# Low Power Long Range Transceiver Module

**Model No.:RFM95W/96W/98W**

## GENERAL DESCRIPTION

The RFM95W/96W/98W transceivers feature the LoRa™ long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using LoRa™ modulation technique RFM95W/96W/98W can achieve a sensitivity of over -140dBm using a low cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRa™ also provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

These devices also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4g. The RFM95W/96W/98W deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.



**RFM95W/96W/98W**

## KEY PRODUCT FEATURES

- LoRa™ Modem.
- 164 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs. 3.3V supply.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -144 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention.
- Fully integrated synthesizer with a resolution of 61 Hz.
- FSK, GFSK, MSK, GMSK, LoRa™ and OOK modulation.
- Built-in bit synchronizer for clock recovery.
- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC.
- Packet engine up to 256 bytes with CRC.
- Built-in temperature sensor and low battery indicator.
- Modue Size：16*16mm

## APPLICATIONS

- Automated Meter Reading.
- Home and Building Automation.
- Wireless Alarm and Security Systems.
- Industrial Monitoring and Control
- Long range Irrigation Systems

## Section Page

1. General Description ..... 9
1.1. Simplified Block Diagram ..... 9
1.2. Product Versions ..... 10
1.3. Pin Diagram ..... 10
1.4. Pin Description ..... 11
2. Electrical Characteristics ..... 12
2.1. ESD Notice ..... 12
2.2. Absolute Maximum Ratings ..... 12
2.3. Operating Range ..... 12
2.4. Chip Specification ..... 13
2.4.1. Power Consumption ..... 13
2.4.2. Frequency Synthesis ..... 13
2.4.3. FSK/OOK Mode Receiver ..... 14
2.4.4. FSK/OOK Mode Transmitter ..... 15
2.4.5. Electrical specification for LoraTM modulation ..... 16
2.4.6. Digital Specification ..... 20
3. RFM95W/96W/98W Features ..... 22
3.1. LoRaTM Modem ..... 22
3.2. FSK/OOK Modem ..... 22
4. RFM95W/96W/98W Digital Electronics ..... 23
4.1. The LoRaTM Modem ..... 23
4.1.1. Link Design Using the LoRaTM Modem ..... 24
4.1.2. LoRaTM Digital Interface ..... 30
4.1.3. Operation of the LoRaTM Modem ..... 32
4.1.4. Frequency Settings ..... 32
4.1.5. Frequency Error Indication ..... 33
4.1.6. LoRa AFC ..... 33
4.1.7. LoRaTM Modem State Machine Sequences ..... 34
4.1.8. Modem Status Indicators ..... 42
4.2. FSK/OOK Modem ..... 42
4.2.1. Bit Rate Setting ..... 42
4.2.2. FSK/OOK Transmission ..... 43
4.2.3. FSK/OOK Reception ..... 44
4.2.4. Operating Modes in FSK/OOK Mode ..... 51
4.2.5. Startup Times ..... 51
4.2.6. Receiver Startup Options ..... 54
4.2.7. Receiver Restart Methods ..... 55
4.2.8. Top Level Sequencer ..... 56
4.2.9. Data Processing in FSK/OOK Mode ..... 61
4.2.10. FIFO ..... 62

## Section Page

4.2.11. Digital IO Pins Mapping ..... 65
4.2.12. Continuous Mode ..... 66
4.2.13. Packet Mode ..... 67
4.2.14. io-homecontrol® Compatibility Mode ..... 75
4.3. SPI Interface ..... 76
5. RFM95W/96W/98W Analog & RF Frontend Electronics ..... 77
5.1. Power Supply Strategy ..... 77
5.2. Low Battery Detector ..... 77
5.3. Frequency Synthesis ..... 77
5.3.1. Crystal Oscillator ..... 77
5.3.2. CLKOUT Output ..... 78
5.3.3. PLL ..... 78
5.3.4. RC Oscillator ..... 78
5.4. Transmitter Description ..... 79
5.4.1. Architecture Description ..... 79
5.4.2. RF Power Amplifiers ..... 79
5.4.3. High Power +20 dBm Operation ..... 80
5.4.4. Over Current Protection ..... 81
5.5. Receiver Description ..... 81
5.5.1. Overview ..... 81
5.5.2. Receiver Enabled and Receiver Active States ..... 81
5.5.3. Automatic Gain Control In FSK/OOK Mode ..... 81
5.5.4. RSSI in FSK/OOK Mode ..... 82
5.5.5. RSSI in LoRaTM Mode ..... 83
5.5.6. Channel Filter ..... 83
5.5.7. Temperature Measurement ..... 84
6. Description of the Registers ..... 85
6.1. Register Table Summary ..... 85
6.2. FSK/OOK Mode Register Map ..... 88
6.3. Band Specific Additional Registers ..... 101
6.4. LoRaTM Mode Register Map ..... 103
7. Application Information ..... 110
7.1. Crystal Resonator Specification ..... 110
7.2. Reset of the Chip ..... 110
7.2.1. POR ..... 110
7.2.2. Manual Reset ..... 111
7.3. Top Sequencer: Listen Mode Examples ..... 111
7.3.1. Wake on Preamble Interrupt ..... 111
7.3.2. Wake on SyncAddress Interrupt ..... 114
7.4. Top Sequencer: Beacon Mode ..... 117
7.4.1. Timing diagram ..... 117

## Section Page

7.4.2. Sequencer Configuration ..... 117
7.5. Example CRC Calculation ..... 119
7.6. Example Temperature Reading ..... 120
7.7. Reference Design ..... 121
8. Packaging Information ..... 122
8.1. Package Outline Drawing ..... 122
8.2. Ordering Information ..... 122
9. Contact Information ..... 123

## Section Page

Table 1. RFM95W/96W/98W Device Variants and Key Parameters ....10
Table 2. Pin Description ....11
Table 3. Absolute Maximum Ratings ....12
Table 4. Operating Range ....12
Table 5. Power Consumption Specification ....13
Table 6. Frequency Synthesizer Specification ....13
Table 7. FSK/OOK Receiver Specification ....14
Table 8. Transmitter Specification ....15
Table 9. LoRa Receiver Specification. ....17
Table 10. Digital Specification ....20
Table 11. Example LoRaTM Modem Performances ....23
Table 12. Range of Spreading Factors ....25
Table 13. Cyclic Coding Overhead ....25
Table 14. LoRa Bandwidth Options ....26
Table 15. LoRaTM Operating Mode Functionality ....32
Table 16. LoRa CAD Consumption Figures ....41
Table 17. DIO Mapping LoRaTM Mode ....42
Table 18. Bit Rate Examples ....43
Table 19. Preamble Detector Settings ....49
Table 20. RxTrigger Settings to Enable Timeout Interrupts ....50
Table 21. Basic Transceiver Modes ....51
Table 22. Receiver Startup Time Summary ....52
Table 23. Receiver Startup Options ....55
Table 24. Sequencer States ....56
Table 25. Sequencer Transition Options ....57
Table 26. Sequencer Timer Settings ....59
Table 27. Status of FIFO when Switching Between Different Modes of the Chip ....63
Table 28. DIO Mapping, Continuous Mode ....65
Table 29. DIO Mapping, Packet Mode ....65
Table 30. CRC Description ....73
Table 31. Power Amplifier Mode Selection Truth Table ....79
Table 32. High Power Settings ....80
Table 33. Operating Range, +20dBm Operation ....80
Table 34. Operating Range, +20dBm Operation ....80
Table 35. Trimming of the OCP Current ....81
Table 36. LNA Gain Control and Performances ....82
Table 37. RssiSmoothing Options ....83
Table 38. Available RxBw Settings ....83
Table 39. Registers Summary ....85
Table 40. Register Map ....88
Table 41. Low Frequency Additional Registers ....101

## Section Page

Table 42. High Frequency Additional Registers .....102
Table 43. Crystal Specification .....110
Table 44. Listen Mode with PreambleDetect Condition Settings .....113
Table 45. Listen Mode with PreambleDetect Condition Recommended DIO Mapping .....113
Table 46. Listen Mode with SyncAddress Condition Settings .....116
Table 47. Listen Mode with PreambleDetect Condition Recommended DIO Mapping .....116
Table 48. Beacon Mode Settings .....118

## Section Page

Figure 1. Block Diagram .....9
Figure 2. Pin Diagrams .....10
Figure 3. RFM95W/96W/98W Block Schematic Diagram .....21
Figure 4. LoRaTM Modem Connectivity .....24
Figure 5. LoRaTM Packet Structure .....27
Figure 6. Interrupts generated in the case of successful frequency hopping communication. .....29
Figure 7. LoRaTM data buffer .....30
Figure 8. LoRaTM modulation transmission sequence .....33
Figure 9. LoRaTM receive sequence .....34
Figure 10. LoRaTM CAD flow.....38
Figure 11. Channel activity detection (CAD) time as a function of spreading factor .....39
Figure 12. Consumption Profile of the LoRa CAD Process .....41
Figure 13. OOK Peak Demodulator Description .....45
Figure 14. Floor Threshold Optimization .....46
Figure 15. Bit Synchronizer Description .....47
Figure 16. FEI Process .....48
Figure 17. Startup Process .....51
Figure 18. Time to Rssi Sample .....53
Figure 19. Tx to Rx Turnaround .....53
Figure 20. Rx to Tx Turnaround .....53
Figure 21. Receiver Hopping .....54
Figure 22. Transmitter Hopping .....54
Figure 23. Timer1 and Timer2 Mechanism .....58
Figure 24. Sequencer State Machine .....60
Figure 25. RFM95W/96W/98W Data Processing Conceptual View .....61
Figure 26. FIFO and Shift Register (SR) .....62
Figure 27. FifoLevel IRQ Source Behavior .....63
Figure 28. Sync Word Recognition .....64
Figure 29. Continuous Mode Conceptual View .....66
Figure 30. Tx Processing in Continuous Mode .....66
Figure 31. Rx Processing in Continuous Mode .....67
Figure 32. Packet Mode Conceptual View .....68
Figure 33. Fixed Length Packet Format .....69
Figure 34. Variable Length Packet Format .....70
Figure 35. Unlimited Length Packet Format .....70
Figure 36. Manchester Encoding/Decoding .....74
Figure 37. Data Whitening Polynomial .....75
Figure 38. SPI Timing Diagram (single access) .....76
Figure 39. TCXO Connection .....77
Figure 40. RF Front-end Architecture Shows the Internal PA Configuration. .....79
Figure 41. Temperature Sensor Response .....84
Figure 42. POR Timing Diagram .....110

## Section Page

Figure 43. Manual Reset Timing Diagram .....111
Figure 44. Listen Mode: Principle .....111
Figure 45. Listen Mode with No Preamble Received .....112
Figure 46. Listen Mode with Preamble Received .....112
Figure 47. Wake On PreambleDetect State Machine .....113
Figure 48. Listen Mode with no SyncAddress Detected .....114
Figure 49. Listen Mode with Preamble Received and no SyncAddress .....114
Figure 50. Listen Mode with Preamble Received & Valid SyncAddress .....115
Figure 51. Wake On SyncAddress State Machine .....115
Figure 52. Beacon Mode Timing Diagram .....117
Figure 53. Beacon Mode State Machine .....117
Figure 54. Example CRC Code .....119
Figure 55. Example Temperature Reading .....120
Figure 56. Reference Schematic.....121
Figure 57. Package Outline Drawing .....122

# 1. General Description

The RFM95W/96W/98W incorporates the LoRa™ spread spectrum modem which is capable of achieving significantly longer range than existing systems based on FSK or OOK modulation. With this new modulation scheme sensitivities 8 dB better than FSK can be achieved with a low-cost, low-tolerance, crystal reference. This increase in link budget provides much longer range and robustness without the need for external amplification. LoRa™ also provides significant advances in selectivity and blocking performance, further improving communication reliability. For maximum flexibility the user may decide on the spread spectrum modulation bandwidth (BW), spreading factor (SF) and error correction rate (CR). Another benefit of the spread modulation is that each spreading factor is orthogonal - thus multiple transmitted signals can occupy the same channel without interfering. This also permits simple coexistence with existing FSK based systems. Standard GFSK, FSK, OOK, and GMSK modulation is also provided to allow compatibility with existing systems or standards such as wireless MBUS and IEEE 802.15.4g.

The RFM95W/96W offers bandwidth options ranging from 125 kHz to 500 kHz with spreading factors ranging from 6 to 12, RFM98W Band3 module options ranging from 20.83kHz to 125kHz with spreading factors ranging from 6 to 12, The RFM95W covers the higher UHF bands. The RFM96W/98W cover the lower UHF bands.

## 1.1. Simplified Block Diagram

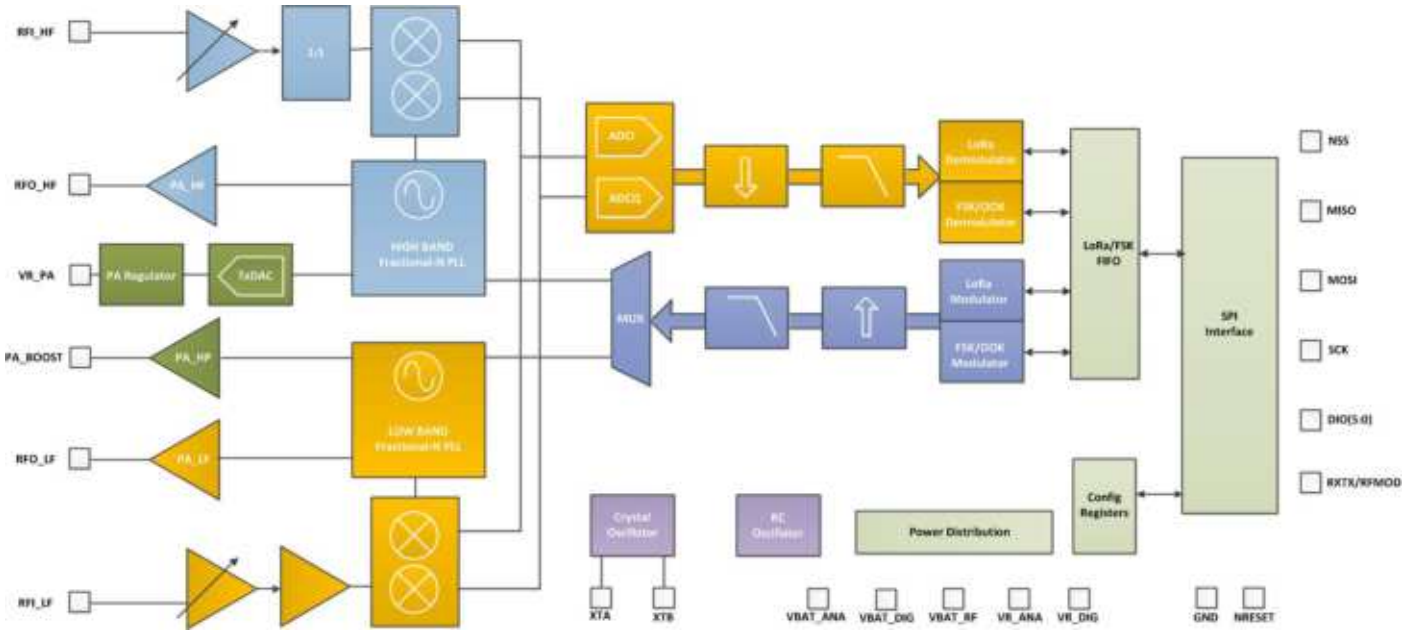



*Figure 1. Block Diagram*

## 1.2. Product Versions

The features of the three product variants are detailed in the following table.

*Table 1 RFM95W/96W/98W Device Variants and Key Parameters*

| Part Number | Frequency Band | Spreading Factor | Bandwidth | Effective Bitrate | Est. Sensitivity |
|---|---|---|---|---|---|
| RFM95W | 868\915MHz | 6 - 12 | 125 - 500 kHz | .293 - 37.5 kbps | -111 to -136 dBm |
| RFM96W/98W | 433\470MHz | 6 - 12 | 62.5- 500 kHz | .1465 - 37.5 kbps | -112 to -140dBm |
| RFM98W | 169MHz | 6 - 12 | 31.25- 125 kHz | 73.24- 9375 bps | -118 to -143 dBm |

## 1.3. Pin Diagram

The following diagram shows the pin arrangement , top view.

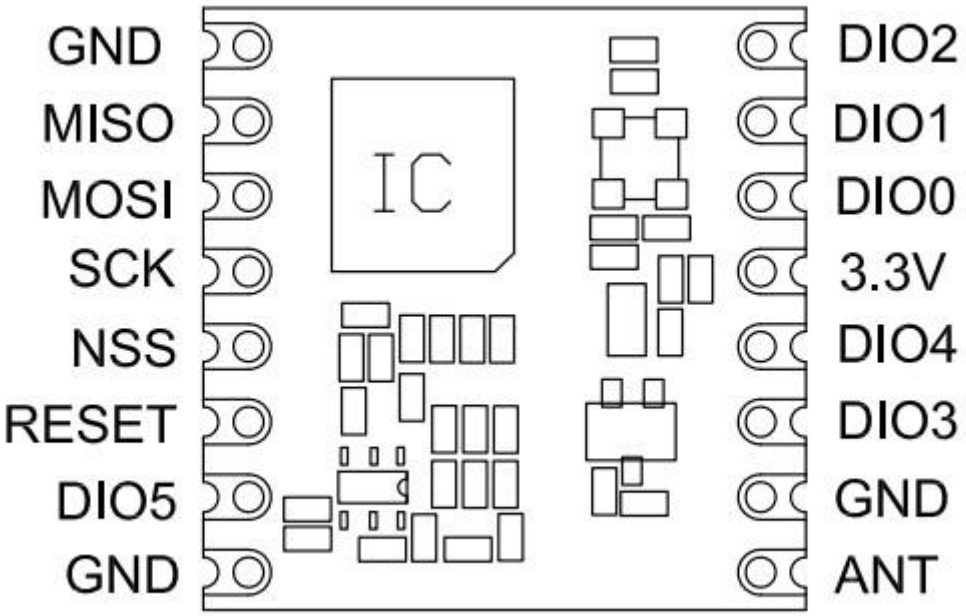



*Figure 2. Pin Diagrams*

## 1.4. Pin Description

*Table 2 Pin Description*

| Number | Name | Type | Description<br>Description Stand Alone Mode |
|---|---|---|---|
| 1 | GND | - | Ground |
| 2 | MISO | O | SPI Data output |
| 3 | MOSI | I | SPI Data input |
| 4 | SCK | I | SPI Clock input |
| 5 | NSS | I | SPI Chip select input |
| 6 | RESET | I | Reset trigger input |
| 7 | DIO5 | I/O | Digital I/O, software configured |
| 8 | GND | - | Ground |
| 9 | ANT | - | RF signal output/input. |
| 10 | GND | - | Ground |
| 11 | DIO3 | I/O | Digital I/O, software configured |
| 12 | DIO4 | I/O | Digital I/O, software configured |
| 13 | 3.3V | - | Supply voltage |
| 14 | DIO0 | I/O | Digital I/O, software configured |
| 15 | DIO1 | I/O | Digital I/O, software configured |
| 16 | DIO2 | I/O | Digital I/O, software configured |

# 2. Electrical Characteristics



## 2.1. ESD Notice

The RFM95W/96W/98W is a high performance radio frequency device. It satisfies:

- Class 2 of the JEDEC standard JESD22-A114-B (Human Body Model) on all pins.
- Class III of the JEDEC standard JESD22-C101C (Charged Device Model) on all pins

It should thus be handled with all the necessary ESD precautions to avoid any permanent damage.

## 2.2. Absolute Maximum Ratings

Stresses above the values listed below may cause permanent device failure. Exposure to absolute maximum ratings for extended periods may affect device reliability.

*Table 3 Absolute Maximum Ratings*

| Symbol | Description | Min | Max | Unit |
|---|---|---|---|---|
| VDDmr | Supply Voltage | -0.5 | 3.9 | V |
| Tmr | Temperature | -55 | +115 | ° C |
| Tj | Junction temperature | - | +125 | ° C |
| Pmr | RF Input Level | - | +10 | dBm |

*Note Specific ratings apply to +20 dBm operation (see Section 5.4.3).*

## 2.3. Operating Range

*Table 4 Operating Range*

| Symbol | Description | Min | Max | Unit |
|---|---|---|---|---|
| VDDop | Supply voltage | 1.8 | 3.7 | V |
| Top | Operational temperature range | -20 | +70 | °C |
| Clop | Load capacitance on digital ports | - | 25 | pF |
| ML | RF Input Level | - | +10 | dBm |

*Note A specific supply voltage range applies to +20 dBm operation (see Section 5.4.3).*

## 2.4. Chip Specification

The tables below give the electrical specifications of the transceiver under the following conditions: Supply voltage VDD=3.3 V, temperature = 25 °C, *FXOSC* = 32 MHz, $F_{RF}$ = 169/434/868/915 MHz (see specific indication), Pout = +13dBm, 2-level FSK modulation without pre-filtering, FDA = 5 kHz, Bit Rate = 4.8 kb/s and terminated in a matched 50 Ohm impedance, shared Rx and Tx path matching., unless otherwise specified.

### 2.4.1. Power Consumption

*Table 5 Power Consumption Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| IDDSL | Supply current in Sleep mode | | - | 0.2 | 1 | uA |
| IDDIDLE | Supply current in Idle mode | RC oscillator enabled | - | 1.5 | - | uA |
| IDDST | Supply current in Standby mode | Crystal oscillator enabled | - | 1.6 | 1.8 | mA |
| IDDFS | Supply current in Synthesizer mode | FSRx | - | 5.8 | - | mA |
| IDDR | Supply current in Receive mode | *LnaBoost* Off, higher bands<br>*LnaBoost* On, higher bands<br>Lower bands | -<br>-<br>- | 10.8<br>11.5<br>12.1 | -<br>-<br>- | mA |
| IDDT | Supply current in Transmit mode with impedance matching | RFOP = +20 dBm, on PA_BOOST<br>RFOP = +17 dBm, on PA_BOOST<br>RFOP = +13 dBm, on RFO_LF/HF pin<br>RFOP = + 7 dBm, on RFO_LF/HF pin | -<br>-<br>-<br>- | 120<br>87<br>29<br>20 | -<br>-<br>-<br>- | mA<br>mA<br>mA<br>mA |

### 2.4.2. Frequency Synthesis

*Table 6 Frequency Synthesizer Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| FR | Synthesizer frequency range | Programmable | 137<br>410<br>862 | -<br>-<br>- | 175<br>525<br>1020 | MHz |
| FXOSC | Crystal oscillator frequency | | - | 32 | - | MHz |
| TS_OSC | Crystal oscillator wake-up time | | - | 250 | - | us |
| TS_FS | Frequency synthesizer wake-up time to PllLock signal | From Standby mode | - | 60 | - | us |
| TS_HOP | Frequency synthesizer hop time at most 10 kHz away from the target frequency | 200 kHz step<br>1 MHz step<br>5 MHz step<br>7 MHz step<br>12 MHz step<br>20 MHz step<br>25 MHz step | -<br>-<br>-<br>-<br>-<br>-<br>- | 20<br>20<br>50<br>50<br>50<br>50<br>50 | -<br>-<br>-<br>-<br>-<br>-<br>- | us<br>us<br>us<br>us<br>us<br>us<br>us |
| FSTEP | Frequency synthesizer step | FSTEP = FXOSC/$2^{19}$ | - | 61.0 | - | Hz |

| | | | | | | |
|---|---|---|---|---|---|---|
| FRC | RC Oscillator frequency | After calibration | - | 62.5 | - | kHz |
| BRF | Bit rate, FSK | Programmable values (1) | 1.2 | - | 300 | kbps |
| BRO | Bit rate, OOK | Programmable | 1.2 | - | 32.768 | kbps |
| BRA | Bit Rate Accuracy | ABS(wanted BR - available BR) | - | - | 250 | ppm |
| FDA | Frequency deviation, FSK (1) | Programmable<br>FDA + BRF/2 =< 250 kHz | 0.6 | - | 200 | kHz |

*Note For Maximum Bit rate the maximum modulation index is 0.5.*

### 2.4.3. FSK/OOK Mode Receiver

All receiver tests are performed with RxBw = 10 kHz (Single Side Bandwidth) as programmed in *RegRxBw*, receiving a PN15 sequence. Sensitivities are reported for a 0.1% BER (with Bit Synchronizer enabled), unless otherwise specified. Blocking tests are performed with an unmodulated interferer. The wanted signal power for the Blocking Immunity, ACR, IIP2, IIP3 and AMR tests is set 3 dB above the receiver sensitivity level.

*Table 7 FSK/OOK Receiver Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| RFS_F_LF | Direct tie of RFI and RFO pins, shared Rx, Tx paths FSK sensitivity, highest LNA gain.<br>Lower frequency bands | FDA = 5 kHz, BR = 1.2 kb/s<br>FDA = 5 kHz, BR = 4.8 kb/s<br>FDA = 40 kHz, BR = 38.4 kb/s*<br>FDA = 20 kHz, BR = 38.4 kb/s**<br>FDA = 62.5 kHz, BR = 250 kb/s*** | -<br>-<br>-<br>-<br>- | -121<br>-117<br>-107<br>-108<br>-95 | -<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| | Split RF paths, the RF switch insertion loss is not accounted for.<br>Lower frequency bands | FDA = 5 kHz, BR = 1.2 kb/s<br>FDA = 5 kHz, BR = 4.8 kb/s<br>FDA = 40 kHz, BR = 38.4 kb/s*<br>FDA = 20 kHz, BR = 38.4 kb/s**<br>FDA = 62.5 kHz, BR = 250 kb/s*** | -<br>-<br>-<br>-<br>- | -123<br>-119<br>-109<br>-110<br>-97 | -<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_F_HF | Direct tie of RFI and RFO pins, shared Rx, Tx paths FSK sensitivity, highest LNA gain.<br>Higher frequency bands | FDA = 5 kHz, BR = 1.2 kb/s<br>FDA = 5 kHz, BR = 4.8 kb/s<br>FDA = 40 kHz, BR = 38.4 kb/s*<br>FDA = 20 kHz, BR = 38.4 kb/s**<br>FDA = 62.5 kHz, BR = 250 kb/s*** | -<br>-<br>-<br>-<br>- | -119<br>-115<br>-105<br>-105<br>-92 | -<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| | Split RF paths, LnaBoost is turned on, the RF switch insertion loss is not accounted for.<br>Higher frequency bands | FDA = 5 kHz, BR = 1.2 kb/s<br>FDA = 5 kHz, BR = 4.8 kb/s<br>FDA = 40 kHz, BR = 38.4 kb/s*<br>FDA = 20 kHz, BR = 38.4 kb/s**<br>FDA = 62.5 kHz, BR = 250 kb/s*** | -<br>-<br>-<br>-<br>- | -123<br>-119<br>-109<br>-109<br>-96 | -<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_O | OOK sensitivity, highest LNA gain shared Rx, Tx paths | BR = 4.8 kb/s<br>BR = 32 kb/s | -<br>- | -117<br>-108 | -<br>- | dBm<br>dBm |
| CCR | Co-Channel Rejection | | - | -9 | - | dB |

| | | | | | | |
|---|---|---|---|---|---|---|
| ACR | Adjacent Channel Rejection | FDA = 5 kHz, BR=4.8kb/s<br>Offset = +/- 25 kHz or +/- 50kHz<br>169MHz Band<br>434 MHz Band<br>8-900 MHz Band | <br><br>-<br>-<br>- | <br><br>59<br>56<br>50 | <br><br>-<br>-<br>- | <br><br>dB<br>dB<br>dB |
| BI_HF | Blocking Immunity, higher bands | Offset = +/- 1 MHz<br>Offset = +/- 2 MHz<br>Offset = +/- 10 MHz | -<br>-<br>- | 71<br>76<br>84 | -<br>-<br>- | dB<br>dB<br>dB |
| BI_LF | Blocking Immunity, lower bands | Offset = +/- 1 MHz<br>Offset = +/- 2 MHz<br>Offset = +/- 10 MHz | -<br>-<br>- | 71<br>72<br>78 | -<br>-<br>- | dB<br>dB<br>dB |
| IIP2 | 2nd order Input Intercept Point<br>Unwanted tones are 20 MHz above the LO | Highest LNA gain | - | +55 | - | dBm |
| IIP3_HF | 3rd order Input Intercept point<br>Unwanted tones are 1MHz and 1.995 MHz above the LO | Higher bands<br>Highest LNA gain G1<br>LNA gain G2, 4dB sensitivity hit | <br>-<br>- | <br>-12.5<br>-8.5 | <br>-<br>- | <br>dBm<br>dBm |
| IIP3_LF | 3rd order Input Intercept point<br>Unwanted tones are 1MHz and 1.995 MHz above the LO | Lower bands<br>Highest LNA gain G1<br>LNA gain G2, 2.5dB sensitivity hit | <br>-<br>- | <br>-22<br>-16 | <br>-<br>- | <br>dBm<br>dBm |
| BW_SSB | Single Side channel filter BW | Programmable | 2.7 | - | 250 | kHz |
| IMR | Image Rejection | Wanted signal 3dB over sensitivity<br>BER=0.1% | - | 48 | - | dB |
| IMA | Image Attenuation | | - | 57 | - | dB |
| DR_RSSI | RSSI Dynamic Range | AGC enabled Min<br>Max | -<br>- | -127<br>0 | -<br>- | dBm<br>dBm |

* *RxBw = 83 kHz (Single Side Bandwidth)*

** *RxBw = 50 kHz (Single Side Bandwidth)*

*** *RxBw = 250 kHz (Single Side Bandwidth)*

### 2.4.4. FSK/OOK Mode Transmitter

*Table 8 Transmitter Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| RF_OP | RF output power in 50 ohms on RFO pin (High efficiency PA). | Programmable with steps<br>Max<br>Min | <br>+11<br>- | <br>+14<br>-1 | <br>-<br>- | <br>dBm<br>dBm |
| ΔRF_OP_V | RF output power stability on RFO pin versus voltage supply. | VDD = 2.5 V to 3.3 V<br>VDD = 1.8 V to 3.7 V | -<br>- | 3<br>8 | -<br>- | dB<br>dB |
| RF_OPH | RF output power in 50 ohms, on PA_BOOST pin (Regulated PA). | Programmable with 1dB steps Max<br>Min | -<br>- | +17<br>+2 | -<br>- | dBm<br>dBm |

| | | | | | | |
|---|---|---|---|---|---|---|
| RF_OPH_MAX | Max RF output power, on PA_BOOST pin | High power mode | - | +20 | - | dBm |
| ΔRF_OPH_V | RF output power stability on PA_-BOOST pin versus voltage supply. | VDD = 2.4 V to 3.7 V | - | +/-1 | - | dB |
| ΔRF_T | RF output power stability versus temperature on PA_BOOST pin. | From T = -40 °C to +85 °C | - | +/-1 | - | dB |
| PHN | Transmitter Phase Noise | 169 MHz band | | | | |
| | | 10kHz Offset | - | -118 | - | dBc/Hz |
| | | 50kHz Offset | - | -118 | - | |
| | | 400kHz Offset | - | -128 | - | |
| | | 1MHz Offset | - | -134 | - | |
| | | 433 MHz band | | | | |
| | | 10kHz Offset | - | -110 | - | dBc/Hz |
| | | 50kHz Offset | - | -110 | - | |
| | | 400kHz Offset | - | -122 | - | |
| | | 1MHz Offset | - | -129 | - | |
| | | 868/915 MHz band | | | | |
| | | 10kHz Offset | - | -103 | - | dBc/Hz |
| | | 50kHz Offset | - | -103 | - | |
| | | 400kHz Offset | - | -115 | - | |
| | | 1MHz Offset | - | -122 | - | |
| ACP | Transmitter adjacent channel power (measured at 25 kHz offset) | BT=1. Measurement conditions as defined by EN 300 220-1 V2.3.1 | - | - | -37 | dBm |
| TS_TR | Transmitter wake up time, to the first rising edge of DCLK | Frequency Synthesizer enabled, *PaRamp* = 10us, BR = 4.8 kb/s | - | 120 | - | us |

### 2.4.5. Electrical specification for Lora™ modulation

The table below gives the electrical specifications for the transceiver operating with Lora™ modulation. Following conditions apply unless otherwise specified:

- Supply voltage = 3.3 V.
- Temperature = 25° C.
- $f_{XOSC}$ = 32 MHz.
- Lower bands: 169 MHz and 433 MHz, higher bands: 868 and 915 MHz
- bandwidth (BW) = 125 kHz.
- Spreading Factor (SF) = 12.
- Error Correction Code (EC) = 4/6.
- Packet Error Rate (PER)= 1%
- CRC on payload enabled.
- Output power = 13 dBm in transmission.
- Payload length = 64 bytes.
- Preamble Length = 12 symbols (programmed register *PreambleLength=8*)
- With matched impedances

*Table 9 LoRa Receiver Specification.*

| Symbol | Description | Conditions | Min. | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| IDDR_L | Supply current in receiver Lora™ mode, LnaBoost off | Lower Bands, BW = 125 kHz<br>Lower Bands, BW = 250 kHz<br>Lower Bands, BW = 500 kHz | -<br>-<br>- | 11.5<br>12.4<br>13.8 | -<br>-<br>- | mA<br>mA<br>mA |
| | | Higher Bands, BW = 125 kHz<br>Higher Bands, BW = 250 kHz<br>Higher Bands, BW = 500 kHz | -<br>-<br>- | 10.3<br>11.1<br>12.6 | -<br>-<br>- | mA<br>mA<br>mA |
| IDDT_L | Supply current in transmitter mode | RFOP = 13 dBm<br>RFOP = 7 dBm | -<br>- | 28<br>20 | -<br>- | mA<br>mA |
| IDDT_H_L | Supply current in transmitter mode with an external impedance transformation | Using PA_BOOST pin<br>RFOP = 17 dBm | - | 90 | - | mA |
| BI_L | Blocking immunity, FRF=868 MHz CW interferer | offset = +/- 1 MHz<br>offset = +/- 2 MHz<br>offset = +/- 10 MHz | - | 89<br>94<br>100 | | dB<br>dB<br>dB |
| IIP3_L_HF | 3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO | Band 1<br>Highest LNA gain G1<br>LNA gain G2, 5dB sensitivity hit | <br>-<br>- | <br>-11<br>-6 | <br>-<br>- | <br>dBm<br>dBm |
| IIP3_L_LF | 3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO | Band 2<br>Highest LNA gain G1<br>LNA gain G2,2.5dB sensitivity hit | <br>-<br>- | <br>-22<br>-15 | <br>-<br>- | <br>dBm<br>dBm |
| IIP2_L | 2nd order input intercept point, highest LNA gain, FRF=868 MHz, CW interferer. | F1 = FRF + 20 MHz<br>F2 = FRF+ 20 MHz + Δf | - | +55 | - | dBm |
| BR_L | Bit rate, Long-Range Mode | From SF6, BW=500kHz to SF12, BW=7.8kHz | 0.018 | - | 37.5 | kbps |
| RFS_L10_HF | RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 10.4 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 11 | -<br>-<br>-<br>- | -131<br>-134<br>-138<br>-146 | -<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm |
| RFS_L62_HF | RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 62.5 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -121<br>-126<br>-129<br>-132<br>-135<br>-137<br>-139 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L125_HF | RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 125 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -118<br>-123<br>-126<br>-129<br>-132<br>-133<br>-136 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |

| Symbol | Description | Conditions | Min. | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| RFS_L250_HF | RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 250 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -115<br>-120<br>-123<br>-125<br>-128<br>-130<br>-133 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L500_HF | RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 500 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -111<br>-116<br>-119<br>-122<br>-125<br>-128<br>-130 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L7.8_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2 or 3, using split Rx/Tx path 7.8 kHz bandwidth | SF = 12<br>SF = 11 | -<br>- | -148<br>-145 | -<br>- | dBm<br>dBm |
| RFS_L10_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2, 10.4 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8 | -<br>-<br>- | -132<br>-136<br>-138 | -<br>-<br>- | dBm<br>dBm<br>dBm |
| RFS_L62_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2, 62.5 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -123<br>-128<br>-131<br>-134<br>-135<br>-137<br>-140 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L125_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2, 125 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -121<br>-125<br>-128<br>-131<br>-134<br>-136<br>-137 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L250_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2 250 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -118<br>-122<br>-125<br>-128<br>-131<br>-133<br>-134 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |
| RFS_L500_LF | RF sensitivity, Long-Range Mode, highest LNA gain, Band 2 500 kHz bandwidth | SF = 6<br>SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>-<br>- | -112<br>-118<br>-121<br>-124<br>-127<br>-129<br>-130 | -<br>-<br>-<br>-<br>-<br>-<br>- | dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm<br>dBm |

| Symbol | Description | Conditions | Min. | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| CCR_LCW | Co-channel rejection<br>Single CW tone = Sens +6 dB<br>1% PER | SF = 7<br>SF = 8<br>SF = 9<br>SF = 10<br>SF = 11<br>SF = 12 | -<br>-<br>-<br>-<br>-<br>- | 5<br>9.5<br>12<br>14.4<br>17<br>19.5 | -<br>-<br>-<br>-<br>-<br>- | dB<br>dB<br>dB<br>dB<br>dB<br>dB |
| CCR_LL | Co-channel rejection | Interferer is a LoRa™ signal using same BW and same SF.<br>Pw = Sensitivity + 3 dB | | -6 | | dB |
| ACR_LCW | Adjacent channel rejection | Interferer is 1.5*BW_L from the wanted signal center frequency<br>1% PER, Single CW tone = Sens + 3 dB<br><br>SF = 7<br>SF = 12 | <br><br><br><br>-<br>- | <br><br><br><br>60<br>72 | <br><br><br><br>-<br>- | <br><br><br><br>dB<br>dB |
| IMR_LCW | Image rejection after calibration. | 1% PER, Single CW tone = Sens +3 dB | - | 66 | - | dB |
| FERR_L | Maximum tolerated frequency offset between transmitter and receiver, no sensitivity degradation, SF6 thru 12 | All BW, +/-25% of BW<br>The tighter limit applies (see below) | | +/-25% | | BW |
| | Maximum tolerated frequency offset between transmitter and receiver, no sensitivity degradation, SF10 thru 12 | SF = 12<br>SF = 11<br>SF = 10 | -50<br>-100<br>-200 | -<br>-<br>- | 50<br>100<br>200 | ppm<br>ppm<br>ppm |

### 2.4.6. Digital Specification

Conditions: Temp = 25° C, VDD = 3.3 V, FXOSC = 32 MHz, unless otherwise specified.

