed-length) Header Mode

Where the payload, coding rate and CRC presence are known in advance the packet duration can be reduced by removing the header (implicit mode). Here the payload length, error coding rate and presence of the payload CRC must be manually configured on both sides of the radio link. After the header section is the payload of a preconfigured length coded at the error rate specified. Note that in all modes of header operation a CRC may optionally be appended to the packet to detect corrupted or invalid packet payloads.

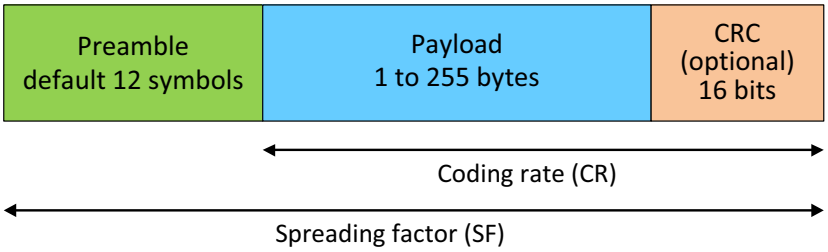


Figure 7-9: LoRa Fixed-length Packet Format

7.4.4 LoRa Time-on-Air

The packet format for the LoRa modem is detailed in Figure 7-8: LoRa Variable-length Packet Format and Figure 7-9: LoRa Fixed-length Packet Format. The equation to obtain Time On Air (ToA) is:

$$ToA = \frac{2^{SF}}{BW} * N_{symbol}$$ with:

- *SF*: Spreading Factor (5 to 12)
- *BW*: Bandwidth (in kHz)
- *ToA*: the Time on Air in ms
- N_{symbol}: number of symbols

The computation of the number of symbols differs depending on the parameters of the modulation.

For CR being legacy coding rate (ie. Not Long Interleaving):

$$N_{symbol} = N_{symbol_{preamble}} + 6.25 + 8 + ceil\left(\frac{\max\left(8 * N_{byte_{payload}} + N_{bit_{CRC}} - 4 * SF + N_{symbol_{header}}, 0\right)}{4 * SF}\right) * (CR + 4)$$

if SF<SF7

$$N_{symbol} = N_{symbol\_preamble} + 4.25 + 8 + ceil\left(\frac{\max\left(8*N_{byte\_payload}+N_{bit\_CRC}-4*SF+8+N_{symbol\_header},0\right)}{4*SF}\right) * (CR + 4)$$

if SF7≤SF≤SF10

$$N_{symbol} = N_{symbol\_preamble} + 4.25 + 8 + ceil\left(\frac{\max\left(8*N_{byte\_payload}+N_{bit\_CRC}-4*SF+8+N_{symbol\_header},0\right)}{4*(SF-2)}\right) * (CR + 4)$$

if SF>SF10

With:

- N_bit_CRC = 16 if CRC activated, 0 if not
- N_symbol_header = 20 if header is variable, 0 if it is fixed
- CR is 1, 2, 3 or 4 for respective coding rates 4/5, 4/6, 4/7 or 4/8

For the Long Interleaving case:

$$N_{symbol} = N_{symbol\_preamble} + 6.25 + 8 + ceil\left(\frac{\max\left(8*N_{byte\_payload}+N_{bit\_CRC}-8*floor\left(\frac{4*SF-N_{symbol\_header}}{8}\right),0\right)*CR}{4*SF}\right)$$

if SF<SF7

$$N_{symbol} = N_{symbol\_preamble} + 4.25 + 8 + ceil\left(\frac{\max\left(8*N_{byte\_payload}+N_{bit\_CRC}-8*floor\left(\frac{4*SF-8-N_{symbol\_header}}{8},0\right)\right)*CR}{4*SF}\right)$$

if SF7≤SF≤SF10

$$N_{symbol} = N_{symbol\_preamble} + 4.25 + 8 + ceil\left(\frac{\max\left(8*N_{byte\_payload}+N_{bit\_CRC}-8*floor\left(\frac{4*SF-8-N_{symbol\_header}}{8},0\right)\right)*CR}{4*(SF-2)}\right)$$

if SF>SF10

With:

- N_bit_CRC = 16 if CRC activated, 0 if not
- N_symbol_header = 20 if header is variable, 0 if it is fixed
- CR is 5, 6, or 8 for respective coding rates 4/5LI, 4/6LI, or 4/8LI

7.5 LoRa Ranging Engine Packet

The ranging operations consist in an exchange, or sequence of exchanges, between a transceiver configured as a ranging Master and a transceiver configured as a ranging Slave. In each exchange the Master generates a ranging packet that is sent over the air and received by the Slave. The Slave then synchronises with the incoming ranging packet and sends a ranging response.

When received by the Master, synchronisation with the ranging response allows the deduction of the time of flight between the Master and the Slave. This in turn is, notionally, converted into distance. It should be noted that the distance reported will be representative of the path travelled by the radio wave rather than the shortest path distance between the Master and the Slave.

The Ranging Engine Packet structure is very similar to the LoRa packet explicit header mode (see Section 7.4.2 "Explicit (Variable-length) Header Mode" on page 43). One reserved bit in the header is simply set to indicate that a packet is a ranging request. The header includes a 32-bit ranging Slave ID. The slave will reject any ranging request that does not have a matching ID.

To afford some flexibility to the system, the Slave may also check a portion of the ranging ID, specifically the least significant 8, 16, 24, or 32 bits.

The time-of-flight reported by the master is available in both raw format - where the result for a single ranging measurement is reported - or in filtered format. Filtering applies a non-linear filtering function to aggregate several ranging exchanges results and improve accuracy. For configuration of the filtering and the ranging parameters please see Section 13.5 "Settings for Ranging" on page 116 for more details.

7.5.1 Ranging Packet Format

The following figure shows the dedicated frames used in ranging exchange:

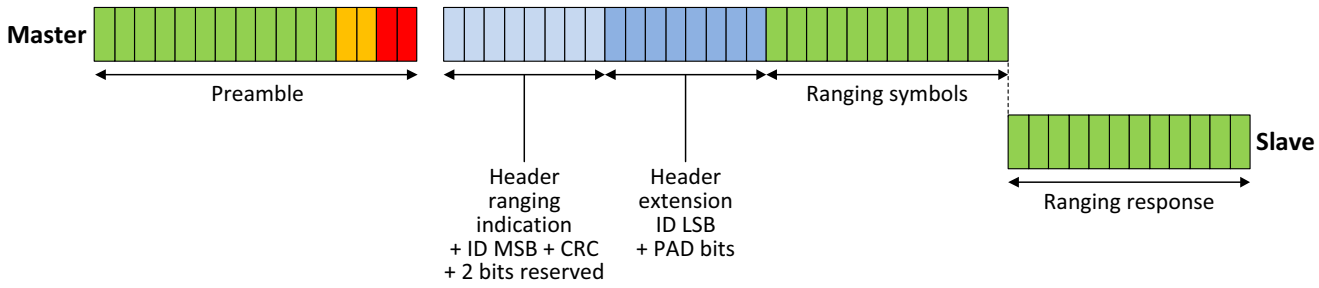


Figure 7-10: Ranging Packet Format

7.5.2 Ranging Master Exchange

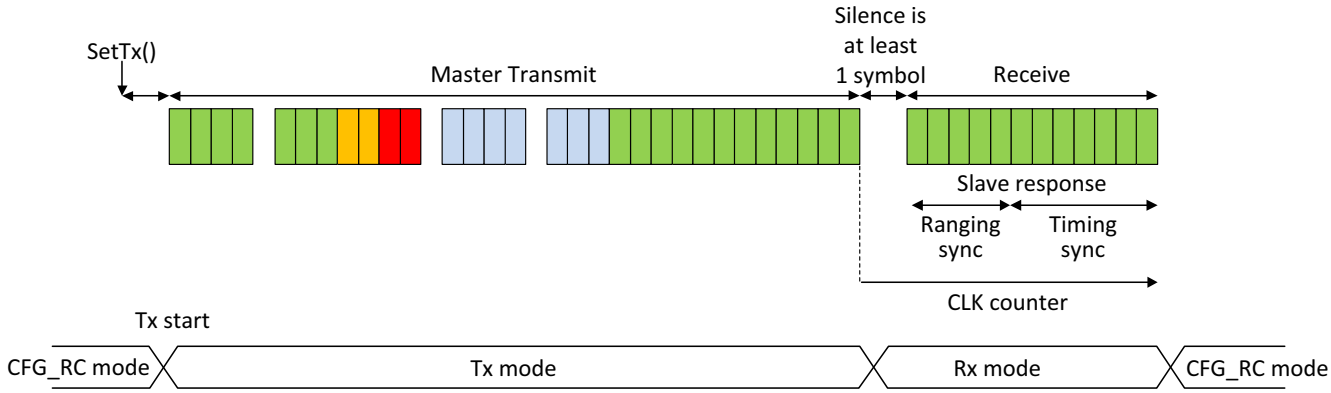


Figure 7-11: Ranging Master Packet Exchange

7.5.3 Ranging Slave Exchange

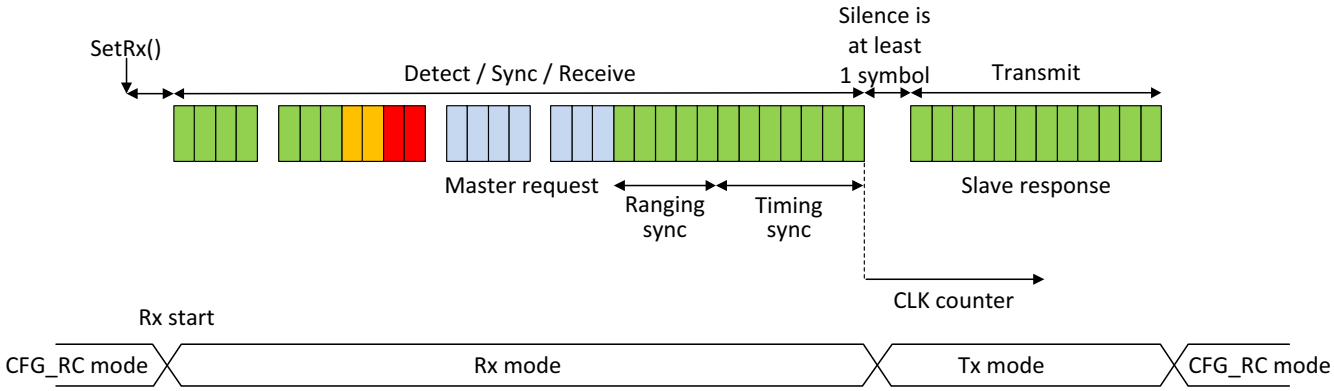


Figure 7-12: Ranging Slave Packet Exchange

8. Data Buffer

The transceiver is equipped with a 256 byte RAM data buffer which is accessible in all modes except sleep mode. This RAM area is fully customizable by the user and allows access to either data for transmission or from the last packet reception. All access to the data buffer is via either the SPI or UART interfaces of Section 9. "Digital Interface and Control" on page 50.

8.1 Principle of Operation

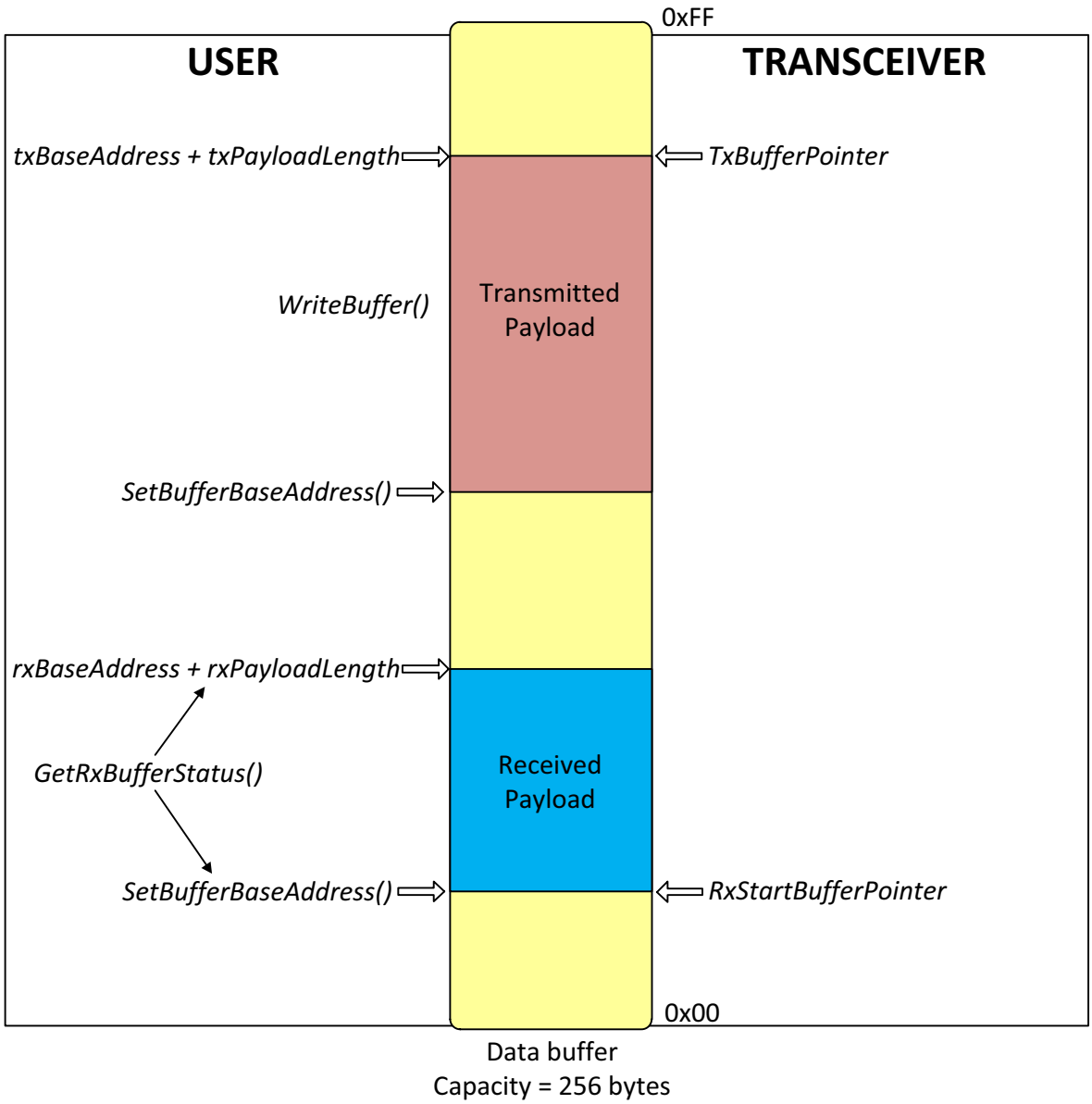


Figure 8-1: Data Buffer Diagram

The data buffer can be configured to store both transmit and receive payloads.

8.2 Receive Operation

In receive mode the rxBaseAddress specifies the buffer offset in memory at which the received packet payload data will be written. The buffer offset of the last byte written in receive mode is then stored in the *RxDataPointer* which is initialized to the value *rxBaseAddress* at the beginning of the reception.

The pointer to the first byte of the last packet received and the packet length can be read via command *GetRxbufferStatus()*.

In single mode, *RxDataPointer* is automatically initialized to *rxBaseAddress* each time the transceiver enters to Rx. In continuous mode the pointer is incremented starting from the previous position. For more details see Section 10.5 "Receive (Rx) Mode" on page 56.

8.3 Transmit Operation

Upon each transition to transmit mode the pointer *TxDataPointer* is initialised to the *txBaseAddress* and is incremented each time a byte is sent over the air. This operation stops once the number of bytes sent equals the *payloadlength* parameter as defined in function *SetPacketParam()*.

8.4 Using the Data buffer

Both, *rxBaseAddress* and *txBaseAddress* are set using the command SetBufferBaseAddress().

By default *rxBaseAddress* and *txBaseAddress* are initialized at address 0x00.

Due to the contiguous nature of the data buffer, the base addresses for Tx and Rx are fully configurable across the 256-byte memory area. Each pointer can be set independently anywhere within the buffer. To exploit the maximum data buffer size in transmit or receive mode, the whole data buffer can be used in each mode by setting the base addresses *txBaseAddress* and *rxBaseAddress* at the bottom of the memory (0x00).

The data buffer is cleared when the device is put in deep Sleep mode (implying no access). The data is retained in all other modes of operation.

It is possible to keep data value in Sleep mode by maintaining the data buffer under retention. However the pointer locations will be lost. In order to retrieve data from sleep retention the user must use default value for base address (for example 0x00 for Rx and 0x80 for Tx) or store *PayloadLengthRx* and *RxStartBufferPointer* before going to Sleep mode.

The data buffer is acceded via SPI or UART using the command *WriteBuffer()* and *ReadBuffer()*. In this function the parameter offset defines the address pointer of the first data to be written or read. Offset zero defines the first position of the data buffer.

Before any read or write operation it is hence necessary to initialize this offset to the corresponding beginning of the buffer. Upon reading or writing to the data buffer the address pointer will then increment automatically.

Two possibilities exist to obtain the offset value:

- First is to use the *rxBaseAddress* value since the user defines it before receiving a payload.
- Second, offset can be initialized with the value of *RxStartBufferPointer* returned by *GetRxbufferStatus* command.

It is important to note that all the received data will be written to the data buffer even if the CRC is invalid, permitting user-defined post processing of corrupted data. It is also important to note that when receiving, if the packet size exceeds the buffer memory allocated for the Rx, it will overwrite the transmit portion of the data buffer.

9. Digital Interface and Control

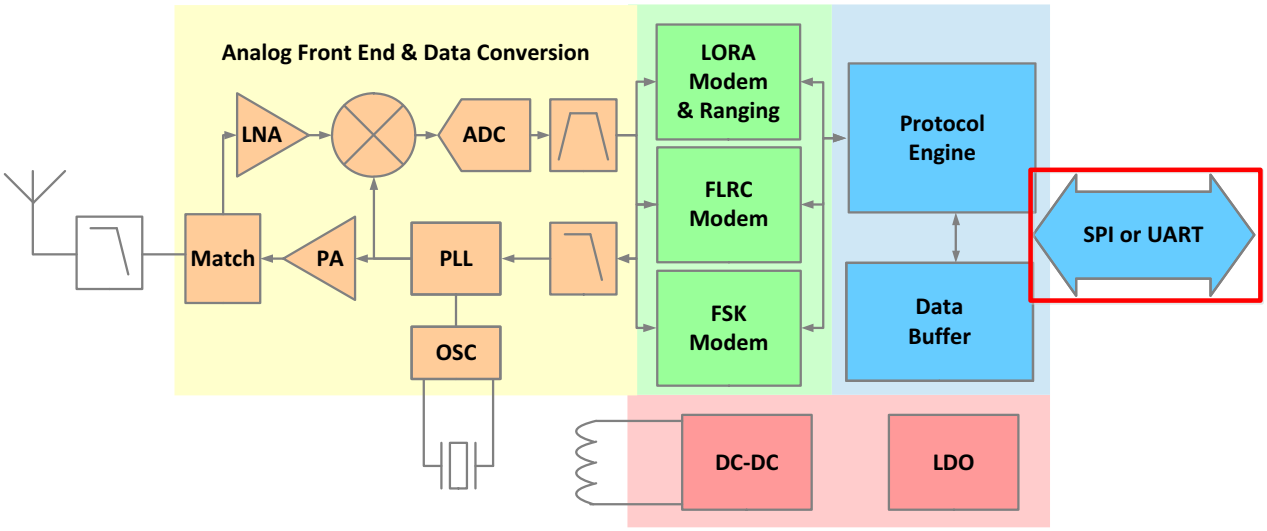


Figure 9-1: Transceiver Block Diagram, Digital Interface Highlighted

The transceiver is controlled via a serial interface (SPI or UART) and a set of general purpose input/output (DIOs). The transceiver uses an Protocol Engine to handle communication and transceiver control (mode switching, API etc...). Through SPI or UART the application sends commands to the internal chip or accesses directly the data memory space. All registers can be accessed by SPI or UART.

9.1 BUSY Pin Communication

In all communications the BUSY Pin is used as busy signal indicating that the transceiver is ready for new command only if this signal is low. See Section 2. "Pin Connections" on page 14.

9.2 Interface Detection

Both interfaces are enabled until the first one receives a valid transaction, this disables the unused interface.

To allow reception by the UART, RTSN needs to be driven low. However, since it is shared with SCK, initially the pin 18 is driven low with a high impedance driver. If the UART interface is detected, pin 18 is driven directly by the on-chip UART; otherwise the pin is configured as input and driven by the external SPI master.

9.3 SPI Interface

The SPI interface gives access to the configuration register via a synchronous full-duplex frame corresponding to CPOL = 0 and CPHA = 0 in Motorola/Freescale nomenclature. Only the slave side is implemented.

An address byte followed by a data byte is sent for a write access whereas an address byte is sent and a read byte is received for the read access. The NSS pin goes low at the beginning of the frame and goes high after the data byte.

MOSI is generated by the master on the falling edge of SCK and is sampled by the slave (i.e. this SPI interface) on the rising edge of SCK. MISO is generated by the slave on the falling edge of SCK.

A transfer is always started by the NSS pin going low. MISO is high impedance when NSS is high.

The SPI runs on the external SCK clock to allow high speed up to 20 MHz

The host terminates an SPI transaction by raising the NSS signal, it does not explicitly send the command length as a parameter. The host must not raise NSS within the bytes of a transaction.