*Table 10 Digital Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IH}$ | Digital input level high | | 0.8 | - | - | VDD |
| $V_{IL}$ | Digital input level low | | - | - | 0.2 | VDD |
| $V_{OH}$ | Digital output level high | Imax = 1 mA | 0.9 | - | - | VDD |
| $V_{OL}$ | Digital output level low | Imax = -1 mA | - | - | 0.1 | VDD |
| $F_{SCK}$ | SCK frequency | | - | - | 10 | MHz |
| $t_{ch}$ | SCK high time | | 50 | - | - | ns |
| $t_{cl}$ | SCK low time | | 50 | - | - | ns |
| $t_{rise}$ | SCK rise time | | - | 5 | - | ns |
| $t_{fall}$ | SCK fall time | | - | 5 | - | ns |
| $t_{setup}$ | MOSI setup time | From MOSI change to SCK rising edge. | 30 | - | - | ns |
| $t_{hold}$ | MOSI hold time | From SCK rising edge to MOSI change. | 20 | - | - | ns |
| $t_{nsetup}$ | NSS setup time | From NSS falling edge to SCK rising edge. | 30 | - | - | ns |
| $t_{nhold}$ | NSS hold time | From SCK falling edge to NSS rising edge, normal mode. | 100 | - | - | ns |
| $t_{nhigh}$ | NSS high time between SPI accesses | | 20 | - | - | ns |
| T_DATA | DATA hold and setup time | | 250 | - | - | ns |

# 3. RFM95W/96W/98W Features

This section gives a high-level overview of the functionality of the RFM95W/96W/98W low-power, highly integrated transceiver. The following figure shows a simplified block diagram of the RFM95W/96W/98W.

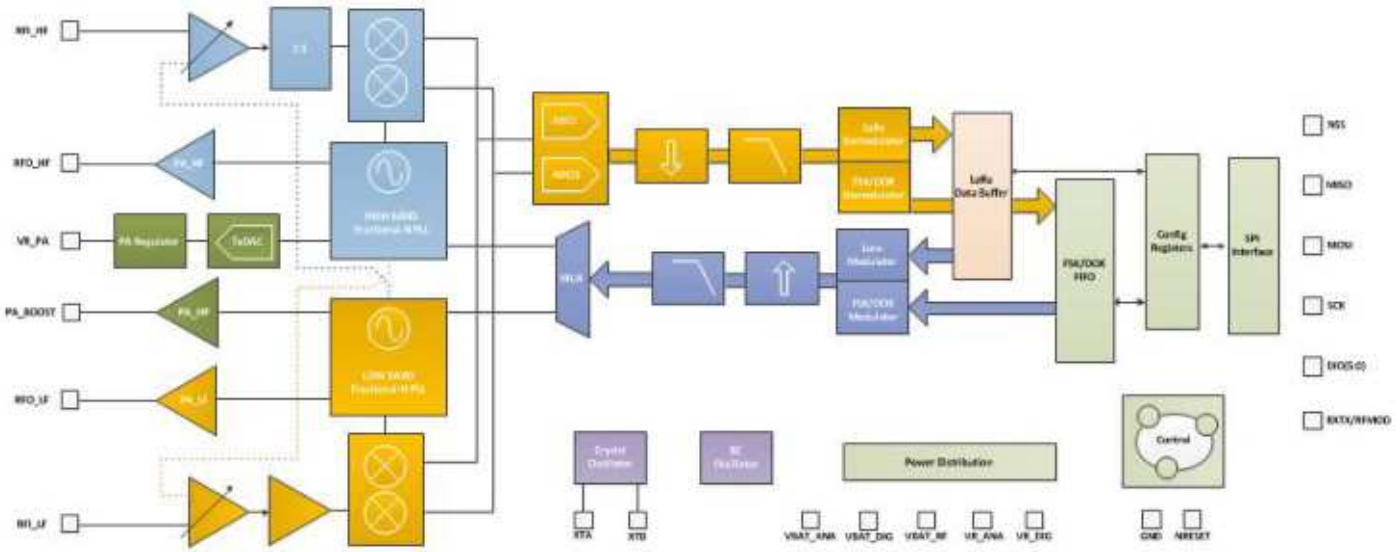


*Figure 3. RFM95W/96W/98W Block Schematic Diagram*

RFM95W/96W/98W is a half-duplex, low-IF transceiver. Here the received RF signal is first amplified by the LNA. The LNA inputs are single ended to minimise the external BoM and for ease of design. Following the LNA inputs, the conversion to differential is made to improve the second order linearity and harmonic rejection. The signal is then down-converted to in- phase and quadrature (I&Q) components at the intermediate frequency (IF) by the mixer stage. A pair of sigma delta ADCs then perform data conversion, with all subsequent signal processing and demodulation performed in the digital domain. The digital state machine also controls the automatic frequency correction (AFC), received signal strength indicator (RSSI) and automatic gain control (AGC). It also features the higher-level packet and protocol level functionality of the top level sequencer (TLS).

The frequency synthesisers generate the local oscillator (LO) frequency for both receiver and transmitter, one covering the lower UHF bands (up to 525 MHz), and the other one covering the upper UHF bands (from 860 MHz). The PLLs are optimized for user-transparent low lock time and fast auto-calibrating operation. In transmission, frequency modulation is performed digitally within the PLL bandwidth. The PLL also features optional pre-filtering of the bit stream to improve spectral purity.

RFM95W/96W/98W feature three distinct RF power amplifiers. Two of those, connected to RFO_LF and RFO_HF, can deliver up to +14 dBm, are unregulated for high power efficiency and can be connected directly to their respective RF receiver inputs via a pair of passive components to form a single antenna port high efficiency transceiver. The third PA, connected to the PA_BOOST pin and can deliver up to +20 dBm via a dedicated matching network. Unlike the high efficiency PAs, this high- stability PA covers all frequency bands that the frequency synthesizer addresses.

RFM95W/96W/98W also include two timing references, an RC oscillator and a 32 MHz crystal oscillator.

All major parameters of the RF front end and digital state machine are fully configurable via an SPI interface which gives access to RFM95W/96W/98W's configuration registers. This includes a mode auto sequencer that oversees the transition and calibration of the RFM95W/96W/98W between intermediate modes of operation in the fastest time possible.

The RFM95W/96W/98W are equipped with both standard FSK and long range spread spectrum (LoRa™) modems. Depending upon the mode selected either conventional OOK or FSK modulation may be employed or the LoRa™ spread spectrum modem.

## 3.1. LoRa™ Modem

The LoRa™ modem uses a proprietary spread spectrum modulation technique. This modulation, in contrast to legacy modulation techniques, permits an increase in link budget and increased immunity to in-band interference. At the same time the frequency tolerance requirement of the crystal reference oscillator is relaxed - allowing a performance increase for a reduction in system cost. For a fuller description of the design trade-offs and operation of the RFM95W/96W/98W please consult Section 4.1 of the datasheet.

## 3.2. FSK/OOK Modem

In FSK/OOK mode the RFM95W/96W/98W supports standard modulation techniques including OOK, FSK, GFSK, MSK and GMSK. The RFM95W/96W/98W is especially suited to narrow band communication thanks the low-IF architecture employed and the built-in AFC functionality. For full information on the FSK/OOK modem please consult Section 4.2 of this document.

# 4. RFM95W/96W/98W Digital Electronics

## 4.1. The LoRa™ Modem

The LoRa™ modem uses spread spectrum modulation and forward error correction techniques to increase the range and robustness of radio communication links compared to traditional FSK or OOK based modulation. Examples of the performance improvement possible, for several possible settings, are summarised in the table below. Here the spreading factor and error correction rate are design variables that allow the designer to optimise the trade-off between occupied bandwidth, data rate, link budget improvement and immunity to interference.

*Table 11 Example **LoRa®** Modem Performances, 868MHz Band*

| Bandwidth (kHz) | Spreading Factor | Coding rate | Nominal Rb (bps) | Sensitivity indication (dBm) | Frequency Reference |
|---|---|---|---|---|---|
| 10.4 | 6 | 4/5 | 782 | -131 | TCXO |
| | 12 | 4/5 | 24 | -147 | |
| 20.8 | 6 | 4/5 | 1562 | -128 | |
| | 12 | 4/5 | 49 | -144 | |
| 62.5 | 6 | 4/5 | 4688 | -121 | XTAL |
| | 12 | 4/5 | 146 | -139 | |
| 125 | 6 | 4/5 | 9380 | -118 | |
| | 12 | 4/5 | 293 | -136 | |

For European operation the range of crystal tolerances acceptable for each sub-band (of the ERC 70-03) is given in the specifications table. For US based operation a frequency hopping mode is available that automates both the LoRa™ spread spectrum and frequency hopping spread spectrum processes.

Another important facet of the LoRa™ modem is its increased immunity to interference. The LoRa™ modem is capable of co-channel GMSK rejection of up to 25 dB. This immunity to interference permits the simple coexistence of LoRa™ modulated systems either in bands of heavy spectral usage or in hybrid communication networks that use LoRa™ to extend range when legacy modulation schemes fail.

### 4.1.1. Link Design Using the LoRa™ Modem

#### *4.1.1.1. Overview*

The LoRa™ modem is setup as shown in the following figure. This configuration permits the simple replacement of the FSK modem with the LoRa™ modem via the configuration register setting *RegOpMode*. This change can be performed on the fly (in Sleep operating mode) thus permitting the use of both standard FSK or OOK in conjunction with the long range capability. The LoRa™ modulation and demodulation process is proprietary, it uses a form of spread spectrum modulation combined with cyclic error correction coding. The combined influence of these two factors is an increase in link budget and enhanced immunity to interference.

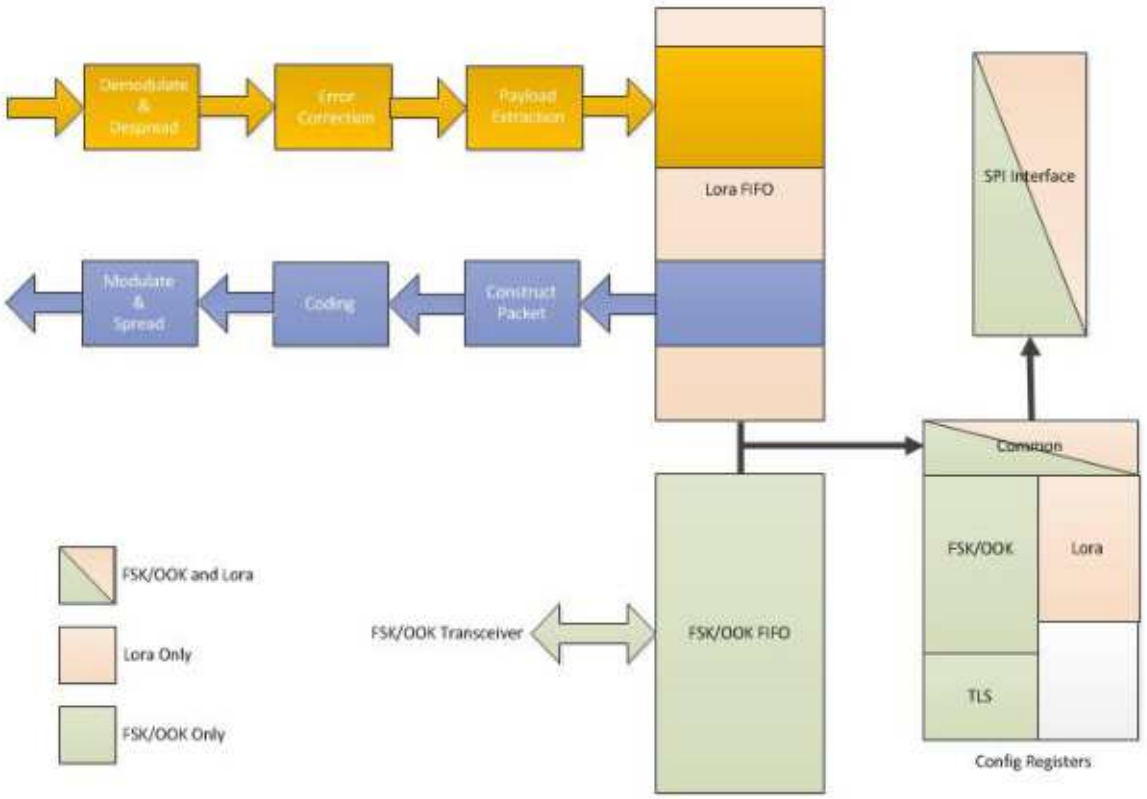



*Figure 4. LoRa™ Modem Connectivity*

A simplified outline of the transmit and receive processes is also shown above. Here we see that the LoRa™ modem has an independent dual port data buffer FIFO that is accessed through an SPI interface common to all modes. Upon selection of LoRa™ mode, the configuration register mapping of the RFM95W/96W/98W changes. For full details of this change please consult the register description of Section 6.

So that it is possible to optimise the LoRa™ modulation for a given application, access is given to the designer to three critical design parameters. Each one permitting a trade off between link budget, immunity to interference, spectral occupancy and nominal data rate. These parameters are spreading factor, modulation bandwidth and error coding rate.

#### *4.1.1.2. Spreading Factor*

The spread spectrum LoRa™ modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the spread information is sent is referred to as the symbol rate (Rs), the ratio between the nominal symbol rate and chip rate is the spreading factor and represents the number of symbols sent per bit of information. The range of values accessible with the LoRa™ modem are shown in the following table.

*Table 12 Range of Spreading Factors*

| ***SpreadingFactor*** **(RegModulationCfg)** | **Spreading Factor (Chips / symbol)** | **LoRa Demodulator SNR** |
|---|---|---|
| 6 | 64 | -5 dB |
| 7 | 128 | -7.5 dB |
| 8 | 256 | -10 dB |
| 9 | 512 | -12.5 dB |
| 10 | 1024 | -15 dB |
| 11 | 2048 | -17.5 dB |
| 12 | 4096 | -20 dB |

Note that the spreading factor, *SpreadingFactor*, must be known in advance on both transmit and receive sides of the link as different spreading factors are orthogonal to each other. Note also the resulting signal to noise ratio (SNR) required at the receiver input. It is the capability to receive signals with negative SNR that increases the sensitivity, so link budget and range, of the LoRa receiver.

**Spreading Factor 6**

SF = 6 Is a special use case for the highest data rate transmission possible with the LoRa modem. To this end several settings must be activated in the RFM95W/96W/98W registers when it is in use:

- Set *SpreadingFactor* = 6 in *RegModemConfig2*
- The header must be set to Implicit mode
- Write bits 2-0 of register address 0x31 to value "0b101"
- Write register address 0x37 to value 0x0C

#### *4.1.1.3. Coding Rate*

To further improve the robustness of the link the LoRa™ modem employs cyclic error coding to perform forward error detection and correction. Such error coding incurs a transmission overhead - the resultant additional data overhead per transmission is shown in the table below.

*Table 13 Cyclic Coding Overhead*

| ***CodingRate*** **(RegTxCfg1)** | **Cyclic Coding Rate** | **Overhead Ratio** |
|---|---|---|
| 1 | 4/5 | 1.25 |
| 2 | 4/6 | 1.5 |
| 3 | 4/7 | 1.75 |
| 4 | 4/8 | 2 |

Forward error correction is particularly efficient in improving the reliability of the link in the presence of interference. So that the coding rate (and so 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. Please consult Section 4.1.1.6 for more information on the LoRa™ packet and header.

#### *4.1.1.4. Signal Bandwidth*

An increase in signal bandwidth permits the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity improvement. There are of course regulatory constraints in most countries on the permissible occupied bandwidth. Contrary to the FSK modem which is described in terms of the single sideband bandwidth, the LoRa™ modem bandwidth refers to the double sideband bandwidth (or total channel bandwidth). The range of bandwidths relevant to most regulatory situations is given in the LoRa™ modem specifications table (see Section 2.4.5).

*Table 14 LoRa Bandwidth Options*

| Bandwidth (kHz) | Spreading Factor | Coding rate | Nominal Rb (bps) |
|---|---|---|---|
| 7.8 | 12 | 4/5 | 18 |
| 10.4 | 12 | 4/5 | 24 |
| 15.6 | 12 | 4/5 | 37 |
| 20.8 | 12 | 4/5 | 49 |
| 31.2 | 12 | 4/5 | 73 |
| 41.7 | 12 | 4/5 | 98 |
| 62.5 | 12 | 4/5 | 146 |
| 125 | 12 | 4/5 | 293 |
| 250 | 12 | 4/5 | 586 |
| 500 | 12 | 4/5 | 1172 |

*Note In the lower band (169 MHz), the 250 kHz and 500 kHz bandwidths are not supported.*

#### *4.1.1.5. LoRa™ Transmission Parameter Relationship*

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

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

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

#### *4.1.1.6. LoRa™ Packet Structure*

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 used in the packet. The packet format is shown in the following figure.

The LoRa™ packet comprises three elements:

- A preamble.
- An optional header.
- The data payload.

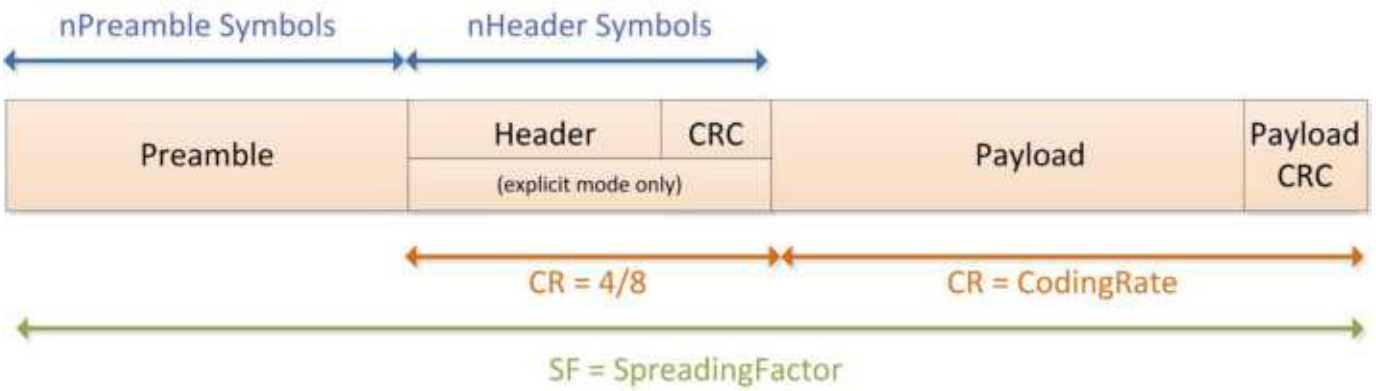



*Figure 5.* LoRa™ *Packet Structure*

**Preamble**

The preamble is used to synchronize receiver with the incoming data flow. By default the packet is configured with a 12 symbol long sequence. This is a programmable variable so the preamble length may be extended, for example in the interest of reducing to receiver duty cycle in receive intensive applications. However, the minimum length suffices for all communication. The transmitted preamble length may be changed by setting the register *PreambleLength* from 6 to 65535, yielding total preamble lengths of 6+4 to 65535+4 symbols, once the fixed overhead of the preamble data is considered. This permits the transmission of a near arbitrarily long preamble sequence.

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

**Header**

Depending upon the chosen mode of operation two types of header are available. The header type is selected by the *ImplictHeaderMode* bit found within the *RegSymbTimeoutMsb* register.

***Explicit Header Mode***

This is the default mode of operation. Here the header provides information on the payload, namely:

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

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

**Implicit Header Mode**

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

*Note* *With SF = 6 selected, implicit header mode is the only mode of operation possible.*

**Explicit Header Mode:**

In Explicit Header Mode, the presence of the CRC at the end of the payload in selected only on the transmitter side through the bit *RxPayloadCrcOn* in the register *RegModemConfig1*.

On the receiver side, the bit *RxPayloadCrcOn* in the register *RegModemConfig1* is not used and once the payload has been received, the user should check the bit *CrcOnPayload* in the register *RegHopChannel.* If the bit *CrcOnPayload* is at '1', the user should then check the Irq Flag *PayloadCrcError* to make sure the CRC is valid.

If the bit *CrcOnPayload* is at '0', it means there was no CRC on the payload and thus the IRQ Flag *PayloadCrcError* will not be trigged even if the payload has errors.

| Explicit Header | Transmitter | Receiver | CRC Status |
|---|---|---|---|
| Value of the bit RxPayloadCrcOn | 0 | 0 | CRC is not checked |
| | 0 | 1 | CRC is not checked |
| | 1 | 0 | CRC is checked |
| | 1 | 1 | CRC is checked |

**Implicit Header Mode;**

In Implicit Header Mode, it is necessary to set the bit RxPayloadCrcOn in the register *RegModemConfig1* on both sides (TX and RX)

| Implicit Header | Transmitter | Receiver | CRC Status |
|---|---|---|---|
| Value of the bit RxPayloadCrcOn | 0 | 0 | CRC is not checked |
| | 0 | 1 | CRC is always wrong |
| | 1 | 0 | CRC is not checked |
| | 1 | 1 | CRC is checked |

**Low Data Rate Optimization**

Given the potentially long duration of the packet at high spreading factors the option is given to improve the robustness of the transmission to variations in frequency over the duration of the packet transmission and reception. The bit *LowDataRateOptimize* increases the robustness of the LoRa link at these low effective data rates. Its use is mandated when the symbol duration exceeds 16ms. Note that both the transmitter and the receiver must have the same setting for *LowDataRateOptimize*.

**Payload**

The packet payload is a variable-length field that contains the actual data coded at the error rate either as specified in the header in explicit mode or in the register settings in implicit mode. An optional CRC may be appended. For more information on the payload and how it is loaded from the data buffer FIFO please see Section 4.1.2.3.

#### 4.1.1.7. Time on air

For a given combination of spreading factor (SF), coding rate (CR) and signal bandwidth (BW) the total on-the-air transmission time of a LoRa™ packet can be calculated as follows. From the definition of the symbol rate it is convenient to define the symbol rate:

$$Ts = \frac{1}{Rs}$$

The LoRa packet duration is the sum of the duration of the preamble and the transmitted packet. The preamble length is calculated as follows:

$$T_{preamble} = (n_{preamble} + 4.25)T_{sym}$$

where $n_{preamble}$ is the programmed preamble length, *PreambleLength*.The payload duration depends upon the header mode that is enabled. The following formulae give the payload duration in implicit (headerless) and explicit (with header) modes.

$$n_{payload} = 8 + max\left(ceil\left[\frac{(8PL - 4SF + 28 + 16CRC - 20IH)}{4(SF - 2DE)}\right](CR + 4), 0\right)$$

With the following dependencies:

- PL is the number of Payload bytes (1 to 255)
- SF is the spreading factor (6 to 12)
- IH=0 when the header is enabled, IH=1 when no header is present
- DE=1 when *LowDataRateOptimize*=1, DE=0 otherwise
- CR is the coding rate (1 corresponding to 4/5, 4 to 4/8)

The Payload duration is then the symbol period multiplied by the number of Payload symbols

$$T_{payload} = n_{payload} \times T_s$$

The time on air, or packet duration, in simply then the sum of the preamble and payload duration.

$$T_{packet} = T_{preamble} + T_{payload}$$

#### 4.1.1.8. Frequency Hopping with LoRa™

Frequency hopping spread spectrum (FHSS) is typically employed when the duration of a single packet could exceed regulatory requirements relating to the maximum permittable channel dwell time. This is most notably the case in US operation where the 902 to 928 MHz ISM band can be used ina frequency hopping mode. To ease implementation of FHSS systems the frequency hopping mode of the LoRa™ modem can be enabled (see *FhssMode* of register *RegTxCfg1*).

**Principle of Operation**

The principle behind the FHSS scheme is that a portion of each LoRa™ packet is transmitted on each hopping channel from a look up table of frequencies managed by the host microcontroller. After a predetermined hopping period the transmitter and receiver change to the next channel in a predefined list of hopping frequencies to continue transmission and reception of the next portion of the packet. The time which the transmission will dwell in any given channel is determined by *HoppingPeriod* which is an integer multiple of symbol periods:

$$HoppingPeriod = Ts \times FreqHoppingPeriod$$

The frequency hopping transmission and reception process starts at channel 0. The preamble and header are transmitted first on channel 0. At the beginning of each transmission the interrupt the channel counter *FhssPresentChannel* is incremented and the interrupt signal *FhssChangeChannel* is generated. The new frequency must then be programmed within the hopping period to ensure it is taken into account for the next hop, the interrupt *FhssChangeChannel* is then to be cleared by writing a logical '1'.

FHSS Reception always starts on channel 0. The receiver waits for a valid preamble detection before starting the frequency hopping process as described above. Note that in the eventuality of header CRC corruption, the receiver will automatically request channel 0 and recommence the valid preamble detection process.

**Timing of Channel Updates**

The interrupt requesting the channel change, *FhssChannelChange,* is generated upon transition to the new frequency. The frequency hopping process is recapitulated in the diagram below:

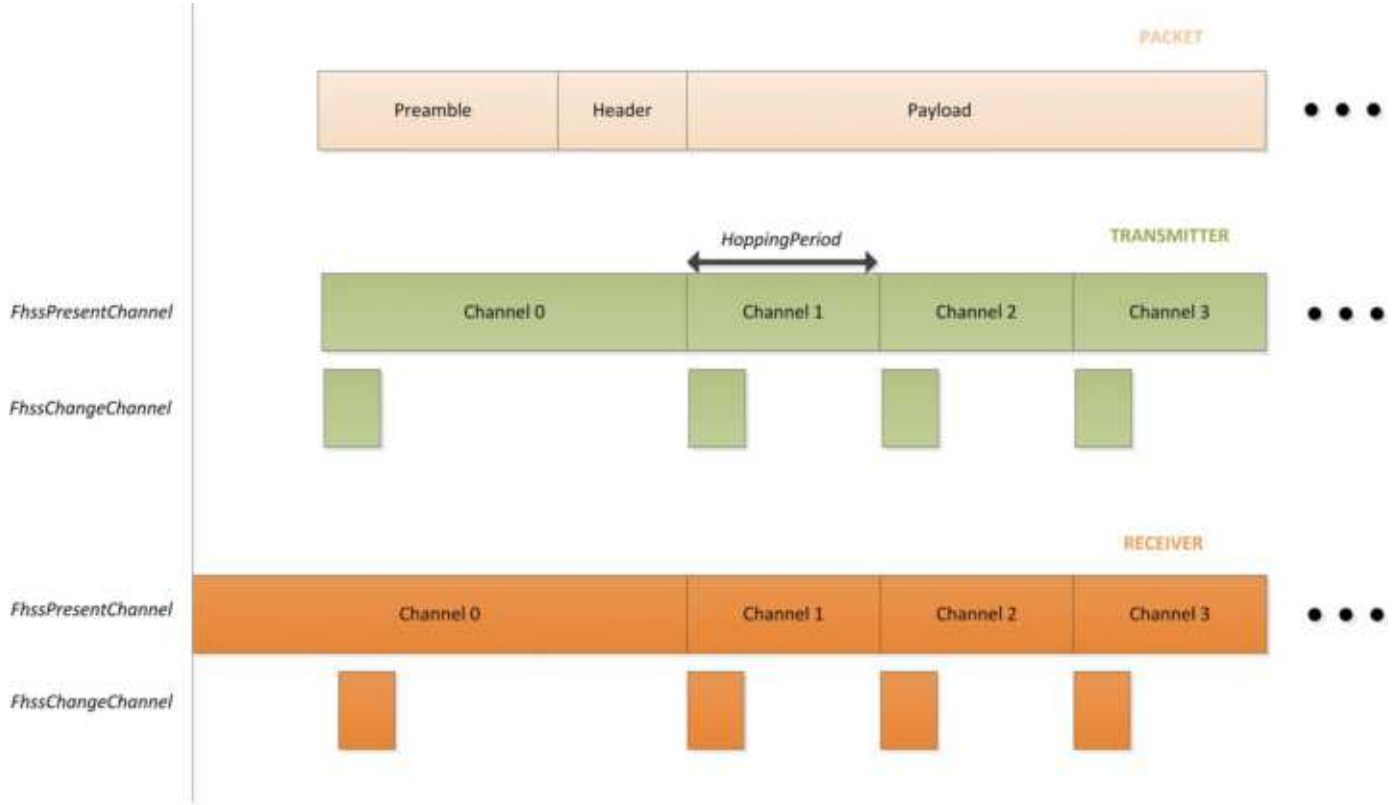



*Figure 6. Interrupts generated in the case of successful frequency hopping communication.*

### 4.1.2. LoRa™ Digital Interface

The LoRa™ modem comprises three types of digital interface, static configuration registers, status registers and a FIFO data buffer. All are accessed through the RFM95W/96W/98W's SPI interface - full details of each type of register are given below. Full listings of the register addresses used for SPI access are given in Section 6.4.

#### *4.1.2.1. LoRa™ Configuration Registers*

Configuration registers are accessed through the SPI interface. Registers are readable in all device mode including Sleep. However, they should be **written only in Sleep and Stand-by modes**. Please note that **the automatic top level sequencer (TLS modes) are not available in LoRa™ mode** and **the configuration register mapping changes as shown in Table 39.** The content of the LoRa™ configuration registers is retained in FSK/OOK mode. For the functionality of mode registers common to both FSK/OOK and LoRa™ mode, please consult the Analog and RF Front End section of this document (Section 5).

#### *4.1.2.2. Status Registers*

Status registers provide status information during receiver operation.

#### *4.1.2.3. LoRa™ Mode FIFO Data Buffer*

**Overview**

The RFM95W/96W/98W is equipped with a 256 byte RAM data buffer which is uniquely accessible in LoRa mode. This RAM area, thereafter reffered to as the FIFO Data buffer, is fully customizable by the user and allows access to the received, or to be transmitted, data. All access to the LoRa™ FIFO data buffer is done via the SPI interface. A diagram of

the user defined memory mapping of the FIFO data buffer is shown below. These FIFO data buffer can be read in all operating modes except sleep and store data related to the last receive operation performed. It is automatically cleared of old content upon each new transition to receive mode.

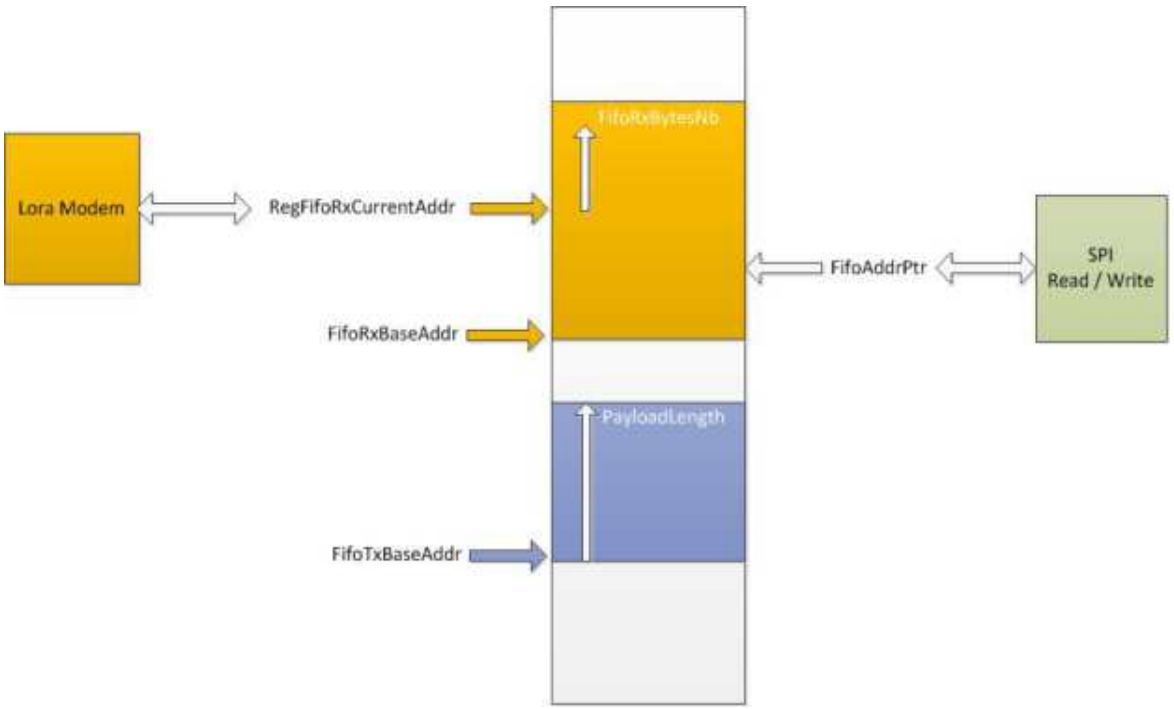



*Figure 7.* LoRa™ *data buffer*

**Principle of Operation**

Thanks to its dual port configuration, it is possible to simultaneously store both transmit and receive information in the FIFO data buffer. The register *FifoTxBaseAddr* specifies the point in memory where the transmit information is stored. Similarly, for receiver operation, the register *FifoRxBaseAddr* indicates the point in the data buffer where information will be written to in event of a receive operation.

By default, the device is configured at power-up so that half of the available memory is dedicated to Rx (*FifoRxBaseAddr* initialized at address 0x00) and the other half is dedicated for Tx (*FifoTxBaseAddr* initialized at address 0x80).

However, due to the contiguous nature of the FIFO 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 FIFO. To exploit the maximum FIFO data buffer size in transmit or receive mode, the whole FIFO data buffer can be used in each mode by setting the base addresses *FifoTxBaseAddr* and *FifoRxBaseAddr* at the bottom of the memory (0x00).

The FIFO data buffer is cleared when the device is put in SLEEP mode, consequently no access to the FIFO data buffer is possible in sleep mode. However, the data in the FIFO data buffer are retained when switching across the other LoRa modes of operation, so that a received packet can be retransmitted with minimum data handling on the controller side. The FIFO data buffer is not self-clearing (unless if the device is put in sleep mode) and the data will only be “erased” when a new set of data is written into the occupied memory location.

The actual location to be read from, or written to, over the SPI interface is defined by the address pointer *FifoAddrPtr*. Before any read or write operation it is hence necessary to initialise this pointer to the corresponding base value. Upon reading or writing to the FIFO data buffer (*RegFifo*) the address pointer will then increment automatically.

The register *FifoRxBytesNb* defines the size of the memory location to be written in the event of a successful receive operation. On the other hand *PayloadLength* indicates the size of the memory location to be transmitted. In implicit header mode, the *FifoRxBytesNb* is not used as the number of payload bytes is known. Otherwise, in explicit header mode, the initial size of the receive buffer is set to the packet length in the received header. The variable *FifoRxCurrentAddr* indicates the location of the last packet received in the FIFO so that the last packet received can be easily read by pointing the *FifoAddrPtr* to this register.

It is important to notice that all the received data will be written to the FIFO data buffer even if the CRC is invalid. This allows for post-processing of received data for debug purposes for instance. It is also imporant 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.

### 4.1.3. Operation of the LoRa™ Modem

#### *4.1.3.1. Operating Mode Control*

The operating modes of the LoRa™ modem are accessed by enabling LoRa™ mode (setting the *LongRangeMode* bit of *RegOpMode*). Depending upon the operating mode selected the range of functionality and register access is given by the following table:

*Table 15 LoRa™ Operating Mode Functionality*

| Operating Mode | Description |
|---|---|
| **SLEEP** | Low-power mode. In this mode only SPI and configuration registers are accessible. Lora FIFO is not accessible.<br>Note that this is the only mode permissible to switch between FSK/OOK mode and LoRa mode. |
| **STAND-BY** | both Crystal oscillator and Lora baseband blocks are turned on.RF part and PLLs are disabled |
| **FSTX** | This is a frequency synthesis mode for transmission. The PLL selected for transmission is locked and active at the transmit frequency. The RF part is off. |
| **FSRX** | This is a frequency synthesis mode for reception. The PLL selected for reception is locked and active at the receive frequency. The RF part is off. |
| **TX** | When activated the RFM95W/96W/98W powers all remaining blocks required for transmit, ramps the PA, transmits the packet and returns to Stand-by mode. |
| **RXCONTINUOUS** | When activated the RFM95W/96W/98W powers all remaining blocks required for reception, processing all received data until a new user request is made to change operating mode. |
| **RXSINGLE** | When activated the RFM95W/96W/98W powers all remaining blocks required for reception, remains in this state until a valid packet has been received and then returns to Stand-by mode. |
| **CAD** | When in CAD mode, the device will check a given channel to detect LoRa preamble signal |

It is possible to access any mode from any other mode by changing the value in the *RegOpMode* register.

### 4.1.4. Frequency Settings

Recalling that the frequency step is given by:

$$F_{STEP} = \frac{F_{XOSC}}{2^{19}}$$

In order to set LO frequency values following registers are available.

Frf is a 24-bit register which defines carrier frequency. The carrier frequency relates to the register contents by following formula:

$$F_{RF} = F_{STEP} \times Frf(23,0)$$

### 4.1.5 Frequency Error Indication

The RFM95W/96W/98W derives its RF center frequency from a crystal reference oscillator which has a finite frequency precision. Errors in reference frequency will manifest themselves as errors of the same proportion from the RF center frequency.

In LoRa receive mode the RFM95W/96W/98W is capable of measuring the frequency offset between the receiver centre frequency and that of an incoming LoRa signal. The modem is intolerant of frequency offsets in the region of +/- 25% of the bandwidth and will accurately report the error over this same range.

In LoRa mode the frequency error, LoRaFeiValue, indicator is a 2's compliment 20-bit value accessible from registers 0x28 to 0x2A. To convert this value to a frequency error in Hertz, the following conversion should be applied:

$$F_{ErrHz} = \frac{LoRaFeiValue \times 2^{24}}{32 \times 10^{6}} \times \frac{BW}{500}$$

Where *BW* is the LoRa modem bandwidth in kHz. It can also be useful to express this same quantity as the equivalent PPM (parts per million) of frequency error.

$$F_{Errppm} = F_{Err-Hz} \times \frac{10^{6}}{F_{RF}}$$

Where $F_{RF}$ is the programmed RF centre frequency of the RFM95W/96W/98W at the time the FEI measurement was made.

### 4.1.6 LoRa AFC

To use the *LoRaFeiValue* information to correct a frequency offset and perform automatic frequency correction two additional steps are necessary:

1) Retune the RF centre frequency This is done by simply reprogramming to the following corrected centre frequency:

$$F_{RF_{new}} = F_{RF} - \frac{F_{RF}}{10^{6}} \times F_{Errppm}$$

2) In addition to this the LoRa modem nust be made aware of the subtle difference in data rate caused by the shift in frequency error. This is done by first scaling the frequency error:

$$offset = 0.95 \times F_{Errppm}$$

The scaled is then converted to an 8 bit 2's compliment number which is written to register 0x27.

### 4.1.7 LoRa™ Modem State Machine Sequences

The sequence for transmission and reception of data to and from the LoRa™ modem, together with flow charts of typical sequences of operation, are detailed below.

**Data Transmission Sequence**

In transmit mode power consumption is optimized by enabling RF, PLL and PA blocks only when packet data needs to be transmitted. Figure 8 shows a typical LoRa™ transmit sequence.

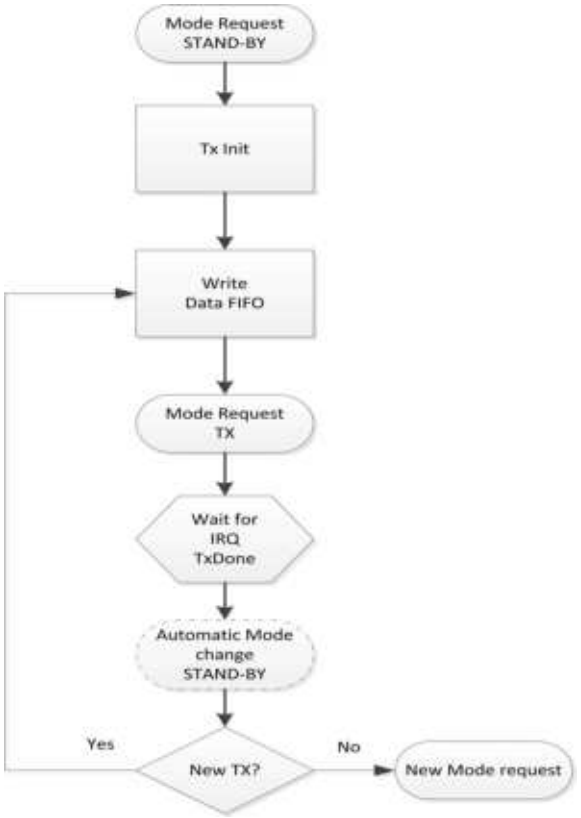



*Figure 8.* LoRa™ *modulation transmission sequence.*

- Static configuration registers can only be accessed in Sleep mode, Stand-by mode or FSTX mode.
- The LoRa™ FIFO can only be filled in Stand-by mode.
- Data transmission is initiated by sending TX mode request.
- Upon completion the *TxDone* interrupt is issued and the radio returns to Stand-by mode.
- Following transmission the radio can be manually placed in Sleep mode or the FIFO refilled for a subsequent Tx operation.

### LoRa™ Transmit Data FIFO Filling

In order to write packet data into FIFO user should:

1 Set *FifoPtrAddr* to *FifoTxPtrBase*.

2 Write *PayloadLength* bytes to the FIFO (*RegFifo*)

### Data Reception Sequence

Figure 9 shows typical LoRa™ receive sequences for both single and continuous receiver modes of operation.

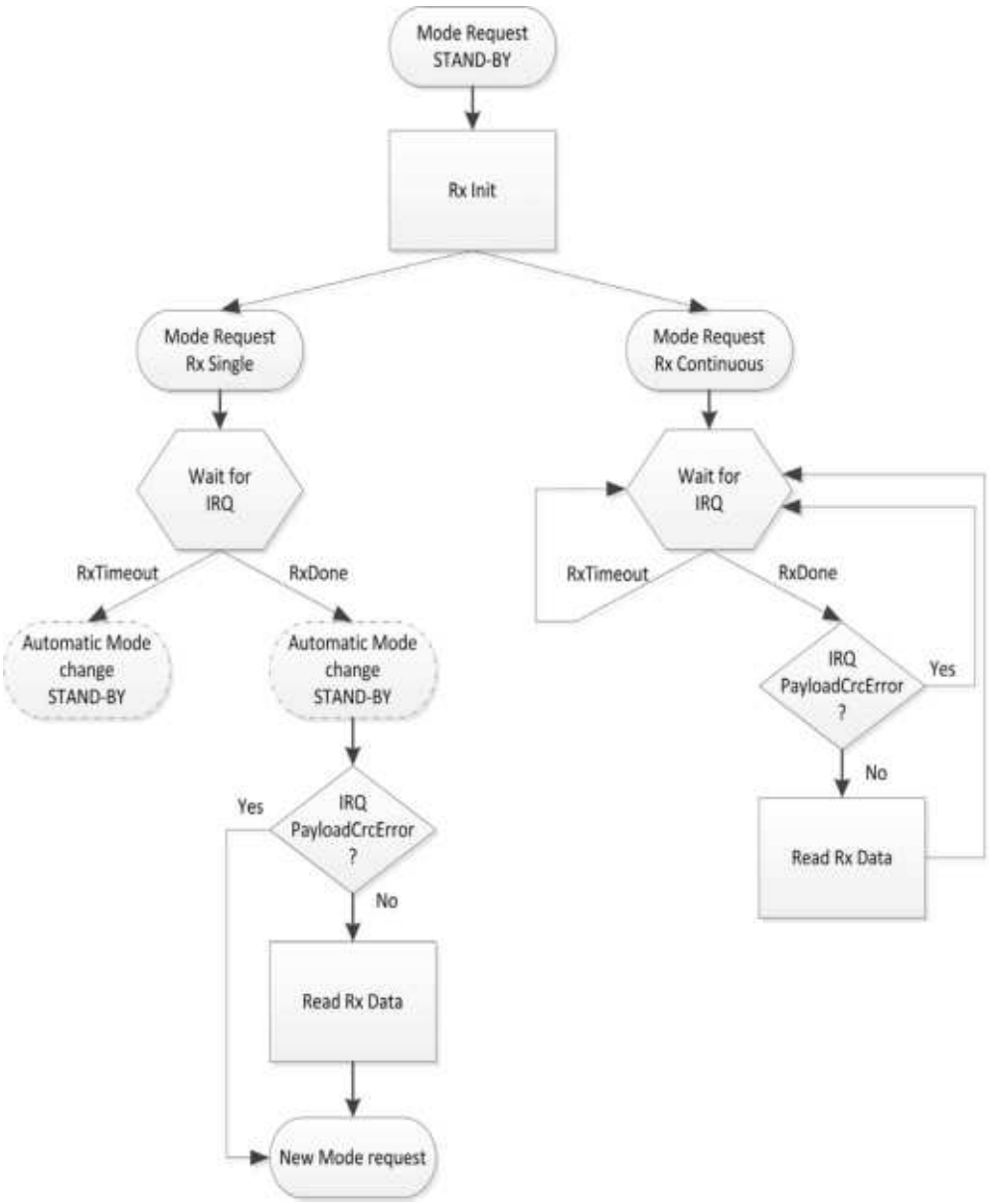



*Figure 9. LoRa™ receive sequence.*

The LORA receive modem can work in two distinct mode

1. Single receive mode
2. Continuous receive mode

Those two modes correspond to different use cases and it is important to understand the subtle differences between them.

***Single Reception Operating Mode***

In this mode, the modem searches for a preamble during a given time window. If a preamble hasn't been found at the end of the time window, the chip generates the RxTimeout interrupt and goes back to stand-by mode . The length of the window (in symbols) is defined by the RegSymbTimeout register and should be in the range of 4 (minimum time for the modem to acquire lock on a preamble) up to 1023 symbols. (The default value being 5). If no preamble is detected during this window the RxTimeout interrupt is generated and the radio goes back to stand-by mode.

At the end of the payload, the RxDone interrupt is generated together with the interrupt PayloadCrcError if the payload CRC is not valid. However, even when the CRC is not valid, the data are written in the FIFO data buffer for post processing. Following the RxDone interrupt the radio goes to stand-by mode.

The modem will also automatically return in stand-by mode when the interrupts RxDone or RxTimeout are generated. Therefore, this mode should only be used when the time window of arrival of the packet is known . In other cases, the RX continuous mode should be used.

In Rx single mode low-power is achieved by turning off PLL and RF blocks as soon as a packet has been received. The flow is as follows:

1 *Set FifoPtrAddr* to *FifoRxPtrBase*.

2 Static configuration register device can be written in either Sleep mode, Stand-by mode or FSRX mode.

3 A single packet receive operation is initiated by selecting the operating mode RXSINGLE.

4 The receiver will then await the reception of a valid preamble. Once received, the gain of the receive chain is set. Following the ensuing reception of a valid header, indicated by the ValidHeader interrupt in explicit mode. The packet reception process commences. Once the reception process is complete the RxDone interrupt is set. The radio then returns automatically to Stand-by mode to reduce power consumption.

5 The receiver status register PayloadCrcError should be checked for packet payload integrity.

6 If a valid packet payload has been received then the FIFO should be read (See Payload Data Extraction below). Should a subsequent single packet reception need to be triggered, then the RXSINGLE operating mode must be re-selected to launch the receive process again - taking care to reset the SPI pointer (FifoPtrAddr) to the base location in memory (FifoRxPtrBase).

***Continuous Reception Operating Mode***

In continuous receive mode the modem scans the channel continuously for a preamble. Each time a preamble is detected the modem detects and tracks it until the packet is received and then carries on waiting for the next preamble.

If the preamble length exceeds the anticipated value set by the registers RegPreambleMsb and RegPreambleLsb (measured in symbol unit), the preamble will be dropped and the search for a preamble restarted. However, this scenario will not be flagged by an interrupt. In continous RX mode, opposite to the single RX mode, when a timeout interrupt is generated, the device will not go in standby mode. In this case, the user must simply clear the interrupt while the device carry on waiting for a valid preamble.

It is also important to note that the demodulated bytes are written in the data buffer memory in the order received. Meaning, the first byte of a new packet is written just after the last byte of the preceding packet. The RX modem address pointer is never reseted as long as this mode is enabled. It is therefore necessary for the controller to handle the address pointer to make sure the FIFO data buffer is never full.

In continuous mode the received packet processing sequence is given below.

1 Whilst in Sleep or Stand-by mode select RXCONT mode.
2 Upon reception of a valid header CRC the *RxDone i*nterrupt is set. The radio remains in RXCONT mode waiting for the next RX LoRa™ packet.
3 The *PayloadCrcError* flag should be checked for packet integrity.
4 If packet has been correctly received the FIFO data buffer can be read (see below).
5 The reception process (steps 2 - 4) can be repeated or receiver operating mode exited as desired.

In continuous mode status information are available only for the last packet received, i.e. the corresponding registers should be read before the next RxDone arrives.

**Payload Data Extraction from FIFO**

In order to retrieve received data from FIFO the user must ensure that *ValidHeader*, *PayloadCrcError, RxDone* and *RxTimeout* interrupts in the status register RegIrqFlags are not asserted to ensure that packet reception has terminated successfully (i.e. no flags should be set).

In case of errors the steps below should be skipped and the packet discarded. In order to retrieve valid received data from the FIFO the user must:

- *FifoNbRxBytes* Indicates the number of bytes that have been received thus far.
- *RegRxDataAddr* Is a dynamic pointer that indicates precisely where the Lora modem received data has been written up to.
- Set *FifoPtrAddr* to *FifoRxCurrentAddr*. This sets the FIFO pointer to the the location of the last packet received in the FIFO. The payload can then be extracted by reading the *RegFifo* address *RegNbRxBytes* times. Alternatively, it is possible to manually point to the location of the last packet received from the start of the current packet by setting *FifoPtrAddr* to *RegRxDataAddr* - *FifoNbRxBytes*. In the same way, packet bytes can then be extracted from FIFO by reading the *RegFifo* address *RegNbRxBytes* times.

**Packet Filtering based on Preamble Start**

The LoRa modem does automatically filter received packets based upon any addressing. However the RFM95W/96W/98W permit software filtering of the received packets based on the contents of the first few bytes of payload. A brief example is given below for a 4 byte address, however, the address length can be selected by the designer.

The objective of the packet filtering process is to determine the presence, or otherwise, of a valid packet designed for the receiver. If the packet is not for the receiver then the radio returns to sleep mode in order to improve battery life.

The software packet filtering process follows the steps below:

- Each time the RxDone interrupt is received, latch the RegFifoRxByteAddr[7:0] register content in a variable , this variable will be called start_address. The RegFifoRxByteAddr[7:0] register of the RFM95W/96W/98W gives in real time the address of the last byte written in the data buffer + 1 (or the address at which the next byte will be written) by the receive LoRa modem . So by doing this , we make sure that the variable start_address always contains the start address of the next packet.

- Upon reception of the interrupt ValidHeader, start polling the RegFifoRxByteAddr[7:0] register until it begins to increment. The speed at which this register will increment depends on the Spreading factor, the error correction code and the modulation bandwidth. (Note that this interrupt is still generated in implicit mode).

- As soon as RegFifoRxByteAddr[7:0] >= start address + 4, the first 4 bytes (address) are stored in the FIFO data buffer. These can be read and tested to see if the packet is destined for the radio and either remaining in Rx mode to receive the packet or returning to sleep mode if not.

**Receiver Timeout Operation**

In either single or continuous LoRa™ reception modes, a receiver timeout functionality is available that permits the receiver to listen for a pre-determined period of time before generating an interrupt signal to indicate that no valid packets have been received. The timer is absolute and commences as soon as the radio is placed in either single or continuous receive mode. The interrupt itself, *RxTimeout*, can be found in the interrupt register *RegIrqFlags*. In Rx Single mode, the device will return in Stanby mode as soon as the interrupt occurs and the interrupt needs to be cleared before to return in Rx Single mode. In Rx Continuous mode, the interrupt will interrupt will simply be raised but the device will stay in Rx Continous mode. It is therefore the responsability on the controller to clear the interrupt while still in Rx Continuous mode. The programmed timeout value is expressed as a multiple of the symbol period and is given by:

$$TimeOut = LoraRxTimeout \cdot Ts$$

**Channel Activity Detection**

The use of a spread spectrum modulation technique presents challenges in determining whether the channel is already in use by a signal that may be below the noise floor of the receiver. The use of the RSSI in this situation would clearly be impracticable. To this end the channel activity detector is used to detect the presence of other LoRa™ signals. Figure 10 shows the channel activity detection (CAD) process:

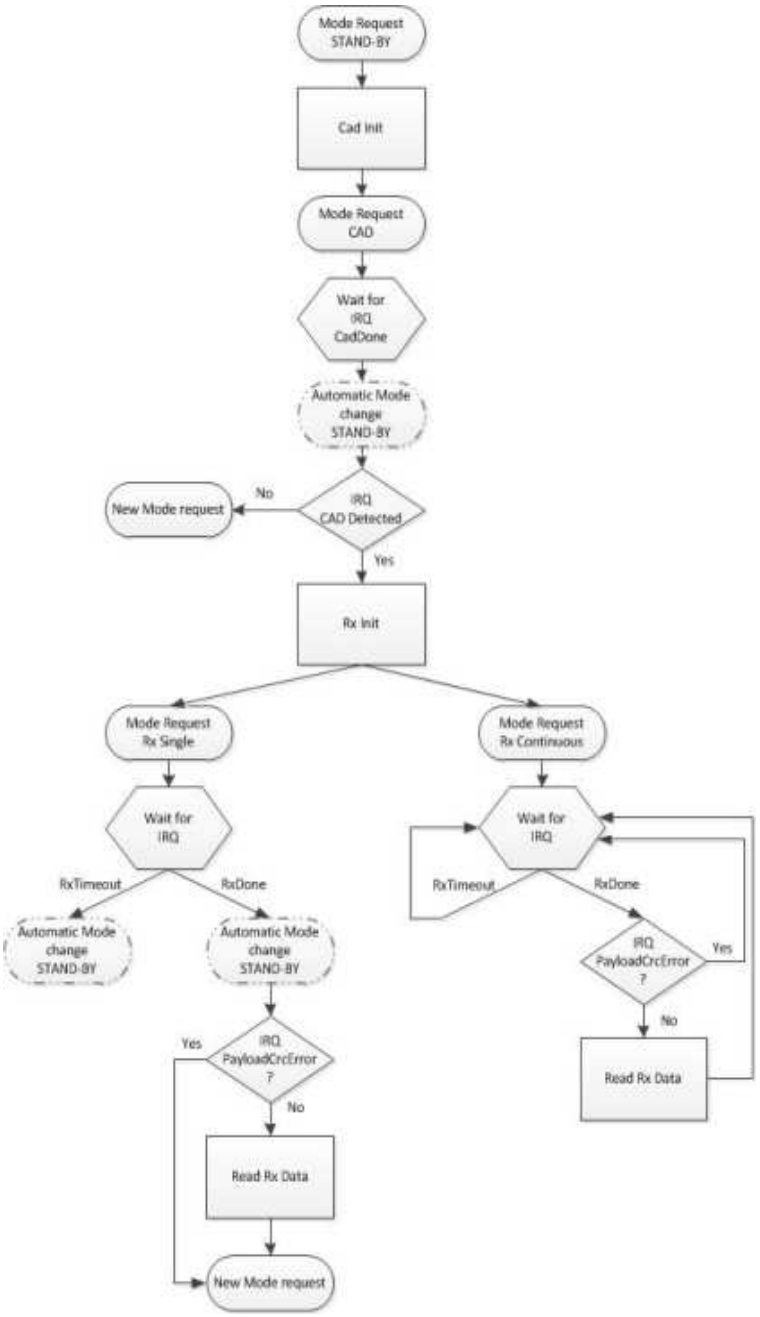



*Figure 10.* LoRa™ *CAD flow*

**Principle of Operation**

The channel activity detection mode is designed to detect a LoRa preamble on the radio channel with the best possible power efficiency. Once in CAD mode, the RFM95W/96W/98W will perform a very quick scan of the band to detect a LoRa packet preamble.

During a CAD the following operations take place:

- The PLL locks
- The radio receiver captures LoRa preamble symbol of data from the channel. The radio current consumption during that phase corresponds to the specified Rx mode current
- The radio receiver and the PLL turn off, and the modem digital processing starts.
- The modem searches for a correlation between the radio captured samples and the ideal preamble waveform. This correlation process takes a little bit less than a symbol period to perform. The radio current consumption during that phase is greatly reduced.
- Once the calculation is finished the modem generates the CadDone interrupt. If the correlation was successful, CadDetected is generated simultaneously.
- The chip goes back to stand-by mode.
- If a preamble was detected, clear the interrupt, then initiate the reception by putting the radio in RX single mode or RX continuous mode.

The time taken for the channel activity detection is dependent upon the LoRa modulation settings used. For a given configuration the typical CAD detection time is shown in the graph below, expressed as a multiple of the LoRa symbol period. Of this period the radio is in receiver mode for ($2^{SF}$ + 32) / BW seconds. For the remainder of the CAD cycle the radio is in a reduced consumption state.

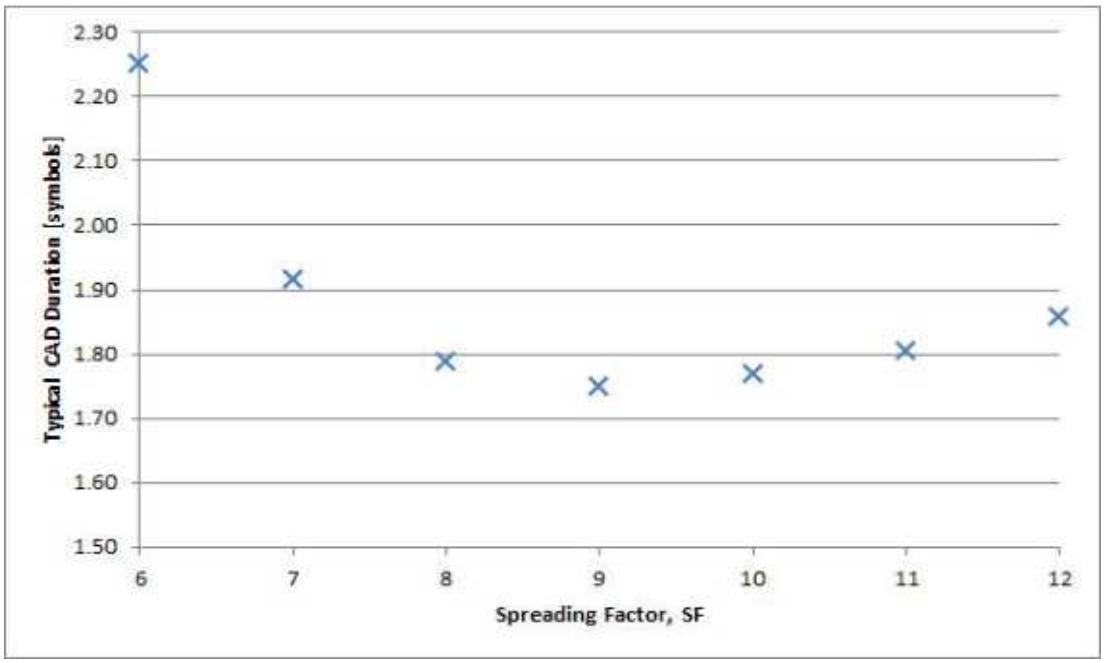



*Figure 11. Channel activity detection (CAD) time as a function of spreading factor*

To illustrate this process and the respective consumption in each mode, the CAD process follows the sequence of events outlined below:

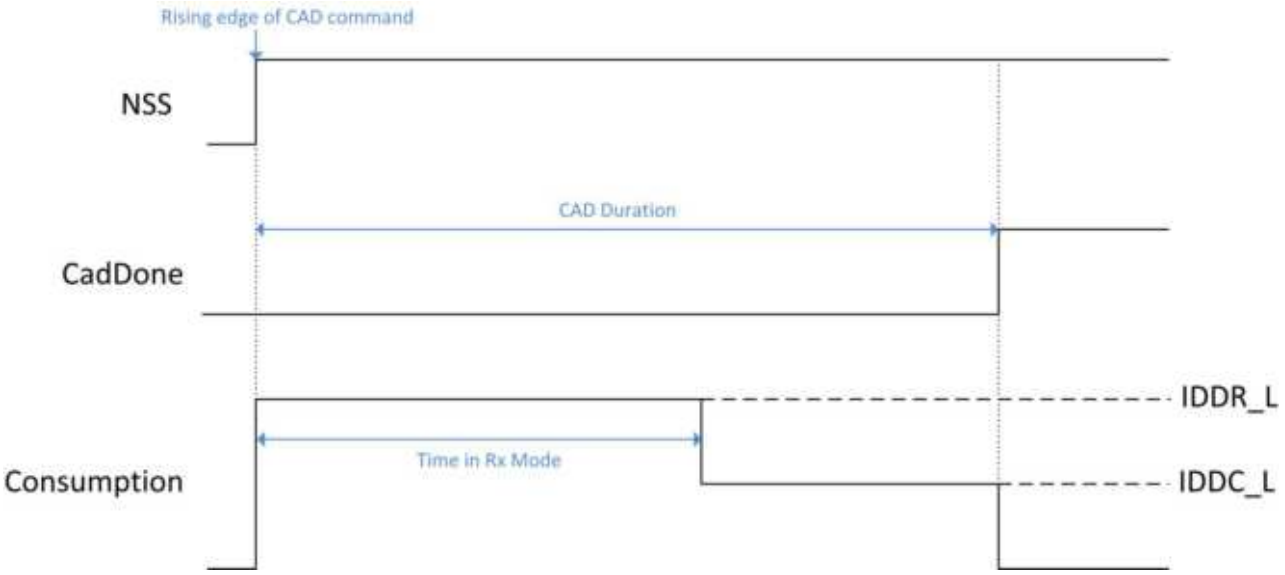



*Figure 12. Consumption Profile of the LoRa CAD Process*

The receiver is then in full receiver mode for just over half of the activity detection, followed by a reduced consumption processing phase where the consumption varies with the LoRa bandwidth as shown in the table below.

*Table 16 LoRa CAD Consumption Figures*

| Bandwidth (kHz) | Full Rx, IDDR_L (mA) | Processing, IDDC_L (mA) |
|---|---|---|
| 7.8 | To be confirmed | |
| 10.4 | | |
| 15.6 | | |
| 20.8 | | |
| 31.2 | | |
| 41.7 | | |
| 62.5 | | |
| 125 | 10.8 | 5.6 |
| 250 | 11.6 | 6.5 |
| 500 | 13 | 8 |

**Digital IO Pin Mapping**

Six of RFM95W/96W/98W's general purpose IO pins are available used in LoRa™ mode. Their mapping is shown below and depends upon the configuration of registers *RegDioMapping1* and *RegDioMapping2.*

*Table 17 DIO Mapping* LoRa™ *Mode*

| Operating Mode | DIOx Mapping | DIO5 | DIO4 | DIO3 | DIO2 | DIO1 | DIO0 |
|---|---|---|---|---|---|---|---|
| ALL | 00 | ModeReady | CadDetected | CadDone | FhssChangeChannel | RxTimeout | RxDone |
| | 01 | ClkOut | PllLock | ValidHeader | FhssChangeChannel | FhssChangeChannel | TxDone |
| | 10 | ClkOut | PllLock | PayloadCrcError | FhssChangeChannel | CadDetected | CadDone |
| | 11 | - | - | - | - | - | - |

### 4.1.8 Modem Status Indicators

The state of the LoRa modem is accessible with the *ModemStatus* bits in *RegModemStat.* They can mostly used for debug in Rx mode and the useful indicators are:

- Bit 0: *Signal Detected* indicates that a valid LoRa preamble has been detected
- Bit 1: *Signal Synchronized* indicates that the end of Preamble has been detected, the modem is in lock
- Bit 3: Header Info Valid toggles high when a valid Header (with correct CRC) is detected

## 4.2. FSK/OOK Modem

### 4.2.1. Bit Rate Setting

The bitrate setting is referenced to the crystal oscillator and provides a precise means of setting the bit rate (or equivalently chip) rate of the radio. In continuous transmit mode (Section 3.2.2) the data stream to be transmitted can be input directly to the modulator via pin 9 (DIO2/DATA) in an asynchronous manner, unless Gaussian filtering is used, in which case the DCLK signal on pin 10 (DIO1/DCLK) is used to synchronize the data stream. See section 4.2.2.3 for details on the Gaussian filter.

In Packet mode or in Continuous mode with Gaussian filtering enabled, the Bit Rate (BR) is controlled by bits *Bitrate* in *RegBitrateMsb and RegBitrateLsb*

$$BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$$

*Note: BitrateFrac bits have* ***no effect*** *(i.e may be considered equal to 0)* ***in OOK*** *modulation mode.*

The quantity *BitrateFrac* is hence designed to allow very high precision (max. 250 ppm programing resolution) for any bitrate in the programmable range. Table 18 below shows a range of standard bitrates and the accuracy to within which they may be reached.

*Table 18 Bit Rate Examples*

| Type | BitRate (15:8) | BitRate (7:0) | (G)FSK (G)MSK | OOK | Actual BR (b/s) |
|---|---|---|---|---|---|
| Classical modem baud rates (multiples of 1.2 kbps) | 0x68 | 0x2B | 1.2 kbps | 1.2 kbps | 1200.015 |
| | 0x34 | 0x15 | 2.4 kbps | 2.4 kbps | 2400.060 |
| | 0x1A | 0x0B | 4.8 kbps | 4.8 kbps | 4799.760 |
| | 0x0D | 0x05 | 9.6 kbps | 9.6 kbps | 9600.960 |
| | 0x06 | 0x83 | 19.2 kbps | 19.2 kbps | 19196.16 |
| | 0x03 | 0x41 | 38.4 kbps | | 38415.36 |
| | 0x01 | 0xA1 | 76.8 kbps | | 76738.60 |
| | 0x00 | 0xD0 | 153.6 kbps | | 153846.1 |
| Classical modem baud rates (multiples of 0.9 kbps) | 0x02 | 0x2C | 57.6 kbps | | 57553.95 |
| | 0x01 | 0x16 | 115.2 kbps | | 115107.9 |
| Round bit rates (multiples of 12.5, 25 and 50 kbps) | 0x0A | 0x00 | 12.5 kbps | 12.5 kbps | 12500.00 |
| | 0x05 | 0x00 | 25 kbps | 25 kbps | 25000.00 |
| | 0x80 | 0x00 | 50 kbps | | 50000.00 |
| | 0x01 | 0x40 | 100 kbps | | 100000.0 |
| | 0x00 | 0xD5 | 150 kbps | | 150234.7 |
| | 0x00 | 0xA0 | 200 kbps | | 200000.0 |
| | 0x00 | 0x80 | 250 kbps | | 250000.0 |
| | 0x00 | 0x6B | 300 kbps | | 299065.4 |
| Watch Xtal frequency | 0x03 | 0xD1 | 32.768 kbps | 32.768 kbps | 32753.32 |

### 4.2.2. FSK/OOK Transmission

#### *4.2.2.1. FSK Modulation*

FSK modulation is performed inside the PLL bandwidth, by changing the fractional divider ratio in the feedback loop of the PLL. The high resolution of the sigma-delta modulator, allows for very narrow frequency deviation. The frequency deviation $F_{DEV}$ is given by:

$$F_{DEV} = F_{STEP} \times Fdev(13,0)$$

To ensure correct modulation, the following limit applies:

$$F_{DEV} = F_{STEP} \times Fdev(13,0)$$

*Note No constraint applies to the modulation index of the transmitter, but the frequency deviation must be set between 600 Hz and 200 kHz.*

#### *4.2.2.2. OOK Modulation*

OOK modulation is applied by switching on and off the power amplifier. Digital control and ramping are available to improve the transient power response of the OOK transmitter.

#### *4.2.2.3. Modulation Shaping*

Modulation shaping can be applied in both OOK and FSK modulation modes, to improve the narrowband response of the transmitter. Both shaping features are controlled with *PaRamp* bits in *RegPaRamp*.

- In FSK mode, a Gaussian filter with BT = 0.5 or 1 is used to filter the modulation stream, at the input of the sigma-delta modulator. If the Gaussian filter is enabled when the RFM95W/96W/98W is in Continuous mode, DCLK signal on pin 10 (DIO1/DCLK) will trigger an interrupt on the uC each time a new bit has to be transmitted. Please refer to section 5.4.2 for details.
- When OOK modulation is used, the PA bias voltages are ramped up and down smoothly when the PA is turned on and off, to reduce spectral splatter.

*Note The transmitter must be restarted if the ModulationShaping setting is changed, in order to recalibrate the built-in filter.*

### 4.2.3. FSK/OOK Reception

#### *4.2.3.1. FSK Demodulator*

The FSK demodulator of the RFM95W/96W/98W is designed to demodulate FSK, GFSK, MSK and GMSK modulated signals. It is most efficient when the modulation index of the signal is greater than 0.5 and below 10:

$$0.5 \le \beta = \frac{2 \times F_{DEV}}{BR} \le 10$$

The output of the FSK demodulator can be fed to the Bit Synchronizer to provide the companion processor with a synchronous data stream in Continuous mode.