If the host sends a command requiring parameters, all parameters must be sent before raising NSS. If not, the transceiver will use unknown values for the missing parameters.

9.3.1 SPI Timing When the Transceiver is in Active Mode

The transceiver is considered to be in active mode when not in Sleep mode. In active mode the transceiver can immediately process standard SPI commands i.e. there is no extra delay needed at the first SPI transaction. The main reason is that, contrary to the behavior when in sleep mode, the transceiver does not have to go through the start-up process.

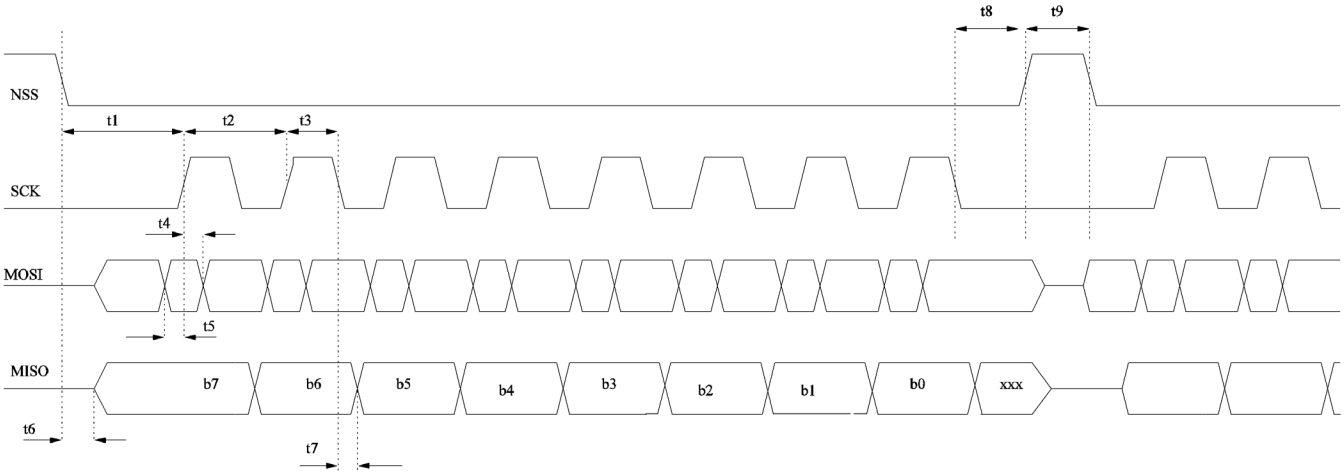


Figure 9-2: SPI Timing Diagram

9.3.2 SPI Timing When the Transceiver Leaves Sleep Mode

The method for the transceiver to leave Sleep mode is to wait for a falling edge of NSS. At the falling edge, all necessary internal regulators are switched ON; the transceiver starts its initialization before being able to accept the first SPI command. This means that the delay between the falling edge of NSS and the first rising edge of SCK must take into account the wake-up sequence and the transceiver initialization.

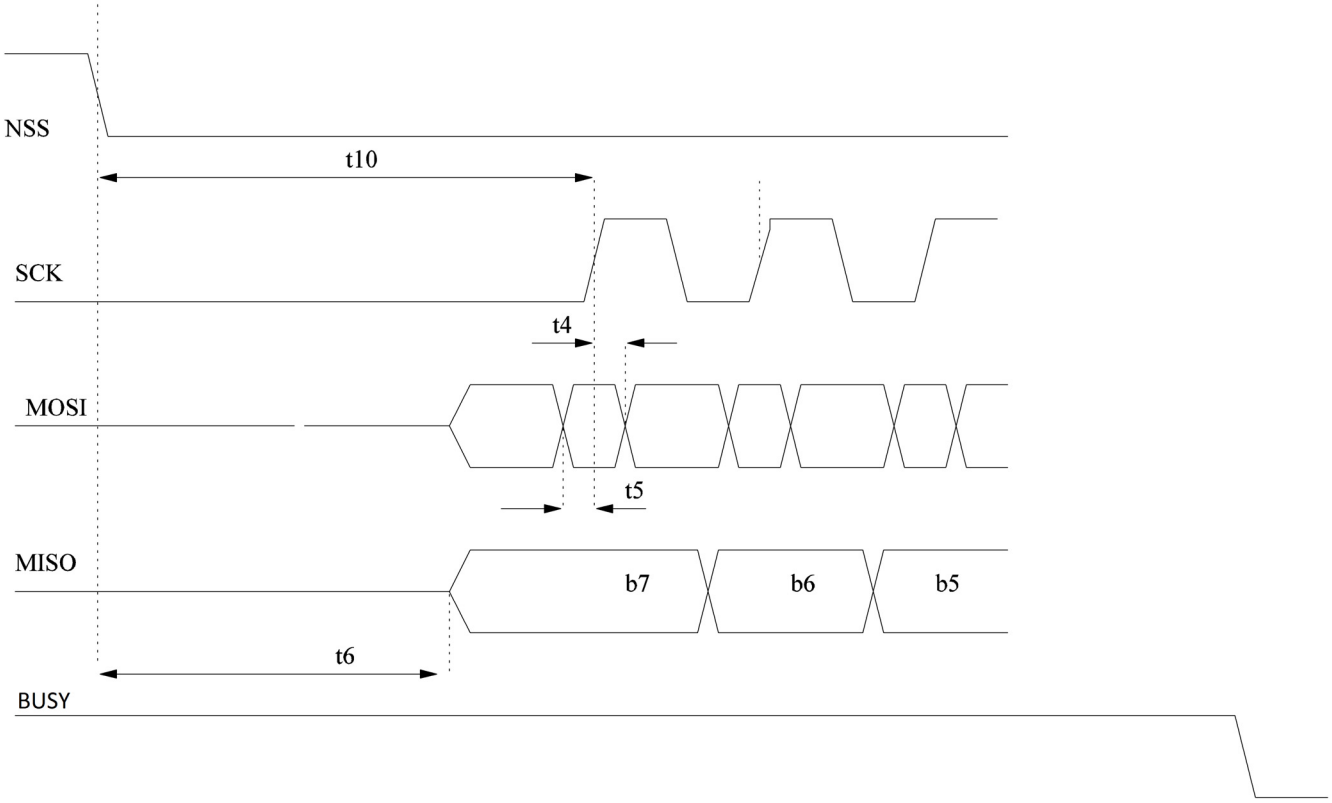


Figure 9-3: SPI Timing Transition

In Sleep mode and during the initialization phase, the busy signal mapped on BUSY pin is set high, indicating to the host that the transceiver is not able to accept a new command. Once the transceiver is in STDBY_RC mode, the busy signal goes low and the host can start sending a command. Note that this is also true for startup at power on or after hard reset.

The values for the SPI timings are visible in Section 9.3.3 "SPI Timings" on page 53.

9.3.3 SPI Timings

The following specifications are given for the typical operating conditions of VBAT_IO = VBAT = 3.3 V, temperature = 25 °C, crystal oscillator frequency = 52 MHz.

All timings are given in next table for Max load cap of 10 pF.

Table 9-1: SPI Timing Requirements

Symbol	Description	Minimum	Typical	Maximum	Unit
t1	NSS falling edge to SCK setup time	25	-	-	ns
t2	SCK period	50	-	-	ns
t3	SCK high time	25	-	-	ns
t4	MOSI to SCK hold time	5	-	-	ns
t5	MOSI to SCK setup time	5	-	-	ns
t6	NSS falling to MISO delay	0	-	15	ns
t7	SCK falling to MISO delay	0	-	15	ns
t8	SCK to NSS rising edge hold time	25	-	-	ns
t9	NSS high time	100	-	-	ns
t10	NSS falling edge to SCK setup time when switching from Sleep to STDBY_RC mode	125	-	-	µs

9.4 UART Interface

The transceiver UART supports the following settings:

- Baud rates: 921.6 k, 460.6 k, 115.2 k, 57.6 k, 38.4 k, 19.2 k, 9.6k
- RTS/CTS flow control
- Parity control: none, odd, even
- 8 bit words
- 1, 2 stop bits
- Rx full, Tx empty, Error (parity, no stop bit) interrupts.

Initially the UART is configured to operate at 115.2 kb/s with a 1 stop bit, even parity, CTS flow control and Least Significant Bit (LSB) arriving first. At start-up the CSTN must be driven low to initiate the communication. Other compatible UART communication settings may then be configured.

In a UART transaction, the host must provide the command length. The device starts processing the transactions as soon as the required bytes have been received. Subsequent bytes are treated as belonging to a new transaction.

9.5 Pin Sharing

The pins between SPI and UART are shared in the following way:

- NSS (IN) / CTSN (IN)
- SCK (IN) / RTSN (OUT)
- MOSI (IN) / RX (IN)
- MISO (OUT) / TX (OUT)

9.6 Multi-Purpose Digital Input/Output (DIO)

The transceiver provides 3 DIOs that can be configured as an interrupt.

The BUSY pin is used as an interrupt and is always an output. The busy interrupt is asserted when the current command has been processed and the device is ready to accept a new one.

Additionally any of the 3 DIOs can be selected as an external interrupt source for the transceiver.

Note: Any of the 3 DIOs can be mapped to any interrupt signal from the transceiver using the SetDioIrqParams() command. For full details please see Section 11.8.1 "SetDioIrqParams" on page 81.

When the application receives an interrupt it can determine the cause by reading the device IRQ register. The interrupt can be cleared using the *ClearIrq()* command.

When the SPI interface is used, the status is sent on every transaction that does not require data to be sent.

When using the UART interface the status can be retrieved via *GetDeviceStatus()* command.

10. Operational Modes

The transceiver features six operating modes, the analog front end and digital blocks that are enabled in each operating mode are explained in the following table:

Table 10-1: SX1280 Operating Modes

Mode	Enabled Blocks
SLEEP	Optional registers, backup regulator, RC32K oscillator, data buffer, data RAM
STDBY_RC	Top regulator (LDO), RC13M oscillator
STDBY_XOSC	Top regulator (DC-DC), RC13M oscillator, XOSC
FS	Frequency synthesizer at Tx frequency
Tx	Frequency synthesizer and transmitter, Modem
Rx	Frequency synthesizer and receiver, Modem

10.1 Startup

At power-up, the transceiver enters its start-up state. The BUSY pin is set to high, indicating that the transceiver is busy and cannot accept a command. When the digital voltage and RC clock are available, the transceiver can boot up and initiate the calibration phase which consists of:

- Calibration of the RC13 MHz with help of the52 MHz crystal. This is needed to properly establish UART communication
- Calibration of the RC 64K with the help of the 52 MHz crystal.
- Calibration of the PLL modulation path
- Calibration of the ADC

Once the calibration has terminated, the transceiver enters STDBY_RC mode. The transceiver is now ready and the BUSY pin goes low, indicating that the device is ready to accept a command from the host.

All results from calibration are stored in the data memory. When the transceiver wakes up from a Sleep mode and the data memory content is preserved, the calibration data is retrieved from memory without repeating the complete procedure.

10.2 Sleep Mode

In this mode only Start-up and Sleep Controller (SCC) block and optionally RC64K and timers are running, memories may be placed under retention. The transceiver may enter in this mode from STDBY_RC state and can leave the Sleep Mode if one of the following events occurs:

- The NSS pin (19) goes low.
- RTC timer generates an End-Of-Count (corresponding to Duty cycled operation). See Section 11.5.6 "SetRxDutyCycle" on page 67.

10.3 Standby Mode

In Standby mode the host should configure the transceiver before going to Rx or Tx modes. By default in this state, the system is clocked by the 13 MHz RC oscillator to reduce power consumption in all other modes, except Sleep mode, the crystal is turned ON. However if the application is time critical, the XOSC block can be turned or left ON.

Crystal oscillator (STDBY_XOSC) or 13 MHz RC oscillator (STDBY_RC) selection in STDBY mode is determined by mode parameter in command *SetStandby(oscillatorMode)* command.

If XOSC is used in conjunction with the DC-DC supply regulation, the DC-DC is automatically powered in STDBY_XOSC mode.

10.4 Frequency Synthesis (FS) Mode

In FS mode, PLL and related regulators are switched ON. The BUSY pin goes low as soon as the PLL is locked.

The radio may be requested to remain in this mode by using the *SetFs()* command.

Since the transceiver uses a low IF architecture, the Rx and Tx frequencies are different. The Rx frequency is equal to Tx frequency minus the intermediate frequency (IF) offset (1.3 MHz by default). In FS mode the frequency to which the PLL is tuned corresponds to the transmit frequency.

10.5 Receive (Rx) Mode

In Rx mode the LNA, MIXER, PLL and selected modem (LoRa/FSK/FLRC) are turned ON. In continuous mode the device remains in Rx mode and looks for incoming packets until the host requests a different mode. In single mode the device returns automatically to STDBY_RC mode.

The transition to receive mode is made by issuing the *SetRx(periodBase, periodBaseCount)* command with the *periodBase* oscillator timebase and *periodBaseCount* number of clock ticks specifying the time-out upon which receive mode will be exited to STDBY_RC mode. The process of periodic reception can be fully automated in the transceiver. This process and the processing specific to each modulation format are described in Section 13.1.3 "Rx Setting and Operations" on page 92.

10.6 Transmit (Tx) Mode

In Tx mode, after ramp-up the Power-Amplifier (PA) transmits the data buffer. The device returns automatically to STDBY_RC after transmitting the packet.

10.7 Transceiver Circuit Modes Graphical Illustration

All of the device operating modes and the states through which each mode selection transitions is shown in the figure below:

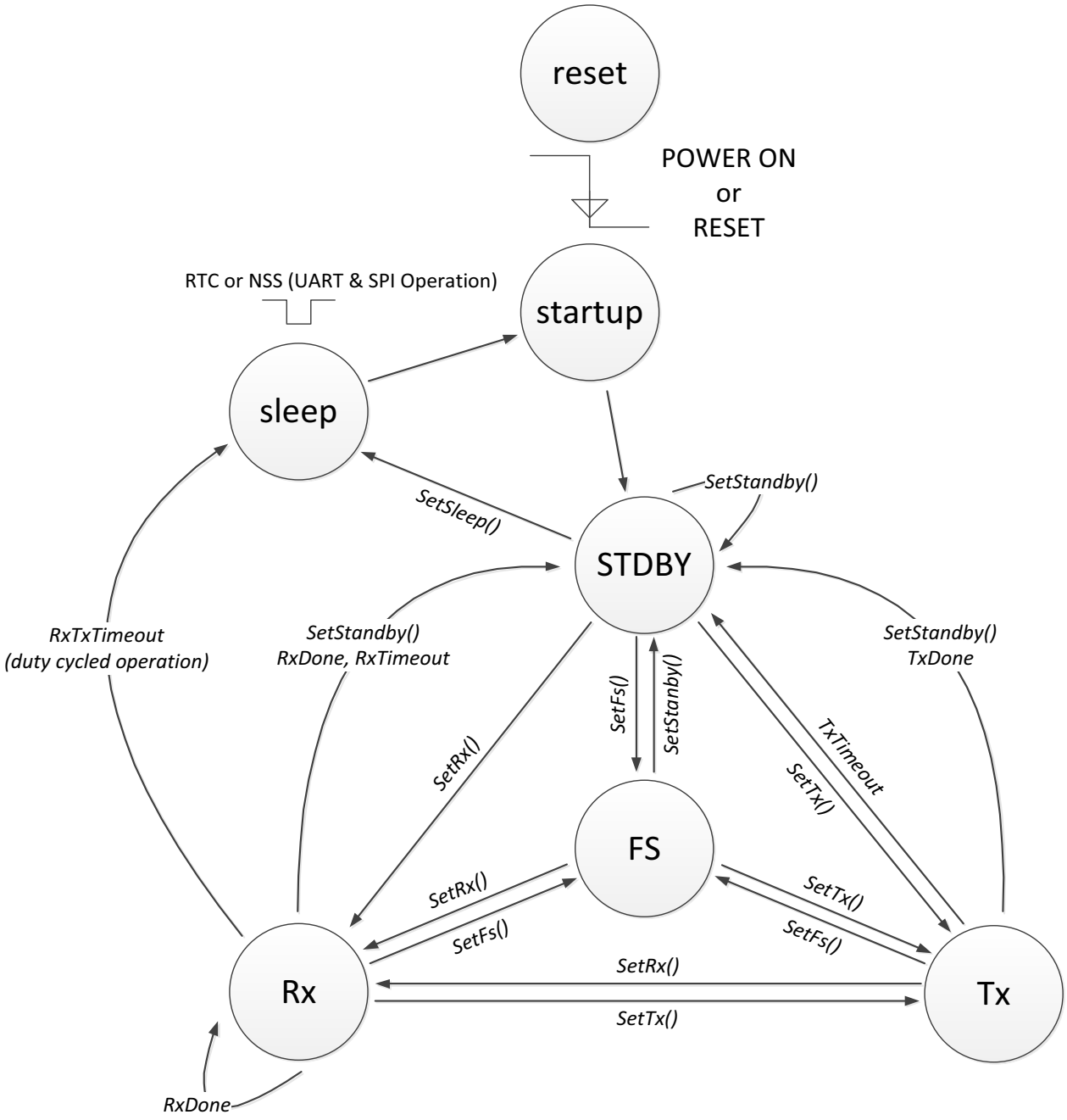


Figure 10-1: Transceiver Circuit Modes

10.8 Active Mode Switching Time

At each transaction with the transceiver (register read/write operation or mode switching) the BUSY pin is set to high during the transaction and while the transceiver is processing the command. The BUSY pin is set back to zero once the transceiver is ready for new commands or has reached a stable mode. In the following figure, the switching time is defined as the time between the rising edge of the NSS ending the SPI transaction and the falling edge of the BUSY pin.

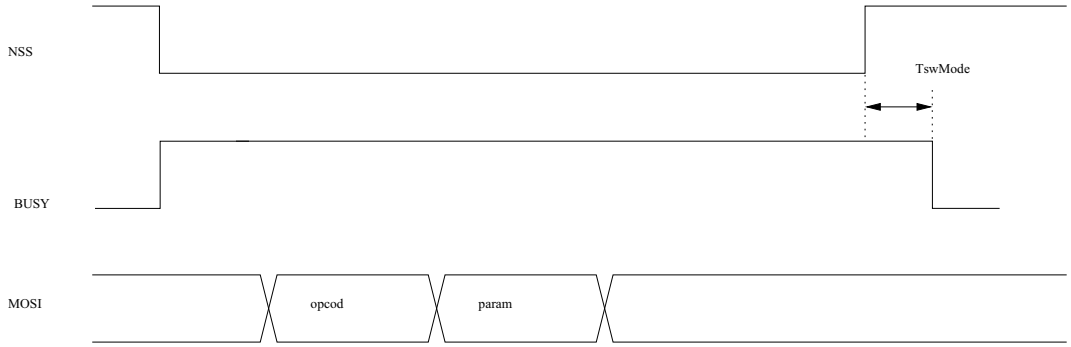


Figure 10-2: Switching Time Definition in Active Mode

Table 10-2: Switching Time (TswMode) for all Possible Transitions

Transition	TswMode Typical Value [µs]	
SLEEP to STDBY_RC	1700	no data retention
SLEEP to STDBY_RC	250	with data retention
STDBY_RC to STDBY_XOSC	53	
STDBY_RC to FS	83	
STDBY_RC to Rx	115	
STDBY_RC to Tx	102	
STDBY_XOSC to FS	40	
STDBY_XOSC to Tx	54	
STDBY_XOSC to Rx	68	
FS to Rx	34	
FS to Tx	27	
Rx to FS	13	
Rx to Tx	39	
Tx to FS	31	
Tx to Rx	60	

In FS, BUSY pin will go low when the PLL is locked.

In Rx, BUSY pin will go to zero as soon as the Rx is up and ready to receive data.

In Tx, BUSY pin will go low when the PA has ramp-up and transmission of preamble starts.

11. Host Controller Interface

Through SPI or UART interface, the host can issue commands to the transceiver or access the data memory space to directly retrieve or write data. In normal operation a reduced number of direct data write operations is required except for data buffer. The user interacts with the transceiver through an API (instruction set).

The transceiver uses a BUSY pin to indicate the status of the transceiver and its ability to receive a command while it is completing its internal processing. Prior to the host transmitting a command, it is thus necessary to check the status of the BUSY pin to ensure that the transceiver is in a state to process it.

Two types of transactions are supported:

- Configuration transaction (STDBY): provides to the host a direct register access i.e. it is used for writing or reading the transceiver configuration registers or the data buffer.
- Command transaction (CMD): requires more complex, non-atomic operations such as packet transmission / reception or mode switching.

11.1 Command Structure

In case of a command that does not pass any parameters, the host sends only the opcode (1 byte) for both SPI and UART interface.

In case of a command that requires parameters:

- In case of a SPI transfer the opcode byte is followed immediately by parameter bytes with NSS rising edge terminating the command.

Table 11-1: SPI interface Command Sequence

Byte	0	[1:n]
Data from host	opcode	parameters
Data to host	status	status

- In case of a UART transfer the opcode byte is followed by length byte (i.e. the number of parameter bytes) and then by parameter bytes.

Table 11-2: UART Interface Command Sequence

Byte	0	1	[2:n]
Data from host	opcode	length	parameters

- In case of a UART buffer write operation, after having sent the LSB of the address, the host must send a byte defining the number of data bytes that will be transmitted. That number of data bytes must then be transmitted.