#### *4.2.3.2. OOK Demodulator*

The OOK demodulator performs a comparison of the RSSI output and a threshold value. Three different threshold modes are available, configured through bits *OokThreshType* in *RegOokPeak*.

The recommended mode of operation is the “Peak” threshold mode, illustrated in Figure 13:

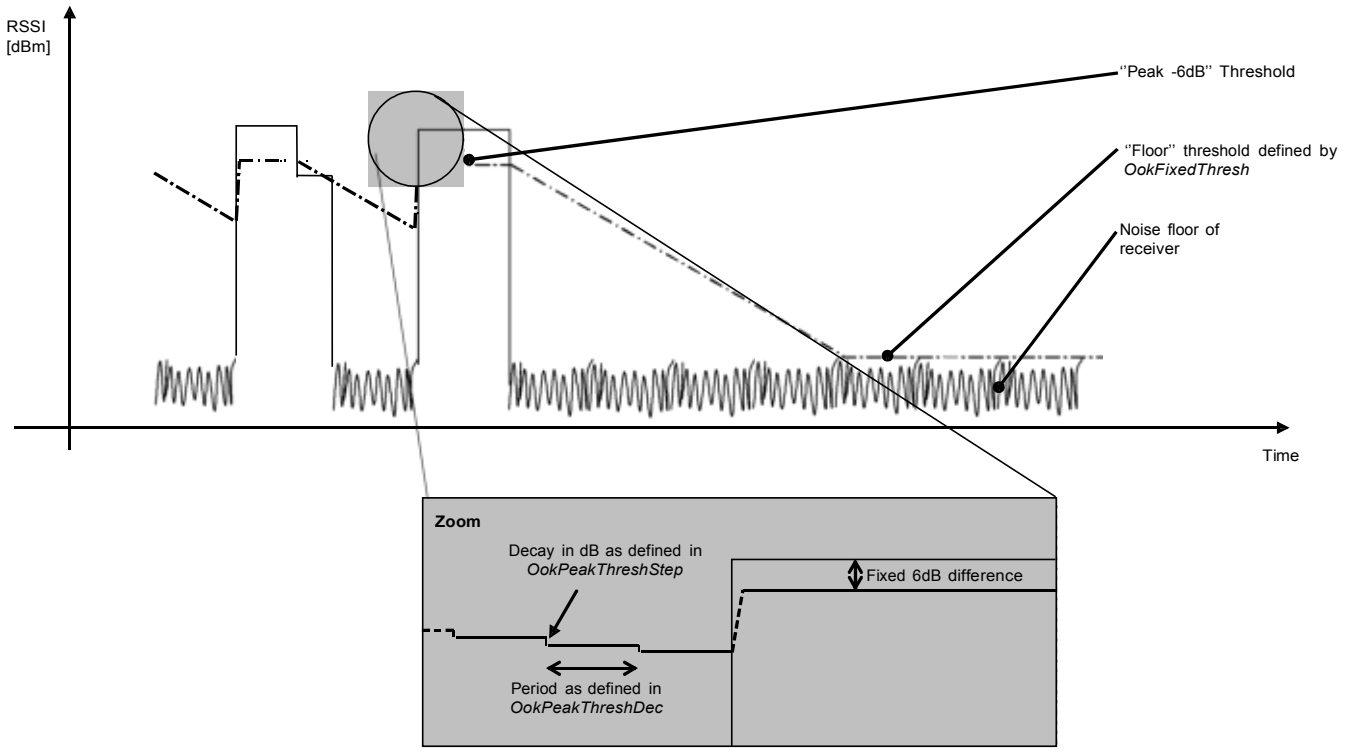



*Figure 13. OOK Peak Demodulator Description*

In peak threshold mode the comparison threshold level is the peak value of the RSSI, reduced by 6dB. In the absence of an input signal, or during the reception of a logical '0', the acquired peak value is decremented by one *OokPeakThreshStep* every *OokPeakThreshDec* period.

When the RSSI output is null for a long time (for instance after a long string of "0" received, or if no transmitter is present), the peak threshold level will continue falling until it reaches the "Floor Threshold", programmed in *OokFixedThresh*.

The default settings of the OOK demodulator lead to the performance stated in the electrical specification. However, in applications in which sudden signal drops are awaited during a reception, the three parameters should be optimized accordingly.

**Optimizing the Floor Threshold**

*OokFixedThresh* determines the sensitivity of the OOK receiver, as it sets the comparison threshold for weak input signals (i.e. those close to the noise floor). Significant sensitivity improvements can be generated if configured correctly.

Note that the noise floor of the receiver at the demodulator input depends on:

- The noise figure of the receiver.
- The gain of the receive chain from antenna to base band.
- The matching - including SAW filter if any.
- The bandwidth of the channel filters.

It is therefore important to note that the setting of *OokFixedThresh* will be application dependant. The following procedure is recommended to optimize *OokFixedThresh*.

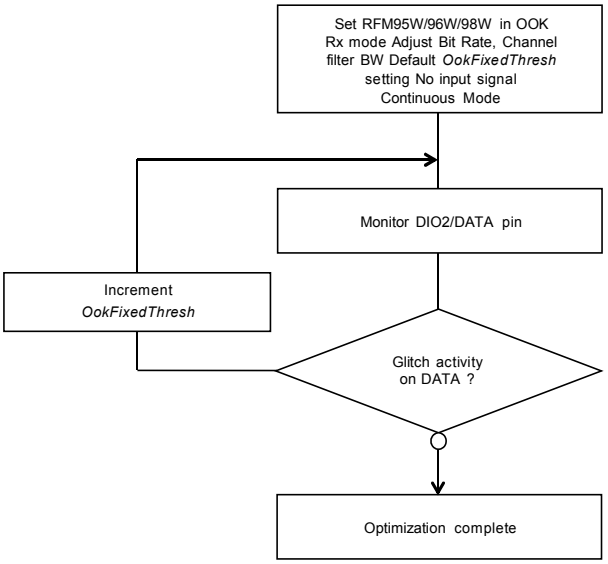



*Figure 14. Floor Threshold Optimization*

The new floor threshold value found during this test should be used for OOK reception with those receiver settings.

**Optimizing OOK Demodulator for Fast Fading Signals**

A sudden drop in signal strength can cause the bit error rate to increase. For applications where the expected signal drop can be estimated, the following OOK demodulator parameters *OokPeakThreshStep* and *OokPeakThreshDec* can be optimized as described below for a given number of threshold decrements per bit. Refer to *RegOokPeak* to access those settings.

**Alternative OOK Demodulator Threshold Modes**

In addition to the Peak OOK threshold mode, the user can alternatively select two other types of threshold detectors:

- Fixed Threshold: The value is selected through *OokFixedThresh*
- Average Threshold: Data supplied by the RSSI block is averaged, and this operation mode should only be used with DC-free encoded data.

*4.2.3.3. Bit Synchronizer*

The bit synchronizer provides a clean and synchronized digital output based upon timing recovery information gleaned from the received data edge transitions. Its output is made available on pin DIO1/DCLK in Continuous mode and can be disabled through register settings. However, for optimum receiver performance, especially in Continuous receive mode, its use is strongly advised.

The Bit Synchronizer is automatically activated in Packet mode. Its bit rate is controlled by *BitRateMsb* and *BitRateLsb* in *RegBitrate.*

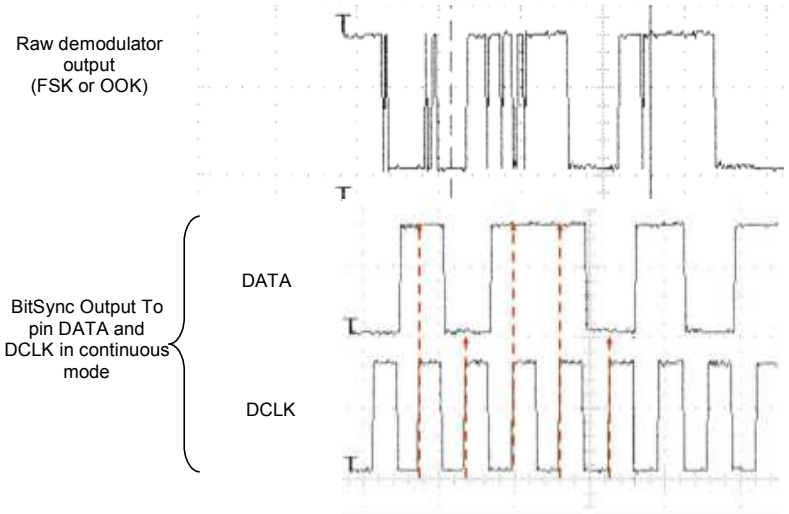



*Figure 15. Bit Synchronizer Description*

To ensure correct operation of the Bit Synchronizer, the following conditions have to be satisfied:

- A preamble (0x55 or 0xAA) of at least 12 bits is required for synchronization, the longer the synchronization phase is the better the ensuing packet detection rate will be.
- The subsequent payload bit stream must have at least one edge transition (either rising or falling) every 16 bits during data transmission.
- The absolute error between transmitted and received bit rate must not exceed 6.5%.

#### *4.2.3.4. Frequency Error Indicator*

This frequency error indicator measures the frequency error between the programmed RF centre frequency and the carrier frequency of the modulated input signal to the receiver. When the FEI is performed, the frequency error is measured and the signed result is loaded in *FeiValue* in *RegFei*, in 2's complement format. The time required for an FEI evaluation is 4 bit periods.

To ensure correct operation of the FEI:

- The measurement must be launched during the reception of preamble.
- The sum of the frequency offset and the 20 dB signal bandwidth must be lower than the base band filter bandwidth. i.e. The whole modulated spectrum must be received.

The 20 dB bandwidth of the signal can be evaluated as follows (double-side bandwidth):

$$BW_{20dB} \quad 2 \quad \left(F_{DEV} \quad 2\right)$$

The frequency error, in Hz, can be calculated with the following formula:

$$FEI = F_{STEP} \times FeiValue$$

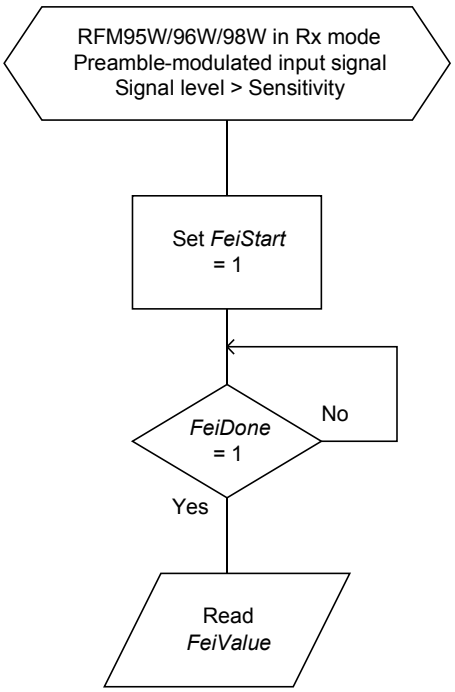



*Figure 16. FEI Process*

#### *4.2.3.5. AFC*

The AFC is based on the FEI measurement, therefore the same input signal and receiver setting conditions apply. When the AFC procedure is performed the *AfcValue* is directly subtracted from the register that defines the frequency of operation of the chip, $F_{RF}$. The AFC is executed each time the receiver is enabled, if *AfcAutoOn* = 1.

When the AFC is enabled (*AfcAutoOn* = 1), the user has the option to:

- Clear the former AFC correction value, if *AfcAutoClearOn* = 1. Allowing the next frequency correction to be performed from the initial centre frequency.
- Start the AFC evaluation from the previously corrected frequency. This may be useful in systems in which the centre frequency experiences cumulative drift - such as the ageing of a crystal reference.

The RFM95W/96W/98W offers an alternate receiver bandwidth setting during the AFC phase allowing the accommodation of larger frequency errors. The setting *RegAfcBw* sets the receive bandwidth during the AFC process. In a typical receiver application the, once the AFC is performed, the radio will revert to the receiver communication or channel bandwidth (*RegRxBw*) for the ensuing communication phase.

Note that the FEI measurement is valid only during the reception of preamble. The provision of the *PreambleDetect* flag can hence be used to detect this condition and allow a reliable AFC or FEI operation to be triggered. This process can be performed automatically by using the appropriate options in *StartDemodOnPreamble* found in the *RegRxConfig* register.

A detailed description of the receiver setup to enable the AFC is provided in section 4.2.6.

#### *4.2.3.6. Preamble Detector*

The Preamble Detector indicates the reception of a carrier modulated with a 0101...sequence. It is insensitive to the frequency offset, as long as the receiver bandwidth is large enough. The size of detection can be programmed from 1 to 3 bytes with *PreambleDetectorSize* in *RegPreambleDetect* as defined in the next table.

*Table 19 Preamble Detector Settings*

| ***PreambleDetectorSize*** | **# of Bytes** |
|---|---|
| 00 | 1 |
| 01 | 2 (recommended) |
| 10 | 3 |
| 11 | reserved |

For normal operation, *PreambleDetectTol* should be set to be set to 10 (0x0A), with a qualifying preamble size of 2 bytes.

The *PreambleDetect* interrupt (either in *RegIrqFlags1* or mapped to a specific DIO) then goes high every time a valid preamble is detected, assuming *PreambleDetectorOn*=1.

The preamble detector can also be used as a gate to ensure that AFC and AGC are performed on valid preamble. See section 4.2.6. for details.

*4.2.3.7. Image Rejection Mixer*

The RFM95W/96W/98W employs an image rejection mixer (IRM) which, uncalibrated, 35 dB image rejection. A low phase noise
PLL is used to perform calibration of the receiver chain. This increases the typical image rejection to 48 dB.

*4.2.3.8.Image and RSSI Calibration*

An automated process is implemented to calibrate the phase and gain imbalances of I and Q receive paths. This calibration enhances image rejection and improves RSSI precision. It is launched under the following circumstances:

- Automatically at Power On Reset or after a Manual Reset of the chip (refer to section 7.2), only for the Low Frequency front-end, and is performed at 434MHz
- Automatically when a pre-defined temperature change is observed, if the option is enabled. A selectable temperature change, set with *TempThreshold* (5, 10, 15 or 20°C), is detected and reported in *TempChange*, if the temperature monitoring is turned On with *TempMonitorOff*=0.This interrupt flag can be used by the application to launch a new image calibration at a convenient time if *AutoImageCalOn*=0, or immediately when this temperature variation is detected, if *AutoImageCalOn*=1
- Upon user request, by setting bit *ImageCalStart* in *RegImageCal*, when the device is in Standby mode

*Notes - The calibration procedure takes approximately 10ms. It is recommended to disable the fully automated (temperature-dependent) calibration, to better control when it is triggered (and avoid unexpected packet losses)*

*- To perform the calibration, the radio must be temporarily returned to FSK/OOK mode*

*- The automatic IQ and RSSI calibration done at POR and Reset is only valid at 434 MHz (the value of RegFrf at POR). To improve accuracy of RSSI and image rejection, this calibration should be replicated at the frequency (ies) of interest, for instance a calibration should be launched with Frf set to 868.3 MHz if the high frequency port supports communication in this frequency band. Conversely if the product is used at 169 MHz, the calibration should be repeated with Frf=169MHz*

*- FormerTemp and TempChange in SX1276/77/79 are frequency-specific and the IC keeps a copy of these variables when switching between the low frequency and the high frequency domains (along with the corresponding calibration values, stored in test registers)*

*- FormerTemp and TempChange cannot be read in Sleep mode (although they are saved). They should be read in Standby mode*

*4.2.3.9. Timeout Function*

The RFM95W/96W/98W includes a Timeout function, which allows it to automatically shut-down the receiver after a receive sequence and therefore save energy.

- *Timeout* interrupt is generated *TimeoutRxRssi x 16 x Tbit* after switching to Rx mode if the *Rssi* flag does not raise within this time frame (*RssiValue* > *RssiThreshold*)
- *Timeout* interrupt is generated *TimeoutRxPreamble x 16 x Tbit* after switching to Rx mode if the *PreambleDetect* flag does not raise within this time frame
- *Timeout* interrupt is generated *TimeoutSignalSync x 16 x Tbit* after switching to Rx mode if the *SyncAddress* flag does not raise within this time frame

This timeout interrupt can be used to warn the companion processor to shut down the receiver and return to a lower power mode. To become active, these timeouts must also be enabled by setting the correct *RxTrigger* parameters in *RegRxConfig:*

*Table 20 RxTrigger Settings to Enable Timeout Interrupts*

| ***Receiver Triggering Event*** | ***RxTrigger (2:0)*** | ***Timeout on Rssi*** | ***Timeout on Preamble*** | ***Timeout on SyncAddress*** |
|---|---|---|---|---|
| None | 000 | Off | Off | Active |
| *Rssi* Interrupt | 001 | Active | Off | |
| *PreambleDetect* | 110 | Off | Active | |
| *Rssi* Interrupt & *PreambleDetect* | 111 | Active | Active | |

### 4.2.4. Operating Modes in FSK/OOK Mode

The RFM95W/96W/98W has several working modes, manually programmed in *RegOpMode*. Fully automated mode selection, packet transmission and reception is also possible using the Top Level Sequencer described in Section 4.2.8.

*Table 21 Basic Transceiver Modes*

| Mode | Selected mode | Symbol | Enabled blocks |
|---|---|---|---|
| 000 | Sleep mode | Sleep | None |
| 001 | Standby mode | Stdby | Top regulator and crystal oscillator |
| 010 | Frequency synthesiser to Tx frequency | FSTx | Frequency synthesizer at Tx frequency (Frf) |
| 011 | Transmit mode | Tx | Frequency synthesizer and transmitter |
| 100 | Frequency synthesiser to Rx frequency | FSRx | Frequency synthesizer at frequency for reception (Frf-IF) |
| 101 | Receive mode | Rx | Frequency synthesizer and receiver |

When switching from a mode to another the sub-blocks are woken up according to a pre-defined optimized sequence.

### 4.2.5. Startup Times

The startup time of the transmitter or the receiver is dependant upon which mode the transceiver was in at the beginning. For a complete description, Figure 17 below shows a complete startup process, from the lower power mode “Sleep”.

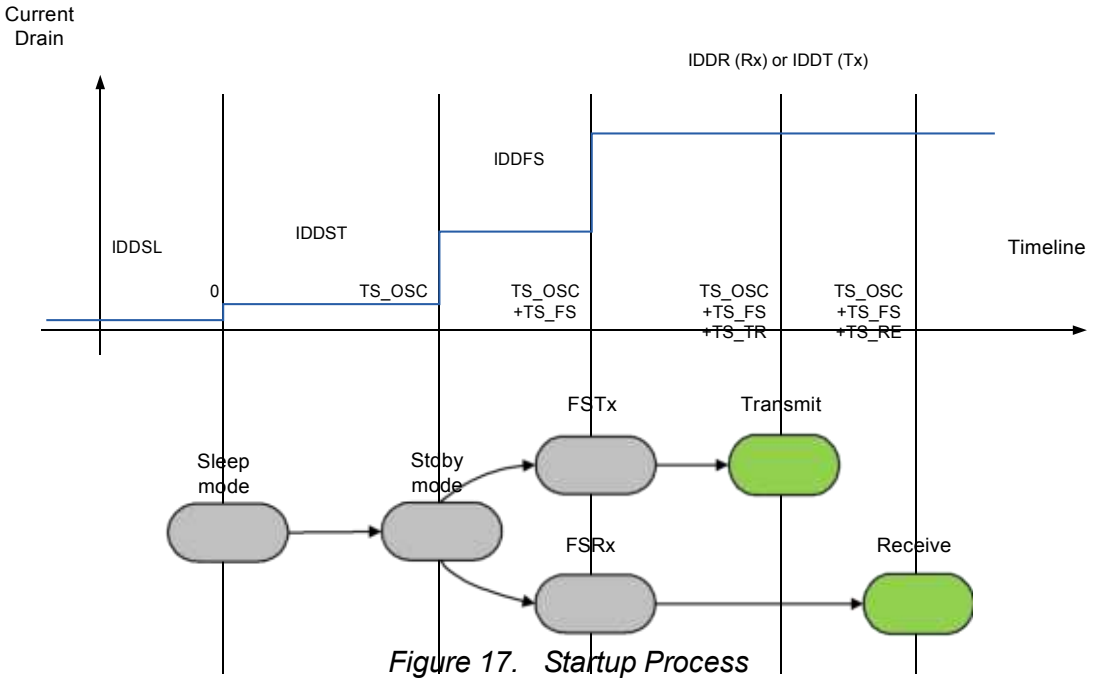



*Figure 17. Startup Process*

TS_OSC is the startup time of the crystal oscillator which depends on the electrical characteristics of the crystal. TS_FS is the startup time of the PLL including systematic calibration of the VCO.

Typical values of TS_OSC and TS_FS are given in Section 2.3.

#### 4.2.5.1. Transmitter Startup Time

The transmitter startup time, TS_TR, is calculated as follows in FSK mode:

$$TS_TR = 5\mu s + 1.25 \times PaRamp + \frac{1}{2} \times Tbit$$

where *PaRamp* is the ramp-up time programmed in *RegPaRamp* and *Tbit* is the bit time.

In OOK mode, this equation can be simplified to the following:

$$TS_TR = 5\mu s + \frac{1}{2} \times Tbit$$

#### 4.2.5.2. Receiver Startup Time

The receiver startup time, TS_RE, only depends upon the receiver bandwidth effective at the time of startup. When AFC is enabled (*AfcAutoOn*=1), *AfcBw* should be used instead of *RxBw* to extract the receiver startup time:

*Table 22 Receiver Startup Time Summary*

| ***RxBw* if *AfcAutoOn=0*<br>*RxBwAfc* if *AfcAutoOn=1*** | **TS_RE<br>(+/-5%)** |
|---|---|
| 2.6 kHz | 2.33 ms |
| 3.1 kHz | 1.94 ms |
| 3.9 kHz | 1.56 ms |
| 5.2 kHz | 1.18 ms |
| 6.3 kHz | 984 us |
| 7.8 kHz | 791 us |
| 10.4 kHz | 601 us |
| 12.5 kHz | 504 us |
| 15.6 kHz | 407 us |
| 20.8 kHz | 313 us |
| 25.0 kHz | 264 us |
| 31.3 kHz | 215 us |
| 41.7 kHz | 169 us |
| 50.0 kHz | 144 us |
| 62.5 kHz | 119 us |
| 83.3 kHz | 97 us |
| 100.0 kHz | 84 us |
| 125.0 kHz | 71 us |
| 166.7 kHz | 85 us |
| 200.0 kHz | 74 us |
| 250.0 kHz | 63 us |

TS_RE or later after setting the device in Receive mode, any incoming packet will be detected and demodulated by the transceiver.

*4.2.5.3. Time to RSSI Evaluation*

The first RSSI sample will be available TS_RSSI after the receiver is ready, in other words TS_RE + TS_RSSI after the receiver was requested to turn on.

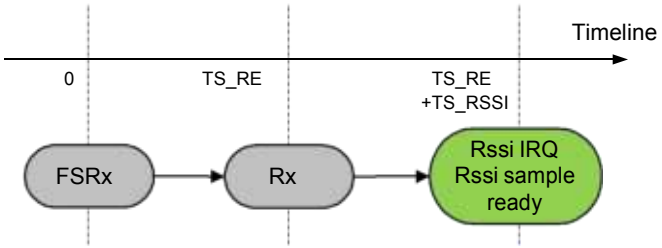



*Figure 18. Time to Rssi Sample*

TS_RSSI depends on the receiver bandwidth, as well as the *RssiSmoothing* option that was selected. The formula used to calculate TS_RSSI is provided in section 2.5.4.

*4.2.5.4. Tx to Rx Turnaround Time*

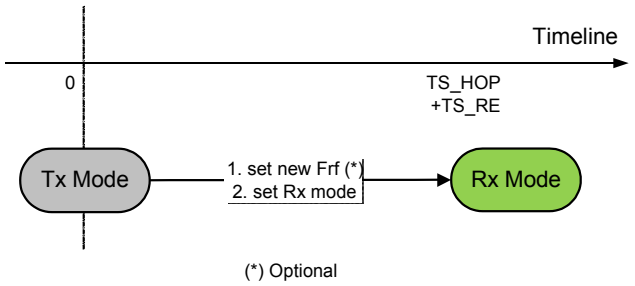



*Figure 19. Tx to Rx Turnaround*

*Note The SPI instruction times are omitted, as they can generally be very small as compared to other timings (up to 10MHz SPI clock).*

*4.2.5.5. Rx to Tx*

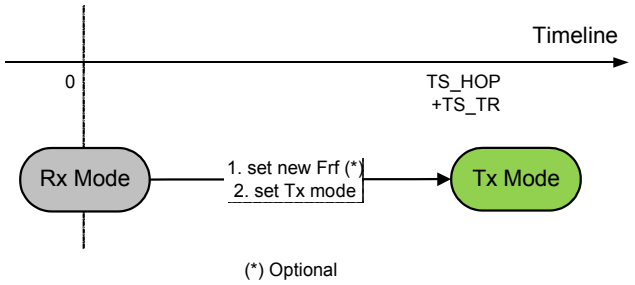



*Figure 20. Rx to Tx Turnaround*

#### *4.2.5.6. Receiver Hopping, Rx to Rx*

Two methods are possible:

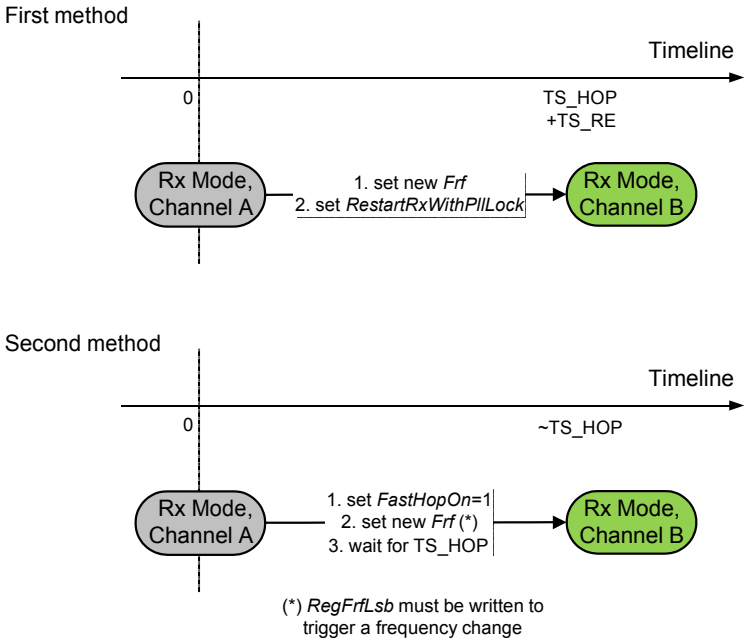



*Figure 21. Receiver Hopping*

The second method is quicker, and should be used if a very quick RF sniffing mechanism is to be implemented.

#### *4.2.5.7. Tx to Tx*

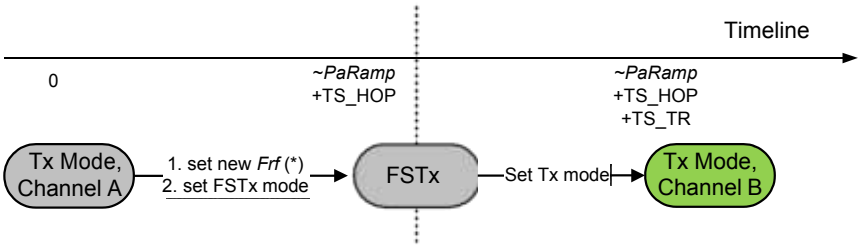



*Figure 22. Transmitter Hopping*

### 4.2.6. Receiver Startup Options

The RFM95W/96W/98W receiver can automatically control the gain of the receive chain (AGC) and adjust the receiver LO
frequency (AFC). Those processes are carried out on a packet-by-packet basis. They occur:

- When the receiver is turned On.
- When the Receiver is restarted upon user request, through the use of trigger bits *RestartRxWithoutPllLock* or *RestartRxWithPllLock*, in *RegRxConfig.*
- When the receiver is automatically restarted after the reception of a valid packet, or after a packet collision.

Automatic restart capabilities are detailed in Section 4.2.7.

The receiver startup options available in RFM95W/96W/98W are described in Table 23.

*Table 23 Receiver Startup Options*

| ***Triggering Event*** | **Realized Function** | ***AgcAutoOn*** | ***AfcAutoOn*** | ***RxTrigger (2:0)*** |
|---|---|---|---|---|
| None | None | 0 | 0 | 000 |
| *Rssi* Interrupt | AGC | 1 | 0 | 001 |
| | AGC & AFC | 1 | 1 | 001 |
| *PreambleDetect* | AGC | 1 | 0 | 110 |
| | AGC & AFC | 1 | 1 | 110 |
| *Rssi* Interrupt & *PreambleDetect* | AGC | 1 | 0 | 111 |
| | AGC & AFC | 1 | 1 | 111 |

When *AgcAutoOn*=0, the LNA gain is manually selected by choosing *LnaGain* bits in *RegLna.*

### 4.2.7. Receiver Restart Methods

The options for restart of the receiver are covered below. This is typically of use to prepare for the reception of a new signal whose strength or carrier frequency is different from the preceding packet to allow the AGC or AFC to be re-evaluated.

#### *4.2.7.1. Restart Upon User Request*

In Receive mode the user can request a receiver restart - this can be useful in conjunction with the use of a Timeout interrupt following a period of inactivity in the channel of interest. Two options are available:

- No change in the Local Oscillator upon restart: the AFC is disabled, and the *Frf* register has not been changed through SPI before the restart instruction: set bit *RestartRxWithoutPllLock* in *RegRxConfig* to 1.
- Local Oscillator change upon restart: if AFC is enabled (*AfcAutoOn*=1), and/or the *Frf* register had been changed during the last Rx period: set bit *RestartRxWithPllLock* in *RegRxConfig* to 1.

*Note ModeReady must be at logic level 1 for a new RestartRx command to be taken into account.*

#### *4.2.7.2. Automatic Restart after valid Packet Reception*

The bits *AutoRestartRxMode* in *RegSyncConfig* control the automatic restart feature of the RFM95W/96W/98W receiver, when a valid packet has been received:

- If *AutoRestartRxMode* = 00, the function is off, and the user should manually restart the receiver upon valid packet reception (see section 4.2.7.1*).*
- If *AutoRestartRxMode* = 01, after the user has emptied the FIFO following a *PayloadReady* interrupt, the receiver will automatically restart itself after a delay of *InterPacketRxDelay*, allowing for the distant transmitter to ramp down, hence avoiding a false RSSI detection on the 'tail' of the previous packet.
- If *AutoRestartRxMode* = 10 should be used if the next reception is expected on a new frequency, i.e. *Frf* is changed after the reception of the previous packet. An additional delay is systematically added, in order for the PLL to lock at a new frequency.

#### *4.2.7.3. Automatic Restart when Packet Collision is Detected*

In receive mode the RFM95W/96W/98W is able to detect packet collision and restart the receiver. Collisions are detected by a sudden rise in received signal strength, detected by the RSSI. This functionality can be useful in network configurations where many asynchronous slaves attempt periodic communication with a single a master node.

The collision detector is enabled by setting bit *RestartRxOnCollision* to 1.

The decision to restart the receiver is based on the detection of RSSI change. The sensitivity of the system can be adjusted in 1 dB steps by using register *RssiCollisionThreshold* in *RegRxConfig*.

### 4.2.8. Top Level Sequencer

Depending on the application, it is desirable to be able to change the mode of the circuit according to a predefined sequence without access to the serial interface. In order to define different sequences or scenarios, a user-programmable state machine, called Top Level Sequencer (Sequencer in short), can automatically control the chip modes.

***NOTE THAT THIS FUNCTIONALITY IS ONLY AVAILABLE IN FSK/OOK MODE.***

The Sequencer is activated by setting the *SequencerStart* bit in *RegSeqConfig1* to 1 in Sleep or Standby mode (called initial mode).

It is also possible to force the Sequencer off by setting the *Stop* bit in *RegSeqConfig1* to 1 at any time.

*Note SequencerStart and Stop bit must never be set at the same time.*

#### *4.2.8.1. Sequencer States*

As shown in the table below, with the aid of a pair of interrupt timers (T1 and T2), the sequencer can take control of the chip operation in all modes.

*Table 24 Sequencer States*

| ***Sequencer State*** | **Description** |
|---|---|
| **SequencerOff State** | The Sequencer is not activated. Sending a *SequencerStart* command will launch it.<br>When coming from **LowPowerSelection** state, the Sequencer will be Off, whilst the chip will return to its initial mode (either Sleep or Standby mode). |
| **Idle State** | The chip is in low-power mode, either *Standby* or *Sleep*, as defined by *IdleMode* in *RegSeqConfig1*. The Sequencer waits only for the *T1* interrupt. |
| **Transmit State** | The transmitter in on. |
| **Receive State** | The receiver in on. |
| **PacketReceived** | The receiver is on and a packet has been received. It is stored in the FIFO. |
| **LowPowerSelection** | Selects low power state (**SequencerOff** or **Idle** State) |
| **RxTimeout** | Defines the action to be taken on a RxTimeout interrupt.<br>RxTimeout interrupt can be a *TimeoutRxRssi*, *TimeoutRxPreamble* or *TimeoutSignalSync* interrupt. |

#### *4.2.8.2. Sequencer Transitions*

The transitions between sequencer states are listed in the forthcoming table.

*Table 25 Sequencer Transition Options*

| ***Variable*** | **Transition** |
|---|---|
| *IdleMode* | Selects the chip mode during **Idle** state:<br>0: *Standby* mode<br>1: *Sleep* mode |
| *FromStart* | Controls the Sequencer transition when the *SequencerStart* bit is set to 1 in *Sleep* or *Standby* mode:<br>00: to **LowPowerSelection**<br>01: to **Receive** state<br>10: to **Transmit** state<br>11: to **Transmit** state on a *FifoThreshold* interrupt |
| *LowPowerSelection* | Selects Sequencer LowPower state after a *to LowPowerSelection* transition<br>0: **SequencerOff** state with chip on Initial mode<br>1: **Idle** state with chip on *Standby* or *Sleep* mode depending on **IdleMode**<br>Note: Initial mode is the chip LowPower mode at Sequencer start. |
| *FromIdle* | Controls the Sequencer transition from the **Idle** state on a *T1* interrupt:<br>0: to **Transmit** state<br>1: to **Receive** state |
| *FromTransmit* | Controls the Sequencer transition from the **Transmit** state:<br>0: to **LowPowerSelection** on a *PacketSent* interrupt<br>1: to **Receive** state on a *PacketSent* interrupt |
| *FromReceive* | Controls the Sequencer transition from the **Receive** state:<br>000 and 111: unused<br>001: to **PacketReceived** state on a *PayloadReady* interrupt<br>010: to **LowPowerSelection** on a *PayloadReady* interrupt<br>011: to **PacketReceived** state on a *CrcOk* interrupt. If CRC is wrong (corrupted packet, with CRC on but CrcAutoClearOn is off), the PayloadReady interrupt will drive the sequencer to RxTimeout state.<br>100: to **SequencerOff** state on a *Rssi* interrupt<br>101: to **SequencerOff** state on a *SyncAddress* interrupt<br>110: to **SequencerOff** state on a *PreambleDetect* interrupt<br>Irrespective of this setting, transition to **LowPowerSelection** on a *T2* interrupt |
| *FromRxTimeout* | Controls the state-machine transition from the **Receive** state on a *RxTimeout* interrupt (and on *PayloadReady* if **FromReceive** = 011):<br>00: to **Receive** state via *ReceiveRestart*<br>01: to **Transmit** state<br>10: to **LowPowerSelection**<br>11: to **SequencerOff** state<br>Note: RxTimeout interrupt is a *TimeoutRxRssi*, *TimeoutRxPreamble* or *TimeoutSignalSync* interrupt. |
| *FromPacketReceived* | Controls the state-machine transition from the **PacketReceived** state:<br>000: to **SequencerOff** state<br>001: to **Transmit** on a *FifoEmpty* interrupt<br>010: to **LowPowerSelection**<br>011: to **Receive** via *FS* mode, if frequency was changed<br>100: to **Receive** state (no frequency change) |

*4.2.8.3. Timers*

Two timers (Timer1 and Timer2) are also available in order to define periodic sequences. These timers are used to generate interrupts, which can trigger transitions of the Sequencer.

*T1* interrupt is generated (Timer1Resolution * Timer1Coefficient) after ***T2* interrupt** or ***SequencerStart***. command.
*T2* interrupt is generated (Timer2Resolution * Timer2Coefficient) after ***T1* interrupt**.

The timers' mechanism is summarized on the following diagram.

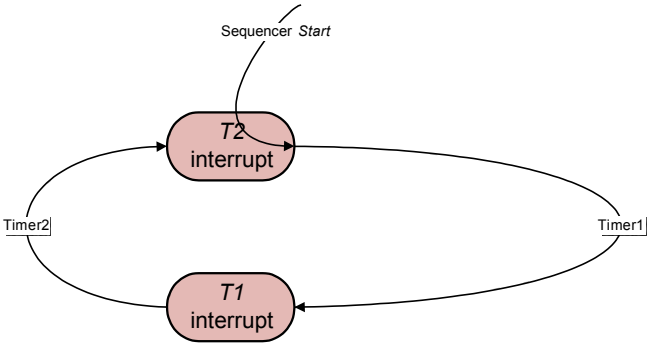



*Figure 23. Timer1 and Timer2 Mechanism*

*Note* *The timer sequence is completed independently of the actual Sequencer state. Thus, both timers need to be on to achieve periodic cycling.*

*Table 26 Sequencer Timer Settings*

| ***Variable*** | **Description** |
|---|---|
| Timer1Resolution | Resolution of Timer1<br>00: disabled<br>01: 64 us<br>10: 4.1 ms<br>11: 262 ms |
| Timer2Resolution | Resolution of Timer2<br>00: disabled<br>01: 64 us<br>10: 4.1 ms<br>11: 262 ms |
| Timer1Coefficient | Multiplying coefficient for Timer1 |
| Timer2Coefficient | Multiplying coefficient for Timer2 |

#### *4.2.8.4. Sequencer State Machine*

The following graphs summarize every possible transition between each Sequencer state. The Sequencer states are highlighted in grey. The transitions are represented by arrows. The condition activating them is described over the transition arrow. For better readability, the start transitions are separated from the rest of the graph.

Transitory states are highlighted in light grey, and exit states are represented in red. It is also possible to force the Sequencer off by setting the *Stop* bit in *RegSeqConfig1* to 1 at any time.

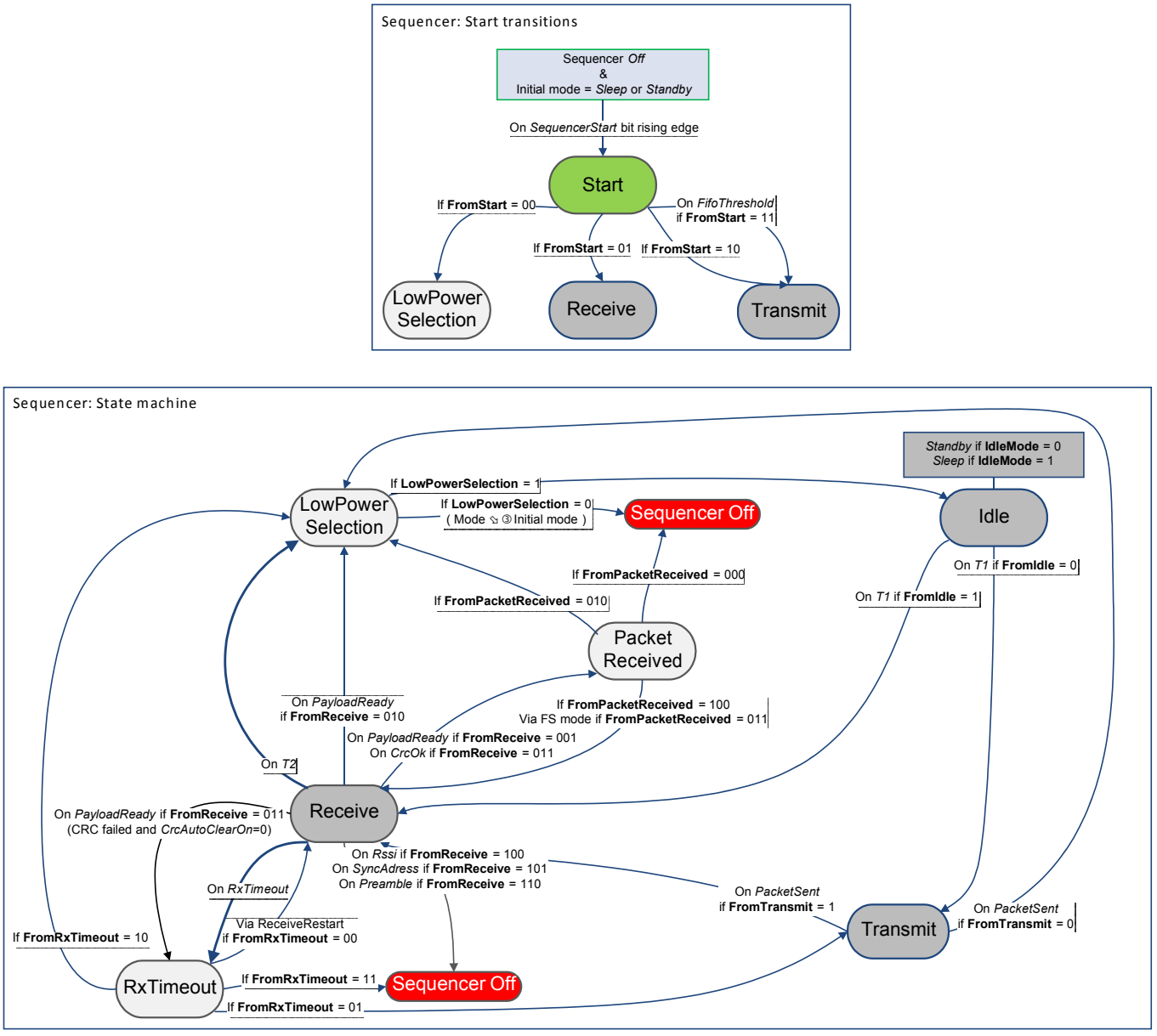



*Figure 24. Sequencer State Machine*

### 4.2.9. Data Processing in FSK/OOK Mode

#### *4.2.9.1. Block Diagram*

Figure below illustrates the RFM95W/96W/98W data processing circuit. Its role is to interface the data to/from the modulator/demodulator and the uC access points (SPI and DIO pins). It also controls all the configuration registers.

The circuit contains several control blocks which are described in the following paragraphs.

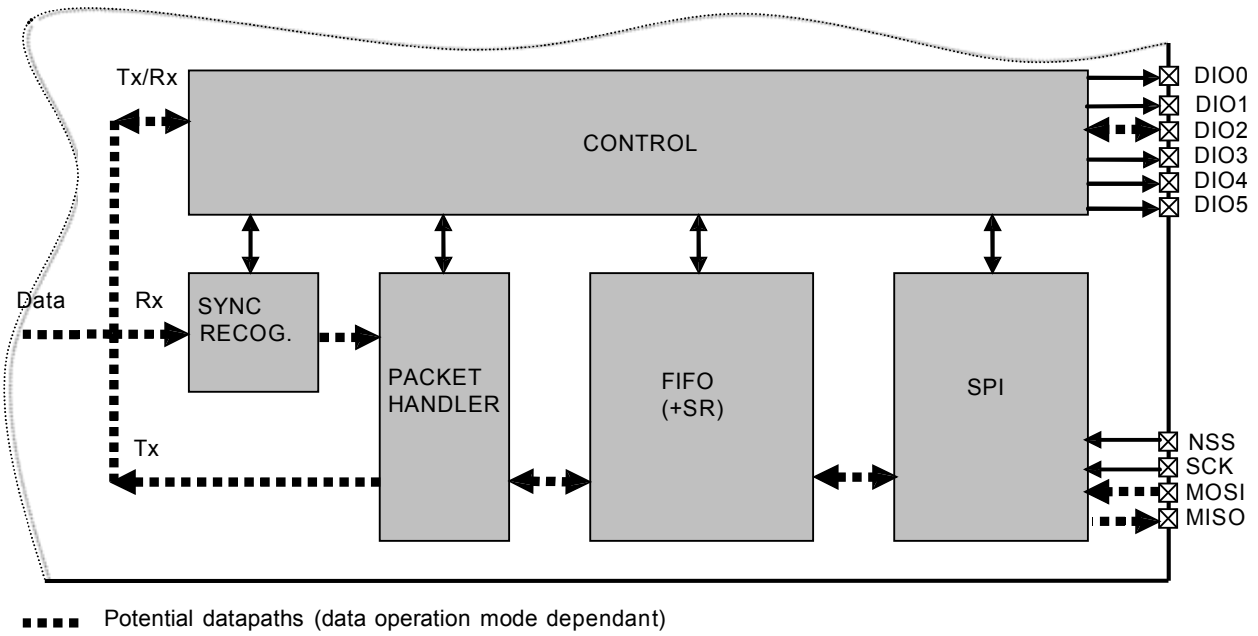



*Figure 25. RFM95W/96W/98W Data Processing Conceptual View*

The RFM95W/96W/98W implements several data operation modes, each with their own data path through the data processing section. Depending on the data operation mode selected, some control blocks are active whilst others remain disabled.

#### *4.2.9.2. Data Operation Modes*

The RFM95W/96W/98W has two different data operation modes selectable by the user:

- Continuous mode: each bit transmitted or received is accessed in real time at the DIO2/DATA pin. This mode may be used if adequate external signal processing is available.
- Packet mode (recommended): user only provides/retrieves payload bytes to/from the FIFO. The packet is automatically built with preamble, Sync word, and optional CRC and DC-free encoding schemes The reverse operation is performed in reception. The uC processing overhead is hence significantly reduced compared to Continuous mode. Depending on the optional features activated (CRC, etc) the maximum payload length is limited to 255, 2047 bytes or unlimited.

Each of these data operation modes is fully described in the following sections.

### 4.2.10. FIFO

**Overview and Shift Register (SR)**

In packet mode of operation, both data to be transmitted and that has been received are stored in a configurable FIFO (First In First Out) device. It is accessed via the SPI interface and provides several interrupts for transfer management.

The FIFO is 1 byte wide hence it only performs byte (parallel) operations, whereas the demodulator functions serially. A shift register is therefore employed to interface the two devices. In transmit mode it takes bytes from the FIFO and outputs them serially (MSB first) at the programmed bit rate to the modulator. Similarly, in Rx the shift register gets bit by bit data from the demodulator and writes them byte by byte to the FIFO. This is illustrated in figure below.

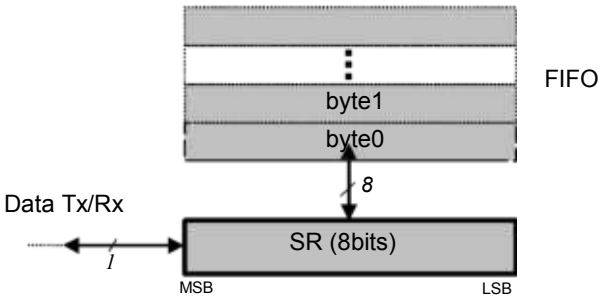



*Figure 26. FIFO and Shift Register (SR)*

*Note* *When switching to Sleep mode, the FIFO can only be used once the ModeReady flag is set (quasi immediate from all modes except from Tx)*

The FIFO size is fixed to 64 bytes.

**Interrupt Sources and Flags**

- *FifoEmpty*: *FifoEmpty* interrupt source is high when byte 0, i.e. whole FIFO, is empty. Otherwise it is low. Note that when retrieving data from the FIFO, *FifoEmpty* is updated on NSS falling edge, i.e. when *FifoEmpty* is updated to low state the currently started read operation must be completed. In other words, *FifoEmpty* state must be checked after each read operation for a decision on the next one (*FifoEmpty* = 0: more byte(s) to read; *FifoEmpty* = 1: no more byte to read).
- *FifoFull*: *FifoFull* interrupt source is high when the last FIFO byte, i.e. the whole FIFO, is full. Otherwise it is low.
- *FifoOverrunFlag*: *FifoOverrunFlag* is set when a new byte is written by the user (in Tx or Standby modes) or the SR (in Rx mode) while the FIFO is already full. Data is lost and the flag should be cleared by writing a 1, note that the FIFO will also be cleared.
- *PacketSent*: *PacketSent* interrupt source goes high when the SR's last bit has been sent.
- *FifoLevel*: Threshold can be programmed by *FifoThreshold* in *RegFifoThresh*. Its behavior is illustrated in figure below.

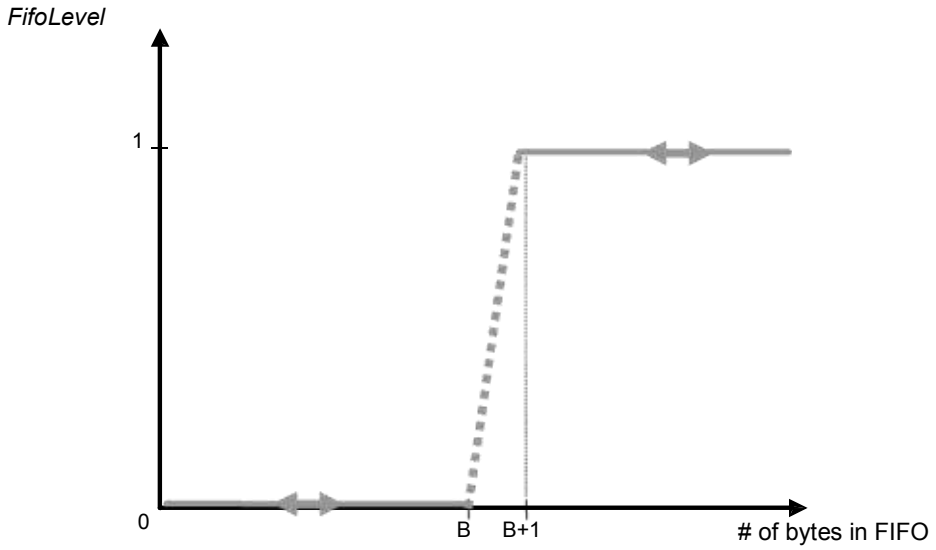



*Figure 27. FifoLevel IRQ Source Behavior*

*Notes - FifoLevel interrupt is updated only after a read or write operation on the FIFO. Thus the interrupt cannot be dynamically updated by only changing the FifoThreshold parameter*

*- FifoLevel interrupt is valid as long as FifoFull does not occur. An empty FIFO will restore its normal operation*

**FIFO Clearing**

Table below summarizes the status of the FIFO when switching between different modes

*Table 27 Status of FIFO when Switching Between Different Modes of the Chip*

| From | To | FIFO status | Comments |
|---|---|---|---|
| Stdby | Sleep | Not cleared | |
| Sleep | Stdby | Not cleared | |
| Stdby/Sleep | Tx | Not cleared | To allow the user to write the FIFO in Stdby/Sleep before Tx |
| Stdby/Sleep | Rx | Cleared | |
| Rx | Tx | Cleared | |
| Rx | Stdby/Sleep | Not cleared | To allow the user to read FIFO in Stdby/Sleep mode after Rx |
| Tx | Any | Cleared | |

*4.2.10.1. Sync Word Recognition*

**Overview**

Sync word recognition (also called Pattern recognition) is activated by setting *SyncOn* in *RegSyncConfig*. The bit synchronizer must also be activated in Continuous mode (automatically done in Packet mode).

The block behaves like a shift register; it continuously compares the incoming data with its internally programmed Sync word and sets *SyncAddressMatch* when a match is detected. This is illustrated in Figure 28 below.

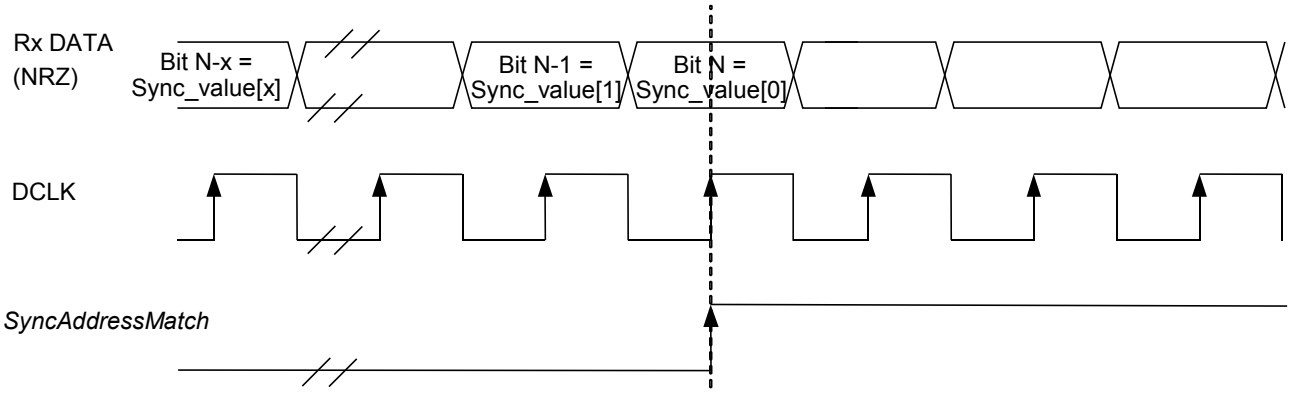



*Figure 28. Sync Word Recognition*

During the comparison of the demodulated data, the first bit received is compared with bit 7 (MSB) of *RegSyncValue1* and the last bit received is compared with bit 0 (LSB) of the last byte whose address is determined by the length of the Sync word.

When the programmed Sync word is detected the user can assume that this incoming packet is for the node and can be processed accordingly.

*SyncAddressMatch* is cleared when leaving Rx or FIFO is emptied.

**Configuration**

- Size: Sync word size can be set from 1 to 8 bytes (i.e. 8 to 64 bits) via *SyncSize* in *RegSyncConfig*. In Packet mode this field is also used for Sync word generation in Tx mode.
- Value: The Sync word value is configured in *SyncValue(63:0)*. In Packet mode this field is also used for Sync word generation in Tx mode.

*Note SyncValue choices containing 0x00 bytes are not allowed*

**Packet Handler**

The packet handler is the block used in Packet mode. Its functionality is fully described in section 4.2.13.

**Control**

The control block configures and controls the full chip's behavior according to the settings programmed in the configuration registers.

### 4.2.11. Digital IO Pins Mapping

Six general purpose IO pins are available on the RFM95W/96W/98W, and their configuration in Continuous or Packet mode is controlled through *RegDioMapping1* and *RegDioMapping2.*

*Table 28 DIO Mapping, Continuous Mode*

| | DIOx Mapping | Sleep | Standby | FSRx/Tx | Rx | Tx |
|---|---|---|---|---|---|---|
| DIO0 | 00 | - | | | SyncAddress | TxReady |
| | 01 | - | | | Rssi / PreambleDetect | - |
| | 10 | - | | | RxReady | TxReady |
| | 11 | - | | | | |
| DIO1 | 00 | - | | | Dclk | |
| | 01 | - | | | Rssi / PreambleDetect | - |
| | 10 | - | | | | |
| | 11 | - | | | | |
| DIO2 | 00 | - | | | Data | |
| | 01 | - | | | Data | |
| | 10 | - | | | Data | |
| | 11 | - | | | Data | |
| DIO3 | 00 | - | | | Timeout | - |
| | 01 | - | | | Rssi / PreambleDetect | - |
| | 10 | - | | | | |
| | 11 | - | TempChange / LowBat | | TempChange / LowBat | |
| DIO4 | 00 | - | | TempChange / LowBat | | |
| | 01 | - | | PllLock | | |
| | 10 | - | | | TimeOut | - |
| | 11 | - | ModeReady | | ModeReady | |
| DIO5 | 00 | ClkOut if RC | ClkOut | | ClkOut | |
| | 01 | - | | PllLock | | |
| | 10 | - | | | Rssi / PreambleDetect | - |
| | 11 | - | ModeReady | | ModeReady | |

*Table 29 DIO Mapping, Packet Mode*

| | DIOx Mapping | Sleep | Standby | FSRx/Tx | Rx | Tx |
|---|---|---|---|---|---|---|
| DIO0 | 00 | - | | | PayloadReady | PacketSent |
| | 01 | - | | | CrcOk | - |
| | 10 | - | | | | |
| | 11 | - | TempChange / LowBat | | TempChange / LowBat | |
| DIO1 | 00 | FifoLevel | | FifoLevel | FifoLevel | |
| | 01 | FifoEmpty | | FifoEmpty | FifoEmpty | |
| | 10 | FifoFull | | FifoFull | FifoFull | |
| | 11 | - | | | | |
| DIO2 | 00 | FifoFull | | FifoFull | FifoFull | |
| | 01 | - | | | RxReady | - |
| | 10 | FifoFull | | | TimeOut | FifoFull |
| | 11 | FifoFull | | | SyncAddress | FifoFull |
| DIO3 | 00 | FifoEmpty | | FifoEmpty | FifoEmpty | |
| | 01 | - | | | | TxReady |
| | 10 | FifoEmpty | | FifoEmpty | FifoEmpty | |
| | 11 | FifoEmpty | | FifoEmpty | FifoEmpty | |
| DIO4 | 00 | - | TempChange / LowBat | | TempChange / LowBat | |
| | 01 | - | | PllLock | | |
| | 10 | - | | | TimeOut | - |
| | 11 | - | | | Rssi / PreambleDetect | - |
| DIO5 | 00 | ClkOut if RC | ClkOut | | ClkOut | |
| | 01 | - | | PllLock | | |
| | 10 | - | | | Data | |
| | 11 | - | ModeReady | | ModeReady | |

### 4.2.12. Continuous Mode

#### *4.2.12.1. General Description*

As illustrated in Figure 29, in Continuous mode the NRZ data to (from) the (de)modulator is directly accessed by the uC on the bidirectional DIO2/DATA pin. The FIFO and packet handler are thus inactive.

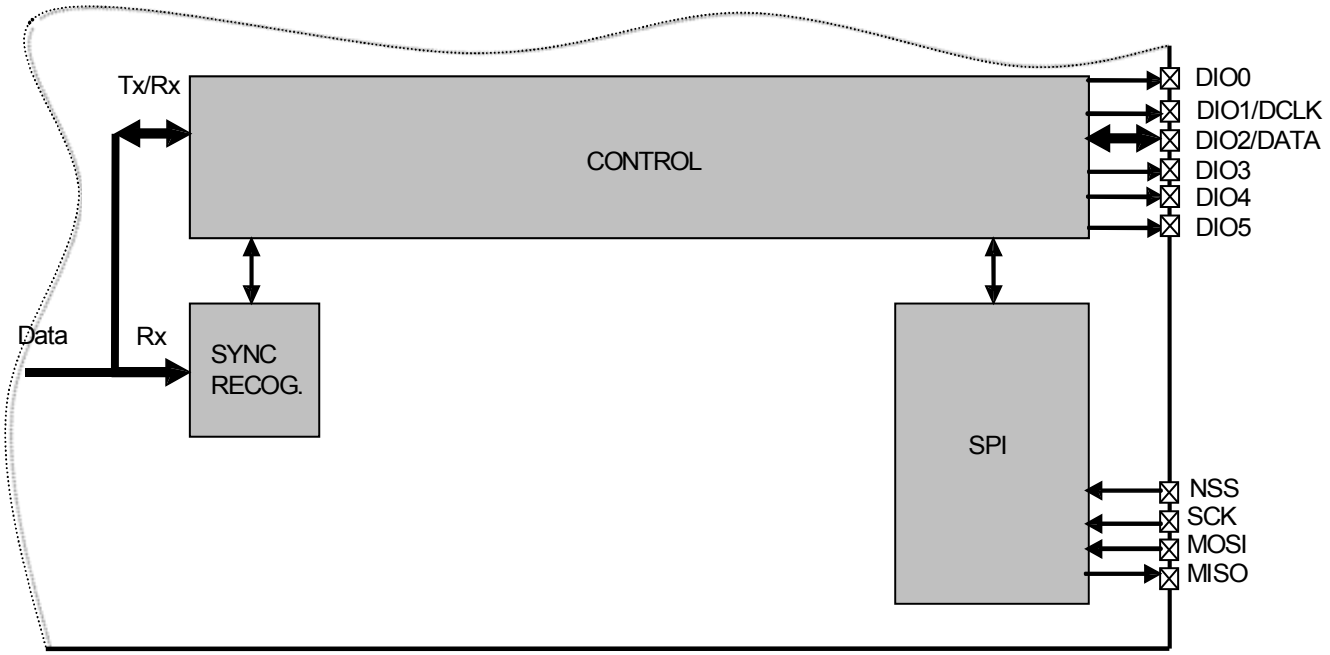



*Figure 29. Continuous Mode Conceptual View*

#### *4.2.12.2. Tx Processing*

In Tx mode, a synchronous data clock for an external uC is provided on DIO1/DCLK pin. Clock timing with respect to the data is illustrated in Figure 30. DATA is internally sampled on the rising edge of DCLK so the uC can change logic state anytime outside the grayed out setup/hold zone.

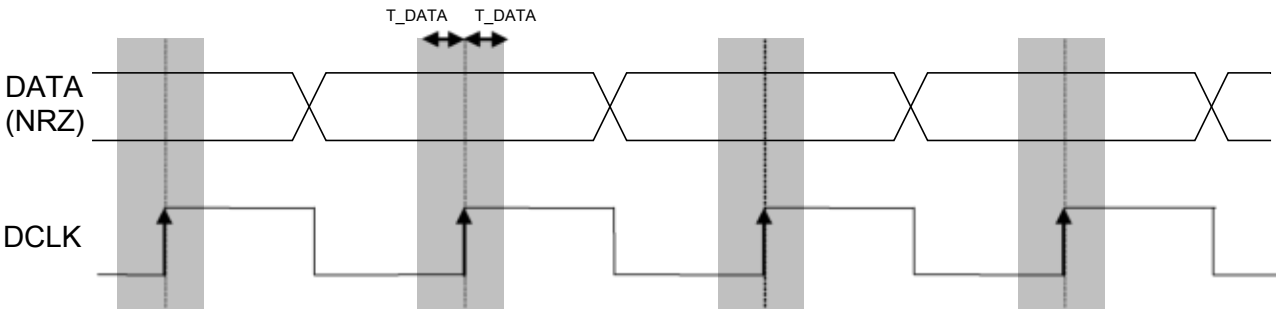



*Figure 30. Tx Processing in Continuous Mode*

*Note the use of DCLK is required when the modulation shaping is enabled (see section 3.4.5).*

*4.2.12.3. Rx Processing*

If the bit synchronizer is disabled, the raw demodulator output is made directly available on DATA pin and no DCLK signal is provided.

Conversely, if the bit synchronizer is enabled, synchronous cleaned data and clock are made available respectively on DIO2/DATA and DIO1/DCLK pins. DATA is sampled on the rising edge of DCLK and updated on the falling edge as illustrated below.

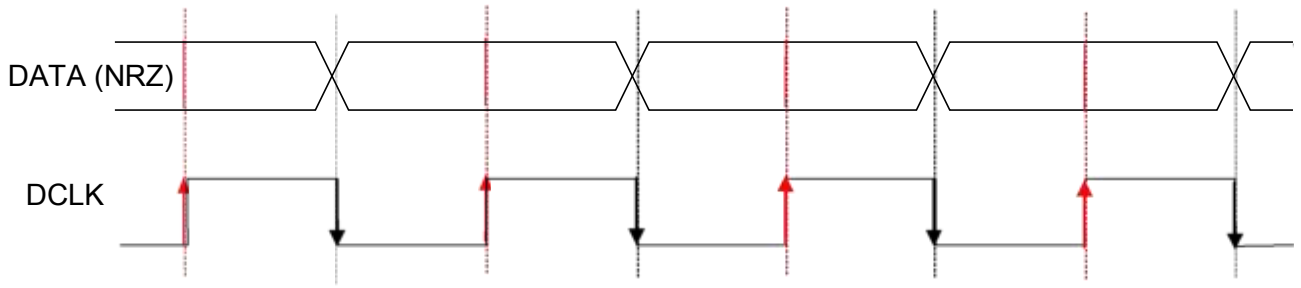



*Figure 31. Rx Processing in Continuous Mode*

*Note* *In Continuous mode it is always recommended to enable the bit synchronizer to clean the DATA signal even if the DCLK signal is not used by the uC (bit synchronizer is automatically enabled in Packet mode).*

### 4.2.13. Packet Mode

*4.2.13.1. General Description*

In Packet mode the NRZ data to (from) the (de)modulator is not directly accessed by the uC but stored in the FIFO and accessed via the SPI interface.

In addition, the RFM95W/96W/98W packet handler performs several packet oriented tasks such as Preamble and Sync word generation, CRC calculation/check, whitening/dewhitening of data, Manchester encoding/decoding, address filtering, etc. This simplifies software and reduces uC overhead by performing these repetitive tasks within the RF chip itself.

Another important feature is ability to fill and empty the FIFO in Sleep/Stdby mode, ensuring optimum power consumption and adding more flexibility for the software.

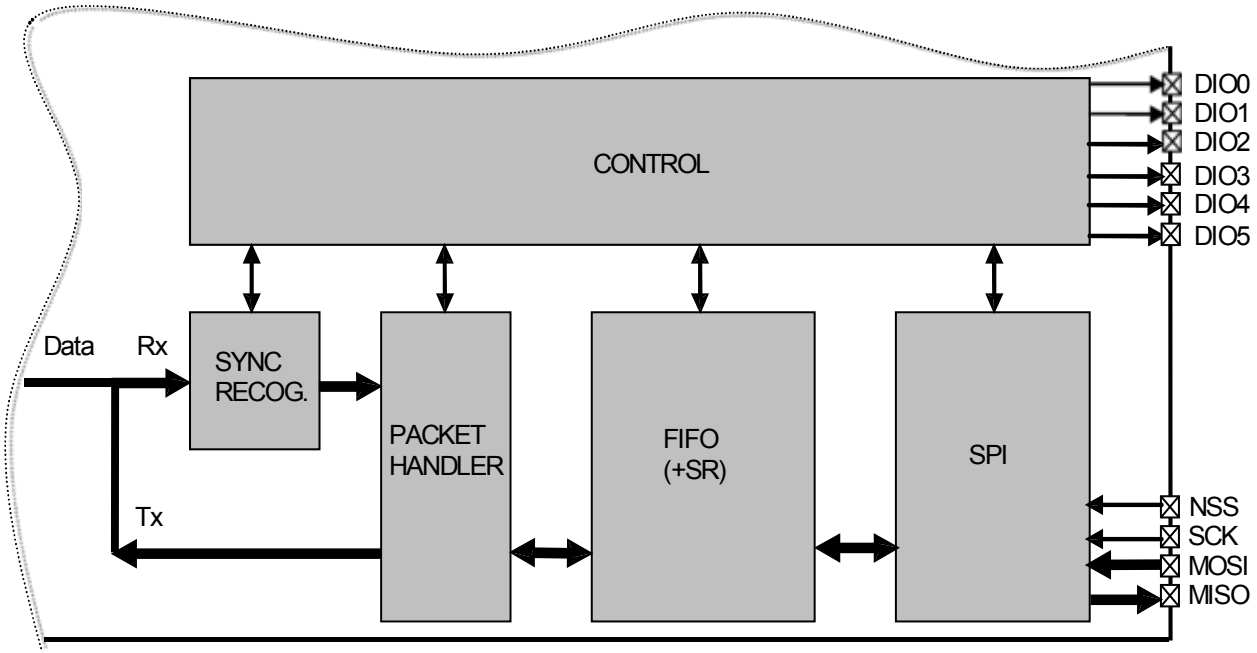



*Figure 32. Packet Mode Conceptual View*

*Note The Bit Synchronizer is automatically enabled in Packet mode.*

*4.2.13.2. Packet Format*

**Fixed Length Packet Format**

Fixed length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to any value greater than 0.

In applications where the packet length is fixed in advance, this mode of operation may be of interest to minimize RF overhead (no length byte field is required). All nodes, whether Tx only, Rx only, or Tx/Rx should be programmed with the same packet length value.

The length of the payload is limited to 2047 bytes.

The length programmed in *PayloadLength* relates only to the payload which includes the message and the optional address byte. In this mode, the payload must contain at least one byte, i.e. address or message byte.

An illustration of a fixed length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Optional Address byte (Node ID)
- Message data
- Optional 2-bytes CRC checksum

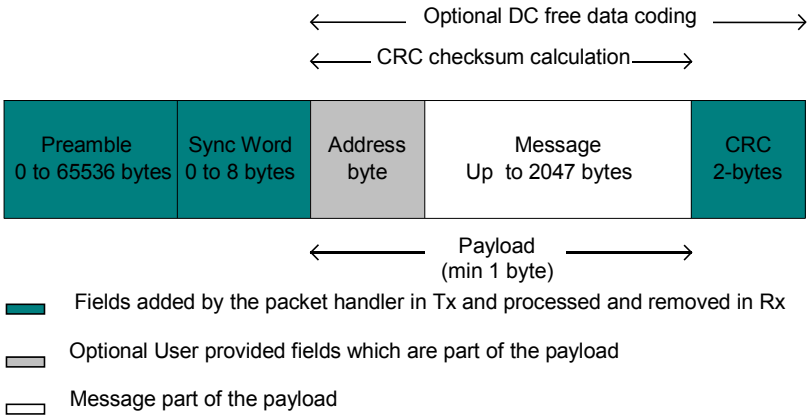



*Figure 33. Fixed Length Packet Format*

**Variable Length Packet Format**

Variable length packet format is selected when bit *PacketFormat* is set to 1.

This mode is useful in applications where the length of the packet is not known in advance and can vary over time. It is then necessary for the transmitter to send the length information together with each packet in order for the receiver to operate properly.

In this mode the length of the payload, indicated by the length byte, is given by the first byte of the FIFO and is limited to 255 bytes. Note that the length byte itself is not included in its calculation. In this mode, the payload must contain at least 2 bytes, i.e. length + address or message byte.

An illustration of a variable length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Length byte
- Optional Address byte (Node ID)
- Message data

- Optional 2-bytes CRC checksum

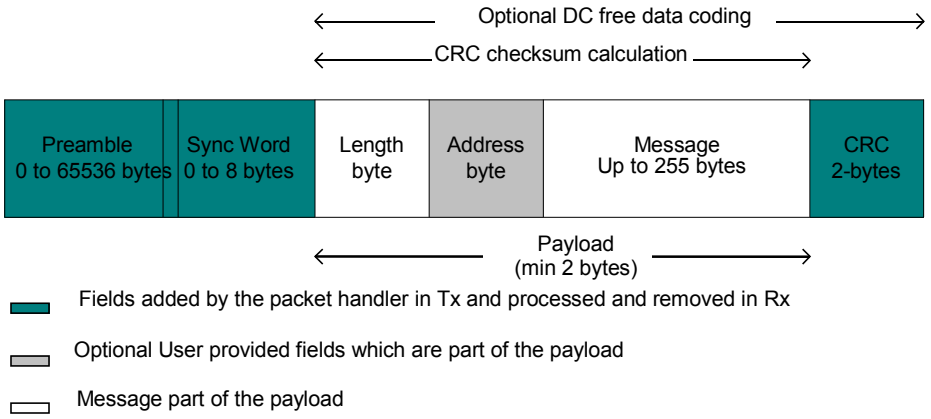



*Figure 34. Variable Length Packet Format*

**Unlimited Length Packet Format**

Unlimited length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to 0. The user can then transmit and receive packet of arbitrary length and *PayloadLength* register is not used in Tx/Rx modes for counting the length of the bytes transmitted/received.