- In case of a UART buffer read operation, after having sent the LSB of the address, the host must send a byte defining the number of data bytes that will be received. The transceiver will then transmit that number of bytes to the host.
- In case of a UART direct register operation, after having sent the opcode and the address, UART has to send the number of bytes to be read/written.

11.2 GetStatus Command

The host can retrieve the transceiver status directly through the *GetStatus()* command: this command can be issued at any time. In case of an SPI interface, the device returns the status on the same transaction; in case of a UART frame, the status is returned by a write transaction following the command. When using the SPI interface, the *GetStatus()* command is not strictly necessary since the device returns status information also on command bytes. The status byte returned is described in the following table:

Table 11-3: Status Byte Definition

7:5	4:2	1	0
Circuit mode	Command status	Reserved	Busy
0x0: Reserved	0x0: Reserved	-	This bit is 1 when the transceiver is processing command or doing internal operation. It reflects the BUSY pin status.
0x1: Reserved	0x1: Transceiver has successfully processed the command[1]		
0x2: STDBY_RC	0x2: Data are available to host[2]		
0x3: STDBY_XOSC	0x3: Command time-out[3]		
0x4: FS	0x4: Command processing error[4]		
0x5: Rx	0x5: Failure to execute command[5]		
0x6: Tx	0x6: Command Tx done[6]		

1. The command has been terminated correctly
2. A packet has been successfully received and data can be retrieved
3. A transaction from the host took too long to complete and triggered an internal watchdog. The watchdog mechanism can be disabled by the host, it is meant to prevent a dead-lock situation. In this case host should resend the command.
4. The transceiver was unable to process command either because of an invalid opcode or because an incorrect number of parameters has been provided.
5. The command was successfully processed, however the transceiver could not execute the command; for instance it was unable to enter the specified device mode or send the requested data,
6. The transmission of the current packet has terminated

The SPI transaction for *GetStatus()* command is given in Table 11-4, and the UART transaction on Table 11-5.

Table 11-4: GetStatus Data Transfert (SPI)

Byte	0
Data from host	Opcode = 0xC0
Data to host	status

Table 11-5: GetStatus Data Transfert (UART)

Byte	0	1
Data from host	Opcode = 0xC0	-
Data to host	-	status

11.3 Register Access Operations

11.3.1 WriteRegister Command

The command *WriteRegister()* writes a block of bytes in a data memory space starting at a specific address. The address is auto incremented after each data byte so that data is stored in contiguous memory locations. The SPI data transfer is described on Table 11-6 and UART data transfer is described on Table 11-7.

Table 11-6: WriteRegister Data Transfer (SPI)

Byte	0	1	2	3	4	...	n
Data from host	Opcode = 0x18	address[15:8]	address[7:0]	data@address	data@address+1	...	data@address+ (n-3)
Data to host	status	status	status	status	status	...	status

Table 11-7: WriteRegister Data Transfer (UART)

Byte	0	1	2	3	4	5	...	n
Host UART Tx	Opcode = 0x18	address[15:8]	address[7:0]	length = (n-4)	data@address	data@address+1	...	data@address+ (n-4)

11.3.2 ReadRegister Command

The command *ReadRegister()* reads a block of data starting at a given address. The address is auto incremented after each byte. The SPI data transfer is described in Table 11-8, and the UART data transfer in Table 11-9. In UART case, the number of data to be read is provided by length parameter.

Note that when using SPI, the host has to send a NOP after sending the 2 bytes of address to start receiving data bytes on the next NOP sent.

Table 11-8: ReadRegister Data Transfer (SPI)

Byte	0	1	2	3	4	5	...	n
Data from host	Opcode = 0x19	address[15:8]	address[7:0]	NOP	NOP	NOP	...	NOP
Data to host	status	status	status	status	data@address	data@address+1	...	data@address+ (n-4)

Table 11-9: ReadRegister Data Transfer (UART)

Byte	0	1	2	3	4	5	...	n
Host UART Tx	Opcode = 0x19	address[15:8]	address[7:0]	length	---	----	...	----
Chip UART Tx	---	----	---	----	data@address	data@address+1	...	data@address+ (n-4)

11.4 Data Buffer Operations

11.4.1 WriteBuffer Command

This function is used to write the data payload to be transmitted. The address is auto-incremented, when the address exceeds 255 it wraps back to 0 due to the circular nature of data buffer. The address starts from the offset given as a parameter of the function. Table 11-10 describes SPI data transfer, and Table 11-11 describes UART data transfer.

Table 11-10: WriteBuffer SPI Data Transfer

Byte	0	1	2	3	...	n
Data from host	Opcode = 0x1A	offset	data@offset	data@offset+1	...	data@offset+(n-2)
Data to host	status	status	status	status	...	status

Table 11-11: WriteBuffer UART Data Transfer

Byte	0	1	2	3	4	..	n
Host UART Tx	Opcode =0x1A	offset	length = n-3	data@address	data@address+1	...	data@address+(n-3)
Chip UART Tx	---	----	---	---	----	---	---

11.4.2 ReadBuffer

This function allows reading (n-3) bytes of payload received starting at offset.

Note the NOP to be sent if using SPI after sending the offset.

Table 11-12: ReadBuffer SPI Data Transfer

Byte	0	1	2	3	4	...	n
Data from host	Opcode = 0x1B	offset	NOP	NOP	NOP	...	NOP
Data to host	status	status	status	data@offset	data@offset+1	...	data@offset+(n-3)

Table 11-13: ReadBuffer UART Data Transfer

Byte	0	1	2	3	4	...	n
Host UART Tx	opcode = 0x1B	offset	length	---	----	...	NOP
Device UART Tx	---	----	---	data@offset	data@offset+1	...	data@offset+(n-4)

11.5 Radio Operation Modes

This chapter describes the command set available for the transceiver. The transaction is given for SPI only but the same commands are available when using UART.

11.5.1 SetSleep

The *SetSleep()* command is used to set the transceiver to Sleep mode with the lowest current consumption possible. This command can be sent only in STDBY mode (STDBY_RC or STDBY_XOSC). After rising edge of NSS, all blocks are switched OFF except backup regulator if needed and the blocks specified in *sleepConfig* parameter.

Table 11-14: SetSleep SPI Data Transfer

Byte	0	1
Data from host	Opcode = 0x84	sleepConfig

In a UART transaction, the host sends the same bytes as for a SPI transaction. The *sleepConfig* argument is defined as:

Table 11-15: Sleep Mode Definition.

sleepConfig[7:4]	sleepConfig[2]	sleepConfig[1]	sleepConfig[0]
Unused	0: Instruction Ram is flushed during Sleep mode (equivalent to a reset)	0: Data buffer is flushed during Sleep Mode	0: Data Ram is flushed during Sleep Mode
	1: instruction RAM in retention mode	1: Data buffer in retention mode	1: Data Ram in retention mode

The transceiver mode will move from SLEEP to STDBY_RC if either a rising edge of NSS or the transceiver RTC has been programmed to wake itself up

When the transceiver enters Sleep mode the contents of the registers are lost. To avoid this, the *SaveContext* command must be performed and the *SetSleep* command must use the *sleepConfig[0]* bit which, when set to 1, allows the register contents to be stored in a data memory location. The data memory is retained and upon transition from SLEEP to STANDBY_RC the registers are populated with the values previously stored in the data memory. This reduces interactions between host and transceiver, useful in systems that regularly put the transceiver to sleep in order to save power.

11.5.2 SetStandby

The command *SetStandby()* is used to set the device in either STDBY_RC or STDBY_XOSC mode which are intermediate levels of power consumption. In this mode, the transceiver may be configured for future RF operations.

After power on or application of a reset, the transceiver will enter in STDBY_RC mode running with a 13 MHz RC clock.

Table 11-16: SetStandby SPI Data Transfer

Byte	0	1
Data from host	Opcode = 0x80	StandbyConfig

Table 11-17: SetStandby UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x80	0x01	StandbyConfig

The *StandbyConfig* byte definition is given in next table:

Table 11-18: StandbyConfig Definition

StandbyConfig	value	description
STDBY_RC	0	Device running on RC 13MHz, set STDBY_RC mode
STDBY_XOSC	1	Device running on XTAL 52MHz, set STDBY_XOSC mode

11.5.3 SetFs

The command *SetFs()* is used to set the device in Frequency Synthesis mode where the PLL is locked to the carrier frequency. This mode is used for test purposes of the PLL and can be considered as an intermediate mode. It is automatically reached when going from STDBY_RC mode to Tx mode or to Rx mode. The data transfer for this command is the same for SPI and UART.

Table 11-19: SetFs Data Transfer

Byte	0
Data from host	Opcode = 0xC1

11.5.4 SetTx

The command *SetTx()* sets the device in Transmit mode.

The IRQ status should be cleared prior to using this command, see Section 11.8.3 "ClearIrqStatus" on page 82.

Table 11-20: SetTx SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x83	periodBase	periodBaseCount[15:8]	periodBaseCount[7:0]

Table 11-21: SetTx UART Data Transfer

Byte	0	1	2	3	4
Data from host	Opcode = 0x83	0x03	periodBase	periodBaseCount[15:8]	periodBaseCount[7:0]

Starting from STDBY_RC mode the oscillator is switched ON followed by the PLL, then the PA (Power Amplifier) is switched ON and the PA regulator starts ramping according to the ramp-up time defined by *SetTxParam()* command. Once the ramp-up is complete the packet handling starts the packet transmission. Once the last bit of the packet has been sent, the PA regulator is ramped down, the PA is switched OFF, the transceiver goes back to STDBY_RC mode and an IRQ TxDone is generated. A TIMEOUT IRQ is triggered if the TxDone IRQ is not generated. The transceiver goes back to STDBY_RC mode after a TIMEOUT IRQ or a TxDone IRQ.

The time-out duration is computed by the formula:

Time-out duration = **periodBase** * **periodBaseCount**

Where *periodBase* is the step of the RTC defined in the next table.

Table 11-22: SetTx Time-out Definition.

periodBase	Time-out step
0x00	15.625 µs
0x01	62.5 µs
0x02	1 ms
0x03	4 ms

periodBaseCount is a 16-bit parameter defining the number of steps used during time-out as defined in the following table:

Table 11-23: SetTx Time-out Duration

periodBaseCount[15:0]	Time-out duration
0x0000	No time-out, Tx Single mode, the device will stay in Tx Mode until the packet is transmitted and returns in STDBY_RC mode upon completion.
Others	Time-out active, the device remains in Tx mode, it returns automatically to STDBY_RC mode on timer end-of-count or when a packet has been transmitted.

11.5.5 SetRx

The command *SetRx()* sets the device in Receiver mode.

The IRQ status should be cleared prior to using this command, see Section 11.8.3 "ClearIrqStatus" on page 82.

Table 11-24: SetRx SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x82	periodBase	periodBaseCount[15:8]	periodBaseCount[7:0]

Table 11-25: SetRx UART Data Transfer

Byte	0	1	2	3	4
Data from host	Opcode = 0x82	0x03	periodBase	periodBaseCount[15:8]	periodBaseCount[7:0]

This command sets the transceiver in Rx mode, waiting for the reception of one or several packets. The Receiver mode operates with a time-out to provide maximum flexibility to the end user. The parameters for time-out duration are:

Time-out duration = **periodBase** * **periodBaseCount**

Where *periodBase* is the step of the RTC as defined in Table 11-22.

periodBaseCount is the number of steps used during time-out as defined in the following table:

Table 11-26: SetRx Time-out Duration

TickNum(15:0)	Time-out duration
0x0000	No time-out. Rx Single mode. The device will stay in Rx mode until a reception occurs and the devices return in STDBY_RC mode upon completion
0xFFFF	Rx Continuous mode. The device remains in Rx mode until the host sends a command to change the operation mode. The device can receive several packets. Each time a packet is received, a "packet received" indication is given to the host and the device will continue to search for a new packet.
Others	Time-out active. The device remains in Rx mode, it returns automatically to STDBY_RC mode on timer end-of-count or when a packet has been received. As soon as a packet is detected, the timer is automatically disabled to allow complete reception of the packet.

11.5.6 SetRxDutyCycle

This command sets the transceiver in sniff mode, so that it regularly looks for new packets (duty cycled operation).

Table 11-27: Duty Cycled Operation SPI Data Transfer

Byte	0	1	2	3	5	6
Data from host	Opcode= 0x94	PeriodBase	rxPeriodBase Count [15:8]	rxPeriodBase Count[7:0]	sleepPeriodBase Count [15:8]	sleepPeriodBase Count [7:0]

Table 11-28: Duty Cycled Operation UART Data Transfer

Byte	0	1	2	3	4	6	7
Data from host	Opcode = 0x94	0x05	PeriodBase	rxPeriodBase Count [15:8]	rxPeriodBase Count[7:0]	sleepPeriodBase Count [15:8]	sleepPeriodBase Count [7:0]

Once this command is sent in STDBY_RC mode, the context (Rx configuration) is saved into the data RAM and the transceiver starts a loop defined by the following steps:

- Enter Rx and listen for a packet for a period of time defined by PeriodBase and rxPeriodBaseCount.
- The transceiver looks for a preamble made of a 0101.... If the preamble is detected, the transceiver looks for a Sync Word and payload.
- If no packet is received during Rx window, the transceiver goes in Sleep mode (with context saved) for a period of time defined by PeriodBase and sleepPeriodBaseCount.

- At the end of the Sleep window, the transceiver leaves the Sleep mode, restores context and enters in the Rx mode after context restoration and listens for a packet during Rx window 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, the transceiver interrupts the host via the RxDone flag and returns to STDBY_RC mode.
- The host issues a *SetStandby()* command during the Rx window (within Sleep the window device is unable to receive commands).

Note that to use the RxDone interrupt, you have to enable the corresponding IRQ prior to enter Duty cycled operation. To enable the RxDone IRQ, refer to the command *SetDioIrqParams()* in Section 11.8.1 "SetDioIrqParams" on page 81.

The Sleep mode duration is defined by:

Sleep Duration = PeriodBase * sleepPeriodBaseCount

The Rx mode duration is defined by

Rx Duration = PeriodBase * rxPeriodBaseCount

Where PeriodBase is defined as periodBase in Table 11-22.

rxPeriodBaseCount and sleepPeriodBaseCount are 16-bit parameters defining the number of steps used to define the Rx duration and Sleep durations. Some specific values for rxPeriodBaseCount are given in Table 11-29.

Table 11-29: Rx Duration Definition.

rxPeriodBaseCount[15:0]	Time-out duration
0x0000	The transceiver waits until a packet is found. Once found, the transceiver goes to STDBY_RC mode after sending an RxDone IRQ to the host
Others	The device will stay in Rx Mode until the end of the timer when the device returns in Sleep mode for Sleep duration

Notice! The command SetLongPreamble must be issued prior to SetRxDutyCycle.

11.5.7 SetLongPreamble

The command (opcode 0x98) sets the transceiver into Long Preamble mode, and can only be used with either the LoRa mode and GFSK mode. In this mode, the behavior of the commands *SetTx*, *SetRx* and *SetRxDutyCycle* is modified as:

- In GFSK only, the *SetTx* arguments do not define a timeout anymore, but the time of the preamble part of GFSK packet. Therefore, there is no *TxTimeout* interrupt generated in GFSK mode. In LoRa, the *SetTx* behavior is not changed.
- In GFSK only with LongPreamble mode, the preamble detection mode is activated. The command *SetRx* can then generate an interrupt for Preamble detection.
- In GFSK and LoRa, the behavior of *RxDutyCycle* is modified so that if a preamble is detected, the Rx window is extended by SleepPeriod + 2 * RxPeriod.

Table 11-30: SetLongPreamble Data Transfer

Byte	0	1
Data from host	Opcode = 0x9B	Enable

11.5.8 SetCAD

The command *SetCAD()* (Channel Activity Detection) can be used only in LoRa packet type. The Channel Activity Detection is a LoRa specific mode of operation where the device searches for a LoRa signal. After search has completed, the device returns to STDBY_RC mode. The length of the search is configured via *SetCadParams()* command.

At the end of search period the device always sends the CadDone IRQ. If a valid signal has been detected it also generates the CadDetected IRQ.

This mode of operation is especially useful in all the applications requiring Listen before Talk.

The UART data transfer and SPI data transfer are the same.

Table 11-31: SetCAD Data Transfer

Byte	0
Data from host	Opcode = 0xC5

11.5.9 SetTxContinuousWave

The command *SetTxContinuousWave()* is a test command to generate a Continuous Wave (RF tone) at a selected frequency and output power. The device remains in Tx Continuous Wave until the host sends a mode configuration command. This command is available for all packet types.

Table 11-32: SetTxContinuousWave Data Transfer

Byte	0
Data from host	Opcode = 0xD1

The UART data transfer and SPI data transfer are the same.

11.5.10 SetTxContinuousPreamble

The command *SetTxContinuousPreamble()* is a test command to generate an infinite sequence of alternating '0's and '1's in GFSK, BLE, or FLRC modulation and symbol 0 in LoRa. The device remains in Tx Continuous Wave until the host sends a mode configuration command.

The UART data transfer and SPI data transfer are the same.

Table 11-33: SetTxContinuousPreamble Data Transfer

Byte	0
Data from host	Opcode = 0xD2

11.5.11 SetAutoTx

BLE requires the transceiver to be able to send back a response 125 µs after a packet reception. This is carried out by sending the command *SetAutoTx()* which allows the transceiver to send a packet at a user programmable time (time) after the end of a packet reception. *SetAutoTx()* must be issued in STDBY_RC mode. The data transfer of *SetAutoTx()* is described in Table 11-34.

Table 11-34: SetAutoTx SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x98	time[15:8]	time[7:0]

Table 11-35: SetAutoTx UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x98	0x02	time[15:8]	time[7:0]

time is expressed in µs. The delay between the end of reception of a packet and the start of the transmission of the next packet is defined by:

$$TxDelay = time + Offset$$

Here Offset is a time needed for the transceiver to switch modes and is equal to 33 µs.

Once this command is issued, each time the transceiver goes in Rx mode, it will automatically switch to Tx and send a packet in a predefined time TxDelay. To have normal operation (going in STDBY_RC after Rx) one needs to use the command SetAutoTx with 0x00 as time argument.

11.5.12 SetAutoFs

This feature modifies the chip behavior so that the state following a Rx or Tx operation is FS and not STDBY (see Section 10.7 "Transceiver Circuit Modes Graphical Illustration" on page 57). This feature is to be used to reduce the switching time between consecutive Rx and/or Tx operations (see Table 10-2: Switching Time (TswMode) for all Possible Transitions).

- To activate the AutoFs feature, use the command *SetAutoFs* with argument *true*
- To deactivate the AutoFs feature, use the command *SetAutoFs* with *false*.

Table 11-36: SetAutoFs SPI Data Transfer

Byte	0	1
Data from host	Opcode = 0x9E	enable=0x01, disable=0x00

Table 11-37: SetAutoFs UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x9E	0x1	enable

11.6 Radio Configuration

11.6.1 SetPacketType

The command *SetPacketType()* sets the transceiver radio frame out of a choice of 6 different packet types. Despite some of them using the same physical modem, they do not all share the same parameters.

Note! The command *SetPacketType()* must be the first of a radio configuration sequence.

Table 11-38: SetPacketType SPI Data Transfer

Byte	0	1
Data from host	Opcode = 0x8A	packetType

Table 11-39: SetPacketType UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x8A	0x01	packetType

The parameter for this command is defined in Table 11-40.

Table 11-40: PacketType Definition

packetType	Value	Modem mode of operation
PACKET_TYPE_GFSK	0x00 [default]	GFSK mode
PACKET_TYPE_LORA	0x01	LoRa mode
PACKET_TYPE_RANGING	0x02	Ranging Engine mode
PACKET_TYPE_FLRC	0x03	FLRC mode
PACKET_TYPE_BLE	0x04	BLE mode
Reserved	>=5	Reserved

Changing from one mode of operation to another is carried by sending the *SetPacketType()* command. The parameters from the previous mode are not kept internally. The switch from one frame to another must be carried out in STDBY_RC mode.

11.6.2 GetPacketType

The command *GetPacketType()* returns the current operating packet type of the radio.

Table 11-41: GetPacketType SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x03	NOP	NOP
Data to host	status	status	packetType

Table 11-42: GetPacketType UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x03	0x01	-
Data to host	-	-	packetType

11.6.3 SetRfFrequency

The command *SetRfFrequency()* is used to set the frequency of the RF frequency mode.

Table 11-43: SetRfFrequency SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x86	rfFrequency[23:16]	rfFrequency[15:8]	rfFrequency[7:0]

Table 11-44: SetRfFrequency UART Data Transfer

Byte	0	1	2	3	4
Data from host	Opcode = 0x86	0x03	rfFrequency[23:16]	rfFrequency[15:8]	rfFrequency[7:0]

The LSB of rfFrequency is equal to the PLL step i.e. 52e6/2^18 Hz, where 52e6 is the crystal frquency in Hz. *SetRfFrequency()* defines the Tx frequency. The Rx frequency is lowered by IF. The IF is by default set to 1.3 MHz.

11.6.4 SetTxParams

This command sets the Tx output power using parameter power and the Tx ramp time using parameter rampTime. This command is available for all packetType.

Table 11-45: SetTxParams SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x8E	power	rampTime

Table 11-46: SetTxParams UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x8E	0x02	power	rampTime

The output power (Pout) is defined by parameter power.

$$P_{out} = -18 + power\ dBm$$

P_{outMax} = - 18 dBm (power = 0)

P_{outMax} = 13 dBm (power = 31)

The desired power amplifier ramp time is defined using rampTime parameter according to Table 11-47.

Table 11-47: RampTime Definition

rampTime	Value	Ramp time (µs)
RADIO_RAMP_02_US	0x00	2
RADIO_RAMP_04_US	0x20	4
RADIO_RAMP_06_US	0x40	6
RADIO_RAMP_08_US	0x60	8
RADIO_RAMP_10_US	0x80	10
RADIO_RAMP_12_US	0xA0	12
RADIO_RAMP_16_US	0xC0	16
RADIO_RAMP_20_US	0xE0	20

11.6.5 SetCadParams

The command *SetCadParams()* defines the number of symbols on which Channel Activity Detected (CAD) operates.

Table 11-48: CAD SPI Data Transfer

Byte	0	1
Data from host	Opcode = 0x88	cadSymbolNum

Table 11-49: CAD UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x88	0x01	cadSymbolNum

The number of symbol to be used is defined in the next table.

Table 11-50: CadSymbolNum Definition

cadSymbolNum	Value	Number of symbols used for CAD
LORA_CAD_01_SYMBOL	0x00	1
LORA_CAD_02_SYMBOLS	0x20	2
LORA_CAD_04_SYMBOLS	0x40	4
LORA_CAD_08_SYMBOLS	0x60	8
LORA_CAD_16_SYMBOLS	0x80	16

Notice: for symbols 1 & 2, there are higher risks of false detection.

11.6.6 SetBufferBaseAddress

This command fixes the base address for the packet handing operation in Tx and Rx mode for all packet types.

Table 11-51: SetBufferBaseAddress SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x8F	txBaseAdress	rxBaseAdress

Table 11-52: SetBufferBaseAddress UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x8F	0x02	txBaseAdress	rxBaseAdress

11.6.7 SetModulationParams

The command *SetModulationParams()* is used to configure the modulation parameters of the radio. The parameters passed by this function will be interpreted depending on the frame type, which should have been set to the required type before calling this function.

Table 11-53: SetModulationParams SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x8B	param[0]	param[1]	param[2]

Table 11-54: SetModulationParams UART Data Transfer

Byte	0	1	2	3	4
Data from host	Opcode = 0x8B	0x03	param[0]	param[1]	param[2]

In GFSK, FLRC and BLE the bitrate and the bandwidth are defined by param[0] parameter as a pair of values, see Section Table 13-1: "Modulation Parameters in GFSK Mode" on page 85. The modulation index is used in conjunction with the bitrate to calculate the Frequency Deviation used for the transmission or reception. The modulation index is defined by param[1] parameter. The BT represents the Gaussian filter which can be used to filter the modulation stream at the transmitter side. BT is defined by param[2] parameter. The parameter's meaning depends on the chosen packet type and will be defined in the chapter dedicated to the selected packet type.

In LoRa packet type, SF corresponds to the Spreading Factor used for the LoRa modulation. SF is defined by param[0] parameter. The BW corresponds to the bandwidth onto which the LoRa signal is spread. BW in LoRa is defined by param[1] parameter.

The LoRa payload is fitted with a forward error correcting mechanism which has several levels of encoding. Coding Rate (CR) is defined by param[2] parameter in LoRa.

The definition of *SetModulationParams()* parameters are summarized in the following table:

Table 11-55: SetModulationParams Parameters Definition

Parameter	BLE and GFSK	FLRC	LoRa and Ranging Engine
modParam1	BitrateBandwidth	BitrateBandwidth	SpreadingFactor
modParam2	ModulationIndex	CodingRate	Bandwidth
modParam3	ModulationShaping	ModulationShaping	CodingRate

11.6.8 SetPacketParams

This command is used to set the parameters of the packet handling block.

Table 11-56: SetPacketParams SPI Data Transfer

Byte	0	1	2	3	4	5	6	7
Data from host	Opcode= 0x8C	SetPacketParam1	SetPacketParam2	SetPacketParam3	SetPacketParam4	SetPacketParam5	SetPacketParam6	SetPacketParam7

Table 11-57: SetPacketParams UART Data Transfer

Byte	0	1	2	3	4	5	6	7	8
Data from host	Opcode = 0x8C	0x07	SetPacketParam1	SetPacketParam2	SetPacketParam3	SetPacketParam4	SetPacketParam5	SetPacketParam6	SetPacketParam7

The packet parameters meaning depends on the chosen packet type. Table 11-58 gives the meaning of the parameters according to the packet types.

Table 11-58: SetPacketParams Parameters Definition.

Parameter	GFSK and FLRC	BLE	LoRa and Ranging Engine
SetPacketParam1	PreambleLength	ConnectionState	PreambleLength
SetPacketParam2	SyncWordLength	CrcLength	HeaderType
SetPacketParam3	SyncWordMatch	BleTestPayload	PayloadLength
SetPacketParam4	HeaderType	Whitening	CRC
SetPacketParam5	PayloadLength	not used	InvertIQ/chirp invert
SetPacketParam6	CrcLength	not used	not used
SetPacketParam7	Whitening	not used	not used

The usage and definition of those parameters are described in the different packet type sections.

11.7 Communication Status Information

These commands return information about the transceiver status, received packet length, reception power and several flags indicating if the packet has been correctly received. The returned parameters are common to all frames except LoRa.

11.7.1 GetRxBufferStatus

This command returns the length of the last received packet (payloadLengthRx) and the address of the first byte received (rxBufferOffset), it is applicable to all modems. The address is an offset relative to the first byte of the data buffer.

Table 11-59: GetRxBufferStatus SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x17	NOP	NOP	NOP
Data to host	status	status	rxPayloadLength	rxStartBufferPointer

Table 11-60: GetRxBufferStatus UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x17	0x02	-	-
Data to host	-	-	rxPayloadLength	rxStartBufferPointer

Note that in LoRa packet type with fixed header (see Section 7.4.3 "Implicit (Fixed-length) Header Mode" on page 44) the *GetRxBufferStatus* always returns 0x00 for *rxPayloadLength*. Indeed, in this configuration, no header is present in the packet so the payload size cannot be extracted from it. However, it is possible to recover the payload size configured in the radio by direct register reading. Hence, reading register 0x901 will return the payload size. The data transfer for register reading is described in Section 11.3.2 "ReadRegister Command" on page 62."

11.7.2 GetPacketStatus

This command is used to retrieve information about the last received packet. The returned parameters are frame-dependent.

Table 11-61: GetPacketStatus SPI Data Transfer

Byte	0	1	2	3	4	5	6
Data from host	Opcode = 0x1D	NOP	NOP	NOP	NOP	NOP	NOP
Data to host	status	status	packetStatus [7:0]	packetStatus [15:8]	packetStatus [23:16]	packetStatus [31:24]	packetStatus [39:32]

Table 11-62: GetPacketStatus UART Data Transfer

Byte	0	1	2	3	4	5	6
Data from host	Opcode = 0x1D	0x05	-	-	-	-	-
Data to host	-	-	packetStatus [0:7]	packetStatus [8:15]	packetStatus [16:23]	packetStatus [24:31]	packetStatus [32:39]

The value returned by *GetPacketStatus()* command is packet-type-dependent, and summarized in Table 11-63.

In the case of LoRa and/or Ranging Engine, there are only 2 bytes returned by the command.

Table 11-63: packetStatus Definition

Parameter	BLE, GFSK, FLRC	LoRa and Ranging Engine
packetStatus[7:0]	RFU	rssiSync
packetStatus[15:8]	rssiSync	snr
packetStatus[16:23]	errors	-
packetStatus[24:31]	status	-
packetStatus[32:39]	sync	-

The meaning of rssiSync and snr is given in the next table. Note that snr is only available in LoRa and Ranging Engine packet types.

Table 11-64: RSSI and SNR Packet Status

Value	Description
rssiSync	RSSI value latched upon the detection of the sync address. Actual signal power is $-(\mathbf{rssiSync})/2$ (dBm)
snr	Estimation of SNR on last packet received.In two's compliment format multiplied by 4. Actual SNR is $(\mathbf{snr})/4$ (dB)

Table 11-65: Status Packet Status Byte

PStatus3	Symbol	Description
bit 7:6	reserved	reserved
bit 5	rxNoAck	NO_ACK field of the received packet. Only applicable in Rx for dynamic length packets.
bit 4:1	reserved	reserved
bit 0	pktSent	Indicates that the packet transmission is complete. Does not signify packet validity. Only applicable in Tx.

Table 11-66: Error Packet Status Byte

Error	Symbol	Description
bit 7	reserved	reserved
bit 6	SyncError	sync address detection status for the current packet Only applicable in Rx when sync address detection is enabled.
bit 5	LengthError	Asserted when the length of the received packet is greater than the Max length defined in the PAYLOAD_LENGTH parameter. Only applicable in Rx for dynamic length packets.
bit 4	CrcError	CRC check status of the current packet. The packet is available anyway in the FIFO. Only applicable in Rx when the CRC check is enabled
bit 3	AbortError	Abort status indicates if the current packet in Rx/Tx was aborted. Applicable both in Rx & Tx.
bit 2	headerReceived	Indicates if the header for the current packet was received. Only applicable in Rx for dynamic length packets
bit 1	packetReceived	Indicates that the packet reception is complete. Does not signify packet validity. Only applicable in Rx.
bit 0	packetCtrlBusy	Indicates that the packet controller is busy. Applicable both in Rx/Tx

Table 11-67: Sync Packet Status Byte

Sync	Symbol	Description
bit 7:3	reserved	reserved
bit 2:0	syncAddrsCode	Code of the sync address detected 000: sync address detection error 001: sync_adrs_1' detected 010: sync_adrs_2', detected 100: sync_adrs_3' detected

11.7.3 GetRssiInst

This command returns the instantaneous RSSI value during reception of the packet. The command is valid for all frames. In LoRa operation, the instantaneous RSSI is updated at every symbol received.

Table 11-68: GetRssiInst SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x1F	NOP	NOP
Data to host	status	status	rssiInst

Table 11-69: GetRssiInst UART Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x1F	0x01	-
Data to host	-	-	rssiInst

Table 11-70: RssiInst Definition

Parameter	Description
rssiInst	Signal power is $(-\mathbf{rssiInst})/2$ (dBm)

11.8 IRQ Handling

In total there are 16 possible interrupt sources depending on the chosen frame and transceiver mode. Each of them can be enabled or masked. In addition, each of them can be mapped to DIO1, DIO2 or DIO3.

Table 11-71: IRQ Register

Bit	IRQ	Description	Packet
0	TxDone	Tx complete	All
1	RxDone	Rx complete	All
2	SyncWordValid	Sync. word valid	GFSK/BLE/FLRC
3	SyncWordError	Sync. word error	FLRC
4	HeaderValid	Header Valid	LoRa/Ranging Engine
5	HeaderError	Header Error	LoRa/Ranging Engine
6	CrcError	CRC error	GFSK/BLE/FLRC/LoRa
7	RangingSlaveResponseDone	Ranging response complete (Slave)	Ranging Engine
8	RangingSlaveRequestDiscard	Ranging request discarded (Slave)	LoRa/Ranging Engine
9	RangingMasterResultValid	Ranging result valid (Master)	Ranging Engine

Table 11-71: IRQ Register

Bit	IRQ	Description	Packet
10	RangingMasterTimeout	Ranging timeout (Master)	Ranging Engine
11	RangingMasterRequestValid	Ranging Request valid (Slave)	Ranging Engine
12	CadDone	Channel activity check complete	LoRa/Ranging Engine
13	CadDetected	Channel activity detected	LoRa/Ranging Engine
14	RxTxTimeout	Rx or Tx timeout	All
15	PreambleDetected	Preamble Detected	All if *SetLongPreamble* is activated

A dedicated 16-bit register called IRQ_reg is used to log IRQ sources. Each position corresponds to one IRQ source as described in the table above. A set of user commands is used to configure IRQ mask, DIOs mapping and IRQ clearing as explained in the next paragraphs.

11.8.1 SetDioIrqParams

This command is used to enable IRQs and to attach IRQs to DIO pins.

Table 11-72: IRQ Mask Definition SPI Data Transfer

Byte	0	1	2	3	4	5	6	7	8
Data from host	OpCode = 0x8D	irqMask [15:8]	irqMask [7:0]	dio1Mask [15:8]	dio1Mask [7:0]	dio2Mask [15:8]	dio2Mask [7:0]	dio3Mask [15:8]	dio3Mask [7:0]

Table 11-73: IRQ Mask Definition UART Data Transfer

Byte	0	1	2	3	4	5	6	7	8	9
Data from host	OpCode = 0x8D	0x08	irqMask [15:8]	irqMask [7:0]	dio1Mask [15:8]	dio1Mask [7:0]	dio2Mask [15:8]	dio2Mask [7:0]	dio3Mask [15:8]	dio3Mask [7:0]

An interrupt is flagged in IRQ register if the corresponding bit in flag register is set. As an example, TxDone can set bit 0 of IRQ register only if bit 0 of IrqMask is set to 1.

The interrupt causes a DIO to be set if the corresponding bit in dioXMask and the irqMask are set. As an example, if bit 0 of irqMask is set to 1 and bit 0 of dio1Mask is set to 1 then a rising edge of IRQ source TxDone will be logged in IRQ register and will appear at the same time on DIO1.

One IRQ can be mapped to all DIOs, one DIO can be mapped to all IRQs (an OR operation is carried out) but some IRQ source will be available only on certain modes of operation and frame type.

11.8.2 GetIrqStatus

This command returns the value of the IRQ register.

Table 11-74: GetIrqStatus SPI Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x15	NOP	NOP	NOP
Data to host	status	status	irqStatus[15:8]	irqStatus[7:0]

Table 11-75: GetIrqStatus UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x15	0x02	-	-
Data to host	-	-	irqStatus[15:8]	irqStatus[7:0]

11.8.3 ClearIrqStatus

This command clears an IRQ flag in IRQ register.

Table 11-76: ClearIrqStatus SPI Data Transfer

Byte	0	1	2
Data from host	Opcode = 0x97	irqMask[15:8]	irqMask[7:0]

Table 11-77: ClearIrqStatus UART Data Transfer

Byte	0	1	2	3
Data from host	Opcode = 0x97	0x02	irqMask[15:8]	irqMask[7:0]

To clear an IRQ flag in IRQ register, one should set to 1 the bit of irqMask corresponding to the same position as the IRQ flag to be cleared. As an example, if bit 0 of irqMask is set to 1 then the IRQ flag at bit 0 for IRQ register is cleared.

If a DIO is mapped to one single IRQ source, the DIO is cleared if the corresponding bit in the IRQ register is cleared. If DIO is the ORed with several IRQ sources, then the DIO remains set to 1 until all bits mapped to the DIO in the IRQ register are cleared.

12. List of Commands

The next table gives the list of commands and the corresponding opcode.

Table 12-1: Transceiver Available Commands

Command	Opcode	Parameters	Return
GetStatus	0xC0	-	status
WriteRegister	0x18	address[15:8], address[7:0], data[0:n]	-
ReadRegister	0x19	address[15:8], address[7:0]	data[0:n-1]
WriteBuffer	0x1A	offset,data[0:n]	-
ReadBuffer	0x1B	offset	data[0:n-1]
SetSleep	0x84	sleepConfig	-
SetStandby	0x80	standbyConfig	-
SetFs	0xC1	-	-
SetTx	0x83	periodBase, periodBaseCount[15:8], periodBaseCount[7:0]	-
SetRx	0x82	periodBase, periodBaseCount[15:8], periodBaseCount[7:0]	-
SetRxDutyCycle	0x94	rxPeriodBase, rxPeriodBaseCount[15:8], rxPeriodBaseCount[7:0], sleepPeriodBase, sleepPeriodBaseCount[15:8], sleepPeriodBaseCount[7:0]	-
SetCad	0xC5	-	-
SetTxContinuousWave	0xD1	-	-
SetTxContinuousPreamble	0xD2	-	-
SetPacketType	0x8A	packetType	-
GetPacketType	0x03	-	packetType
SetRfFrequency	0x86	rfFrequency[23:16],rfFrequency[15:8], rfFrequency[7:0]	-
SetTxParams	0x8E	power, rampTime	-
SetCadParams	0x88	cadSymbolNum	-
SetBufferBaseAddress	0x8F	txBaseAddress, rxBaseAddress	-
SetModulationParams	0x8B	modParam1, modParam2, modParam3	-

Table 12-1: Transceiver Available Commands

Command	Opcode	Parameters	Return
SetPacketParams	0x8C	packetParam1, packetParam2, packetParam3, packetParam4, packetParam5, packetParam6, packetParam7	-
GetRxBufferStatus	0x17	-	payloadLength, rxBufferOffset
GetPacketStatus	0x1D	-	packetStatus[39:32], packetStatus[31:24], packetStatus[23:16], packetStatus[15:8], packetStatus[7:0]
GetRssiInst	0x1F	-	rssiInst
SetDioIrqParams	0x8D	irqMask[15:8], irqMask[7:0], dio1Mask[15:8],dio1Mask[7:0], dio2Mask[15:8], dio2Mask[7:0], dio3Mask[15:8], dio3Mask[7:0]	-
GetIrqStatus	0x15	-	irqStatus[15:8], irqStatus[7:0]
ClrIrqStatus	0x97	irqMask[15:8], irqMask[7:0]	-
SetRegulatorMode	0x96	regulatorMode	-
SetSaveContext	0xD5	-	-
SetAutoFS	0x9E	0x00: disable or 0x01: enable	-
SetAutoTx	0x98	time	-
SetPerfCounterMode	0x9C	perfCounterMode	-
SetLongPreamble	0x9B	enable	-
SetUartSpeed	0x9D	uartSpeed	-
SetRangingRole	0xA3	0x00=Slave or 0x01=Master	-

13. Transceiver Operation

13.1 GFSK Packet

13.1.1 Common Transceiver Settings

After power up or hard reset the transceiver runs a brief calibration procedure then goes into STDBY_RC mode, indicated by a low state on BUSY pin. From this state the steps (the order is important) needed to either send, or receive, a GFSK format FSK packet are indicated below:

1. If not in STDBY_RC mode, then go to this mode by sending the command:

   SetStandby(STDBY_RC)

2. Define the GFSK packet by sending the command:

   SetPacketType(PACKET_GFSK)

3. Define the RF frequency by sending the command:

   SetRfFrequency(rfFrequency)

The LSB of rfFrequency is equal to the PLL step i.e. 52e6/2^18 Hz. *SetRfFrequency()* defines the Tx frequency.

In Rx the frequency is lowered by the intermediate frequency (IF). The IF is set to 1.3 MHz by default.

4. Indicate the addresses where the packet handler will read (txBaseAddress in Tx) or write (rxBaseAddress in Rx) the first byte of the data payload by sending the command:

   SetBufferBaseAddress(txBaseAddress, rxBaseAddress)

   Note that txBaseAddress and rxBaseAddress are offset relative to the beginning of the data memory map.

5. Define the modulation parameters by sending command

   SetModulationParams(modParam1, modParam2, modParam3)

   The bitrate and bandwidth are configures via the modParam1 setting.

Table 13-1: Modulation Parameters in GFSK Mode

Parameter	Symbol	Value	Bitrate (Mb/s)	Bandwidth (MHz DSB)
modParam1	GFSK_BLE_BR_2_000_BW_2_4	0x04	2	2.4
modParam1	GFSK_BLE_BR_1_600_BW_2_4	0x28	1.6	2.4
modParam1	GFSK_BLE_BR_1_000_BW_2_4	0x4C	1	2.4
modParam1	GFSK_BLE_BR_1_000_BW_1_2	0x45	1	1.2
modParam1	GFSK_BLE_BR_0_800_BW_2_4	0x70	0.8	2.4

Table 13-1: Modulation Parameters in GFSK Mode

Parameter	Symbol	Value	Bitrate (Mb/s)	Bandwidth (MHz DSB)
modParam1	GFSK_BLE_BR_0_800_BW_1_2	0x69	0.8	1.2
modParam1	GFSK_BLE_BR_0_500_BW_1_2	0x8D	0.5	1.2
modParam1	GFSK_BLE_BR_0_500_BW_0_6	0x86	0.5	0.6
modParam1	GFSK_BLE_BR_0_400_BW_1_2	0xB1	0.4	1.2
modParam1	GFSK_BLE_BR_0_400_BW_0_6	0xAA	0.4	0.6
modParam1	GFSK_BLE_BR_0_250_BW_0_6	0xCE	0.25	0.6
modParam1	GFSK_BLE_BR_0_250_BW_0_3	0xC7	0.25	0.3
modParam1	GFSK_BLE_BR_0_125_BW_0_3	0xEF	0.125	0.3

Table 13-2: Modulation Parameters in GFSK Mode

Parameter	Symbol	Value	Modindex
modParam2	MOD_IND_0_35	0x00	0.35
modParam2	MOD_IND_0_5	0x01	0.5
modParam2	MOD_IND_0_75	0x02	0.75
modParam2	MOD_IND_1_00	0x03	1
modParam2	MOD_IND_1_25	0x04	1.25
modParam2	MOD_IND_1_50	0x05	1.5
modParam2	MOD_IND_1_75	0x06	1.75
modParam2	MOD_IND_2_00	0x07	2
modParam2	MOD_IND_2_25	0x08	2.25
modParam2	MOD_IND_2_50	0x09	2.5
modParam2	MOD_IND_2_75	0x0A	2.75
modParam2	MOD_IND_3_00	0x0B	3
modParam2	MOD_IND_3_25	0x0C	3.25
modParam2	MOD_IND_3_50	0x0D	3.5
modParam2	MOD_IND_3_75	0x0E	3.75
modParam2	MOD_IND_4_00	0x0F	4

Table 13-3: Modulation Parameters in GFSK Mode

Parameter	Symbol	Value	BT
modParam3	BT_OFF	0x00	No filtering
modParam3	BT_1_0	0x10	1
modParam3	BT_0_5	0x20	0.5

6. Define the packet settings to be used by sending the command:

SetPacketParams(param[0], param[1], param[2], param[3], param[4], param[5], param[6])

- packetParam1 = PreambleLength
- packetParam2 = defines the number of bytes used for Sync Word (SyncWordLength).
- packetParam3 = defines the number of correlators to be used by SyncWordMatch
- packetParam4 = HeaderType
- packetParam5 = PayloadLength
- packetParam6 = CrcLength
- packetParam7 = Whitening

Table 13-4: Preamble Length Definition in GFSK Packet

Parameter	Symbol	Value	Preamble length in bits
packetParam1	PREAMBLE_LENGTH_04_BITS	0x00	4
packetParam1	PREAMBLE_LENGTH_08_BITS	0x10	8
packetParam1	PREAMBLE_LENGTH_12_BITS	0x20	12
packetParam1	PREAMBLE_LENGTH_16_BITS	0x30	16
packetParam1	PREAMBLE_LENGTH_20_BITS	0x40	20
packetParam1	PREAMBLE_LENGTH_24_BITS	0x50	24
packetParam1	PREAMBLE_LENGTH_28_BITS	0x60	28
packetParam1	PREAMBLE_LENGTH_32_BITS	0x70	32

The minimum preamble length when AGC is used should be 8 bits for a bit rate of 1 Mb/s. For other bit rates, the minimum number of preamble bits must be at least 16 bits.