In Tx the data is transmitted depending on the *TxStartCondition* bit. On the Rx side the data processing features like Address filtering, Manchester encoding and data whitening are not available if the sync pattern length is set to zero (*SyncOn = 0*). The filling of the FIFO in this case can be controlled by the bit *FifoFillCondition*. The CRC detection in Rx is also not supported in this mode of the packet handler, however CRC generation in Tx is operational. The interrupts like *CrcOk* & *PayloadReady* are not available either.

An unlimited length packet shown below is made up of the following fields:

- Preamble (1010...).
- Sync word (Network ID).
- Optional Address byte (Node ID).
- Message data
- Optional 2-bytes CRC checksum (Tx only)

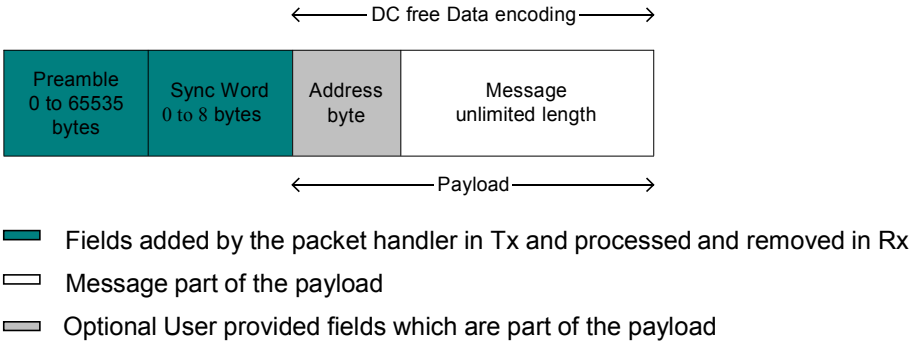



*Figure 35. Unlimited Length Packet Format*

*4.2.13.3. Tx Processing*

In Tx mode the packet handler dynamically builds the packet by performing the following operations on the payload available in the FIFO:

- Add a programmable number of preamble bytes
- Add a programmable Sync word
- Optionally calculating CRC over complete payload field (optional length byte + optional address byte + message) and appending the 2 bytes checksum.
- Optional DC-free encoding of the data (Manchester or whitening)

Only the payload (including optional address and length fields) is required to be provided by the user in the FIFO.

The transmission of packet data is initiated by the Packet Handler only if the chip is in Tx mode and the transmission condition defined by *TxStartCondition* is fulfilled. If transmission condition is not fulfilled then the packet handler transmits a preamble sequence until the condition is met. This happens only if the preamble length /= 0, otherwise it transmits a zero or one until the condition is met to transmit the packet data.

The transmission condition itself is defined as:

- if *TxStartCondition* = 1, the packet handler waits until the first byte is written into the FIFO, then it starts sending the preamble followed by the sync word and user payload
- If *TxStartCondition* = 0, the packet handler waits until the number of bytes written in the FIFO is equal to the number defined in *RegFifoThresh* + 1
- If the condition for transmission was already fulfilled i.e. the FIFO was filled in Sleep/Stdby then the transmission of packet starts immediately on enabling Tx

*4.2.13.4. Rx Processing*

In Rx mode the packet handler extracts the user payload to the FIFO by performing the following operations:

- Receiving the preamble and stripping it off
- Detecting the Sync word and stripping it off
- Optional DC-free decoding of data
- Optionally checking the address byte
- Optionally checking CRC and reflecting the result on *CrcOk*.

Only the payload (including optional address and length fields) is made available in the FIFO.

When the Rx mode is enabled the demodulator receives the preamble followed by the detection of sync word. If fixed length packet format is enabled then the number of bytes received as the payload is given by the *PayloadLength* parameter.

In variable length mode the first byte received after the sync word is interpreted as the length of the received packet. The internal length counter is initialized to this received length. The *PayloadLength* register is set to a value which is greater than the maximum expected length of the received packet. If the received length is greater than the maximum length stored in *PayloadLength* register the packet is discarded otherwise the complete packet is received.

If the address check is enabled then the second byte received in case of variable length and first byte in case of fixed length is the address byte. If the address matches to the one in the *NodeAddress* field, reception of the data continues otherwise it's stopped. The CRC check is performed if *CrcOn* = 1 and the result is available in *CrcOk* indicating that the

CRC was successful. An interrupt (*PayloadReady*) is also generated on DIO0 as soon as the payload is available in the FIFO. The payload available in the FIFO can also be read in Sleep/Standby mode.

If the CRC fails the *PayloadReady* interrupt is not generated and the FIFO is cleared. This function can be overridden by setting *CrcAutoClearOff* = 1, forcing the availability of *PayloadReady* interrupt and the payload in the FIFO even if the CRC fails.

#### *4.2.13.5. Handling Large Packets*

When *PayloadLength* exceeds FIFO size (64 bytes) whether in fixed, variable or unlimited length packet format, in addition to *PacketSent* in Tx and *PayloadReady* or *CrcOk* in Rx, the FIFO interrupts/flags can be used as described below:

- For Tx:

FIFO can be prefilled in Sleep/Standby but must be refilled “on-the-fly” during Tx with the rest of the payload.

1) Pre-fill FIFO (in Sleep/Standby first or directly in Tx mode) until *FifoThreshold* or *FifoFull* is set

2) In Tx, wait for *FifoThreshold* or *FifoEmpty* to be set (i.e. FIFO is nearly empty)

3) Write bytes into the FIFO until *FifoThreshold* or *FifoFull* is set.

4) Continue to step 2 until the entire message has been written to the FIFO (*PacketSent* will fire when the last bit of the packet has been sent).

- For Rx:

FIFO must be unfilled “on-the-fly” during Rx to prevent FIFO overrun.

1) Start reading bytes from the FIFO when *FifoEmpty* is cleared or *FifoThreshold* becomes set.

2) Suspend reading from the FIFO if *FifoEmpty* fires before all bytes of the message have been read

3) Continue to step 1 until *PayloadReady* or *CrcOk* fires

4) Read all remaining bytes from the FIFO either in Rx or Sleep/Standby mode

#### *4.2.13.6. Packet Filtering*

The RFM95W/96W/98W packet handler offers several mechanisms for packet filtering, ensuring that only useful packets are made available to the uC, reducing significantly system power consumption and software complexity.

**Sync Word Based**

Sync word filtering/recognition is used for identifying the start of the payload and also for network identification. As previously described, the Sync word recognition block is configured (size, value) in *RegSyncConfig* and *RegSyncValue(i)* registers. This information is used, both for appending Sync word in Tx, and filtering packets in Rx.

Every received packet which does not start with this locally configured Sync word is automatically discarded and no interrupt is generated.

When the Sync word is detected, payload reception automatically starts and *SyncAddressMatch* is asserted.

*Note Sync Word values containing 0x00 byte(s) are forbidden*

**Address Based**

Address filtering can be enabled via the *AddressFiltering* bits. It adds another level of filtering, above Sync word (i.e. Sync must match first), typically useful in a multi-node networks where a network ID is shared between all nodes (Sync word) and each node has its own ID (address).

Two address based filtering options are available:

- *AddressFiltering* = *01*: Received address field is compared with internal register *NodeAddress*. If they match then the packet is accepted and processed, otherwise it is discarded.
- *AddressFiltering* = *10*: Received address field is compared with internal registers *NodeAddress* and *BroadcastAddress*. If either is a match, the received packet is accepted and processed, otherwise it is discarded. This additional check with a constant is useful for implementing broadcast in a multi-node networks

Please note that the received address byte, as part of the payload, is not stripped off the packet and is made available in the FIFO. In addition, *NodeAddress* and *AddressFiltering* only apply to Rx. On Tx side, if address filtering is expected, the address byte should simply be put into the FIFO like any other byte of the payload.

As address filtering requires a Sync word match, both features share the same interrupt flag *SyncAddressMatch*.

**Length Based**

In variable length Packet mode, *PayloadLength* must be programmed with the maximum payload length permitted. If received length byte is smaller than this maximum then the packet is accepted and processed, otherwise it is discarded.

Please note that the received length byte, as part of the payload, is not stripped off the packet and is made available in the FIFO.

To disable this function the user should set the value of the *PayloadLength* to 2047.

**CRC Based**

The CRC check is enabled by setting bit *CrcOn* in *RegPacketConfig1*. It is used for checking the integrity of the message.

- On Tx side a two byte CRC checksum is calculated on the payload part of the packet and appended to the end of the message
- On Rx side the checksum is calculated on the received payload and compared with the two checksum bytes received. The result of the comparison is stored in bit *CrcOk.*

By default, if the CRC check fails then the FIFO is automatically cleared and no interrupt is generated. This filtering function can be disabled via *CrcAutoClearOff* bit and in this case, even if CRC fails, the FIFO is not cleared and only *PayloadReady* interrupt goes high. Please note that in both cases, the two CRC checksum bytes are stripped off by the packet handler and only the payload is made available in the FIFO. Two CRC implementations are selected with bit *CrcWhiteningType*.

*Table 30 CRC Description*

| Crc Type | *CrcWhiteningType* | Polynomial | Seed Value | Complemented |
|---|---|---|---|---|
| **CCITT** | 0 (default) | $X^{16} + X^{12} + X^5 + 1$ | 0x1D0F | Yes |
| **IBM** | 1 | $X^{16} + X^{15} + X^2 + 1$ | 0xFFFF | No |

A C code implementation of each CRC type is proposed in Application Section 7.

*4.2.13.7. DC-Free Data Mechanisms*

The payload to be transmitted may contain long sequences of 1's and 0's, which introduces a DC bias in the transmitted signal. The radio signal thus produced has a non uniform power distribution over the occupied channel bandwidth. It also introduces data dependencies in the normal operation of the demodulator. Thus it is useful if the transmitted data is random and DC free.

For such purposes, two techniques are made available in the packet handler: Manchester encoding and data whitening.

*Note Only one of the two methods can be enabled at a time.*

**Manchester Encoding**

Manchester encoding/decoding is enabled if *DcFree = 01* and can only be used in Packet mode.

The NRZ data is converted to Manchester code by coding '1' as “10” and '0' as “01”.

In this case, the maximum chip rate is the maximum bit rate given in the specifications section and the actual bit rate is half the chip rate.

Manchester encoding and decoding is only applied to the payload and CRC checksum while preamble and Sync word are kept NRZ. However, the chip rate from preamble to CRC is the same and defined by *BitRate* in *RegBitRate* (Chip Rate = Bit Rate NRZ = 2 x Bit Rate Manchester).

Manchester encoding/decoding is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

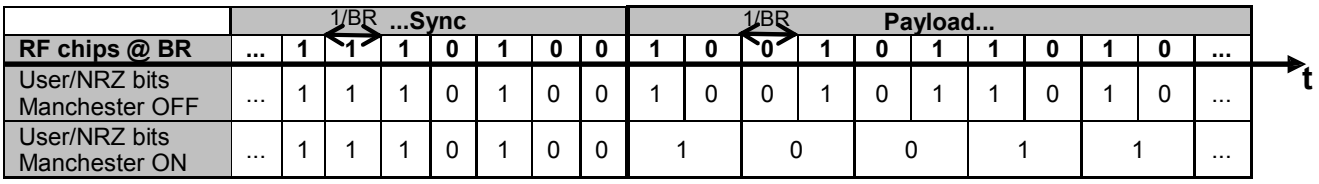



*Figure 36. Manchester Encoding/Decoding*

**Data Whitening**

Another technique called whitening or scrambling is widely used for randomizing the user data before radio transmission. The data is whitened using a random sequence on the Tx side and de-whitened on the Rx side using the same sequence. Comparing to Manchester technique it has the advantage of keeping NRZ data rate i.e. actual bit rate is not halved.

The whitening/de-whitening process is enabled if *DcFree = 10*. A 9-bit LFSR is used to generate a random sequence. The payload and 2-byte CRC checksum is then XORed with this random sequence as shown below. The data is de-whitened on the receiver side by XORing with the same random sequence.

Payload whitening/de-whitening is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

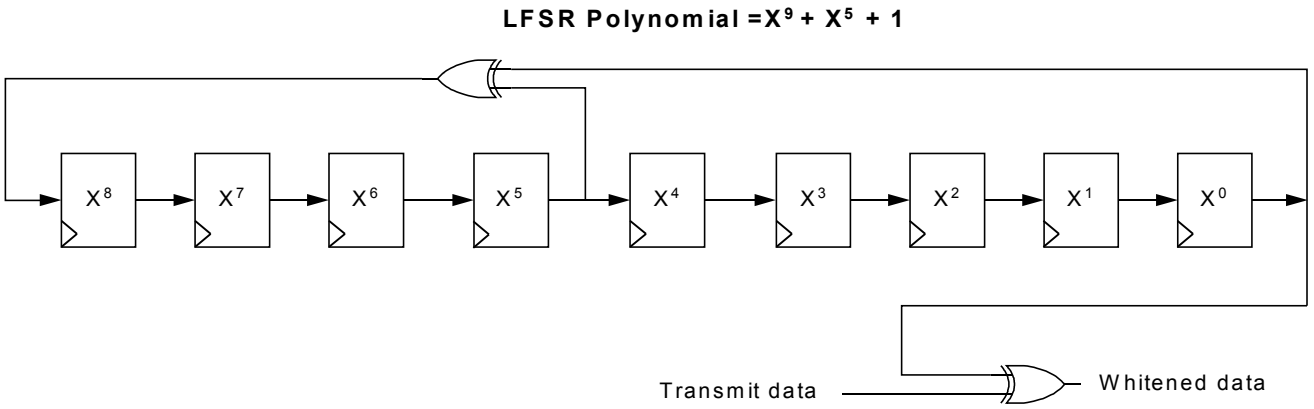



*Figure 37. Data Whitening Polynomial*

*4.2.13.8. Beacon Tx Mode*

In some short range wireless network topologies a repetitive message, also known as beacon, is transmitted periodically by a transmitter. The Beacon Tx mode allows for the re-transmission of the same packet without having to fill the FIFO multiple times with the same data.

When *BeaconOn* in *RegPacketConfig2* is set to 1, the FIFO can be filled only once in Sleep or Stdby mode with the required payload. After a first transmission, *FifoEmpty* will go high as usual, but the FIFO content will be restored when the chip exits Transmit mode. *FifoEmpty*, *FifoFull* and *FifoLevel* flags are also restored.

This feature is only available in Fixed packet format, with the Payload Length smaller than the FIFO size. The control of the chip modes (Tx-Sleep-Tx....) can either be undertaken by the microcontroller, or be automated in the Top Sequencer. See example in section 4.2.13.8.

The Beacon Tx mode is exited by setting *BeaconOn* to 0, and clearing the FIFO by setting *FifoOverrun* to 1.

### 4.2.14. io-homecontrol® Compatibility Mode

The RFM95W/96W/98W features a io-homecontrol® compatibility mode. Please contact your local Hope RF representative for details on its implementation.

## 4.3. SPI Interface

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

Three access modes to the registers are provided:

- SINGLE access: 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.
- BURST access: the address byte is followed by several data bytes. The address is automatically incremented internally between each data byte. This mode is available for both read and write accesses. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.
- FIFO access: if the address byte corresponds to the address of the FIFO, then succeeding data byte will address the FIFO. The address is not automatically incremented but is memorized and does not need to be sent between each data byte. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.

The figure below shows a typical SPI single access to a register.

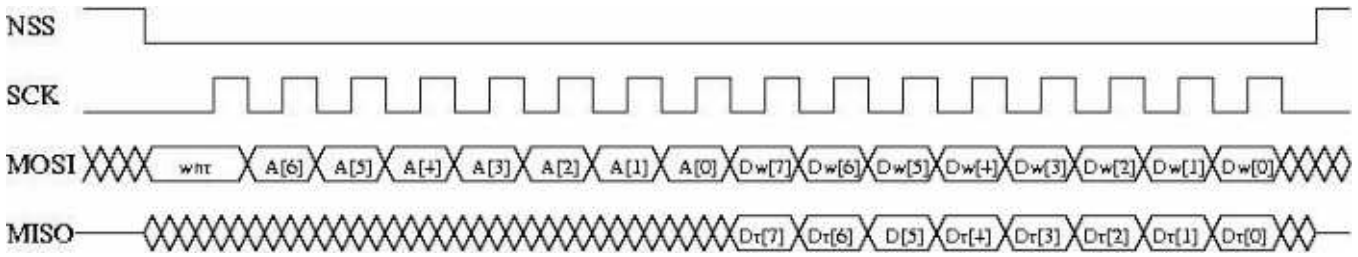



*Figure 38. SPI Timing Diagram (single access)*

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 first byte is the address byte. It is comprises:

- A wnr bit, which is 1 for write access and 0 for read access.
- Then 7 bits of address, MSB first.

The second byte is a data byte, either sent on MOSI by the master in case of a write access or received by the master on MISO in case of read access. The data byte is transmitted MSB first.

Proceeding bytes may be sent on MOSI (for write access) or received on MISO (for read access) without a rising NSS edge and re-sending the address. In FIFO mode, if the address was the FIFO address then the bytes will be written / read at the FIFO address. In Burst mode, if the address was not the FIFO address, then it is automatically incremented for each new byte received.

The frame ends when NSS goes high. The next frame must start with an address byte. The SINGLE access mode is therefore a special case of FIFO / BURST mode with only 1 data byte transferred.

During the write access, the byte transferred from the slave to the master on the MISO line is the value of the written register before the write operation.

# 5. RFM95W/96W/98W Analog & RF Frontend Electronics

## 5.1. Power Supply Strategy

The RFM95W/96W/98W employs an internal voltage regulation scheme which provides stable operating voltage, and hence device characteristics, over the full industrial temperature and operating voltage range of operation. This includes up to
+17 dBm of RF output power which is maintained from 1.8 V to 3.7 V and +20 dBm from 2.4 V to 3.7 V.

The RFM95W/96W/98W can be powered from any low-noise voltage source via pins VBAT_ANA, VBAT_RF and VBAT_DIG. Decoupling capacitors should be connected, as suggested in the reference design of the applications section of this document, on VR_PA, VR_DIG and VR_ANA pins to ensure correct operation of the built-in voltage regulators.

## 5.2. Low Battery Detector

A low battery detector is also included allowing the generation of an interrupt signal in response to the supply voltage dropping below a programmable threshold that is adjustable through the register *RegLowBat*. The interrupt signal can be mapped to any of the DIO pins by programming *RegDioMapping*.

## 5.3. Frequency Synthesis

### 5.3.1. Crystal Oscillator

The crystal oscillator is the main timing reference of the RFM95W/96W/98W. It is used as the reference for the PLL's frequency synthesis and as the clock signal for all digital processing.

The crystal oscillator startup time, TS_OSC, depends on the electrical characteristics of the crystal reference used, for more information on the electrical specification of the crystal see Section 2.3. The crystal connects to the Pierce oscillator on pins XTA and XTB. The RFM95W/96W/98W optimizes the startup time and automatically triggers the PLL when the oscillator signal is stable.

### 5.3.2. CLKOUT Output

The reference frequency, or a fraction of it, can be provided on DIO5 (pin 13) by modifying bits *ClkOut* in *RegDioMapping2*. Two typical applications of the CLKOUT output include:

- To provide a clock output for a companion processor, thus saving the cost of an additional oscillator. CLKOUT can be made available in any operation mode except Sleep mode and is automatically enabled at power on reset.
- To provide an oscillator reference output. Measurement of the CLKOUT signal enables simple software trimming of the initial crystal tolerance.

*Note* *To minimize the current consumption of the RFM95W/96W/98W, please ensure that the CLKOUT signal is disabled when not required.*

### 5.3.3. PLL

The local oscillator of the RFM95W/96W/98W is derived from two almost identical fractional-N PLLs that are referenced to the crystal oscillator circuit. Both PLLs feature a programmable bandwidth setting where one of four discrete preset bandwidths may be accessed.

The RFM95W/96W/98W PLL uses a 19-bit sigma-delta modulator whose frequency resolution, constant over the whole frequency range, is given by:

$$F_{STEP} = \frac{F_{XOSC}}{2^{19}}$$

The carrier frequency is programmed through *RegFrf*, split across addresses 0x06 to 0x08:

$$F_{RF} = F_{STEP} \times Frf(23,0)$$

*Note* *The Frf setting is split across 3 bytes. A change in the center frequency will only be taken into account when the least significant byte FrfLsb in RegFrfLsb is written. This allows the potential for user generation of m-ary FSK at very low bit rates. This is possible where frequency modulation is achieved by direct programming of the programmed RF centre frequency. To enable this functionality set the FastHopOn bit of register RegPllHop.*

### 5.3.4. RC Oscillator

All timing operations in the low-power Sleep state of the Top Level Sequencer rely on the accuracy of the internal low-power RC oscillator. This oscillator is automatically calibrated at the device power-up not requiring any user input.

## 5.4. Transmitter Description

The transmitter of RFM95W/96W/98W comprises the frequency synthesizer, modulator (both LoRa™ and FSK/OOK) and power amplifier blocks, together with the DC biasing and ramping functionality that is provided through the VR_PA block.

### 5.4.1. Architecture Description

The architecture of the RF front end is shown in the following diagram.

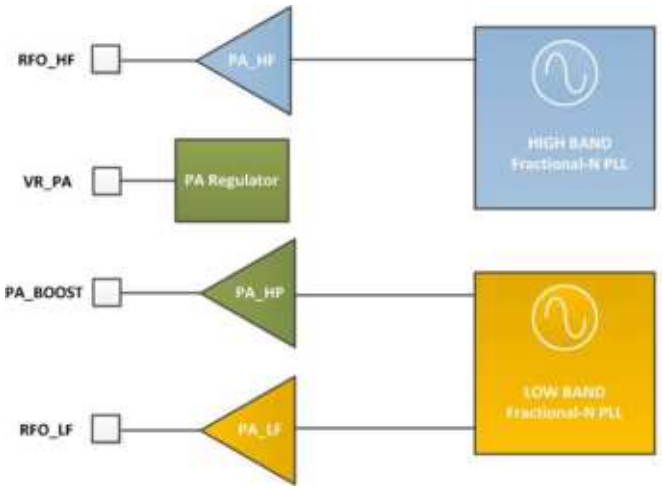



Figure 40. RF Front-end Architecture Shows the Internal PA Configuration.

### 5.4.2. RF Power Amplifiers

PA_HF and PA_LF are high efficiency amplifiers capable of yielding RF power programmable in 1 dB steps from -4 to +14dBm directly into a 50 ohm load with low current consumption. PA_LF covers the lower bands (up to 525 MHz), whilst PA_HF will cover the upper bands (from 860 MHz). The output power is sensitive to the power supply voltage, and typically their performance is expressed at 3.3V.

PA_HP (High Power), connected to the PA_BOOST pin, covers all frequency bands that the chip addresses. It permits continuous operation at up to +17 dBm and duty cycled operation at up to +20dBm. For full details of operation at +20dBm please consult Section 5.4.3

*Table 31 Power Amplifier Mode Selection Truth Table*

| ***PaSelect*** | **Mode** | **Power Range** | **Pout Formula** |
|---|---|---|---|
| 0 | PA_HF or PA_LF on RFO_HF or RFO_LF | -4 to +15dBm | Pout=Pmax-(15-OutputPower)<br>Pmax=10.8+0.6*MaxPower [dBm] |
| 1 | PA_HP on PA_BOOST, any frequency | +2 to +17dBm | Pout=17-(15-OutputPower) [dBm] |

*Notes - For +20 dBm restrictions on operation please consult the following section.*

*- To ensure correct operation at the highest power levels ensure that the current limiter OcpTrim is adjusted to permit delivery of the requisite supply current.*

*- If the PA_BOOST pin is not used it may be left floating.*

### 5.4.3. High Power +20 dBm Operation

The RFM95W/96W/98W have a high power +20 dBm capability on PA_BOOST pin, with the following settings:

*Table 32 High Power Settings*

| ***Register*** | ***Address*** | ***Value for High Power*** | ***Default value PA_HF/LF or +17dBm*** | **Description** |
|---|---|---|---|---|
| *RegPaDac* | 0x4d | 0x87 | 0x84 | Set Pmax to +20dBm for PA_HP |

*Notes - High Power settings must be turned off when using PA_LF or PA_HF*

*- The Over Current Protection limit should be adapted to the actual power level, in RegOcp*

Specific Absolute Maximum Ratings and Operating Range restrictions apply to the +20 dBm operation. They are listed in Table 33 and Table 34.

*Table 33 Operating Range, +20dBm Operation*

| **Symbol** | **Description** | **Min** | **Max** | **Unit** |
|---|---|---|---|---|
| DC_20dBm | Duty Cycle of transmission at +20 dBm output | - | 1 | % |
| VSWR_20dBm | Maximum VSWR at antenna port, +20 dBm output | - | 3:1 | - |

*Table 34 Operating Range, +20dBm Operation*

| **Symbol** | **Description** | **Min** | **Max** | **Unit** |
|---|---|---|---|---|
| VDDop_20dBm | Supply voltage, +20 dBm output | 2.4 | 3.7 | V |

The duty cycle of transmission at +20 dBm is limited to 1%, with a maximum VSWR of 3:1 at antenna port, over the standard operating range [-40;+85°C]. For any other operating condition, contact your Hope RF representative.

### 5.4.4. Over Current Protection

The power amplifiers of RFM95W/96W/98W are protected against current over supply in adverse RF load conditions by the over current protection block. This has the added benefit of protecting battery chemistries with limited peak current capability and minimising worst case PA consumption in battery life calculation. The current limiter value is controlled by the *OcpTrim* bits in *RegOcp*, and is calculated according to the following formulae:

*Table 35 Trimming of the OCP Current*

| ***OcpTrim*** | **$I_{MAX}$** | **Imax Formula** |
|---|---|---|
| 0 to 15 | 45 to 120 mA | 45 + 5**OcpTrim* [mA] |
| 16 to 27 | 130 to 240 mA | -30 + 10**OcpTrim* [mA] |
| 27+ | 240 mA | *240 mA* |

*Note Imax sets a limit on the current drain of the Power Amplifier only, hence the maximum current drain of the* RFM95W/96W/98W *is equal to Imax +* $I_{FS}$.

## 5.5. Receiver Description

### 5.5.1. Overview

The RFM95W/96W/98W features a digital receiver with the analog to digital conversion process being performed directly following the LNA-Mixers block. In addition to the LoRa™ modulation scheme the low-IF receiver is able to demodulate ASK, OOK, (G)FSK and (G)MSK modulation. All filtering, demodulation, gain control, synchronization and packet handling is performed digitally allowing a high degree of programmable flexibility. The receiver also has automatic gain calibration, this improves the precision of RSSI measurement and enhances image rejection.

### 5.5.2. Receiver Enabled and Receiver Active States

In the receiver operating mode two states of functionality are defined. Upon initial transition to receiver operating mode the receiver is in the 'receiver-enabled' state. In this state the receiver awaits for either the user defined valid preamble or RSSI detection criterion to be fulfilled. Once met the receiver enters 'receiver-active' state. In this second state the received signal is processed by the packet engine and top level sequencer. For a complete description of the digital functions of the RFM95W/96W/98W receiver please see Section 4 of the datasheet.

### 5.5.3. Automatic Gain Control In FSK/OOK Mode

The AGC feature allows receiver to handle a wide Rx input dynamic range from the sensitivity level up to maximum input level of 0dBm or more, whilst optimizing the system linearity.

The following table shows typical NF and IIP3 performances for the RFM95W/96W/98W LNA gains available.

*Table 36 LNA Gain Control and Performances*

| *RX input level (Pin)* | *Gain Setting* | *LnaGain* | *Relative LNA Gain [dB]* | NF Band 3/2/1 [dB] | IIP3 Band 3/2/1 [dBm] |
|---|---|---|---|---|---|
| **Pin <= AgcThresh1** | G1 | '001' | 0 dB | 4/5.5/7 | -15/-22/-11 |
| **AgcThresh1 < Pin <= AgcThresh2** | G2 | '010' | -6 dB | 6.5/8/12 | -11/-15/-6 |
| **AgcThresh2 < Pin <= AgcThresh3** | G3 | '011' | -12 dB | 11/12/17 | -11/-12/0 |
| **AgcThresh3 < Pin <= AgcThresh4** | G4 | '100' | -24 dB | 20/21/27 | 2/3/9 |
| **AgcThresh4 < Pin <= AgcThresh5** | G5 | '110' | -26 dB | 32/33/35 | 10/10/14 |
| **AgcThresh5 < Pin** | G6 | '111' | -48 dB | 44/45/43 | 11/12/14 |

### 5.5.4. RSSI in FSK/OOK Mode

The RSSI provides a measure of the incoming signal power at RF input port, measured within the receiver bandwidth. The signal power is available in *RssiValue*. This value is absolute in units of dBm and with a resolution of 0.5 dB. The formula below relates the register value to the absolute input signal level at the RF input port:

$$RssiValue = -2 \cdot RF\,level\,[dBm] + RssiOffset\,[dB]$$

The RSSI value can be compensated to take into account the loss in the matching network or even the gain of an additional LNA by using *RssiOffset*. The offset can be chosen in 1 dB steps from -16 to +15 dB. When compensation is applied, the effective signal strength is read as follows:

$$RSSI[dBm] = -\frac{RssiValue}{2}$$

The RSSI value is smoothed on a user defined number of measured RSSI samples. The precision of the RSSI value is related to the number of RSSI samples used. *RssiSmoothing* selects the number of RSSI samples from a minimum of 2 samples up to 256 samples in increments of power of 2. Table 37 gives the estimation of the RSSI accuracy for a 10 dB SNR and response time versus the number of RSSI samples programmed in *RssiSmoothing*.

*Table 37 RssiSmoothing Options*

| *RssiSmoothing* | Number of Samples | Estimated Accuracy | Response Time |
|---|---|---|---|
| '000' | 2 | ± 6 dB | $\frac{2^{(RssiSmoothing+1)}}{4 \cdot RxBw[kHz]}[ms]$ |
| '001' | 4 | ± 5 dB | |
| '010' | 8 | ± 4 dB | |
| '011' | 16 | ± 3 dB | |
| '100' | 32 | ± 2 dB | |
| '101' | 64 | ± 1.5 dB | |
| '110' | 128 | ± 1.2 dB | |
| '111' | 256 | ± 1.1 dB | |

The RSSI is calibrated when the image and RSSI calibration process is launched.

### 5.5.5. RSSI in LoRa™ Mode

The RSSI values reported by the LoRa™ modem differ from those expressed by the FSK/OOK modem. The following formula shows the method used to interpret the LoRa™ RSSI values:

RSSI (dBm) = -157 + *Rssi*, (when using the High Frequency (HF) port)

or

RSSI (dBm) = -164 + *Rssi*, (when using the Low Frequency (LF) port)

The same formula can be re-used to evaluate the signal strength of the received packet:

Packet Strength (dBm) = -157 + *Rssi*, (when using the High Frequency (HF) port)

or

Packet Strength (dBm) = -164 + *Rssi*, (when using the Low Frequency (LF) port)

Due to the nature of the LoRa modulation, it is possible to receive packets below the noise floor. In this situation, the SNR is used in conjunction of the PacketRssi to compute the signal strength of the received packet:

Packet Strength (dBm) = -157 + *PacketRssi* + *PacketSnr* * 0.25 (when using the HF port and SNR < 0)

or

Packet Strength (dBm) = -164 + *PacketRssi* + *PacketSnr* * 0.25 (when using the LF port and SNR < 0)

Note:

1. *PacketRssi* (in RegPktRssiValue), is an averaged version of *Rssi* (in RegRssiValue). *Rssi* can be read at any time (during packet reception or not), and should be averaged to give more precise results.

2. The constants, -157 and -164, may vary with the front-end setup of the RFM95W/96W/98W (*LnaBoost* =1 or 0, presence of an external LNA, mismatch at the LNA input…). It is recommended to adjust these values with a single-point calibration procedure to increase RSSI accuracy.

3. As signal strength increases (RSSI>-100dBm), the linearity of PacketRssi is not guaranteed and results will diverge from the ideal 1dB/dB ideal curve. When very good RSSI precision is required over the whole dynamic range of the receiver, two options are proposed:

- *Rssi* in RegRssiValue offers better linearity. *Rssi* can be sampled during the reception of the payload (between ValidHeader and RxDone IRQ), and used to extract a more high-signal RSSI measurement

When SNR>=0, the standard formula can be adjusted to correct the slope: RSSI = -157+16/15 * PacketRssi (or RSSI = -164+16/15 * PacketRssi)

### 5.5.6. Channel Filter

The role of the channel filter is to reject noise and interference outside of the wanted channel. The RFM95W/96W/98W channel filtering is implemented with a 16-tap finite impulse response (FIR) filter. Rejection of the filter is high enough that the filter stop-band performance is not the dominant influence on adjacent channel rejection performance. This is instead limited by the RFM95W/96W/98W PLL phase noise.

*Note* *To respect sampling criterion in the decimation chain of the receiver, the communication bit rate cannot be set at a higher than twice the single side receiver bandwidth (BitRate < 2 x RxBw)*

The programmed single side bandwidth *RxBw* of the channel filter is determined by the parameters *RxBwMant* and *RxBwExp* in *RegRxBw:*

$$RxBw = \frac{FXOSC}{RxBwMant \times 2^{RxBwExp+2}}$$

The following channel filter bandwidths are hence accessible in the case of a 32 MHz reference oscillator.

*Table 38 Available RxBw Settings*

| ***RxBwMant*** **(binary/value)** | ***RxBwExp*** **(decimal)** | ***RxBw (kHz)*** **FSK / OOK** |
|---|---|---|
| 10b / 24 | 7 | 2.6 |
| 01b / 20 | 7 | 3.1 |
| 00b / 16 | 7 | 3.9 |
| 10b / 24 | 6 | 5.2 |
| 01b / 20 | 6 | 6.3 |
| 00b / 16 | 6 | 7.8 |
| 10b / 24 | 5 | 10.4 |
| 01b / 20 | 5 | 12.5 |
| 00b / 16 | 5 | 15.6 |
| 10b / 24 | 4 | 20.8 |
| 01b / 20 | 4 | 25.0 |
| 00b / 16 | 4 | 31.3 |
| 10b / 24 | 3 | 41.7 |
| 01b / 20 | 3 | 50.0 |
| 00b / 16 | 3 | 62.5 |
| 10b / 24 | 2 | 83.3 |
| 01b / 20 | 2 | 100.0 |
| 00b / 16 | 2 | 125.0 |
| 10b / 24 | 1 | 166.7 |
| 01b / 20 | 1 | 200.0 |
| 00b / 16 | 1 | 250.0 |
| Other settings | | reserved |

### 5.5.7. Temperature Measurement

A stand alone temperature measurement block is used in order to measure the temperature in any mode except Sleep and Standby. It is enabled by default, and can be stopped by setting *TempMonitorOff* to 1. The result of the measurement is stored in *TempValue* in *RegTemp*.

Due to process variations, the absolute accuracy of the result is +/- 10 °C. Higher precision requires a calibration procedure at a known temperature. The figure below shows the influence of just such a calibration process. For more information, including source code, please consult the applications Section of this document.