Table 13-5: Sync Word Length Definition in GFSK Packet

Parameter	Symbol	Value	Sync Word size in bytes
packetParam2	SYNC_WORD_LEN_1_B	0x00	1
packetParam2	SYNC_WORD_LEN_2_B	0x02	2
packetParam2	SYNC_WORD_LEN_3_B	0x04	3
packetParam2	SYNC_WORD_LEN_4_B	0x06	4
packetParam2	SYNC_WORD_LEN_5_B	0x08	5

Thanks to its 3 correlators, the transceiver can search for several synchronization words at the same time:

Table 13-6: Sync Word Combination in GFSK Packet

Parameter	Symbol	Value	Sync Word combination to use
packetParam3	RADIO_RX_MATCH_SYNCWORD_OFF	0x00	Disable Sync Word
packetParam3	RADIO_RX_MATCH_SYNCWORD_1	0x10	SyncWord1
packetParam3	RADIO_RX_MATCH_SYNCWORD_2	0x20	SyncWord2
packetParam3	RADIO_RX_MATCH_SYNCWORD_1_2	0x30	SyncWord1 or SyncWord2
packetParam3	RADIO_RX_MATCH_SYNCWORD_3	0x40	SyncWord3
packetParam3	RADIO_RX_MATCH_SYNCWORD_1_3	0x50	SyncWord1 or SyncWord3
packetParam3	RADIO_RX_MATCH_SYNCWORD_2_3	0x60	SyncWord2 or SyncWord3
packetParam3	RADIO_RX_MATCH_SYNCWORD_1_2_3	0x70	SyncWord1, SyncWord2 or SyncWord3

Table 13-7: Packet Type Definition in GFSK Packet

Parameter	Symbol	Value	Packet Length mode
packetParam4	RADIO_PACKET_FIXED_LENGTH	0x00	FIXED LENGTH MODE
packetParam4	RADIO_PACKET_VARIABLE_LENGTH	0x20	VARIABLE LENGTH MODE

The payload length is defined by param[4] parameter. This parameter is used by the packet handler in Tx to send the exact number of bytes. In Rx variable length mode, the packet handler will filter-out all packets with size greater than Payloadlength.

Table 13-8: Payload Length Definition in GFSK Packet

Parameter	Symbol	Value	description
packetParam5	PAYLOAD_LENGTH	[0... 255]	Payload length in bytes

Using the GFSK packet, the CRC can be calculated on 1 or 2 bytes or ignored. This is defined using parameter param[5].

Table 13-9: CRC Definition in GFSK Packet

Parameter	Symbol	Value	CRC type
packetParam6	RADIO_CRC_OFF	0x00	No CRC
packetParam6	RADIO_CRC_1_BYTES	0x10	CRC field used 1 byte
packetParam6	RADIO_CRC_2_BYTES	0x20	CRC field uses 2 bytes

The whitening may be enabled in parameter param[6].

Table 13-10: Whitening Enabling in GFSK Packet

Parameter	Symbol	Value	Whitening mode
packetParam7	WHITENING_ENABLE	0x00	WHITENING ENABLE
packetParam7	WHITENING_DISABLE	0x08	WHITENING DISABLE

7. Define Sync Word value

Additionally the user may define the 32 bits of the synchronization word (SyncWord1, SyncWord2, SyncWord3). This is carried out by sending the *WriteRegister()* command, the next table gives the address for the Sync Word.

Table 13-11: Sync Word Definition in GFSK Packet

Sync Word	Bytes	Address
SyncWord1	SyncWord1(39:32)	0x09CE
	SyncWord1(31:24)	0x09CF
	SyncWord1(23:16)	0x09D0
	SyncWord1(15:8)	0x09D1
	SyncWord1(7:0)	0x09D2

Table 13-11: Sync Word Definition in GFSK Packet

Sync Word	Bytes	Address
SyncWord2	SyncWord2(39:32)	0x09D3
	SyncWord2(31:24)	0x09D4
	SyncWord2(23:16)	0x09D5
	SyncWord2(15:8)	0x09D6
	SyncWord2(7:0)	0x09D7
SyncWord3	SyncWord3(39:32)	0x09D8
	SyncWord3(31:24)	0x09D9
	SyncWord3(23:16)	0x09DA
	SyncWord3(15:8)	0x09DB
	SyncWord3(7:0)	0x09DC

The seed used for CRC needs also to be modified for certain applications. This is carried out by direct register access using the command *WriteReg()*.

Table 13-12: CRC Initialisation Registers

Parameter	Bytes	Address
CrcInit	CRC init value MSB	0x9c8
	CRC init value MSB	0x9c9

The CRC polynomial can also be modified by direct register access using the command *WriteReg()*.

Table 13-13: CRC Polynomial Definition

Parameter	Bytes	Address	Description
CrcPolynomial	CRC polynomial MSB	0x9C6	Defines the LSB byte of the 16-bit CRC polynomial or Defines the 8-bit CRC polynomial For example to program the following polynomial P16(x) = x16 + x12 + x5 + 1 Initialize the crc_polynomial(15:0) = 0x1021 To program the following polynomial P8(x) = x8 + x2 + x + 1 Initialize the crc_polynomial(7:0) = 0x07
	CRC polynomial LSB	0x9C7	

13.1.2 Tx Setting and Operations

1. Define output power and ramp time by sending the command:

SetTxParams(power,ramptime)

2. Send the payload to the data buffer by sending the command:

WriteBuffer(offset,*data)

where *data is a pointer to the payload and offset is the address at which the first byte of the payload will be located in the FIFO. The offset will correspond to txBaseAddress in normal operation.

3. Configure the DIOs and Interrupt sources (IRQs) by using command:

SetDioIrqParams(IrqMask,Dio1Mask,Dio2Mask,Dio3Mask)

In a typical Tx operation the user can select one or several IRQ sources:

- TxDone IRQ to indicate the end of packet transmission. The transceiver will be in STDBY_RC mode.
- RxTxTimeout (optional) to prevent deadlock. The transceiver will return automatically to STDBY_RC mode if a timeout occurs.

4. Once configured, set the transceiver in transmitter mode to start transmission using command:

SetTx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

If a timeout is desired, set the periodBaseCount to a non-zero value. This timeout can be used to avoid deadlock.

Wait for IRQ TxDone or RxTxTimeout

Once a packet has been sent, or a timeout has occurred, the transceiver goes automatically to STDBY_RC mode.

5. Optionally check the packet status to make sure that the packet has been sent properly, by using the command:

GetPacketStatus()

In this case only parameter packetStatus[3] is useful.

Table 13-14: PacketStatus[3] in GFSK Packet

PacketStatus[3]	Symbol	Description
bit 7:1	Reserved	Reserved
bit 0	PktSent	Indicates that the packet transmission is complete. Only signifies the completion of transmit process and not the packet validity. Only applicable in Tx.

6. Clear TxDone or RxTxTimeout IRQ by sending the command:

ClrIrqStatus(irqMask[15:8], irqMask[7:0])

This command will reset the flag for which the corresponding bit position in irqMask is set to 1.

13.1.3 Rx Setting and Operations

1. Configure the DIOs and Interrupt sources (IRQs) by using command

SetDioIrqParams(irqMask, dio1Mask, dio2Mask, dio3Mask)

In typical GFSK Rx operation one can select one or several IRQ sources:

- RxDone to indicate a packet has been detected. This IRQ does not mean that the packet is valid (size or CRC correct). The user must check the packet status to ensure that the valid packed is received.
- SyncWordValid to indicate that a Sync Word has been detected.
- CrcError to indicate that the received packet has a CRC error
- RxTxTimeout to indicate that no packet has been detected in a given time frame defined by timeout parameter in the SetRx() command.

Map these IRQs to one or more DIOs as desired.

2. Once configured, set the transceiver in receiver mode to start reception using command:

SetRx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

Depending on *periodBaseCount*, 3 possible Rx behaviours are possible:

- *periodBaseCount* is set to 0, then no Timeout, Rx Single mode, the device will stay in Rx mode until a reception occurs and the devices return in STDBY_RC mode upon completion
- *periodBaseCount* is set to 0xFFFF, Rx Continuous mode, the device remains in Rx mode until the host sends a command to change the operation mode. The device can receive several packets. Each time a packet is received, a packet received indication is given to the host and the device will continue to search for a new packet.
- *periodBaseCount* is set to another value, then Timeout is active. The device remains in Rx mode; it returns automatically to STDBY_RC Mode on timer end-of-count or when a packet has been received. As soon as a packet is detected, the timer is automatically disabled to allow complete reception of the packet.

3. In typical cases, use a timeout and wait for IRQs RxDone or RxTxTimeout.

   If IRQs RxDone rises, the transceiver goes to STDBY_RC mode if single mode is used (timeout set to a value different from 0xFFFF). If Continuous mode is used (timeout set to 0xFFFF) the transceiver stays in Rx and continues to listen for a new packet.

4. Check the packet status to make sure that the packet has been received properly, by using the command:

GetPacketStatus()

The command returns the following parameters:

- *RssiSync*: RSSI value at the time the Sync Word has been detected. Actual signal power is –RssiSync/2 (dBm)
- *packetStatus2*: Gives information about the last packet received as described in the next table
- *packetStatus3*: Used in Tx to indicate end of transmission
- *packetStatus4*: Indicates which correlator has detected the Sync Word

Table 13-15: PacketStatus[2] in GFSK Packet

PacketStatus[2]	Symbol	Description
bit 7	Reserved	Reserved
bit 6	SyncError	Sync address detection status for the current packet Only applicable in Rx when sync address detection is enabled.
bit 5	LengthError	Asserted when the length of the received packet is greater than the Max length defined in the PAYLOAD_LENGTH parameter. Only applicable in Rx for dynamic length packets.
bit 4	CrcError	CRC check status of the current packet. The packet is available anyway in the FIFO. Only applicable in Rx when the CRC check is enabled
bit 3	AbortError	Abort status indicates if the current packet in Rx/Tx was aborted. Applicable in Rx & Tx.
bit 2	HeaderReceived	Indicates if the header for the current packet was received. Only applicable in Rx for dynamic length packets
bit 1	PacketReceived	Indicates that the packet reception is complete. Does not signify packet validity. Only applicable in Rx.
bit 0	PacketCtrlBusy	Indicates that the packet controller is busy. Applicable both in Rx/Tx

Table 13-16: PacketStatus[4] in GFSK Mode Packet

PacketStatus[4]	Symbol	Description	Value
bit 7:3	Reserved	Reserved	
bit 2:0	SyncAdrsCode	Code of the sync address detected	000: sync address detection error 001: sync_adrs_1' detected 010: sync_adrs_2', detected 100: sync_adrs_3' detected

5. Once all checks are complete, then clear the IRQs by sending the command:

   ClrIrqStatus(irqMask)

   This command will reset the flag for which the corresponding bit position in irqMask is set to 1. Note a DIO can be mapped to several IRQ sources (ORed with IRQ sources). The DIO will go to zero once all corresponding IRQ flags have been set to zero.

6. Get packet length and start address of the received payload issuing the command:

   GetRxbufferStatus()

   This command returns the length of the last received packet (payloadLength) and the address of the first byte received (rxBufferOffset). It is applicable to all modems.

7. Read the data buffer using the command:

   ReadBuffer(offset, payloadLength)

Where offset is equal to rxBufferOffset and the length of payload to receive is payloadLength.

13.2 BLE Packet

13.2.1 Common Transceiver Settings

After power up or hard reset the transceiver runs a calibration procedure and goes to STDBY_RC mode indicated by a low state on BUSY pin. From this state the steps are:

1. If not in STDBY_RC mode, then go to this mode by using command:

SetStandby(STDBY_RC)

2. Define BLE packet by sending command:

SetPacketType(PACKET_BLE)

3. Define the RF frequency by sending command:

SetRfFrequency(rfFrequency)

The LSB of rfFrequency is equal to the PLL step i.e. 52 MHz / 2^18. *SetRfFrequency()* defines the Tx frequency.

In Rx the frequency is lowered by the intermediate frequency (IF). The IF is by default set to 1.3 MHz.

4. Indicate the addresses where the packet handler will read (txBaseAddress in Tx) or write (rxBaseAddress in Rx) the first byte of the data payload by sending the command:

SetBufferBaseAddress(txBaseAddress, rxBaseAddress)

5. Define the modulation parameter by sending command:

SetModulationParams(modParam1,modParam2,modParam3)

- param[0]: bit rate and bandwidth definition.
- param[1]: modulation index definition.
- param[2]: pulse shaping definition

In BLE case of different bit rates, modulation index and BT than the standard can be used with the packet.

Table 13-17: Modulation Parameters in BLE and GFSK Mode

Parameter	Symbol	Value	BR [Mb/s]	BW [MHz DSB]
modParam1	BLE_BR_2_000_BW_2_4	0x04	2	2.4
modParam1	BLE_BR_1_600_BW_2_4	0x28	1.6	2.4
modParam1	BLE_BR_1_000_BW_2_4	0x4C	1	2.4
modParam1	BLE_BR_1_000_BW_1_2	0x45	1	1.2
modParam1	BLE_BR_0_800_BW_2_4	0x70	0.8	2.4
modParam1	BLE_BR_0_800_BW_1_2	0x69	0.8	1.2
modParam1	BLE_BR_0_500_BW_1_2	0x8D	0.5	1.2
modParam1	BLE_BR_0_500_BW_0_6	0x86	0.5	0.6
modParam1	BLE_BR_0_400_BW_1_2	0xB1	0.4	1.2

Table 13-17: Modulation Parameters in BLE and GFSK Mode

Parameter	Symbol	Value	BR [Mb/s]	BW [MHz DSB]
modParam1	BLE_BR_0_400_BW_0_6	0xAA	0.4	0.6
modParam1	BLE_BR_0_250_BW_0_6	0xCE	0.25	0.6
modParam1	BLE_BR_0_250_BW_0_3	0xC7	0.25	0.3
modParam1	BLE_BR_0_125_BW_0_3	0xEF	0.125	0.3

Table 13-18: Modulation Parameters in BLE and GFSK Mode

Parameter	Symbol	Value	Modindex
modParam2	MOD_IND_0_35	0x00	0.35
modParam2	MOD_IND_0_5	0x01	0.5
modParam2	MOD_IND_0_75	0x02	0.75
modParam2	MOD_IND_1	0x03	1
modParam2	MOD_IND_1_25	0x04	1.25
modParam2	MOD_IND_1_5	0x05	1.5
modParam2	MOD_IND_1_75	0x06	1.75
modParam2	MOD_IND_2	0x07	2
modParam2	MOD_IND_2_25	0x08	2.25
modParam2	MOD_IND_2_5	0x09	2.5
modParam2	MOD_IND_2_75	0x0A	2.75
modParam2	MOD_IND_3	0x0B	3
modParam2	MOD_IND_3_25	0x0C	3.25
modParam2	MOD_IND_3_5	0x0D	3.5
modParam2	MOD_IND_3_75	0x0E	3.75
modParam2	MOD_IND_4	0x0F	4

Table 13-19: Modulation Parameters in BLE and GFSK Mode

Parameter	Symbol	Value	BT
modParam3	BT_DIS	0x00	No filtering
modParam3	BT_1	0x10	1
modParam3	BT_0_5	0x20	0.5

6. Define the packet format to be used by sending the command:

SetPacketParams(packetParam[0],packetParam[1],packetParam[2],packetParam[3])

- packetParam1 = ConnectionState
- packetParam2 = CrcLength
- packetParam3 = BleTestPayload
- packetParam4 = Whitening

Note that although this command can accept up to 7 arguments, in BLE mode *SetPacketParams* can accept only 4.

Table 13-20: Connection State Definition in BLE Packet

Parameter	Symbol	Value	BLE State
packetParam1	BLE_MASTER_SLAVE	0x00	Master slave
packetParam1	BLE_ADVERTISER	0x02	Advertiser
packetParam1	BLE_Tx_TEST_MODE	0x04	Test Tx mode
packetParam1	BLE_RX_TEST_MODE	0x06	Test Rx mode
packetParam1	BLE_RXTx_TEST_MDOE	0x08	Dedicated Rx Tx test mode

Table 13-21: CRC Definition in BLE Packet

Parameter	Symbol	Value	Packet Length Mode
packetParam2	BLE_CRC_OFF	0x00	No CRC
packetParam2	BLE_CRC_3B	0x10	CRC field used 3bytes

Table 13-22: Tx Test Packet Payload in Test Mode for BLE Packet

Parameter	Symbol	Value	Payload Content
packetParam3	BLE_PRBS_9	0x00	Pseudo Random Binary Sequence based on 9th degree polynomial P7(x) = x9+ x5+ 1 PRBS9 sequence '111111111000001111 0....' (in transmission order)
packetParam3	BLE_EYELONG_1_0	0x04	Repeated '11110000' (in transmission order) sequence
packetParam3	BLE_ EYESHORT_1_0	0x08	Repeated '10101010' (in transmission order) sequence
packetParam3	BLE_PRBS_15	0x0C	Pseudo Random Binary Sequence based on 15th degree polynomial P15(x) = x15+ x14+ x13 + x12+ x2+ x + 1
packetParam3	BLE_ ALL_1	0x10	Repeated '11111111' (in transmission order) sequence
packetParam3	BLE_ ALL_0	0x14	Repeated '11111111' (in transmission order) sequence
packetParam3	BLE_ EYELONG_0_1	0x18	Repeated '00001111' (in transmission order) sequence
packetParam3	BLE_ EYESHORT_0_1	0x1C	Repeated '01010101' (in transmission order) sequence

Table 13-23: Whitening Enabling in BLE Packet

Parameter	Symbol	Value	Whitening Mode
packetParam4	BLE_WHITENING_ENABLE	0x00	WHITENING ENABLE
packetParam4	BLE_WHITENING_DISABLE	0x08	WHITENING DISABLE

7. Define Sync Word value

In addition to these parameters, the user needs to define the 32-bit synchronization word SyncWord1. This is carried out by sending the *WriteRegister()* command, the next table gives the address for the Sync Word.

Table 13-24: Sync Word Definition in BLE Packet

Sync Word	Bytes	Address
SyncWord1	SyncWord1(31:24)	0x09CF
	SyncWord1(23:16)	0x09D0
	SyncWord1(15:8)	0x09D1
	SyncWord1(7:0)	0x09D2

The seed used for CRC needs also to be modified for certain applications. This is carried out by direct register access by sending the function *WriteRegister()*.

Table 13-25: CRC Initialisation Registers

Parameter	Bytes	Address
CrcInit	CRC init value	0x9C7 (MSB)
		0x9C8
		0x9C9 (LSB)

13.2.2 Tx Setting and Operations

1. Define the output power and ramp time by sending the command:

SetTxParam(power,ramptime)

2. Contrarily to other modems, the payload to be written in BLE mode in the data buffer of the SX1280 chip must contain a BLE header. The BLE header to add at the beginning of the payload must correspond to the BLE mode selected at step 6 in Section 13.2.1 "Common Transceiver Settings" on page 94 . See Figure 7-5: PDU Header Format for the header definition.

Send the payload to the data buffer by issueing the command:

WriteBuffer(offset,data)

where data is the payload containing the BLE header to be sent and offset is the address at which the first byte of the payload will be located in the FIFO.

3. Configure the DIOs and Interrupt sources (IRQs) by using command:

SetDioIrqParams(irqMask, dio1Mask, dio2Mask, dio3Mask)

In typical Tx operation one can select one or several IRQ sources:

- TxDone IRQ to indicate the end of packet transmission. The transceiver will be in STDBY_RC mode.
- RxTxTimeout (optional) to make sure no deadlock can happen. The transceiver will return automatically to STDBY_RC mode if a timeout occurs.

4. Once configured, set the transceiver in transmitter mode to start transmission using command:

SetTx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

If a timeout is desired, set *periodBaseCount* to a value different from zero. This timeout can be used to avoid deadlock.

Wait for IRQ TxDone or RxTxTimeout

Once a packet has been sent or a timeout occurred, the transceiver goes automatically to STDBY_RC mode

5. Optionally check the packet status to make sure that the packet has been sent properly. by issuing the command:

GetPacketStatus()

In this case only parameter *packetStatus3* is useful.

Table 13-26: PacketStatus3 in BLE Packet

PacketStatus3	Symbol	Description
bit 7:1	reserved	Reserved
bit 0	PktSent	Indicates that the packet transmission is complete. Does not signifiy packet validity. Only applicable in Tx.

6. Clear TxDone or RxTxTimeout IRQ by sending the command:

ClrIrqStatus(irqMask)

This command will reset the flag for which the corresponding bit position in *irqMask* is set to 1.

13.2.3 Rx Setting and Operations

1. Configure the DIOs and Interrupt sources (IRQs) by using command:

SetDioIrqParams(irqMask, dio1Mask, dio2Mask, dio3Mask)

In typical BLE Rx operation one can select one or several IRQ sources

- RxDone to indicate a packet has been detected. This IRQ does not mean that the packet is valid (size or CRC correct). The user must check the packet status to ensure that the valid packed is received.
- SyncWordValid to indicate that a Sync Word has been detected.
- CrcError to indicate that the received packet has a CRC error
- RxTxTimeout to indicate that no packet has been detected in a given time packet defined by the timeout parameter in the SetRx() command.

Map these IRQs to one DIO (DIO1 or DIO2 or DIO3).

2. Once configured, set the transceiver in receiver mode to start reception using command:

SetRx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

Depending on *periodBaseCount*, 3 possible Rx behaviors are possible:

- *periodBaseCount* is set to 0, then no Timeout, Rx Single mode, the device will stay in Rx mode until a reception occurs and the devices return in STDBY_RC mode upon completion
- *periodBaseCount* is set to 0xFFFF, Rx Continuous mode, the device remains in Rx mode until the host sends a command to change the operation mode. The device can receive several packets. Each time a packet is received, a packet received indication is given to the host and the device will continue to search for a new packet.
- *periodBaseCount* is set to another value, then Timeout is active. The transceiver remains in Rx mode; it returns automatically to STDBY_RC Mode on timer end-of-count or when a packet has been received. As soon as a packet is detected, the timer is automatically disabled to allow complete reception of the packet.

3. In typical cases, use a timeout and wait for IRQ RxDone or RxTxTimeout.

If IRQ RxDone is asserted, the transceiver goes to STDBY_RC mode if single mode is used (timeout set to a value different from 0xFFFF). If Continuous mode is used (timeout set to 0xFFFF) the transceiver stays in Rx and continues to listen for a new packet.

4. Check the packet status to make sure that the packet has been correctly received by using the command:

GetPacketStatus()

The command returns the following parameters:

- *RssiSync*: RSSI value at the time the Sync Word has been detected;
- *packetStatus2*: Gives information about the last packet received as described in the next table;
- *packetStatus3*: In BLE packet, this status indicates in Tx mode if a packet has been sent or not;

Table 13-27: PacketStatus2 in BLE Mode

PacketStatus2	Symbol	Description
bit 7	Reserved	Reserved
bit 6	SyncError	Sync address detection status for the current packet Only applicable in Rx when sync address detection is enabled.

Table 13-27: PacketStatus2 in BLE Mode

PacketStatus2	Symbol	Description
bit 5	lengthError	Asserted when the length of the received packet is greater than the Max length defined in the PAYLOAD_LENGTH parameter. Only applicable in Rx for dynamic length packets.
bit 4	CrcError	CRC check status of the current packet. The packet is available anyway in the FIFO. Only applicable in Rx when the CRC is enabled
bit 3	AbortError	Abort status indicates if the current packet in Rx/Tx was aborted. Applicable both in Rx & Tx.
bit 2	HeaderReceived	Indicates if the header for the current packet was received. Only applicable in Rx for dynamic length packets
bit 1	PacketReceived	Indicates that the packet reception is complete. Does not signify packet validity. Only applicable in Rx.
bit 0	PacketCtrlBusy	Indicates that the packet controller is busy. Applicable both in Rx/Tx

- *packetStatus4*: Indicates which correlator has detected the Sync Word. In case of BLE, only sync_adrs_1 is used.

Table 13-28: PacketStatus4 in BLE Mode

PacketStatus4	Symbol	Description
bit 7:3	Reserved	Reserved
bit 2:0	SyncSdrsCode	Code of the sync address detected 0x0: sync address detection error 0x1: sync_adrs_1' detected others: reserved

5. Once all checks are complete, clear IRQs by sending the command:

ClrIrqStatus(irqMask)

This command will reset the flag for which the corresponding bit position in *irqMask* is set to 1. Note a a DIO can be mapped to several IRQ sources (ORed with IRQ sources). The DIO will go to zero once IRQ flag has been set to zero.

6. Get packet length and start address of the received payload issuing the command:

GetRxbufferStatus()

This command returns the length of the last received packet (*payloadLength*) and the address of the first byte received (*rxBufferOffset*) It is applicable to all modems. The address is an offset relative to the first byte of the data buffer.

7. Read the data buffer using the command:

ReadBuffer(offset, payloadLength)

Where offset is equal to *rxBufferOffset*.

13.2.4 BLE Specific Functions

13.2.4.1 SetAutoTx()

One additional command is available to ease the implementation of the BLE packet. BLE requires that the transceiver is able to send back a response 125 µs after a packet reception. This is carried out by sending the command *SetAutoTx()* that allows the transceiver to send a packet after a user programmable time (time) after the end of a packet reception.

SetAutoTx(time) must be issued in STDBY_RC mode.

Table 13-29: SetAutoTx Mode

Byte	0	1	2
Data from host	Opcode = 0x98	time(15:8)	time(7:0)

time is in µs. The delay between the end of reception of a packet and the start of the transmission of the next packet is defined by:

$$TxDelay = time - offset$$

Where *offset* is a time needed for the transceiver to switch modes and is equal to 33 µs.

Once this command is issued, each time the transceiver receives a packet, it will automatically switch to Tx and transmit a packet after a predefined time.

If the user wants to have normal operation (going in STDBY_RC after Tx) he needs to send the command *SetAutoTx()* with the time parameter set to zero.

13.3 FLRC Packet

13.3.1 Common Transceiver Settings

After power up or hard reset the transceiver runs a calibration procedure and goes to STDBY_RC mode indicated by a low state on BUSY pin. From this state the steps are

1. If not in STDBY_RC mode, then go to this mode by sending the command:

SetStandby(STDBY_RC)

2. Define the GFSK packet type by sending the command:

SetPacketType(PACKET_FLRC)

3. Define the RF frequency by sending the command:

SetRfFrequency(rfFrequency)

The LSB of rfFrequency is equal to the PLL step i.e. 52e6/2^18 Hz. *SetRfFrequency()* defines the Tx frequency.

In Rx the frequency is lowered by the intermediate frequency (IF). The IF is by default set 1.3 MHz.

4. Indicate the addresses where the packet handler will read (*txBaseAddress* in Tx) or write (*rxBaseAddress* in Rx) the first byte of the data payload by sending the command:

SetBufferBaseAddress(txBaseAddress, rxBaseAddress)

Note that *txBaseAddress* and *rxBaseAddress* are offsets from the beginning of the data memory map.

5. Define the modulation parameter by sending command:

SetModulationParams(modParam1, modParam2, modParam3)

The bit rate and bandwidth are linked via param[0]. The coding rate used in error correction mechanism is defined in param[1] and the BT is defined in param[2].

Table 13-30: Modulation Parameters in FLRC Mode: Bandwidth and Bit Rate

Parameter	Symbol	Value	Bit Rate [Mb/s]	Bandwidth [MHz DSB]
modParam1	FLRC_BR_1_300_BW_1_2	0x45	1.3	1.2
modParam1	FLRC_BR_1_000_BW_1_2	0x69	1.04	1.2
modParam1	FLRC_BR_0_650_BW_0_6	0x86	0.65	0.6
modParam1	FLRC_BR_0_520_BW_0_6	0xAA	052	0.6
modParam1	FLRC_BR_0_325_BW_0_3	0xC7	0.325	0.3
modParam1	FLRC_BR_0_260_BW_0_3	0xEB	0.26	0.3

Table 13-31: Modulation Parameters in FLRC Mode: Coding Rate

Parameter	Symbol	Value	Coding rate
modParam2	FLRC_CR_1_2	0x00	½
modParam2	FLRC_CR_3_4	0x02	¾
modParam2	FLRC_CR_1_0	0x04	1
modParam2	Reserved	Greater or equal to 3	Reserved

Table 13-32: Modulation Parameters in FLRC Mode: BT

Parameter	Symbol	Value	BT
modParam3	BT_DIS	0x00	No filtering
modParam3	BT_1	0x10	1
modParam3	BT_0_5	0x20	0.5

6. Define the packet format to be used by sending the command:

SetPacketParams(packetParam1, packetParam2, packetParam3, packetParam4, packetParam5, packetParam6, packetParam7)

- packetParam1 = AGCPreambleLength
- packetParam2 = SyncWordLength
- packetParam3 = SyncWordMatch
- packetParam4 = PacketType
- packetParam5 = PayloadLength
- packetParam6 = CrcLength
- packetParam7 = Whitening

Table 13-33: AGC Preamble Length Definition in FLRC Packet

Parameter	Symbol	Value	Preamble length in bits
packetParam1	PREAMBLE_LENGTH_4_BITS	0x00	Reserved
packetParam1	PREAMBLE_LENGTH_8_BITS	0x10	8
packetParam1	PREAMBLE_LENGTH_12_BITS	0x20	12
packetParam1	PREAMBLE_LENGTH_16_BITS	0x30	16
packetParam1	PREAMBLE_LENGTH_20_BITS	0x40	20

Table 13-33: AGC Preamble Length Definition in FLRC Packet

Parameter	Symbol	Value	Preamble length in bits
packetParam1	PREAMBLE_LENGTH_24_BITS	0x50	24
packetParam1	PREAMBLE_LENGTH_28_BITS	0x60	28
packetParam1	PREAMBLE_LENGTH_32_BITS	0x70	32

The minimum preamble length when AGC is used should be 8 bits for bit rate of 1 Mb/s. For other bit rates, the minimum number of preamble bits must be at least 16 bits.

The number of bytes used for Sync Word is defined by packetParam2. The user can rely on the built-in 21-bit preamble always required to detect start of packet or add 4 additional Sync Word for address detection in case of multiple devices.

Table 13-34: Sync Word Length Definition in FLRC Packet

Parameter	Symbol	Value	Sync Word size in bytes
packetParam2	FLRC_SYNC_NOSYNC	0x00	21 bits preamble
packetParam2	FLRC_SYNC_WORD_LEN_P32S	0x04	21 bits preamble + 32 bits Sync Word

With 3 correlators, the transceiver can search for several Sync Word at the time. The combination of Sync Word detection is defined by parameters *PacketParam3*.

Table 13-35: Sync Word Combination in FLRC Packet

Parameter	Symbol	Value	Sync Word combination to use
packetParam3	RX_DISABLE_SYNC_WORD	0x00	Disable Sync Word
packetParam3	RX_MATCH_SYNC_WORD_1	0x10	SyncWord1
packetParam3	RX_MATCH_SYNC_WORD_2	0x20	SyncWord2
packetParam3	RX_MATCH_SYNC_WORD_1_2	0x30	SyncWord1 or SyncWord2
packetParam3	RX_MATCH_SYNC_WORD_3	0x40	SyncWord3
packetParam3	RX_MATCH_SYNC_WORD_1_3	0x50	SyncWord1 or SyncWord3
packetParam3	RX_MATCH_SYNC_WORD_2_3	0x60	SyncWord2 or SyncWord3
packetParam3	RX_MATCH_SYNC_WORD_1_2_3	0x70	SyncWord1 or SyncWord2 or SyncWord3

The payload length is defined by packetParam4 parameter. This parameter is used by the packet handler in Tx to send the exact number of bytes. In Rx, in variable length mode, the packet handler will filter-out all packets with size greater than Payload length. Note that the minimum payload length is 6 bytes.

Table 13-36: Packet Type Definition in FLRC Packet

Parameter	Symbol	Value	Packet Length mode
packetParam4	PACKET_FIXED_LENGTH	0x00	FIXED LENGTH MODE
packetParam4	PACKET_VARIABLE_LENGTH	0x20	VARIABLE LENGTH MODE

Table 13-37: Payload Length Definition in FLRC Packet

Parameter	Symbol	Value	Description
packetParam5	PAYLOAD_LENGTH	[6 ... 127]	Payload length in bytes

In FLRC mode, the CRC can be calculated on 2, 3 or 4 bytes or ignored. This is defined using parameter *param[5]*.

Table 13-38: CRC Definition in FLRC Packet

Parameter	Symbol	Value	CRC type
packetParam6	CRC_OFF	0x00	No CRC
packetParam6	CRC_1_BYTE	0x10	CRC field used 1 byte
packetParam6	CRC_2_BYTE	0x20	CRC field uses 2 bytes
packetParam6	CRC_3_BYTE	0x30	CRC field uses 3 bytes

The seed used for CRC needs also to be modified for certain applications. This is carried out by direct register access by sending the function *WriteRegister()*.

Table 13-39: CRC Initialisation Registers

Parameter	Bytes	Address
CrcInit	CRC init value MSB	0x9c8
	CRC init value LSB	0x9c9

The CRC polynomial can also be modified by direct register access using the command *WriteRegister()*.

Table 13-40: CRC Polynomial Definition

Parameter	Bytes	Address	Description
CrcPolynomial	CRC polynomial MSB	0x9C6	Defines the LSB byte of the 16-bit CRC polynomial or Defines the 8-bit CRC polynomial For example to program the following polynomial P16(x) = x16 + x12 + x5 + 1 Initialize the crc_polynomial[15:0] = 0x1021 To program the following polynomial P8(x) = x8 + x2 + x + 1 Initialize the crc_polynomial[7:0] = 0x07
	CRC polynomial LSB	0x9C7	

In FLRC packet type, it is not possible to enable whitening. You must always set the value of packetParam7 to *disabled*.

Table 13-41: Whitening Definition in FLRC Packet

Parameter	Symbol	Value	Description
packetParam7	WHITENING	0x08	Whitening disabled

7. Define Sync Word value

In addition to these parameters, the user needs to define the synchronization word (*SyncWord1*, *SyncWord2*, *SyncWord3*). This is carried out by sending the *WriteRegister()* command. The table below gives the address for the Sync Word..

Table 13-42: Sync Word Definition in FLRC Packet

Sync Word	Bytes	Address
SyncWord1	SyncWord1[31:24]	0x09CF
	SyncWord1[23:16]	0x09D0
	SyncWord1[15:8]	0x09D1
	SyncWord1[7:0]	0x09D2
SyncWord2	SyncWord2[31:24]	0x09D4
	SyncWord2[23:16]	0x09D5
	SyncWord2[15:8]	0x09D6
	SyncWord2[7:0]	0x09D7
SyncWord3	SyncWord3[31:24]	0x09D9
	SyncWord3[23:16]	0x09DA
	SyncWord3[15:8]	0x09DB
	SyncWord3[7:0]	0x09DC

13.3.2 Tx Setting and Operations

1. Define output power and ramp time by sending the command:

SetTxParam(power,rampTime)

2. Send the payload to the data buffer by sending the command:

WriteBuffer(offset,data)

where *data* is the payload to be sent and *offset* is the address at which the first byte of the payload will be located in the buffer. Offset will correspond to *txBaseAddress* in normal operation.

3. Configure the DIOs and Interrupt sources (IRQs) by sending the command:

SetDioIrqParams(irqMask,dio1Mask,dio2Mask,dio3Mask)

In a typical Tx operation one or several IRQ sources may be selected:

- TxDone IRQ to indicate the end of packet transmission. The transceiver will be in STDBY_RC mode.
- RxTxTimeout (optional) to make sure no deadlock can happen. The transceiver will return automatically to STDBY_RC mode if a timeout occurs.

4. Once configured, set the transceiver in transmitter mode to start transmission using command:

SetTx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

If a timeout is desired, set *periodBaseCount* value different to zero. This timeout can be used to avoid deadlock.

Wait for IRQ TxDone or RxTxTimeout. Once a packet has been sent or a timeout occurs, the transceiver goes automatically to STDBY_RC mode.

5. Optionally check the packet status to make sure that the packet has been sent properly by using the command:

GetPacketStatus()

In this case only the parameter *packetStatus3* is useful.

Table 13-43: PacketStatus3 in FLRC Packet

PacketStatus3	Symbol	Description
bit 7:6	rxpid	PID field of the received packet Only applicable in Rx for dynamic length packets
bit 5	rx_no_ack	NO_ACK field of the received packet Only applicable in Rx for dynamic length packets.
bit 4	rxpiderr	PID check status for the current packet rxpid(N) = rxpid(N-1) and crc_ checksum(N) =crc_checksum(N-1) Only applicable in Rx for dynamic length packets when rxpid_filter_enable = '1'
bit 3:1	reserved	reserved
bit 0	PktSent	Indicates that the packet transmission is complete. Only signifies the completion of transmit process and not the packet validity. Only applicable in Tx.

6. Clear TxDone or RxTxTimeout IRQ by sending the command:

ClrIrqStatus(irqMask)

This command will reset the flag for which the corresponding bit position in *irqMask* is set to 1.

13.3.3 Rx Setting and Operations

1. Configure the DIOs and Interrupt sources (IRQs) by sending the command:

SetDioIrqParams(IrqMask,Dio1Mask,Dio2Mask,Dio3Mask)

In a typical FLRC Rx operation one or several IRQ sources may be selected:

- RxDone to indicate a packet has been detected. This IRQ does not mean that the packet is valid (size or CRC correct). The user must check the packet status to ensure that the valid packed is received.
- SyncWordValid to indicate that a Sync Word has been detected.
- CrcError to indicate that the received packet has a CRC error
- RxTxTimeout to indicate that no packet has been detected in a given time frame defined by timeout parameter in the *SetRx()* command.

Map these IRQs to one DIO (DIO1 or DIO2 or DIO3).

2. Once configured, set the transceiver in receiver mode to start reception using command:

SetRx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

Depending on *periodBaseCount*, 3 possible Rx behaviour are possible:

- *periodBaseCount* is set to 0, then no timeout, Rx Single mode, the device will stay in Rx mode until a reception occurs and the devices return in STDBY_RC mode upon completion.
- *periodBaseCount* is set 0xFFFF, Rx Continuous mode, the device remains in Rx mode until the host sends a command to change the operation mode. The device can receive several packets. Each time a packet is received, a packet reception indication is given to the host and the device will continue to search for a new packet.
- *periodBaseCount* is set to another value, then Timeout is active. The device remains in Rx mode; it returns automatically to STDBY_RC Mode on timer end-of-count or when a packet has been received. As soon as a packet is detected, the timer is automatically disabled to allow complete reception of the packet.

3. Typically, use a timeout and wait for IRQ RxDone or RxTxTimeout.

If IRQ RxDone rises, the transceiver goes to STDBY_RC mode if single mode is used (timeout set to a value different from 0xFFF). If Continuous mode is used (timeout set to 0xFFFF) the transceiver stays in Rx and continue to listen for a new packet.

4. Check the packet status to make sure that the packet has been received properly, by sending the command:

GetPacketStatus()

The command returns the following parameters:

- *RssiSync*: RSSI value at the time the Sync Word has been detected. Actual signal power is –RssiSync/2 (dBm)
- *packetStatus2*: Gives information about the last packet received as described in the next table
- *packetStatus3*: In FLRC packet, this status indicates in Tx mode if a packet has been sent or not
- *packetStatus4*: Indicates which correlator has detected the Sync Word

Table 13-44: PacketStatus2 in FLRC Packet

PacketStatus2	Symbol	Description
bit 7	Reserved	Reserved
bit 6	SyncError	Sync address detection status for the current packet Only applicable in Rx when sync address detection is enabled.
bit 5	LengthError	Asserted when the length of the received packet is greater than the Max length defined in the PAYLOAD_LENGTH parameter. Only applicable in Rx for dynamic length packets.
bit 4	CrcError	CRC check status of the current packet. The packet is available anyway in the FIFO. Only applicable in Rx when the CRC is enabled
bit 3	AbortError	Abort status indicates if the current packet in Rx/Tx was aborted. Applicable both in Rx & Tx.
bit 2	HeaderReceived	Indicates if the header for the current packet was received. Only applicable in Rx for dynamic length packets
bit 1	PacketReceived	Indicates that the packet reception is complete. Does not signify packet validity. Only applicable in Rx.
bit 0	PacketCtrlBusy	Indicates that the packet controller is busy. Applicable both in Rx/Tx

Table 13-45: PacketStatus3 in FLRC Packet

PacketStatus3	Symbol	Description
bit 7:6	rxpid	PID field of the received packet Only applicable in Rx for dynamic length packets
bit 5	rx_no_ack	NO_ACK field of the received packet Only applicable in Rx for dynamic length packets.
bit 4	rxpiderr	PID check status for the current packet rxpid(N) = rxpid(N-1) and crc_ checksum(N) =crc_checksum(N-1) Only applicable in Rx for dynamic length packets when rxpid_filter_enable = '1'
bit 3:1	Reserved	Reserved
bit 0	PktSent	Indicates that the packet transmission is complete. Does not signify packet validity. Only applicable in Tx.