Example temperature curve, typical device

| | Correction Factor | | 15 |
|---|---|---|---|
| Actual Temp [Celsius] | RegTemp [Dec] | Temp before calibration [°C] | Temp after calibration [°C] |
| 85 | 181 | 74 | 89 |
| 75 | 190 | 65 | 80 |
| 65 | 201 | 54 | 69 |
| 55 | 211 | 44 | 59 |
| 45 | 222 | 33 | 48 |
| 35 | 232 | 23 | 38 |
| 25 | 245 | 10 | 25 |
| 15 | 0 | 0 | 15 |
| 5 | 10 | -10 | 5 |
| -5 | 21 | -21 | -6 |
| -15 | 33 | -33 | -18 |
| -25 | 44 | -44 | -29 |
| -35 | 56 | -56 | -41 |
| -40 | 63 | -63 | -48 |

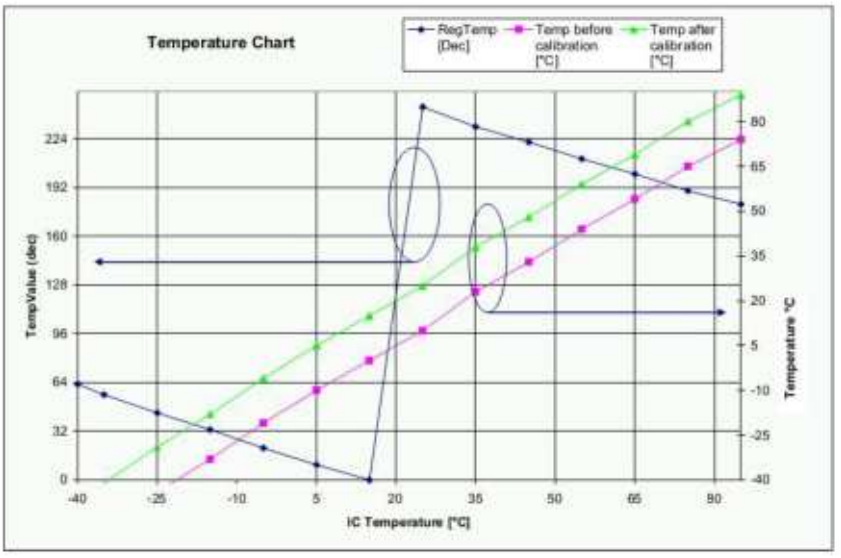



*Figure 41. Temperature Sensor Response*

# 6. Description of the Registers

The register mapping depends upon whether FSK/OOK or LoRa™ mode has been selected. The following table summarises the location and function of each register and gives an overview of the changes in register mapping between both modes of operation.

## 6.1. Register Table Summary

*Table 39 Registers Summary*

| Address | Register Name | | Reset (POR) | Default (FSK) | Description | |
|---|---|---|---|---|---|---|
| | FSK/OOK Mode | LoRa™ Mode | | | FSK Mode | LoRa™ Mode |
| 0x00 | RegFifo | | 0x00 | | FIFO read/write access | |
| 0x01 | RegOpMode | | 0x01 | | Operating mode & LoRa™ / FSK selection | |
| 0x02 | RegBitrateMsb | Unused | 0x1A | | Bit Rate setting, Most Significant Bits | |
| 0x03 | RegBitrateLsb | | 0x0B | | Bit Rate setting, Least Significant Bits | |
| 0x04 | RegFdevMsb | | 0x00 | | Frequency Deviation setting, Most Significant Bits | |
| 0x05 | RegFdevLsb | | 0x52 | | Frequency Deviation setting, Least Significant Bits | |
| 0x06 | RegFrfMsb | | 0x6C | | RF Carrier Frequency, Most Significant Bits | |
| 0x07 | RegFrfMid | | 0x80 | | RF Carrier Frequency, Intermediate Bits | |
| 0x08 | RegFrfLsb | | 0x00 | | RF Carrier Frequency, Least Significant Bits | |
| 0x09 | RegPaConfig | | 0x4F | | PA selection and Output Power control | |
| 0x0A | RegPaRamp | | 0x09 | | Control of PA ramp time, low phase noise PLL | |
| 0x0B | RegOcp | | 0x2B | | Over Current Protection control | |
| 0x0C | RegLna | | 0x20 | | LNA settings | |
| 0x0D | RegRxConfig | RegFifoAddrPtr | 0x08 | 0x0E | AFC, AGC, ctrl | FIFO SPI pointer |
| 0x0E | RegRssiConfig | RegFifoTxBaseAddr | 0x02 | | RSSI | Start Tx data |
| 0x0F | RegRssiCollision | RegFifoRxBaseAddr | 0x0A | | RSSI Collision detector | Start Rx data |
| 0x10 | RegRssiThresh | FifoRxCurrentAddr | 0xFF | | RSSI Threshold control | Start address of last packet received |
| 0x11 | RegRssiValue | RegIrqFlagsMask | n/a | n/a | RSSI value in dBm | Optional IRQ flag mask |
| 0x12 | RegRxBw | RegIrqFlags | 0x15 | | Channel Filter BW Control | IRQ flags |
| 0x13 | RegAfcBw | RegRxNbBytes | 0x0B | | AFC Channel Filter BW | Number of received bytes |
| 0x14 | RegOokPeak | RegRxHeaderCntValueMsb | 0x28 | | OOK demodulator | Number of valid headers received |
| 0x15 | RegOokFix | RegRxHeaderCntValueLsb | 0x0C | | Threshold of the OOK demod | |
| 0x16 | RegOokAvg | RegRxPacketCntValueMsb | 0x12 | | Average of the OOK demod | Number of valid packets received |
| 0x17 | Reserved17 | RegRxPacketCntValueLsb | 0x47 | | - | |
| 0x18 | Reserved18 | RegModemStat | 0x32 | | - | Live LoRaTM modem status |
| 0x19 | Reserved19 | RegPktSnrValue | 0x3E | | - | Espimation of last packet SNR |
| 0x1A | RegAfcFei | RegPktRssiValue | 0x00 | | AFC and FEI control | RSSI of last packet |

| Address | Register Name | | Reset (POR) | Default (FSK) | Description | |
|---|---|---|---|---|---|---|
| | FSK/OOK Mode | LoRa™ Mode | | | FSK Mode | LoRa™ Mode |
| 0x1B | RegAfcMsb | RegRssiValue | 0x00 | n/a | Frequency correction value of the AFC | Current RSSI |
| 0x1C | RegAfcLsb | RegHopChannel | 0x00 | n/a | | FHSS start channel |
| 0x1D | RegFeiMsb | RegModemConfig 1 | 0x00 | n/a | Value of the calculated frequency error | Modem PHY config 1 |
| 0x1E | RegFeiLsb | RegModemConfig 2 | 0x00 | n/a | | Modem PHY config 2 |
| 0x1F | RegPreambleDetect | RegSymbTimeoutLsb | 0x40 | 0xAA | Settings of the Preamble Detector | Receiver timeout value |
| 0x20 | RegRxTimeout1 | RegPreambleMsb | 0x00 | | Timeout Rx request and RSSI | Size of preamble |
| 0x21 | RegRxTimeout2 | RegPreambleLsb | 0x00 | | Timeout RSSI and *PayloadReady* | |
| 0x22 | RegRxTimeout3 | RegPayloadLength | 0x00 | | Timeout RSSI and *SyncAddress* | LoRaTM payload length |
| 0x23 | RegRxDelay | RegMaxPayloadLength | 0x00 | | Delay between Rx cycles | LoRaTM maximum payload length |
| 0x24 | RegOsc | RegHopPeriod | 0x05 | 0x07 | RC Oscillators Settings, CLKOUT frequency | FHSS Hop period |
| 0x25 | RegPreambleMsb | RegFifoRxByteAddr | 0x00 | | Preamble length, MSB | Address of last byte written in FIFO |
| 0x26 | RegPreambleLsb | RegModemConfig3 | 0x03 | | Preamble length, LSB | Modem PHY config 3 |
| 0x27 | RegSyncConfig | RESERVED | 0x93 | | Sync Word Recognition control | RESERVED |
| 0x28 | RegSyncValue1 | RegFeiMsb | 0x55 | 0x01 | Sync Word bytes 1 | Estimated frequency error |
| 0x29 | RegSyncValue2 | RegFeiMid | 0x55 | 0x01 | Sync Word bytes 2 | |
| 0x2A | RegSyncValue3 | RegFeiLsb | 0x55 | 0x01 | Sync Word bytes 3 | |
| 0x2B | RegSyncValue4 | RESERVED | 0x55 | 0x01 | Sync Word bytes 4 | RESERVED |
| 0x2C | RegSyncValue5 | RegRssiWideband | 0x55 | 0x01 | Sync Word bytes 5 | Wideband RSSI measurement |
| 0x2D-0x2E | RegSyncValue6-7 | RESERVED | 0x55 | 0x01 | Sync Word bytes, 6 to 7 | RESERVED |
| 0x2F | RegSyncValue8 | RegIfFreq1 | 0x55 | 0x01 | Sync Word byte 8 | Optimize receiver |
| 0x30 | RegPacketConfig1 | RegIfFreq2 | 0x90 | | Packet mode settings | |
| 0x31 | RegPacketConfig2 | RegDetectOptimize | 0x40 | | Packet mode settings | LoRa detection Optimize for SF6 |
| 0x32 | RegPayloadLength | RESERVED | 0x40 | | Payload length setting | RESERVED |
| 0x33 | RegNodeAdrs | RegInvertIQ | 0x00 | | Node address | Invert LoRa I and Q signals |
| 0x34 | RegBroadcastAdrs | RESERVED | 0x00 | | Broadcast address | RESERVED |
| 0x35 | RegFifoThresh | | 0x0F | 0x1F | Fifo threshold, Tx start condition | |
| 0x36 | RegSeqConfig1 | RegHighBwOptimize1 | 0x00 | | Top level Sequencer settings | Sensitivity optimisation for 500 kHz bandwidth |
| 0x37 | RegSeqConfig2 | RegDetectionThreshold | 0x00 | | Top level Sequencer settings | LoRa detection threshold for SF6 |
| 0x38 | RegTimerResol | RESERVED | 0x00 | | Timer 1 and 2 resolution control | RESERVED |

| Address | Register Name | | Reset (POR) | Default (FSK) | Description | |
|---|---|---|---|---|---|---|
| | FSK/OOK Mode | LoRa™ Mode | | | FSK Mode | LoRa™ Mode |
| 0x39 | RegTimer1Coef | RegSyncWord | 0xF5 | 0x12 | Timer 1 setting | LoRa Sync Word |
| 0x3A | RegTimer2Coef | RegHighBwOpti-mize2 | 0x20 | | Timer 2 setting | Sensitivity optimisation for 500 kHz bandwidth |
| 0x3B | RegImageCal | RegInvertIQ2 | 0x82 | 0x02 | Image calibration engine con-trol | Optimize for inverted IQ |
| 0x3C | RegTemp | RESERVED | - | | Temperature Sensor value | RESERVED |
| 0x3D | RegLowBat | | 0x02 | | Low Battery Indicator Settings | |
| 0x3E | RegIrqFlags1 | | 0x80 | | Status register: PLL Lock state, Timeout, RSSI | |
| 0x3F | RegIrqFlags2 | | 0x40 | | Status register: FIFO handling flags, Low Battery | |
| 0x40 | RegDioMapping1 | | 0x00 | | Mapping of pins DIO0 to DIO3 | |
| 0x41 | RegDioMapping2 | | 0x00 | | Mapping of pins DIO4 and DIO5, ClkOut frequency | |
| 0x42 | RegVersion | | 0x12 | | Semtech ID relating the silicon revision | |
| 0x44 | RegPllHop | Unused | 0x2D | | Control the fast frequency hop-ping mode | Unused |
| 0x4B | RegTcxo | | 0x09 | | TCXO or XTAL input setting | |
| 0x4D | RegPaDac | | 0x84 | | Higher power settings of the PA | |
| 0x5B | RegFormerTemp | | - | | Stored temperature during the former IQ Calibration | |
| 0x5D | RegBitRateFrac | Unused | 0x00 | | Fractional part in the Bit Rate division ratio | Unused |
| 0x61 | RegAgcRef | | 0x13 | | Adjustment of the AGC thresholds | |
| 0x62 | RegAgcThresh1 | | 0x0E | | | |
| 0x63 | RegAgcThresh2 | | 0x5B | | | |
| 0x64 | RegAgcThresh3 | | 0xDB | | | |
| 0x70 | RegPll | | 0xD0 | | Control of the PLL bandwidth | |
| others | RegTest | | - | | Internal test registers. Do not overwrite | |

*Note - Reset values are automatically refreshed in the chip at Power On Reset*

*- Default values are the Hope RF recommended register values, optimizing the device operation*

*- Registers for which the Default value differs from the Reset value are denoted by a * in the tables of section 6.2*

## 6.2. FSK/OOK Mode Register Map

This section details the RFM95W/96W/98W register mapping and the precise contents of each register in FSK/OOK mode. Convention: r: read, w: write, t:trigger, c: clear

*Table 40 Register Map*

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegFifo (0x00) | 7-0 | Fifo | rw | 0x00 | FIFO data input/output |
| Registers for Common settings | | | | | |
| RegOpMode (0x01) | 7 | LongRangeMode | r | 0x00 | 0 → FSK/OOK Mode<br>1→ LoRa™ Mode<br>This bit can be modified only in Sleep mode. A write operation on other device modes is ignored. |
| | 6-5 | ModulationType | rw | 0x00 | Modulation scheme:<br>00 → FSK<br>01 → OOK<br>10 → 11 → reserved |
| | 4 | reserved | r | 0x0 | reserved |
| | 3 | LowFrequencyModeOn | rw | 0x01 | Access Low Frequency Mode registers (from address 0x61 on)<br>0 → High Frequency Mode (access to HF test registers)<br>1 → Low Frequency Mode (access to LF test registers) |
| | 2-0 | Mode | rw | 0x01 | Transceiver modes<br>000 → Sleep mode<br>001 → Stdby mode<br>010 → FS mode TX (FSTx)<br>011 → Transmitter mode (Tx)<br>100 → FS mode RX (FSRx)<br>101 → Receiver mode (Rx)<br>110 → reserved<br>111 → reserved |
| RegBitrateMsb (0x02) | 7-0 | BitRate(15:8) | rw | 0x1a | MSB of Bit Rate (chip rate if Manchester encoding is enabled) |
| RegBitrateLsb (0x03) | 7-0 | BitRate(7:0) | rw | 0x0b | LSB of bit rate (chip rate if Manchester encoding is enabled)<br>$BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$<br>Default value: 4.8 kb/s |
| RegFdevMsb (0x04) | 7-6 | reserved | rw | 0x00 | reserved |
| | 5-0 | Fdev(13:8) | rw | 0x00 | MSB of the frequency deviation |
| RegFdevLsb (0x05) | 7-0 | Fdev(7:0) | rw | 0x52 | LSB of the frequency deviation<br>$Fdev = Fstep \times Fdev(15,0)$<br>Default value: 5 kHz |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegFrfMsb (0x06) | 7-0 | Frf(23:16) | rw | 0x6c | MSB of the RF carrier frequency |
| RegFrfMid (0x07) | 7-0 | Frf(15:8) | rw | 0x80 | MSB of the RF carrier frequency |
| RegFrfLsb (0x08) | 7-0 | Frf(7:0) | rw | 0x00 | LSB of RF carrier frequency<br>$Frf = Fstep \times Frf(23;0)$<br>Default value: 434.000 MHz<br>The RF frequency is taken into account internally only when:<br>- entering FSRX/FSTX modes<br>- re-starting the receiver |
| Registers for the Transmitter | | | | | |
| RegPaConfig (0x09) | 7 | PaSelect | rw | 0x00 | Selects PA output pin<br>0 → RFO pin. Maximum power of +14 dBm<br>1 → PA_BOOST pin. Maximum power of +20 dBm |
| | 6-4 | MaxPower | rw | 0x04 | Select max output power: Pmax=10.8+0.6*MaxPower[dBm] |
| | 3-0 | OutputPower | rw | 0x0f | Pout=Pmax-(15-OutputPower) if PaSelect = 0 (RFO pins)<br>Pout=17-(15-OutputPower) if PaSelect = 1 (PA_BOOST pin) |
| RegPaRamp (0x0A) | 7 | unused | r | 0x00 | unused |
| | 6-5 | ModulationShaping | rw | 0x00 | Data shaping:<br>In FSK:<br>1 → no shaping<br>2 → Gaussian filter BT = 1.0<br>10 → Gaussian filter BT = 0.5<br>11 → Gaussian filter BT = 0.3<br>In OOK:<br>1 → no shaping<br>2 → filtering with fcutoff = bit_rate<br>10 → filtering with fcutoff = 2*bit_rate (for bit_rate < 125 kb/s)<br>11 → reserved |
| | 4 | reserved | rw | 0x00 | reserved |
| | 3-0 | PaRamp | rw | 0x09 | Rise/Fall time of ramp up/down in FSK<br>0000 → 3.4 ms<br>0001 → 2 ms<br>0010 → 1 ms<br>0011 → 500 us<br>0100 → 250 us<br>0101 → 125 us<br>0110 → 100 us<br>0111 → 62 us<br>1000 → 50 us<br>1001 → 40 us (d)<br>1010 → 31 us<br>1011 → 25 us<br>1100 → 20 us<br>1101 → 15 us<br>1110 → 12 us<br>1111 → 10 us |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegOcp (0x0B) | 7-6 | unused | r | 0x00 | unused |
| | 5 | OcpOn | rw | 0x01 | Enables overload current protection (OCP) for the PA:<br>0 → OCP disabled<br>1 → OCP enabled |
| | 4-0 | OcpTrim | rw | 0x0b | Trimming of OCP current:<br>$I_{max}$ = 45+5*OcpTrim [mA] if OcpTrim <= 15 (120 mA)/<br>$I_{max}$ = -30+10*OcpTrim [mA] if 15 < OcpTrim <= 27 (130 to 240 mA)<br>$I_{max}$ = 240mA for higher settings<br>Default $I_{max}$ = 100mA |
| Registers for the Receiver | | | | | |
| RegLna (0x0C) | 7-5 | LnaGain | rw | 0x01 | LNA gain setting:<br>000 → reserved<br>001 → G1 = highest gain<br>010 → G2 = highest gain – 6 dB<br>011 → G3 = highest gain – 12 dB<br>100 → G4 = highest gain – 24 dB<br>101 → G5 = highest gain – 36 dB<br>110 → G6 = highest gain – 48 dB<br>111 → reserved<br>Note:<br>Reading this address always returns the current LNA gain (which may be different from what had been previously selected if AGC is enabled. |
| | 4-3 | LnaBoostLf | rw | 0x00 | Low Frequency (RFI_LF) LNA current adjustment<br>00 → Default LNA current<br>Other → Reserved |
| | 2 | reserved | rw | 0x00 | reserved |
| | 1-0 | LnaBoostHf | rw | 0x00 | High Frequency (RFI_HF) LNA current adjustment<br>00 → Default LNA current<br>11 → Boost on, 150% LNA current |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegRxConfig (0x0d) | 7 | RestartRxOnCollision | rw | 0x00 | Turns on the mechanism restarting the receiver automatically if it gets saturated or a packet collision is detected<br>0 → No automatic Restart<br>1 → Automatic restart On |
| | 6 | RestartRxWithoutPllLock | wt | 0x00 | Triggers a manual Restart of the Receiver chain when set to 1. Use this bit when there is no frequency change, RestartRxWithPllLock otherwise. |
| | 5 | RestartRxWithPllLock | wt | 0x00 | Triggers a manual Restart of the Receiver chain when set to 1. Use this bit when there is a frequency change, requiring some time for the PLL to re-lock. |
| | 4 | AfcAutoOn | rw | 0x00 | 0 → No AFC performed at receiver startup<br>1 → AFC is performed at each receiver startup |
| | 3 | AgcAutoOn | rw | 0x01 | 0 → LNA gain forced by the LnaGain Setting<br>1 → LNA gain is controlled by the AGC |
| | 2-0 | RxTrigger | rw | 0x06 * | Selects the event triggering AGC and/or AFC at receiver startup. See Table 24 for a description. |
| RegRssiConfig (0x0e) | 7-3 | RssiOffset | rw | 0x00 | Signed RSSI offset, to compensate for the possible losses/gains in the front-end (LNA, SAW filter...)<br>1dB / LSB, 2's complement format |
| | 2-0 | RssiSmoothing | rw | 0x02 | Defines the number of samples taken to average the RSSI result:<br>000 → 2 samples used<br>001 → 4 samples used<br>10 → 8 samples used<br>11 → 16 samples used<br>100 → 32 samples used<br>101 → 64 samples used<br>110 → 128 samples used<br>111 → 256 samples used |
| RegRssiCollision (0x0f) | 7-0 | RssiCollisionThreshold | rw | 0x0a | Sets the threshold used to consider that an interferer is detected, witnessing a packet collision. 1dB/LSB (only RSSI increase)<br>Default: 10dB |
| RegRssiThresh (0x10) | 7-0 | RssiThreshold | rw | 0xff | RSSI trigger level for the Rssi interrupt:<br>- RssiThreshold / 2 [dBm] |
| RegRssiValue (0x11) | 7-0 | RssiValue | r | - | Absolute value of the RSSI in dBm, 0.5dB steps.<br>RSSI = - RssiValue/2 [dBm] |
| RegRxBw (0x12) | 7 | unused | r | - | unused |
| | 6-5 | reserved | rw | 0x00 | reserved |
| | 4-3 | RxBwMant | rw | 0x02 | Channel filter bandwidth control:<br>1 → RxBwMant = 16 10 → RxBwMant = 24<br>2 → RxBwMant = 20 11 → reserved |
| | 2-0 | RxBwExp | rw | 0x05 | Channel filter bandwidth control |
| RegAfcBw (0x13) | 7-5 | reserved | rw | 0x00 | reserved |
| | 4-3 | RxBwMantAfc | rw | 0x01 | RxBwMant parameter used during the AFC |
| | 2-0 | RxBwExpAfc | rw | 0x03 | RxBwExp parameter used during the AFC |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegOokPeak (0x14) | 7-6 | reserved | rw | 0x00 | reserved |
| | 5 | BitSyncOn | rw | 0x01 | Enables the Bit Synchronizer.<br>0 → Bit Sync disabled (not possible in Packet mode)<br>1 → Bit Sync enabled |
| | 4-3 | OokThreshType | rw | 0x01 | Selects the type of threshold in the OOK data slicer:<br>00 → fixed threshold 10 → average mode<br>01 → peak mode (default) 11 → reserved |
| | 2-0 | OokPeakTheshStep | rw | 0x00 | Size of each decrement of the RSSI threshold in the OOK demodulator:<br>000 → 0.5 dB 001 → 1.0 dB<br>010 → 1.5 dB 011 → 2.0 dB<br>100 → 3.0 dB 101 → 4.0 dB<br>110 → 5.0 dB 111 → 6.0 dB |
| RegOokFix (0x15) | 7-0 | OokFixedThreshold | rw | 0x0C | Fixed threshold for the Data Slicer in OOK mode<br>Floor threshold for the Data Slicer in OOK when Peak mode is used |
| RegOokAvg (0x16) | 7-5 | OokPeakThreshDec | rw | 0x00 | Period of decrement of the RSSI threshold in the OOK demodulator:<br>000 → once per chip 001 → once every 2 chips<br>010 → once every 4 chips 011 → once every 8 chips<br>100 → twice in each chip 101 → 4 times in each chip<br>110 → 8 times in each chip 111 → 16 times in each chip |
| | 4 | reserved | rw | 0x01 | reserved |
| | 3-2 | OokAverageOffset | rw | 0x00 | Static offset added to the threshold in average mode in order to reduce glitching activity (OOK only):<br>00 → 0.0 dB 10 → 4.0 dB<br>01 → 2.0 dB 11 → 6.0 dB |
| | 1-0 | OokAverageThreshFilt | rw | 0x02 | Filter coefficients in average mode of the OOK demodulator:<br>00 → $f_C$ ≈ chip rate / 32.π 01 → $f_C$ ≈ chip rate / 8.π<br>10 → $f_C$ ≈ chip rate / 4.π 11 → $f_C$ ≈ chip rate / 2.π |
| RegRes17 to RegRes19 | 7-0 | reserved | rw | 0x47<br>0x32<br>0x3E | reserved. Keep the Reset values. |
| RegAfcFei (0x1a) | 7-5 | unused | r | - | unused |
| | 4 | AgcStart | wt | 0x00 | Triggers an AGC sequence when set to 1. |
| | 3 | reserved | rw | 0x00 | reserved |
| | 2 | unused | - | - | unused |
| | 1 | AfcClear | wc | 0x00 | Clear AFC register set in Rx mode. Always reads 0. |
| | 0 | AfcAutoClearOn | rw | 0x00 | Only valid if AfcAutoOn is set<br>0 → AFC register is not cleared at the beginning of the automatic AFC phase<br>1 → AFC register is cleared at the beginning of the automatic AFC phase |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegAfcMsb (0x1b) | 7-0 | AfcValue(15:8) | rw | 0x00 | MSB of the AfcValue, 2's complement format. Can be used to overwrite the current AFC value |
| RegAfcLsb (0x1c) | 7-0 | AfcValue(7:0) | rw | 0x00 | LSB of the AfcValue, 2's complement format. Can be used to overwrite the current AFC value |
| RegFeiMsb (0x1d) | 7-0 | FeiValue(15:8) | rw | - | MSB of the measured frequency offset, 2's complement. Must be read before RegFeiLsb. |
| RegFeiLsb (0x1e) | 7-0 | FeiValue(7:0) | rw | - | LSB of the measured frequency offset, 2's complement<br>*Frequency error* = FeiValue x Fstep |
| RegPreambleDetect (0x1f) | 7 | PreambleDetectorOn | rw | 0x01 * | Enables Preamble detector when set to 1. The AGC settings supersede this bit during the startup / AGC phase.<br>0 → Turned off<br>1 → Turned on |
| | 6-5 | PreambleDetectorSize | rw | 0x01 * | Number of Preamble bytes to detect to trigger an interrupt<br>00 → 1 byte 10 → 3 bytes<br>01 → 2 bytes 11 → Reserved |
| | 4-0 | PreambleDetectorTol | rw | 0x0A * | Number or chip errors tolerated over PreambleDetectorSize.<br>4 chips per bit. |
| RegRxTimeout1 (0x20) | 7-0 | TimeoutRxRssi | rw | 0x00 | *Timeout* interrupt is generated *TimeoutRxRssi**16*$T_{bit}$ after switching to Rx mode if *Rssi* interrupt doesn't occur (i.e. *RssiValue* > *RssiThreshold)*<br>0x00: *TimeoutRxRssi* is disabled |
| RegRxTimeout2 (0x21) | 7-0 | TimeoutRxPreamble | rw | 0x00 | *Timeout* interrupt is generated *TimeoutRxPreamble**16*$T_{bit}$ after switching to Rx mode if *Preamble* interrupt doesn't occur<br>0x00: *TimeoutRxPreamble* is disabled |
| RegRxTimeout3 (0x22) | 7-0 | TimeoutSignalSync | rw | 0x00 | *Timeout* interrupt is generated *TimeoutSignalSync**16*$T_{bit}$ after the Rx mode is programmed, if *SyncAddress* doesn't occur<br>0x00: *TimeoutSignalSync* is disabled |
| RegRxDelay (0x23) | 7-0 | InterPacketRxDelay | rw | 0x00 | Additional delay before an automatic receiver restart is launched:<br>Delay = InterPacketRxDelay*4*Tbit |
| RC Oscillator registers | | | | | |
| RegOsc (0x24) | 7-4 | unused | r | - | unused |
| | 3 | RcCalStart | wt | 0x00 | Triggers the calibration of the RC oscillator when set. Always reads 0. RC calibration must be triggered in Standby mode. |
| | 2-0 | ClkOut | rw | 0x07 * | Selects CLKOUT frequency:<br>000 → FXOSC<br>001 → FXOSC / 2<br>10 → FXOSC / 4<br>11 → FXOSC / 8<br>100 → FXOSC / 16<br>101 → FXOSC / 32<br>110 → RC (automatically enabled)<br>111 → OFF |
| Packet Handling registers | | | | | |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegPreambleMsb (0x25) | 7-0 | PreambleSize(15:8) | rw | 0x00 | Size of the preamble to be sent (from *TxStartCondition* fulfilled). (MSB byte) |
| RegPreambleLsb (0x26) | 7-0 | PreambleSize(7:0) | rw | 0x03 | Size of the preamble to be sent (from *TxStartCondition* fulfilled). (LSB byte) |
| RegSyncConfig (0x27) | 7-6 | AutoRestartRxMode | rw | 0x02 | Controls the automatic restart of the receiver after the reception of a valid packet (PayloadReady or CrcOk):<br>1 → Off<br>2 → On, without waiting for the PLL to re-lock<br>10 → On, wait for the PLL to lock (frequency changed)<br>11 → reserved |
| | 5 | PreamblePolarity | rw | 0x00 | Sets the polarity of the Preamble<br>0 → 0xAA (default)<br>1 → 0x55 |
| | 4 | SyncOn | rw | 0x01 | Enables the Sync word generation and detection:<br>0 → Off<br>1 → On |
| | 3 | reserved | rw | 0x00 | reserved |
| | 2-0 | SyncSize | rw | 0x03 | Size of the Sync word:<br>(*SyncSize* + 1) bytes, (*SyncSize*) bytes if *ioHomeOn*=1 |
| RegSyncValue1 (0x28) | 7-0 | SyncValue(63:56) | rw | 0x01 * | 1st byte of Sync word. (MSB byte)<br>Used if *SyncOn* is set. |
| RegSyncValue2 (0x29) | 7-0 | SyncValue(55:48) | rw | 0x01 * | 2nd byte of Sync word<br>Used if *SyncOn* is set and *(SyncSize +1)* >=2. |
| RegSyncValue3 (0x2a) | 7-0 | SyncValue(47:40) | rw | 0x01 * | 3rd byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* >=3. |
| RegSyncValue4 (0x2b) | 7-0 | SyncValue(39:32) | rw | 0x01 * | 4th byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* >=4. |
| RegSyncValue5 (0x2c) | 7-0 | SyncValue(31:24) | rw | 0x01 * | 5th byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* >=5. |
| RegSyncValue6 (0x2d) | 7-0 | SyncValue(23:16) | rw | 0x01 * | 6th byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* >=6. |
| RegSyncValue7 (0x2e) | 7-0 | SyncValue(15:8) | rw | 0x01 * | 7th byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* >=7. |
| RegSyncValue8 (0x2f) | 7-0 | SyncValue(7:0) | rw | 0x01 * | 8th byte of Sync word.<br>Used if *SyncOn* is set and *(SyncSize +1)* =8. |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegPacketConfig1 (0x30) | 7 | PacketFormat | rw | 0x01 | Defines the packet format used:<br>0 → Fixed length<br>1 → Variable length |
| | 6-5 | DcFree | rw | 0x00 | Defines DC-free encoding/decoding performed:<br>00 → None (Off)<br>01 → Manchester<br>10 → Whitening<br>11 → reserved |
| | 4 | CrcOn | rw | 0x01 | Enables CRC calculation/check (Tx/Rx):<br>0 → Off<br>1 → On |
| | 3 | CrcAutoClearOff | rw | 0x00 | Defines the behavior of the packet handler when CRC check fails:<br>0 → Clear FIFO and restart new packet reception. No *PayloadReady* interrupt issued.<br>1 → Do not clear FIFO. *PayloadReady* interrupt issued. |
| | 2-1 | AddressFiltering | rw | 0x00 | Defines address based filtering in Rx:<br>00 → None (Off)<br>01 → Address field must match *NodeAddress*<br>10 → Address field must match *NodeAddress* or *BroadcastAddress*<br>11 → reserved |
| | 0 | CrcWhiteningType | rw | 0x00 | Selects the CRC and whitening algorithms:<br>0 → CCITT CRC implementation with standard whitening<br>1 → IBM CRC implementation with alternate whitening |
| RegPacketConfig2 (0x31) | 7 | unused | r | - | unused |
| | 6 | DataMode | rw | 0x01 | Data processing mode:<br>0 → Continuous mode<br>1 → Packet mode |
| | 5 | IoHomeOn | rw | 0x00 | Enables the io-homecontrol® compatibility mode<br>0 → Disabled<br>1 → Enabled |
| | 4 | IoHomePowerFrame | rw | 0x00 | reserved - Linked to io-homecontrol® compatibility mode |
| | 3 | BeaconOn | rw | 0x00 | Enables the Beacon mode in Fixed packet format |
| | 2-0 | PayloadLength(10:8) | rw | 0x00 | Packet Length Most significant bits |
| RegPayloadLength (0x32) | 7-0 | PayloadLength(7:0) | rw | 0x40 | If PacketFormat = 0 (fixed), payload length.<br>If PacketFormat = 1 (variable), max length in Rx, not used in Tx. |
| RegNodeAdrs (0x33) | 7-0 | NodeAddress | rw | 0x00 | Node address used in address filtering. |
| RegBroadcastAdrs (0x34) | 7-0 | BroadcastAddress | rw | 0x00 | Broadcast address used in address filtering. |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegFifoThresh (0x35) | 7 | TxStartCondition | rw | 0x01 * | Defines the condition to start packet transmission:<br>0 → *FifoLevel* (i.e. the number of bytes in the FIFO exceeds *FifoThreshold)*<br>1 → *FifoEmpty goes low*(i.e. at least one byte in the FIFO) |
| | 6 | unused | r | - | unused |
| | 5-0 | FifoThreshold | rw | 0x0f | Used to trigger *FifoLevel* interrupt, when:<br>number of bytes in FIFO >= FifoThreshold + 1 |
| Sequencer registers | | | | | |
| RegSeqConfig1 (0x36) | 7 | SequencerStart | wt | 0x00 | Controls the top level Sequencer<br>When set to '1', executes the "Start" transition.<br>The sequencer can only be enabled when the chip is in Sleep or Standby mode. |
| | 6 | SequencerStop | wt | 0x00 | Forces the Sequencer Off.<br>Always reads '0' |
| | 5 | IdleMode | rw | 0x00 | Selects chip mode during the state:<br>0: Standby mode<br>1: Sleep mode |
| | 4-3 | FromStart | rw | 0x00 | Controls the Sequencer transition when *SequencerStart* is set to 1 in Sleep or Standby mode:<br>00: to LowPowerSelection<br>01: to Receive state<br>10: to Transmit state<br>11: to Transmit state on a *FifoLevel* interrupt |
| | 2 | LowPowerSelection | rw | 0x00 | Selects the Sequencer LowPower state after a *to LowPowerSelection* transition:<br>0: SequencerOff state with chip on Initial mode<br>1: Idle state with chip on *Standby* or *Sleep* mode depending on *IdleMode*<br>*Note: Initial mode is the chip LowPower mode at Sequencer Start.* |
| | 1 | FromIdle | rw | 0x00 | Controls the Sequencer transition from the Idle state on a T1 interrupt:<br>0: to Transmit state<br>1: to Receive state |
| | 0 | FromTransmit | rw | 0x00 | Controls the Sequencer transition from the Transmit state:<br>0: to LowPowerSelection on a *PacketSent* interrupt<br>1: to Receive state on a *PacketSent* interrupt |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegSeqConfig2 (0x37) | 7-5 | FromReceive | rw | 0x00 | Controls the Sequencer transition from the Receive state<br>000 and 111: unused<br>001: to PacketReceived state on a *PayloadReady* interrupt<br>010: to LowPowerSelection on a *PayloadReady* interrupt<br>011: to PacketReceived state on a *CrcOk* interrupt (1)<br>100: to SequencerOff state on a *Rssi* interrupt<br>101: to SequencerOff state on a *SyncAddress* interrupt<br>110: to SequencerOff state on a *PreambleDetect* interrupt<br><br>Irrespective of this setting, transition to LowPowerSelection on a T2 interrupt<br><br>(1) If the CRC is wrong (corrupted packet, with CRC on but *CrcAutoClearOn*=0), the *PayloadReady* interrupt will drive the sequencer to RxTimeout state. |
| | 4-3 | FromRxTimeout | rw | 0x00 | Controls the state-machine transition from the Receive state on a *RxTimeout* interrupt (and on *PayloadReady* if FromReceive = 011):<br>00: to Receive State, via ReceiveRestart<br>01: to Transmit state<br>10: to LowPowerSelection<br>11: to SequencerOff state<br><br>*Note: RxTimeout interrupt is a TimeoutRxRssi, TimeoutRxPreamble or TimeoutSignalSync interrupt* |
| | 2-0 | FromPacketReceived | rw | 0x00 | Controls the state-machine transition from the PacketReceived state:<br>000: to SequencerOff state<br>001: to Transmit state on a *FifoEmpty* interrupt<br>010: to LowPowerSelection<br>011: to Receive via FS mode, if frequency was changed<br>100: to Receive state (no frequency change) |
| RegTimerResol (0x38) | 7-4 | unused | r | - | unused |
| | 3-2 | Timer1Resolution | rw | 0x00 | Resolution of Timer 1<br>00: Timer1 disabled<br>01: 64 us<br>10: 4.1 ms<br>11: 262 ms |
| | 1-0 | Timer2Resolution | rw | 0x00 | Resolution of Timer 2<br>00: Timer2 disabled<br>01: 64 us<br>10: 4.1 ms<br>11: 262 ms |
| RegTimer1Coef (0x39) | 7-0 | Timer1Coefficient | rw | 0xf5 | Multiplying coefficient for Timer 1 |
| RegTimer2Coef (0x3a) | 7-0 | Timer2Coefficient | rw | 0x20 | Multiplying coefficient for Timer 2 |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| Service registers | | | | | |
| RegImageCal (0x3b) | 7 | AutoImageCalOn | rw | 0x00 * | Controls the Image calibration mechanism<br>0 → Calibration of the receiver depending on the temperature is disabled<br>1 → Calibration of the receiver depending on the temperature enabled. |
| | 6 | ImageCalStart | wt | - | Triggers the IQ and RSSI calibration when set in Standby mode. |
| | 5 | ImageCalRunning | r | 0x00 | Set to 1 while the Image and RSSI calibration are running.<br>Toggles back to 0 when the process is completed |
| | 4 | unused | r | - | unused |
| | 3 | TempChange | r | 0x00 | IRQ flag witnessing a temperature change exceeding TempThreshold since the last Image and RSSI calibration:<br>0 → Temperature change lower than TempThreshold<br>1 → Temperature change greater than TempThreshold |
| | 2-1 | TempThreshold | rw | 0x01 | Temperature change threshold to trigger a new I/Q calibration<br>00 → 5 °C<br>01 → 10 °C<br>10 → 15 °C<br>11 → 20 °C |
| | 0 | TempMonitorOff | rw | 0x00 | Controls the temperature monitor operation:<br>0 → Temperature monitoring done in all modes except Sleep and Standby<br>1 → Temperature monitoring stopped. |
| RegTemp (0x3c) | 7-0 | TempValue | r | - | Measured temperature<br>-1°C per Lsb<br>Needs calibration for absolute accuracy |
| RegLowBat (0x3d) | 7-4 | unused | r | - | unused |
| | 3 | LowBatOn | rw | 0x00 | Low Battery detector enable signal<br>0 → LowBat detector disabled<br>1 → LowBat detector enabled |
| | 2-0 | LowBatTrim | rw | 0x02 | Trimming of the LowBat threshold:<br>000 → 1.695 V<br>001 → 1.764 V<br>010 → 1.835 V (d)<br>011 → 1.905 V<br>100 → 1.976 V<br>101 → 2.045 V<br>110 → 2.116 V<br>111 → 2.185 V |
| Status registers | | | | | |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegIrqFlags1 (0x3e) | 7 | ModeReady | r | - | Set when the operation mode requested in *Mode*, is ready<br>- Sleep: Entering Sleep mode<br>- Standby: XO is running<br>- FS: PLL is locked<br>- Rx: RSSI sampling starts<br>- Tx: PA ramp-up completed<br>Cleared when changing the operating mode. |
| | 6 | RxReady | r | - | Set in Rx mode, after RSSI, AGC and AFC.<br>Cleared when leaving Rx. |
| | 5 | TxReady | r | - | Set in Tx mode, after PA ramp-up.<br>Cleared when leaving Tx. |
| | 4 | PllLock | r | - | Set (in FS, Rx or Tx) when the PLL is locked.<br>Cleared when it is not. |
| | 3 | Rssi | rwc | - | Set in Rx when the *RssiValue* exceeds *RssiThreshold.*<br>Cleared when leaving Rx or setting this bit to 1. |
| | **2** | Timeout | r | - | Set when a timeout occurs<br>Cleared when leaving Rx or FIFO is emptied. |
| | 1 | PreambleDetect | rwc | - | Set when the Preamble Detector has found valid Preamble.<br>bit clear when set to 1 |
| | 0 | SyncAddressMatch | rwc | - | Set when Sync and Address (if enabled) are detected.<br>Cleared when leaving Rx or FIFO is emptied.<br>This bit is read only in Packet mode, rwc in Continuous mode |
| RegIrqFlags2 (0x3f) | 7 | FifoFull | r | - | Set when FIFO is full (i.e. contains 66 bytes), else cleared. |
| | 6 | FifoEmpty | r | - | Set when FIFO is empty, and cleared when there is at least 1 byte in the FIFO. |
| | 5 | FifoLevel | r | - | Set when the number of bytes in the FIFO strictly exceeds *FifoThreshold*, else cleared. |
| | 4 | FifoOverrun | rwc | - | Set when FIFO overrun occurs. (except in Sleep mode)<br>Flag(s) and FIFO are cleared when this bit is set. The FIFO then becomes immediately available for the next transmission / reception. |
| | 3 | PacketSent | r | - | Set in Tx when the complete packet has been sent.<br>Cleared when exiting Tx |
| | 2 | PayloadReady | r | - | Set in Rx when the payload is ready (i.e. last byte received and CRC, if enabled and *CrcAutoClearOff* is cleared, is Ok). Cleared when FIFO is empty. |
| | 1 | CrcOk | r | - | Set in Rx when the CRC of the payload is Ok. Cleared when FIFO is empty. |
| | 0 | LowBat | rwc | - | Set when the battery voltage drops below the Low Battery threshold. Cleared only when set to 1 by the user. |
| IO control registers | | | | | |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegDioMapping1 (0x40) | 7-6 | Dio0Mapping | rw | 0x00 | Mapping of pins DIO0 to DIO5<br><br>See Table 18 for mapping in LoRa mode<br><br>See Table 29 for mapping in Continuous mode<br>SeeTable 30 for mapping in Packet mode |
| | 5-4 | Dio1Mapping | rw | 0x00 | |
| | 3-2 | Dio2Mapping | rw | 0x00 | |
| | 1-0 | Dio3Mapping | rw | 0x00 | |
| RegDioMapping2 (0x41) | 7-6 | Dio4Mapping | rw | 0x00 | |
| | 5-4 | Dio5Mapping | rw | 0x00 | |
| | 3-1 | reserved | rw | 0x00 | reserved. Retain default value |
| | 0 | MapPreambleDetect | rw | 0x00 | Allows the mapping of either *Rssi* Or *PreambleDetect* to the DIO pins, as summarized on Table 29 and Table 30<br>0 → *Rssi* interrupt<br>1 → *PreambleDetect* interrupt |
| Version register | | | | | |
| RegVersion (0x42) | 7-0 | Version | r | 0x12 | Version code of the chip. Bits 7-4 give the full revision number; bits 3-0 give the metal mask revision number. |
| Additional registers | | | | | |
| RegPllHop (0x44) | 7 | FastHopOn | rw | 0x00 | Bypasses the main state machine for a quick frequency hop.<br>Writing RegFrfLsb will trigger the frequency change.<br>0 → Frf is validated when FSTx or FSRx is requested<br>1 → Frf is validated triggered when RegFrfLsb is written |
| | 6-0 | reserved | rw | 0x2d | reserved |
| RegTcxo (0x4b) | 7-5 | reserved | rw | 0x00 | reserved. Retain default value |
| | 4 | TcxoInputOn | rw | 0x00 | Controls the crystal oscillator<br>0 → Crystal Oscillator with external Crystal<br>1 → External clipped sine TCXO AC-connected to XTA pin |
| | 3-0 | reserved | rw | 0x09 | Reserved. Retain default value. |
| RegPaDac (0x4d) | 7-3 | reserved | rw | 0x10 | reserved. Retain default value |
| | 2-0 | PaDac | rw | 0x04 | Enables the +20dBm option on PA_BOOST pin<br>0x04 → Default value<br>0x07 → +20dBm on PA_BOOST when OutputPower=1111 |
| RegFormerTemp (0x5b) | 7-0 | FormerTemp | rw | - | Temperature saved during the latest IQ (RSSI and Image) calibration. Same format as *TempValue* in *RegTemp*. |
| RegBitrateFrac (0x5d) | 7-4 | unused | r | 0x00 | unused |
| | 3-0 | BitRateFrac | rw | 0x00 | Fractional part of the bit rate divider (Only valid for FSK)<br>If *BitRateFrac*> 0 then:<br>$BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$ |