Table 13-46: PacketStatus4 in FLRC Packet

PacketStatus4	Symbol	Description
bit 7:3	Reserved	Reserved
bit 2:0	SyncSdrsCode	Code of the sync address detected 000: sync address detection error 001: sync_adrs_1' detected 010: sync_adrs_2', detected 100: sync_adrs_3' detected

5. Once all checks are complete, clear the IRQs by sending the command:

ClrIrqStatus(irqMask)

This command will reset the flag for which the corresponding bit position in *irqMask* is set to 1. Note a a DIO can be mapped to several IRQ sources (ORed with IRQ sources). The DIO will go to zero once IRQ flag has been set to zero.

6. Get the packet length and the start address of the received payload by sending the command:

GetRxBufferStatus()

This command returns the length of the last received packet (*payloadLength*) and the address of the first byte received (*rxBufferOffset*) It is applicable to all modems. The address is an offset relative to the first byte of the data buffer.

7. Read the data buffer using the command:

ReadBuffer(offset, payloadLength)

Where offset is equal to *rxBufferOffset* and the command contains *payloadLength*.

13.4 LoRa Packet

13.4.1 Common Transceiver Settings for LoRa

After power up or hard reset the transceiver runs a calibration procedure and goes to STDBY_RC mode indicated by a low state on BUSY pin. From this state the steps are

1. If not in STDBY_RC mode, then go to this mode by sending the command:

SetStandby(STDBY_RC)

2. Define the LoRa packet type by sending the command:

SetPacketType(PACKET_LORA)

3. Define the RF frequency by sending the command:

SetRfFrequency(rfFrequency)

The LSB of *rfFrequency* is equal to the PLL step i.e. 52e6/2^18 Hz. *SetRfFrequency()* defines the Tx frequency.

In Rx the frequency is lowered by the intermediate frequency (IF). The IF is by default set 1.65 MHz.

4. Indicate the addresses where the packet handler will read (*txBaseAddress* in Tx) or write (*rxBaseAddress* in Rx) the first byte of the data payload by sending the command:

SetBufferBaseAddress(txBaseAddress, rxBaseAddress)

Note that *txBaseAddress* and *rxBaseAddress* are offset relative to the beginning of the data memory map.

5. Define the modulation parameter by sending the command:

SetModulationParams(modParam1,modParam2, modParam3)

modParam1 defines the signal BW, *modeParam2* defines SF and *modParam3* defines the coding rate (CR).

Table 13-47: Modulation Parameters in LoRa Mode

Parameter	Symbol	Value	Spreading factor
modParam1	LORA_SF_5	0x50	5
modParam1	LORA_SF_6	0x60	6
modParam1	LORA_SF_7	0x70	7
modParam1	LORA_SF_8	0x80	8
modParam1	LORA_SF_9	0x90	9
modParam1	LORA_SF_10	0xA0	10
modParam1	LORA_SF_11	0xB0	11
modParam1	LORA_SF_12	0xC0	12

Note, after *SetModulationParams* command:

- If the Spreading Factor selected is SF5 or SF6, it is required to use *WriteRegister(0x925, 0x1E)*
- If the Spreading Factor is SF7 or SF-8 then the command *WriteRegister(0x925, 0x37)* must be used
- If the Spreading Factor is SF9, SF10, SF11 or SF12, then the command *WriteRegister(0x925, 0x32)* must be used

Table 13-48: Modulation Parameters in LoRa Mode

Parameter	Symbol	Value	Bandwidth [kHz]
modParam2	LORA_BW_1600	0x0A	1625.0
modParam2	LORA_BW_800	0x18	812.5
modParam2	LORA_BW_400	0x26	406.25
modParam2	LORA_BW_200	0x34	203.125

Table 13-49: Modulation Parameters in LoRa Mode

Parameter	Symbol	Value	Coding rate
modParam3	LORA_CR_4_5	0x01	4/5
modParam3	LORA_CR_4_6	0x02	4/6
modParam3	LORA_CR_4_7	0x03	4/7
modParam3	LORA_CR_4_8	0x04	4/8
modParam3	LORA_CR_LI_4_5	0x05	4/5*
modParam3	LORA_CR_LI_4_6	0x06	4/6*
modParam3	LORA_CR_LI_4_7	0x07	4/8*

* A new interleaving scheme has been implemented to increase robustness to burst interference and/or strong doppler events. The FEC has been kept the same to limit the impact on complexity. Long interleaving re-uses the memory used during detection. This memory is used for both encoding and decoding when long interleaving is set.

Long interleaving is selected by setting *modParam3* = 0x5, 0x6 or 0x7(LORA_CR_LI_4_5,LORA_CR_LI_4_6 or LORA_CR_LI_4_7). Coding rate is signaled in header. Previously, only values 0x0 to 0x4 were valid.

The coding rate values respectively 4/5, 4/6, 4/8. So LORA_CR_LI_4_5 is like LORA_CR_4_5 , LORA_CR_LI_4_6 like LORA_CR_4_6, LORA_CR_LI_4_7 like LORA_CR_4_8 with a different interleaving scheme.

Long interleaving is compatible with implicit header. Scrambling occurs with long interleaving, as with legacy interleaving.

Note: there is a limitation on maximum payload length for LORA_CR_LI_4_7. Payload length should not exceed 253 bytes if CRC is enabled.

6. Define the packet format to be used by sending the command:

SetPacketParams(pktParam1, pktParam2, pktParam3, pktParam4, pktParam5)

- *packetParam1* = PreambleLength, which defines the preamble length (in symbols) to be used mainly by the packet handling in Tx mode.
- *packetParam2* = HeaderType
- *packetParam3* = PayloadLength
- *packetParam4* = CRC

- *packetParam5* = InvertIQ/chirp invert
- *packetParam1* defines the preamble length number expressed in LoRa symbols. Recommended value is 12 symbols.

Table 13-50: Preamble Definition in LoRa or Ranging

Parameter	Symbol	Value	Preamble length in symbols
packetParam1(3:0)	LORA_PBLE_LEN_MANT	[1:15]	preamble length = LORA_PBLE_LEN_MANT*2^(LORA_PBLE_LEN_EXP)
packetParam1(7:4)	LORA_PBLE_LEN_EXP	[1:15]	

The type of packet is defined by parameter PacketParam2. For fixed-length packet, no header is visible and the implicit header is used. In variable length packet, the explicit header mode is used.

Table 13-51: Packet Type Definition in LoRa or Ranging Packet

Parameter	Symbol	Value	Header mode
packetParam2	EXPLICIT_HEADER	0x00	EXPLICIT HEADER
packetParam2	IMPLICIT_HEADER	0x80	IMPLICIT HEADER

The payload length is defined in *packetParam3*.

Table 13-52: Payload Length Definition in LoRa Packet

Parameter	Symbol	Value	PayloadLength
packetParam3	PayloadLength	[1….255]	PayloadLength

Note : there is a limitation on maximum payload length for LORA_CR_LI_4_7. Payload length should not exceed 253 bytes if CRC is enabled.

The CRC usage is defined in *packetParam4*.

Table 13-53: CRC Enabling in LoRa Packet

Parameter	Symbol	Value	CRC mode
packetParam4	LORA_CRC_ENABLE	0x20	CRC ENABLE
packetParam4	LORA_CRC_DISABLE	0x00	CRC DISABLE

The IQ swapping is defined by *PacketParam5*.

Table 13-54: IQ Swapping in LoRa or Ranging Packet

Parameter	Symbol	Value	LoRa IQ swap
packetParam5	LORA_IQ_STD	0x40	IQ as defined
packetParam5	LORA_IQ_INVERTED	0x00	IQ swapped

13.4.2 Tx Setting and Operations

1. Define the output power and ramp time by sending the command:

SetTxParam(power,rampTime)

2. Send the payload to the data buffer by sending the command:

WriteBuffer(offset,*data)

where **data* is a pointer to the payload and *offset* is the address at which the first byte of the payload will be located in the buffer. Offset will correspond to *txBaseAddress* in normal operation.

3. Configure the DIOs and Interrupt sources (IRQs) by sending the command:

SetDioIrqParams(irqMask,dio1Mask,dio2Mask,dio3Mask)

In a typical Tx operation the user can select one or several IRQ sources:

- TxDone IRQ to indicate the end of packet transmission. The transceiver will be in STDBY_RC mode.
- RxTxTimeout (optional) to make sure no deadlock can happen. The transceiver will return automatically to STDBY_RC mode if a timeout occurs.

4. Once configured, set the transceiver in transmitter mode to start transmission by sending the command:

SetTx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

If a timeout is desired, set *periodBaseCount* to a non-zero value. This timeout can be used to avoid deadlock.

Wait for IRQ TxDone or RxTxTimeout

Once a packet has been sent or a timeout has occurred, the transceiver goes automatically to STDBY_RC mode.

5. Clear TxDone or RxTxTimeout IRQ by sending the command:

ClrIrqStatus(irqStatus)

This command will reset the flag for which the corresponding bit position in *irqStatus* is set to 1.

13.4.3 Rx Setting and Operations

1. Configure the DIOs and Interrupt sources (IRQs) by using command:

SetDioIrqParams(irqMask,dio1Mask,dio2Mask,dio3Mask)

In a typical LoRa Rx operation the user could select one or several of the following IRQ sources:

- RxDone to indicate a packet has been detected. This IRQ does not mean that the packet is valid (size or CRC correct). The user must check the packet status to ensure that a valid packed has been received.
- SyncWordValid to indicate that a Sync Word has been detected.
- CrcError to indicate that the received packet has a CRC error
- RxTxTimeout to indicate that no packet has been detected in a given time frame defined by timeout parameter in the SetRx() command.

2. Once configured, set the transceiver in receiver mode to start reception using command:

SetRx(periodBase, periodBaseCount[15:8], periodBaseCount[7:0])

Depending on *periodBaseCount*, 3 possible Rx behaviors are possible:

- *periodBaseCount* is set 0, then no Timeout, Rx Single mode, the device will stay in Rx mode until a reception occurs and the device returns to STDBY_RC mode upon completion.

- *periodBaseCount* is set 0xFFFF, Rx Continuous mode, the device remains in Rx mode until the host sends a command to change the operation mode. The device can receive several packets. Each time a packet is received, a packet received indication is given to the host and the device will continue to search for a new packet.
- *periodBaseCount* is set to another value, then Timeout is active. The device remains in Rx mode; it returns automatically to STDBY_RC Mode on timer end-of-count or when a packet has been received. As soon as a packet is detected, the timer is automatically disabled to allow complete reception of the packet.

3. In typical cases, use a timeout and wait for IRQ RxDone or RxTxTimeout.

If IRQ RxDone is asserted, the transceiver goes to STDBY_RC mode if single mode is used (timeout set to a value different from 0xFFFF). If Continuous mode is used (timeout set to 0xFFFF) the transceiver stays in Rx and continues to listen for a new packet.

4. Check the packet status to make sure that the packet has been received properly, by sending the command:

GetPacketStatus()

The command returns the following parameters:

- *SnrPkt* Estimation of SNR on last packet received. In two's compliment format multiplied by 4.
  Actual SNR in dB =SnrPkt/4

5. Once all checks are complete, clear IRQs by sending the command:

ClrIrqStatus(irqMask)

This command will reset the flag for which the corresponding bit position in *irqMask* is set to 1. Note a a DIO can be mapped to several IRQ sources (ORed with IRQ sources). The DIO will go to zero once IRQ flag has been set to zero.

6. Get the packet length and start address of the received payload by sending the command:

GetRxBufferStatus()

This command returns the length of the last received packet (*payloadLengthRx*) and the address of the first byte received (*rxStartBufferPointer*). It is applicable to all modems. The address is an offset relative to the first byte of the data buffer.

7. Read the data buffer using the command:

ReadBuffer(offset, payloadLengthRx)

Where *offset* is equal to *rxStartBufferPointer* and *payloadLengthRx* is the size of buffer to read.

8. Optionally, the frequency error indicator (FEI) can be read from register 0x0954 (MSB) 0x0955, 0x0956 (LSB). The FEI is expressed as a 20 bit 2's compliment number. This must be converted from two's compliment to a signed FEI reading then, in turn, can be converted to a frequency error in Hz using the following formula:

$$FrequencyError[Hz] = 1.55x\frac{SignedFeiReading}{\frac{1600}{BW[kHz]}}$$

13.5 Settings for Ranging

A basic ranging operation is an exchange of specifically formatted LoRa messages between a single radio configured as a Master radio and a separate SX1280 configured as a Slave.

The following section will introduce the SX1280 configuration settings required for ranging operations. These configuration steps must be reproduced identically on the Master radio and on the Slave radio, except where explicitly stated otherwise.

13.5.1 Ranging Device Setting

The ranging specific settings for master device operation are given in the next section.

1. If not in STDBY_RC mode, then go to this mode by sending the command:

SetStandby(STDBY_RC)

2. Set the packet type to ranging by sending the command:

SetPacketType(PACKET_TYPE_RANGING)

3. Set the modulation parameters for the ranging operation by sending the command:

SetModulationParams(modParamSF,modParamBW, modParamCR)

The definition of the three arguments of the *SetModulationParams* is the same as for LoRa settings.

However, for ranging operation, the use of SF11 and SF12 is not permitted. Similarly, the bandwidth configuration for ranging operations is restricted to the the values 406.25 kHz, 812.5 kHz and 1625 kHz.

The following table summarizes the acceptable values for *SetModulationParams* command (the three arguments can be combined in any way):

Table 13-55: Ranging Device Modulation Parameters

modParamSF	modParamBW	modParamCR
LORA_SF_5	LORA_BW_400	LORA_CR_4_5
LORA_SF_6	LORA_BW_800	LORA_CR_4_6
LORA_SF_7	LORA_BW_1600	LORA_CR_4_7
LORA_SF_8		LORA_CR_4_8
LORA_SF_9		LORA_CR_4_5
LORA_SF_10		LORA_CR_4_6

4. Set the packet parameters by the command:

SetPacketParams(preambleLength, headerType, payloadLength, crcMode, invertIq)

The signification of the arguments is similar to the one of LoRa SetPacketParams usage.

5. Set the RF frequency to use by the command:

SetRfFrequency(rfFrequency)

The rfFrequency is to be provided as a number of PLL step (ie. 52e6/(2^18) Hz). SetRfFrequency() defines the Tx frequency. In Rx the frequency is lowered by the intermediate frequency (IF). The IF is by default set to 1.65 MHz.

6. Set the Tx parameters by:

SetTxParams(txPower, rampTime)

7. During ranging operation, multiple slaves and multiple masters can be within range of communication. However, the ranging operation must use only one slave and one master. To help slaves distinguish the master to respond to and other masters within range, and to address a specific slave, the ranging requests contain an address field which is checked by slave on ranging request reception.

On slave only, use the *WriteRegister* command to set the address the slave can respond to. The registers to write are given by the following table:

Table 13-56: Slave Ranging Request Address Definition

Slave Ranging req address	Address
RangingRangingAddress[31:24]	0x916
RangingRangingAddress[23:16]	0x917
RangingRangingAddress[15:8]	0x918
RangingRangingAddress[7:0]	0x919

The slave also requires the number of address bits to be set by issuing a *WriteRegister* command with the following parameters:

Table 13-57: Register Address Bit Definition

Register Address	Field	Description
0x931	7:6	0x0: 8 bits
		0x1: 16 bits
		0x2: 24 bits
		0x3: bits

The Master also needs to use the same address, as this is the address to which the ranging request will be issued.It is set by issuing the *WriteRegister* command to the following registers:

Table 13-58: Master Ranging Request Address Definition

Master Ranging Request Address	Address
RangingRequestAddress[31:24]	0x912
RangingRequestAddress[23:16]	0x913
RangingRequestAddress[15:8]	0x914
RangingRequestAddress[7:0]	0x915

8. Set the IRQ that should be generated by the radio for ranging operations using the command:

SetDioIrqParams(irqMask, dio1Mask, dio2Mask, dio3Mask)

The IRQs to be activated depend of the ranging role of the sx1280. For the Master typical ranging operations require the following IRQs:

- RangingMasterResultValid
- RangingMasterResultTimeout

For the Slave the typical IRQs are:

- RangingSlaveResponseDone
- RangingSlaveRequestDiscarded

9. The ranging process needs a calibration value to compensate the RxTx delay offset. The calibration value is set by calling *WriteRegister* command on the following registers:

Table 13-59: Calibration Value in Register

Calibration Value	Register
Calibration[15:8]	0x92C
Calibration[7:0]	0x92D

10. The role of the SX1280 in ranging operation must be explicitly given issuing the following command:

SetRangingRole(role)

Where role value is provided by the following table:

Table 13-60: Ranging Role Value

Ranging Role	Value
Master	0x01
Slave	0x00

11. Finally, use the following commands to start the ranging procedure:

- On Slave side: SetRx(periodBaseRx, periodCountRx)
- On Master side: SetTx(periodBaseTx, periodCountTx)

If there is no timing constraint in the application level, it is advised to use the continuous mode (ie. *PeriodCount*=0xFFFF)

The ranging modem automatically manages the transition from Rx to Tx for Slave, and from Tx to Rx for Master.

12. The ranging results are accessible only from the Master. When Master generates the IRQ RangingMasterResultValid, several ranging results are available:

- Raw ranging results
- Averaged ranging results
- Debiased ranging results
- Filtered ranging results

All ranging results are accessible by reading the three registers below:

Table 13-61: Register Result Address

Register Address	Ranging Result
0x961	RangingResult[23:16]
0x962	RangingResult[15:8]
0x963	RangingResult[7:0]

All the ranging results are stored in these three registers and are to be read through a *ReadRegister* command at the given addresses. The selection of the ranging result to read is done by setting bit 5 and bit 4 of register **RangingResMux** at address 0x924 according to the following:.

Table 13-62: Ranging Result Type Selection

RangingResMux(5:4)	Ranging Output Type	Conversion to Distance
00	Raw result, it can be negative	Distance [m] = RangingResult*150/(2^12*BW) with BW in MHz
01	Average result, it can be negative	Distance [m] = RangingResult*20/100
10	Debiased result it cannot be negative	Distance [m] = RangingResult*20/100
11	Filtered result, it cannot be negative	Distance [m] = RangingResult*20/100

Due to the particular usage of the ranging result register, the following procedure is required to read the ranging result:

1. Set the radio in Xoscilator mode with

SetStandby(STDBY_XOSC)

2. Enable clock in LoRa memory:

WriteRegister(0x97F, ReadRegister(0x97F) | (1 << 1));

3. Set the ranging type and read the ranging registers as usual:

WriteRegister(0x0924, (ReadRegister(0x0924) & 0xCF) | (((resultType) & 0x03) << 4));

valLsb = ((ReadRegister(0x0961) << 16) | (ReadRegister(0x0962) << 8) | (ReadRegister(0x0963)));

4. Set back the radio in config mode:

SetStandby(STDBY_RC)

13.5.2 Ranging Operation as State Machines

The ranging operation can be summarized with the following simple state machines:

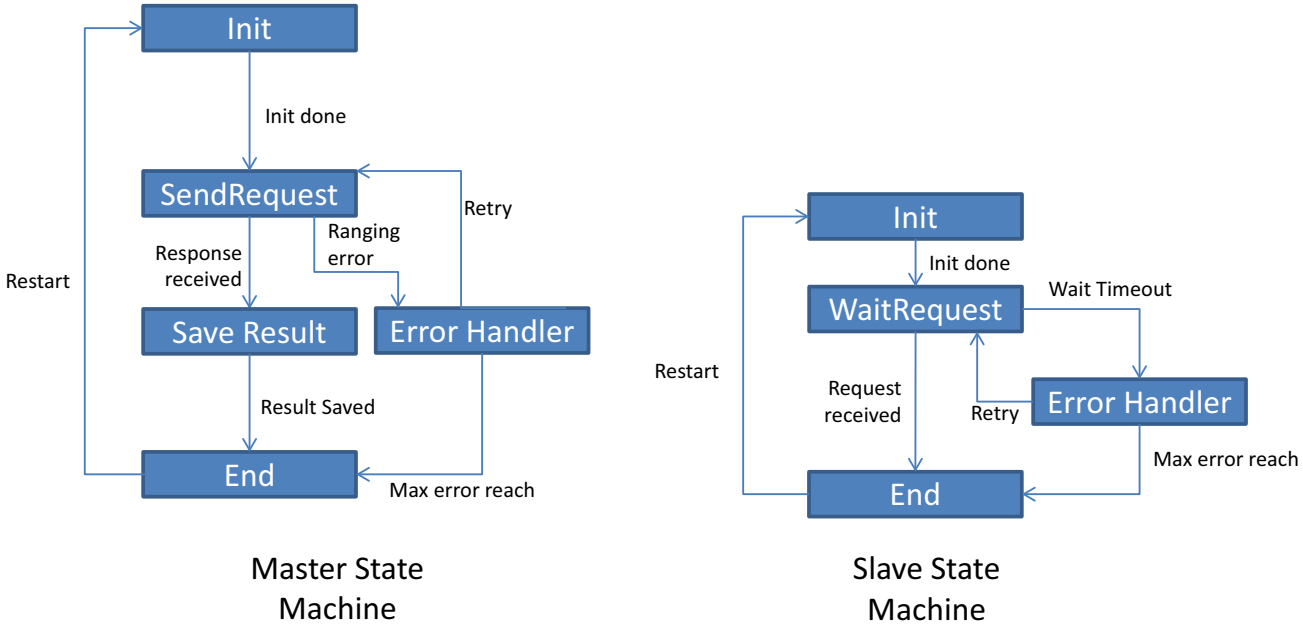


Figure 13-1: Ranging State Machines

These state machines do not represent the internal behavior of the chip, but the steps the user has to implement in order to use the basic ranging estimation.

The radio configuration described in steps 1) to 10) outlined previously belong to the Init state.

The SetTx and SetRx calls described in step 11) are executed in states *SendRequest* and *WaitRequest*.

The reading of the ranging results is performed in the state *SaveResult* of Master state machine. It is also the place where the frequency correction will be applied.

Note that this frequency correction needs a prior evaluation of the Estimated Frequency Error on Slave side, which is not represented here.

The Error Handler state is application-dependent. This state should be reached in case of *RangingMasterResultTimeout* or *RangingSlaveRequestDiscarded* IRQs. Here it is proposed to retry sending/receiving ranging request until a maximum number of errors has been reached.

Note: the reception of the ranging response on Master side and broadcast of the response on Slave side is automatically handled by the SX1280 chip.

13.6 Miscellaneous Functions

13.6.1 SetRegulatorMode Command

By default only LDO is used. This is useful in low cost applications where the cost of an extra inductor needed for DC-DC converter is prohibitive. Using only LDO implies that the Rx or Tx current is doubled. This command allows the user to specify if DC-DC or LDO is used for power regulation.

Table 13-63: Power Regulation Selection SPI Data Transfer

Byte	0	1
Data from host	Opcode= 0x96	regModeParam

Table 13-64: Power Regulation Selection UART Data Transfer

Byte	0	1	2
Data from host	Opcode= 0x96	0x01	regModeParam

Table 13-65: RegModeParam Definition

RegModeParam value	0	1
Regulator used	Only LDO used for all modes	DC-DC used for STDBY_XOSC, FS, Rx and Tx modes LDO used for STDBY_RC

13.6.2 Context Saving

Upon transition to Sleep mode the contents of the transceiver registers will be lost. The configuration of the radio can be automatically restored using the *SetSaveContext()* command. This stores the present context of the radio register values to the Data RAM within the Protocol Engine for restoration upon wake-up. The operation sequence is as follows:

- Once the device has been configured, the host must send *SaveContext()* before *SetSleep()* command.
- Upon issuing the *SetSleep()* command it is necessary that the Data RAM is to be retained.
- When the transceiver wakes up from sleep, it checks to see if the Data RAM has been saved and also if a valid register dump exists (i.e. the save context command has been issued). If both conditions are met the registers will be restored.

  Note that in duty cycled operation the save context is automatically invoked. The Rx buffer pointer used for payload data is reset to the default value of 0x00 when exiting Sleep mode.

Table 13-66: SetSaveContext Data Transfer

Byte	0
Data from host	Opcode = 0xD5

The *SetSaveContext()* data transfer is the same for both SPI and UART transfers.

14. Reference Design and Application Schematics

14.1 Reference Design

14.1.1 Application Design Schematic

The long range 2.4 GHz application circuit is shown below:

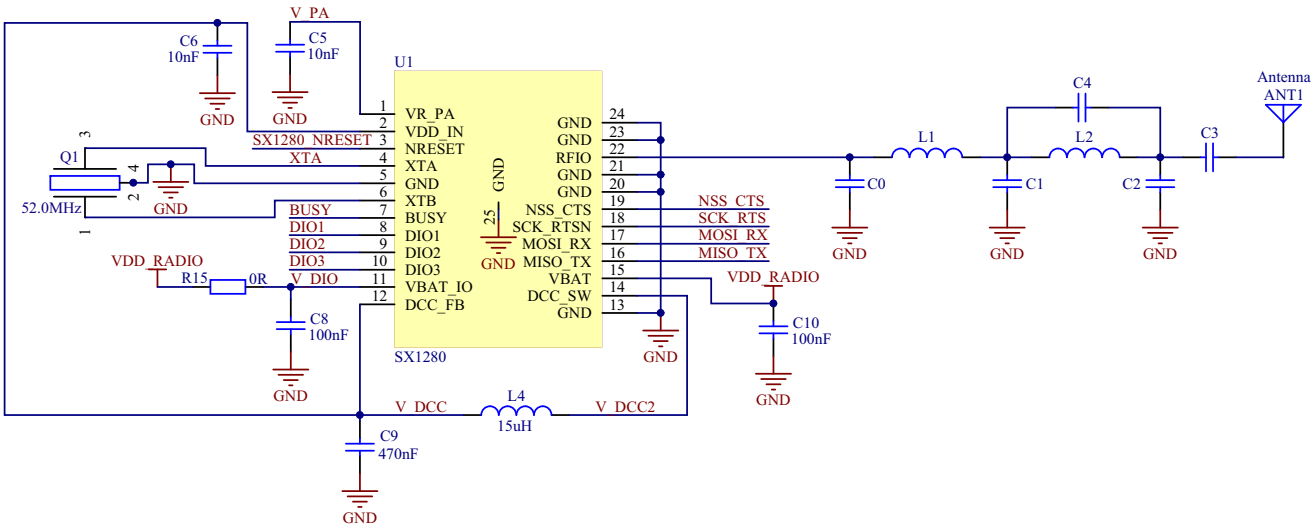


Figure 14-1: Transceiver Application Design Schematic

14.1.2 Reference Design BOM

Table 14-1: Reference Design BOM

RefDes	MPN	Geom	Value	Description
U1	SX1280	VQFN24 4x4mm	SX1280	SX1280 2.4 GHz High Link Budget Transceiver with LoRa technology
R15	CRCW04020000Z0ED	0402	0 Ω	Thick Film Resistor ±1%, 1/16W
C0	GRM1555C1HR80BA01D	0402	0.8 pF	Multilayer ceramic capacitors C0G ± 0.1 pF, 50 V
C1	GRM1555C1H1R2BA01D	0402	1.2 pF	Multilayer ceramic capacitors C0G ± 0.1 pF, 50 V
C2	GRM1555C1H1R2BA01D	0402	1.2 pF	Multilayer ceramic capacitors C0G ± 0.1 pF, 50 V
C3	GRM1555C1H101JA01D	0402	100 pF	Multilayer ceramic capacitors C0G ± 5 %, 50 V
C4	GRM1555C1HR50WA01D	0402	0.5 pF	Multilayer ceramic capacitors C0G ± 0.05 pF, 50 V
C5	GRM155R71E103KA01D	0402	10 nF	Multilayer ceramic capacitors X7R ± 10 %, 25 V
C6	GRM155R71E103KA01D	0402	10 nF	Multilayer ceramic capacitors X7R ± 10 %, 25 V
C8	GRM155R71C104KA88D	0402	100 nF	Multilayer ceramic capacitors X7R ± 10 %, 16 V
C9	GRM155R61A474KE15D	0402	470 nF	Multilayer ceramic capacitors X5R ± 10 %, 10 V
C10	GRM155R71C104KA88D	0402	100 nF	Multilayer ceramic capacitors X7R ± 10 %, 16 V
L1	LQW15AN3N0B80D	0402	3.0 nH	Wirewound Inductor ± 0.1 nH
L2	LQW15AN2N5C00D	0402	2.5 nH	Wirewound Inductor ± 0.2 nH
L4	MLZ2012M150W	0805	15 µH	MLZ2012 Multilayer Shielded Inductor ± 5 %
Q1	NX2016SA-52.000000MHZ	NX2016SA	52.000 MHz	Crystal unit, NDK Ref: EXS00A-CS07103, Tol. ± 10 ppm, Cload = 10 pF

14.1.3 Reference Design PCB

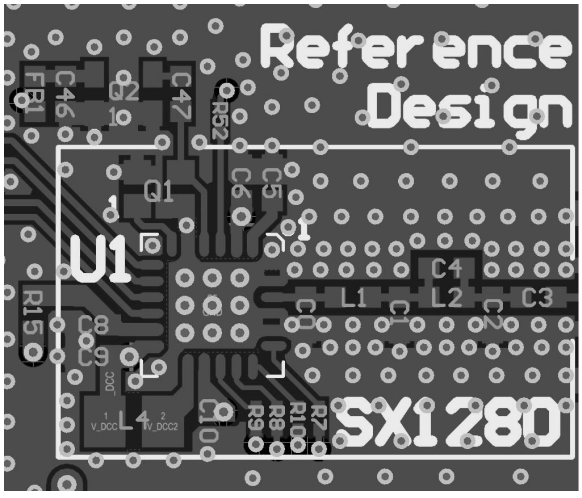


Figure 14-2: Long Range Reference Design PCB Layout

14.2 Application Design with optional TCXO

Although the modulation parameters of the transceiver are designed to tolerate typical frequency drifts associated with the use of industry standard tolerance crystal reference oscillator components, an external Temperature Compensated Crystal Oscillator (TCXO) can be used. The figure below shows how the TCXO should be AC-coupled to the XTA input of transceiver. Please consult the crystal manufacturer for full details of the components required specific to your TCXO.

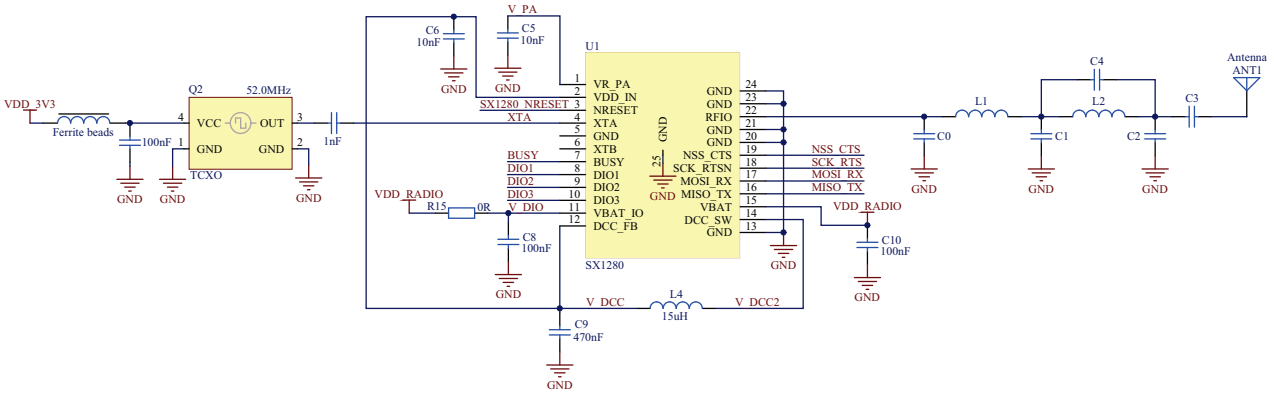


Figure 14-3: Application Schematic with Optional TCXO

14.3 Application Design with Low Drop Out Regulator

The following schematic shows the connection for use of the LDO in place of the DC-DC converter.

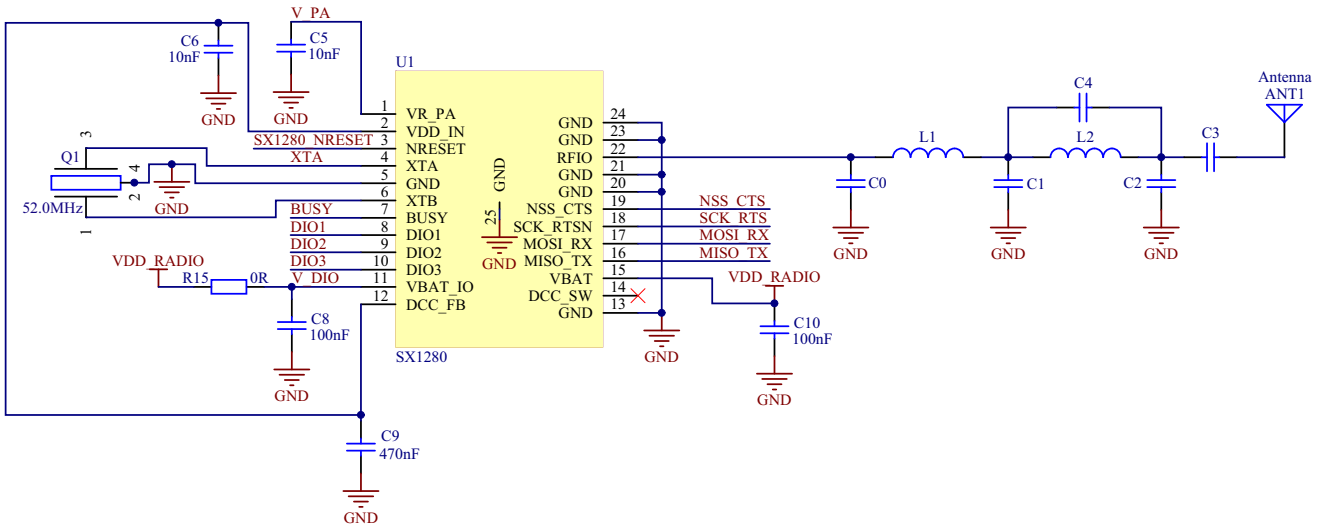


Figure 14-4: Application Schematic with Low Drop Out Regulator Schematic

14.4 Sleep Mode Consumption

To attain the low Sleep mode consumption figures stated in the specification it is necessary to ensure that the following states are presented by the host controller to the SX1280:

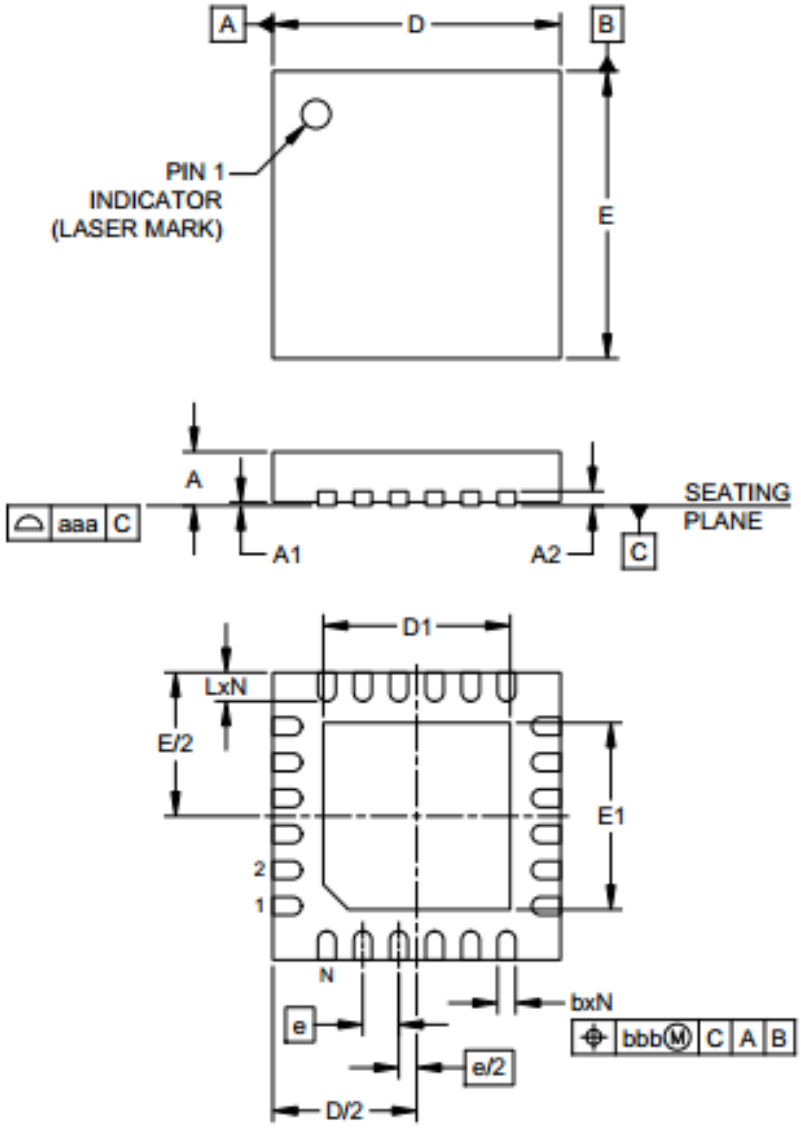
Table 14-2: Host Settings for Minimizing Sleep Mode Consumption

Transceiver Pin	State
NSS	Logical '1'
SCK	Logical '0'
MOSI	Logical '1' or '0'
MISO	High Impedance
NRESET	Logical '1'
BUSY	Output
DIO1	Output
DIO2	Output
DIO3	Output

15. Packaging Information

15.1 Package Outline Drawing

The transceiver is delivered in a 4x4mm QFN package with 0.5mm pitch:



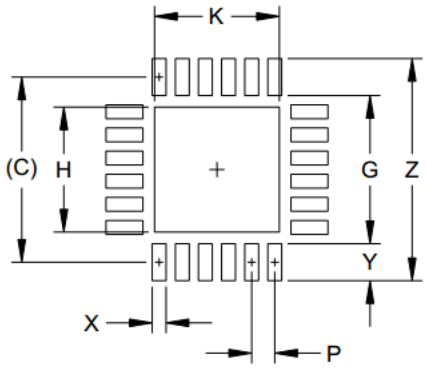
DIMENSIONS			
DIM	MILLIMETERS		
	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A2		(0.20)	
b	0.18	0.25	0.30
D	3.90	4.00	4.10
D1	2.55	2.60	2.65
E	3.90	4.00	4.10
E1	2.55	2.60	2.65
e		0.50 BSC	
L	0.35	0.40	0.45
N		24	
aaa		0.08	
bbb		0.10	

NOTES:
1. CONTROLLING DIMENSIONS ARE IN MILLIMETERS (ANGLES IN DEGREES).
2. COPLANARITY APPLIES TO THE EXPOSED PAD AS WELL AS THE TERMINALS.

Figure 15-1: QFN 4x4 Package Outline Drawing

15.2 Land Pattern

The recommended land pattern is as follows:



DIMENSIONS	
DIM	MILLIMETERS
C	(4.00)
G	3.20
H	2.70
K	2.70
P	0.50
X	0.30
Y	0.80
Z	4.80

NOTES:

1. CONTROLLING DIMENSIONS ARE IN MILLIMETERS (ANGLES IN DEGREES).
2. THIS LAND PATTERN IS FOR REFERENCE PURPOSE ONLY.
   CONSULT YOUR MANUFACTURING GROUP TO ENSURE YOUR COMPANY'S
   MANUFACTURING GUIDELINES ARE MET.
3. THERMAL VIAS IN THE LAND PATTERN OF THE EXPOSED PAD SHALL BE CONNECTED
   TO A SYSTEM GROUND PLANE. FAILURE TO DO SO MAY COMPROMISE THE THERMAL
   AND/OR FUNCTIONAL PERFORMANCE OF THE DEVICE.
4. SQUARE PACKAGE - DIMENSIONS APPLY IN BOTH " X " AND " Y " DIRECTIONS.

Figure 15-2: QFN 4x4mm Land Pattern

15.3 Reflow Profiles

Reflow process instructions are available from the Semtech website, at the following address:

http://www.semtech.com/quality/ir_reflow_profiles.html

The transceiver uses a QFN24 4x4mm package, also named MLP package.

15.4 Thermal Impedance

The Package QFN 24L 4X4 E-pad (Device: SX12801MLTRT) mounted on 4 layers JEDEC PCB with 9 thermal vias in still air is able to dissipate the required amount of power 0.20W at the ambient temperature of 25°C while keeping the maximum junction temperature of die below 125°C.

Θ_{JA} of the corresponding package in still air is computed as 50.3 °C / W.

Ψ_{JT} of the corresponding package is computed as 0.3 °C / W.

Θ_{JC} of the corresponding package is computed as 30.0 °C / W.

Θ_{JB} of the corresponding package is computed as 13.3 °C / W.

Glossary

List of Acronyms and their Meaning

Acronym	Meaning
ACR	Adjacent Channel Rejection
ADC	Analog-to-Digital Converter
AFC	Automatic Frequency Correction
AGC	Automatic Gain Control
API	Application Programming Interface
β	Modulation Index
BLE	Bluetooth Low Energy
BR	Bit Rate
BT	Bandwidth-Time bit period product
BW	BandWidth
CAD	Channel Activity Detection
CMD	Command Transaction
CPOL	Clock Polarity
CPHA	Clock Phase
CR	Coding Rate
CRC	Cyclical Redundancy Check
CW	Continuous Wave
DIO	Digital Input / Output
FEC	Forward Error Correction
FLRC	Fast Long Range Communication
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying
GMSK	Gaussian Minimum Shift Keying
IRQ	Interrupt Request
LDO	Low-Dropout
LLID	Logical Link Identifier
LNA	Low-Noise Amplifier
LO	Local Oscillator
LoRa	Long Range Communication

List of Acronyms and their Meaning

Acronym	Meaning
LSB	Least Significant Bit
MD	More Data
MIC	Message Integrity Check
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
MSB	Most Significant Bit
MSK	Minimum-Shift Keying
NESN	Next Expected Sequence Number
NOP	No Operation
NRZ	Non-Return-to-Zero
NSS	Slave Select active low
OOK	On-Off Keying
PA	Power Amplifier
PDU	Protocol Data Unit
PER	Packet Error Rate
PID	Product Identification
PLL	Phase-Locked Loop
PRNG	Pseudo-Random Number Generation
RFU	Reserved for Future Use
RTC	Real-Time Clock
RTSN	Request to Send
SCK	Serial Clock
SF	Spreading Factor
SN	Sequence Number
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
STDBY	Standby
TCXO	Temperature Compensated Crystal Oscillator
UART	Universal Asynchronous Receiver/Transmitter
XOSC	Crystal Oscillator

Contact Information

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