| Name (Address) | Bits | Variable Name | Mode | Default value | FSK/OOK Description |
|---|---|---|---|---|---|
| RegAgcRef (0x61) | 7-6 | unused | r | - | unused |
| | 5-0 | AgcReferenceLevel | rw | 0x19 | Sets the floor reference for all AGC thresholds:<br>AGC Reference[dBm]=<br>-174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel*<br>SNR = 8dB, fixed value |
| RegAgcThresh1 (0x62) | 7-5 | unused | r | - | unused |
| | 4-0 | AgcStep1 | rw | 0x0c | Defines the 1st AGC Threshold |
| RegAgcThresh2 (0x63) | 7-4 | AgcStep2 | rw | 0x04 | Defines the 2nd AGC Threshold: |
| | 3-0 | AgcStep3 | rw | 0x0b | Defines the 3rd AGC Threshold: |
| RegAgcThresh3 (0x64) | 7-4 | AgcStep4 | rw | 0x0c | Defines the 4th AGC Threshold: |
| | 3-0 | AgcStep5 | rw | 0x0c | Defines the 5th AGC Threshold: |

## 6.3. Band Specific Additional Registers

The registers in the address space from 0x61 to 0x73 are specific for operation in the lower frequency bands (below 525 MHz), or in the upper frequency bands (above 860 MHz). Their programmed value may differ, and are retained when switching from lower to high frequency and vice-versa. The access to the band specific registers is granted by enabling or disabling the bit 3 LowFrequencyModeOn of the RegOpMode register. By default, the bit LowFrequencyModeOn is at '1' indicating that the registers are configured for the low frequency band.

*Table 41 Low Frequency Additional Registers*

| Name (Address) | Bits | Variable Name | Mode | Default value | Low Frequency Additional Registers |
|---|---|---|---|---|---|
| RegAgcRefLf (0x61) | 7-6 | unused | r | - | unused |
| | 5-0 | AgcReferenceLevel | rw | 0x19 | Sets the floor reference for all AGC thresholds:<br>AGC Reference[dBm]=<br>-174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel*<br>SNR = 8dB, fixed value |
| RegAgcThresh1Lf (0x62) | 7-5 | unused | r | - | unused |
| | 4-0 | AgcStep1 | rw | 0x0c | Defines the 1st AGC Threshold |
| RegAgcThresh2Lf (0x63) | 7-4 | AgcStep2 | rw | 0x04 | Defines the 2nd AGC Threshold: |
| | 3-0 | AgcStep3 | rw | 0x0b | Defines the 3rd AGC Threshold: |
| RegAgcThresh3Lf (0x64) | 7-4 | AgcStep4 | rw | 0x0c | Defines the 4th AGC Threshold: |
| | 3-0 | AgcStep5 | rw | 0x0c | Defines the 5th AGC Threshold: |
| RegPllLf (0x70) | 7-6 | PllBandwidth | rw | 0x03 | Controls the PLL bandwidth:<br>00 → 75 kHz 10 → 225 kHz<br>01 → 150 kHz 11 → 300 kHz |
| | 5-0 | reserved | rw | 0x10 | reserved. Retain default value |

*Table 42 High Frequency Additional Registers*

| Name (Address) | Bits | Variable Name | Mode | Default value | Low Frequency Additional Registers |
|---|---|---|---|---|---|
| RegAgcRefHf (0x61) | 7-6 | unused | r | - | unused |
| | 5-0 | AgcReferenceLevel | rw | 0x1c | Sets the floor reference for all AGC thresholds:<br>AGC Reference[dBm]=<br>-174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel*<br>SNR = 8dB, fixed value |
| RegAgcThresh1Hf (0x62) | 7-5 | unused | r | - | unused |
| | 4-0 | AgcStep1 | rw | 0x0e | Defines the 1st AGC Threshold |
| RegAgcThresh2Hf (0x63) | 7-4 | AgcStep2 | rw | 0x05 | Defines the 2nd AGC Threshold: |
| | 3-0 | AgcStep3 | rw | 0x0b | Defines the 3rd AGC Threshold: |
| RegAgcThresh3Hf (0x64) | 7-4 | AgcStep4 | rw | 0x0c | Defines the 4th AGC Threshold: |
| | 3-0 | AgcStep5 | rw | 0x0c | Defines the 5th AGC Threshold: |
| RegPllHf (0x70) | 7-6 | PllBandwidth | rw | 0x03 | Controls the PLL bandwidth:<br>00 → 75 kHz 10 → 225 kHz<br>01 → 150 kHz 11 → 300 kHz |
| | 5-0 | reserved | rw | 0x10 | reserved. Retain default value |

## 6.4. LoRa™ Mode Register Map

This section details the RFM95W/96W/98W register mapping and the precise contents of each register in LoRa™ mode.

It is essential to understand that the LoRa modem is controlled independently of the FSK modem. Therefore, care should be taken when accessing the registers, especially as some register may have the same name in LoRa or FSK mode.

The LoRa registers are only accessible when the device is set in Lora mode (and, in the same way, the FSK register are only accessible in FSK mode). However, in some cases, it may be necessary to access some of the FSK register while in LoRa mode. To this aim, the *AccesSharedReg* bit was created in the *RegOpMode* register. This bit, when set to '1', will grant access to the FSK register 0x0D up to the register 0x3F. Once the setup has been done, it is strongly recommended to clear this bit so that LoRa register can be accessed normally.

Convention: r: read, w: write, c : set to clear and t: trigger.

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegFifo (0x00) | 7-0 | Fifo | rw | 0x00 | LoRa™ base-band FIFO data input/output. FIFO is cleared an not accessible when device is in SLEEP mode |
| Common Register Settings | | | | | |
| RegOpMode (0x01) | 7 | LongRangeMode | rw | 0x0 | 0 → FSK/OOK Mode<br>1 → LoRa™ Mode<br>This bit can be modified only in Sleep mode. A write operation on other device modes is ignored. |
| | 6 | AccessSharedReg | rw | 0x0 | This bit operates when device is in Lora mode; if set it allows access to FSK registers page located in address space (0x0D:0x3F) while in LoRa mode<br>0 → Access LoRa registers page 0x0D: 0x3F<br>1 → Access FSK registers page (in mode LoRa) 0x0D: 0x3F |
| | 5-4 | reserved | r | 0x00 | reserved |
| | 3 | LowFrequencyModeOn | rw | 0x01 | Access Low Frequency Mode registers<br>0 → High Frequency Mode (access to HF test registers)<br>1 → Low Frequency Mode (access to LF test registers) |
| | 2-0 | Mode | rwt | 0x01 | Device modes<br>000 → SLEEP<br>001 → STDBY<br>010 → Frequency synthesis TX (FSTX)<br>011 → Transmit (TX)<br>100 → Frequency synthesis RX (FSRX)<br>101 → Receive continuous (RXCONTINUOUS)<br>110 → receive single (RXSINGLE)<br>111 → Channel activity detection (CAD) |
| (0x02) | 7-0 | reserved | r | 0x00 | - |
| (0x03) | 7-0 | reserved | r | 0x00 | - |
| (0x04) | 7-0 | reserved | rw | 0x00 | - |
| (0x05) | 7-0 | reserved | r | 0x00 | - |
| RegFrMsb (0x06) | 7-0 | Frf(23:16) | rw | 0x6c | MSB of RF carrier frequency |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegFrMid (0x07) | 7-0 | Frf(15:8) | rw | 0x80 | MSB of RF carrier frequency |
| RegFrLsb (0x08) | 7-0 | Frf(7:0) | rwt | 0x00 | LSB of RF carrier frequency<br>$f_{RF} = \frac{F(XOSC) \cdot Frf}{2^{19}}$<br>Resolution is 61.035 Hz if F(XOSC) = 32 MHz. Default value is 0x6c8000 = 434 MHz. Register values must be modified only when device is in SLEEP or STAND-BY mode. |
| Registers for RF blocks | | | | | |
| RegPaConfig (0x09) | 7 | PaSelect | rw | 0x00 | Selects PA output pin<br>0 → RFO pin. Output power is limited to +14 dBm.<br>1 → PA_BOOST pin. Output power is limited to +20 dBm |
| | 6-4 | MaxPower | rw | 0x04 | Select max output power: Pmax=10.8+0.6*MaxPower[dBm] |
| | 3-0 | OutputPower | rw | 0x0f | Pout=Pmax-(15-OutputPower) if PaSelect = 0 (RFO pin)<br>Pout=17-(15-OutputPower) if PaSelect = 1 (PA_BOOST pin) |
| RegPaRamp (0x0A) | 7-5 | unused | r | - | unused |
| | 4 | reserved | rw | 0x00 | reserved |
| | 3-0 | PaRamp(3:0) | rw | 0x09 | Rise/Fall time of ramp up/down in FSK<br>0000 → 3.4 ms<br>0001 → 2 ms<br>0010 → 1 ms<br>0011 → 500 us<br>0100 → 250 us<br>0101 → 125 us<br>0110 → 100 us<br>0111 → 62 us<br>1000 → 50 us<br>1001 → 40 us<br>1010 → 31 us<br>1011 → 25 us<br>1100 → 20 us<br>1101 → 15 us<br>1110 → 12 us<br>1111 → 10 us |
| RegOcp (0x0B | 7-6 | unused | r | 0x00 | unused |
| | 5 | OcpOn | rw | 0x01 | Enables overload current protection (OCP) for PA:<br>0 → OCP disabled<br>1 → OCP enabled |
| | 4-0 | OcpTrim | rw | 0x0b | Trimming of OCP current:<br>Imax = 45+5*OcpTrim [mA] if OcpTrim <= 15 (120 mA) /<br>Imax = -30+10*OcpTrim [mA] if 15 < OcpTrim <= 27 (130 to 240 mA)<br>Imax = 240mA for higher settings<br>Default Imax = 100mA |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegLna (0x0C) | 7-5 | LnaGain | rwx | 0x01 | LNA gain setting:<br>000 → not used<br>001 → G1 = maximum gain<br>010 → G2<br>011 → G3<br>100 → G4<br>101 → G5<br>110 → G6 = minimum gain<br>111 → not used |
| | 4-3 | LnaBoostLf | rw | 0x00 | Low Frequency (RFI_LF) LNA current adjustment<br>00 → Default LNA current<br>Other → Reserved |
| | 2 | reserved | rw | 0x00 | reserved |
| | 1-0 | LnaBoostHf | rw | 0x00 | High Frequency (RFI_HF) LNA current adjustment<br>00 → Default LNA current<br>11 → Boost on, 150% LNA current |
| Lora page registers | | | | | |
| RegFifoAddrPtr (0x0D) | 7-0 | FifoAddrPtr | rw | 0x00 | SPI interface address pointer in FIFO data buffer. |
| RegFifoTxBa seAddr (0x0E) | 7-0 | FifoTxBaseAddr | rw | 0x80 | write base address in FIFO data buffer for TX modulator |
| RegFifoRxBa seAddr (0x0F) | 7-0 | FifoRxBaseAddr | rw | 0x00 | read base address in FIFO data buffer for RX demodulator |
| RegFifoRxCu rrentAddr (0x10) | 7-0 | FifoRxCurrentAddr | r | n/a | Start address (in data buffer) of last packet received |
| RegIrqFlags Mask (0x11) | 7 | RxTimeoutMask | rw | 0x00 | Timeout interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 6 | RxDoneMask | rw | 0x00 | Packet reception complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 5 | PayloadCrcErrorMask | rw | 0x00 | Payload CRC error interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 4 | ValidHeaderMask | rw | 0x00 | Valid header received in Rx mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 3 | TxDoneMask | rw | 0x00 | FIFO Payload transmission complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 2 | CadDoneMask | rw | 0x00 | CAD complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 1 | FhssChangeChannelMask | rw | 0x00 | FHSS change channel interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags |
| | 0 | CadDetectedMask | rw | 0x00 | Cad Detected Interrupt Mask: setting this bit masks the corresponding IRQ in RegIrqFlags |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegIrqFlags (0x12) | 7 | RxTimeout | rc | 0x00 | Timeout interrupt: writing a 1 clears the IRQ |
| | 6 | RxDone | rc | 0x00 | Packet reception complete interrupt: writing a 1 clears the IRQ |
| | 5 | PayloadCrcError | rc | 0x00 | Payload CRC error interrupt: writing a 1 clears the IRQ |
| | 4 | ValidHeader | rc | 0x00 | Valid header received in Rx: writing a 1 clears the IRQ |
| | 3 | TxDone | rc | 0x00 | FIFO Payload transmission complete interrupt: writing a 1 clears the IRQ |
| | 2 | CadDone | rc | 0x00 | CAD complete: write to clear: writing a 1 clears the IRQ |
| | 1 | FhssChangeChannel | rc | 0x00 | FHSS change channel interrupt: writing a 1 clears the IRQ |
| | 0 | CadDetected | rc | 0x00 | Valid Lora signal detected during CAD operation: writing a 1 clears the IRQ |
| RegRxNbBytes (0x13) | 7-0 | FifoRxBytesNb | r | n/a | Number of payload bytes of latest packet received |
| RegRxHeaderCntValueMsb (0x14) | 7-0 | ValidHeaderCntMsb(15:8) | r | n/a | Number of valid headers received since last transition into Rx mode, MSB(15:8). Header and packet counters are reseted in Sleep mode. |
| RegRxHeaderCntValueLsb (0x15) | 7-0 | ValidHeaderCntLsb(7:0) | r | n/a | Number of valid headers received since last transition into Rx mode, LSB(7:0). Header and packet counters are reseted in Sleep mode. |
| RegRxPacketCntValueMsb (0x16) | 7-0 | ValidPacketCntMsb(15:8) | rc | n/a | Number of valid packets received since last transition into Rx mode, MSB(15:8). Header and packet counters are reseted in Sleep mode. |
| RegRxPacketCntValueLsb (0x17) | 7-0 | ValidPacketCntLsb(7:0) | r | n/a | Number of valid packets received since last transition into Rx mode, LSB(7:0). Header and packet counters are reseted in Sleep mode. |
| RegModemStat (0x18) | 7-5 | RxCodingRate | r | n/a | Coding rate of last header received |
| | 4 | ModemStatus | r | '1' | Modem clear |
| | 3 | | r | '0' | Header info valid |
| | 2 | | r | '0' | RX on-going |
| | 1 | | r | '0' | Signal synchronized |
| | 0 | | r | '0' | Signal detected |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegPktSnrValue (0x19) | 7-0 | PacketSnr | r | n/a | Estimation of SNR on last packet received. In two's compliment format mutiplied by 4.<br>$SNR[dB] = \frac{PacketSnr[twos\ complement]}{4}$ |
| RegPktRssiValue (0x1A) | 7-0 | PacketRssi | r | n/a | RSSI of the latest packet received (dBm):<br>RSSI[dBm] = -157 + Rssi (using HF output port, SNR >= 0)<br>or<br>RSSI[dBm] = -164 + Rssi (using LF output port, SNR >= 0)<br>(see section 5.5.5 for details) |
| RegRssiValue (0x1B) | 7-0 | Rssi | r | n/a | Current RSSI value (dBm)<br>RSSI[dBm] = -157 + Rssi (using HF output port)<br>or<br>RSSI[dBm] = -164 + Rssi (using LF output port)<br>(see section 5.5.5 for details) |
| RegHopChannel (0x1C) | 7 | PllTimeout | r | n/a | PLL failed to lock while attempting a TX/RX/CAD operation<br>1 → PLL did not lock<br>0 → PLL did lock |
| | 6 | CrcOnPayload | r | n/a | CRC Information extracted from the received packet header (Explicit header mode only)<br>0 → Header indicates CRC off<br>1 → Header indicates CRC on |
| | 5-0 | FhssPresentChannel | r | n/a | Current value of frequency hopping channel in use. |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegModemConfig1 (0x1D) | 7-4 | Bw | rw | 0x07 | Signal bandwidth:<br>0000 → 7.8 kHz<br>0001 → 10.4 kHz<br>0010 → 15.6 kHz<br>0011 → 20.8kHz<br>0100 → 31.25 kHz<br>0101 → 41.7 kHz<br>0110 → 62.5 kHz<br>0111 → 125 kHz<br>1000 → 250 kHz<br>1001 → 500 kHz<br>other values → reserved<br>In the lower band (169MHz), signal bandwidths 8&9 are not supported) |
| | 3-1 | CodingRate | rw | '001' | Error coding rate<br>001 → 4/5<br>010 → 4/6<br>011 → 4/7<br>100 → 4/8<br>All other values → reserved<br>In implicit header mode should be set on receiver to determine expected coding rate. See 4.1.1.3 |
| | 0 | ImplicitHeaderModeOn | rw | 0x0 | 0 → Explicit Header mode<br>1 → Implicit Header mode |
| RegModemConfig2 (0x1E) | 7-4 | SpreadingFactor | rw | 0x07 | SF rate (expressed as a base-2 logarithm)<br>6 → 64 chips / symbol<br>7 → 128 chips / symbol<br>8 → 256 chips / symbol<br>9 → 512 chips / symbol<br>10 → 1024 chips / symbol<br>11 → 2048 chips / symbol<br>12 → 4096 chips / symbol<br>other values reserved. |
| | 3 | TxContinuousMode | rw | 0 | 0 → normal mode, a single packet is sent<br>1 → continuous mode, send multiple packets across the FIFO (used for spectral analysis) |
| | 2 | RxPayloadCrcOn | rw | 0x00 | Enable CRC generation and check on payload:<br>0 → CRC disable<br>1 → CRC enable<br>If CRC is needed, RxPayloadCrcOn should be set:<br>- in Implicit header mode: on Tx and Rx side<br>- in Explicit header mode: on the Tx side alone (recovered from the header in Rx side) |
| | 1-0 | SymbTimeout(9:8) | rw | 0x00 | RX Time-Out MSB |
| RegSymbTimeoutLsb (0x1F) | 7-0 | SymbTimeout(7:0) | rw | 0x64 | RX Time-Out LSB<br>RX operation time-out value expressed as number of symbols:<br>$TimeOut = SymbTimeout \cdot Ts$ |

| Name (Address) | Bits | Variable Name | Mode | Reset | LoRa™ Description |
|---|---|---|---|---|---|
| RegPreambleMsb (0x20) | 7-0 | PreambleLength(15:8) | rw | 0x0 | Preamble length MSB, = PreambleLength + 4.25 Symbols<br>See 4.1.1 for more details. |
| RegPreambleLsb (0x21) | 7-0 | PreambleLength(7:0) | rw | 0x8 | Preamble Length LSB |
| RegPayloadLength (0x22) | 7-0 | PayloadLength(7:0) | rw | 0x1 | Payload length in bytes. The register needs to be set in implicit header mode for the expected packet length. A 0 value is not permitted |
| RegMaxPayloadLength (0x23) | 7-0 | PayloadMaxLength(7:0) | rw | 0xff | Maximum payload length; if header payload length exceeds value a header CRC error is generated. Allows filtering of packet with a bad size. |
| RegHopPeriod (0x24) | 7-0 | FreqHoppingPeriod(7:0) | rw | 0x0 | Symbol periods between frequency hops. (0 = disabled). 1st hop always happen after the 1st header symbol |
| RegFifoRxByteAddr (0x25) | 7-0 | FifoRxByteAddrPtr | r | n/a | Current value of RX databuffer pointer (address of last byte written by Lora receiver) |
| RegModemConfig3 (0x26) | 7-4 | Unused | r | 0x00 | |
| | 3 | LowDataRateOptimize | rw | 0x00 | 0 → Disabled<br>1 → Enabled; mandated for when the symbol length exceeds 16ms |
| | 2 | AgcAutoOn | rw | 0x00 | 0 → LNA gain set by register LnaGain<br>1 → LNA gain set by the internal AGC loop |
| | 1-0 | Reserved | rw | 0x00 | Reserved |
| (0x27) | 7-0 | PpmCorrection | rw | 0x00 | Data rate offset value, used in conjunction with AFC |
| RegFeiMsb (0x28) | 7-4 | Reserved | r | n/a | Reserved |
| | 3-0 | FreqError(19:16) | r | 0x0 | Estimated frequency error from modem<br>MSB of RF Frequency Error<br>$F_{Error} = \frac{FreqError \times 2^{24}}{F_{xtal}} \times \frac{BW[kHz]}{500}$ |
| RegFeiMid (0x29) | 7-0 | FreqError(15:8) | r | 0x0 | Middle byte of RF Frequency Error |
| RegFeiLsb (0x2A) | 7-0 | FreqError(7:0) | r | 0x0 | LSB of RF Frequency Error |
| (0x2B) | - | Reserved | r | n/a | Reserved |
| RegRssiWideband (0x2C) | 7-0 | RssiWideband(7:0) | r | n/a | Wideband RSSI measurement used to locally generate a random number |
| (0x2D) - (0x2E) | - | Reserved | r | n/a | Reserved |
| 0x2F | 7-0 | IfFreq2 | rw | 0x20 | See errata note |

# 7. Application Information

## 7.1. Crystal Resonator Specification

Table 43 shows the crystal resonator specification for the crystal reference oscillator circuit of the RFM95W/96W/98W. This specification covers the full range of operation of the RFM95W/96W/98W and is employed in the reference design.

*Table 43 Crystal Specification*

| Symbol | Description | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| FXOSC | XTAL Frequency | | - | 32 | - | MHz |
| RS | XTAL Serial Resistance | | - | 30 | TBC | ohms |
| C0 | XTAL Shunt Capacitance | | - | 2.8 | TBC | pF |
| CFOOT | External Foot Capacitance | On each pin XTA and XTB | 8 | 15 | 22 | pF |
| CLOAD | Crystal Load Capacitance | | 6 | - | 12 | pF |

*Notes - the initial frequency tolerance, temperature stability and ageing performance should be chosen in accordance with the target operating temperature range and the receiver bandwidth selected.*

*- the loading capacitance should be applied externally, and adapted to the actual Cload specification of the XTAL.*

## 7.2. Reset of the Chip

A power-on reset of the RFM95W/96W/98W is triggered at power up. Additionally, a manual reset can be issued by controlling pin 6.

### 7.2.1. POR

If the application requires the disconnection of VDD from the RFM95W/96W/98W, despite of the extremely low Sleep Mode current, the user should wait for 10 ms from of the end of the POR cycle before commencing communications over the SPI bus. Pin 7 (NRESET) should be left floating during the POR sequence.

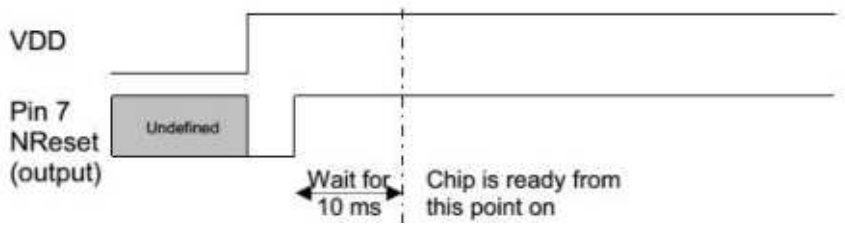



*Figure 42. POR Timing Diagram*

Please note that any CLKOUT activity can also be used to detect that the chip is ready.

### 7.2.2. Manual Reset

A manual reset of the RFM95W/96W/98W is possible even for applications in which VDD cannot be physically disconnected. Pin
7 should be pulled low for a hundred microseconds, and then released. The user should then wait for 5 ms before using the chip.

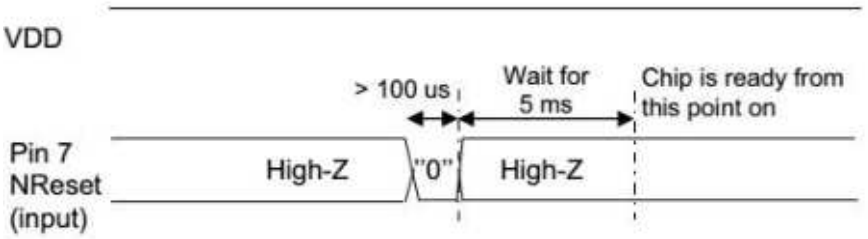



*Figure 43. Manual Reset Timing Diagram*

*Note whilst pin 7 is driven low, an over current consumption of up to one milliampere can be seen on VDD.*

## 7.3. Top Sequencer: Listen Mode Examples

In this scenario, the circuit spends most of the time in Idle mode, during which only the RC oscillator is on. Periodically the receiver wakes up and looks for incoming signal. If a wanted signal is detected, the receiver is kept on and data are analyzed. Otherwise, if there was no wanted signal for a defined period of time, the receiver is switched off until the next receive period.

During Listen mode, the Radio stays most of the time in a Low Power mode, resulting in very low average power consumption. The general timing diagram of this scenario is given in Figure 44.

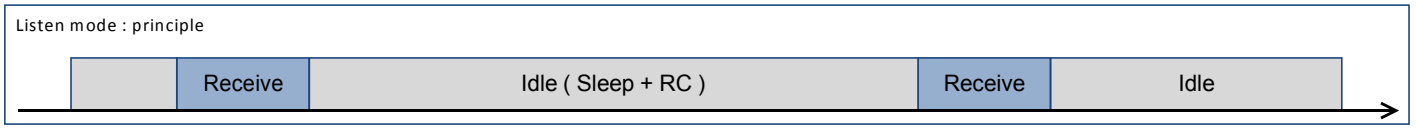



*Figure 44. Listen Mode: Principle*

An interrupt request is generated on a packet reception. The user can then take appropriate action.

Depending on the application and environment, there are several ways to implement Listen mode:

- Wake on a *PreambleDetect* interrupt
- Wake on *a SyncAddress* interrupt
- Wake on *a PayloadReady* interrupt

### 7.3.1. Wake on Preamble Interrupt

In one possible scenario, the sequencer polls for a Preamble detection. If a preamble signal is detected, the sequencer is switched off and the circuit stays in Receive mode until the user switches modes. Otherwise, the receiver is switched off until the next Rx period.

*7.3.1.1. Timing Diagram*

When no signal is received, the circuit wakes every Timer1 + Timer2 and switches to Receive mode for a time defined by Timer2, as shown on the following diagram. If no Preamble is detected, it then switches back to Idle mode, i.e. Sleep mode with RC oscillator on.

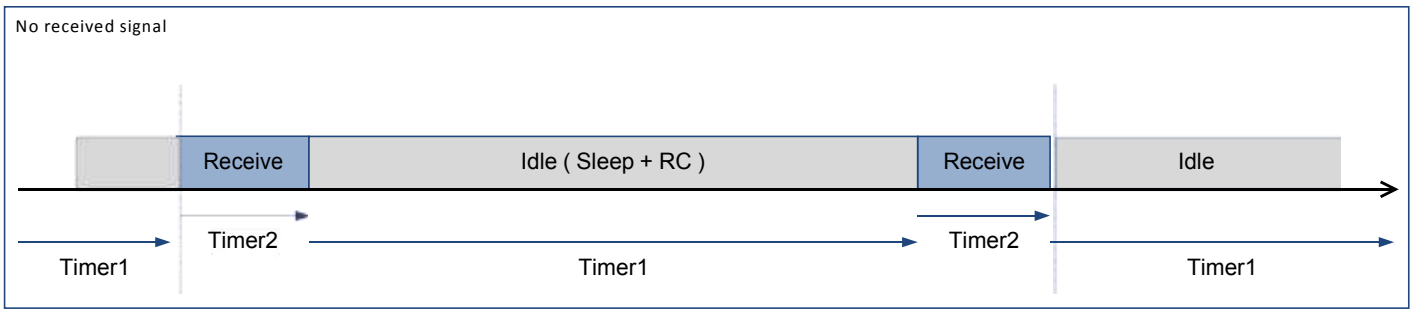



*Figure 45. Listen Mode with No Preamble Received*

If a Preamble signal is detected, the Sequencer is switched off. The *PreambleDetect* signal can be mapped to DIO4, in order to request the user's attention. The user can then take appropriate action.

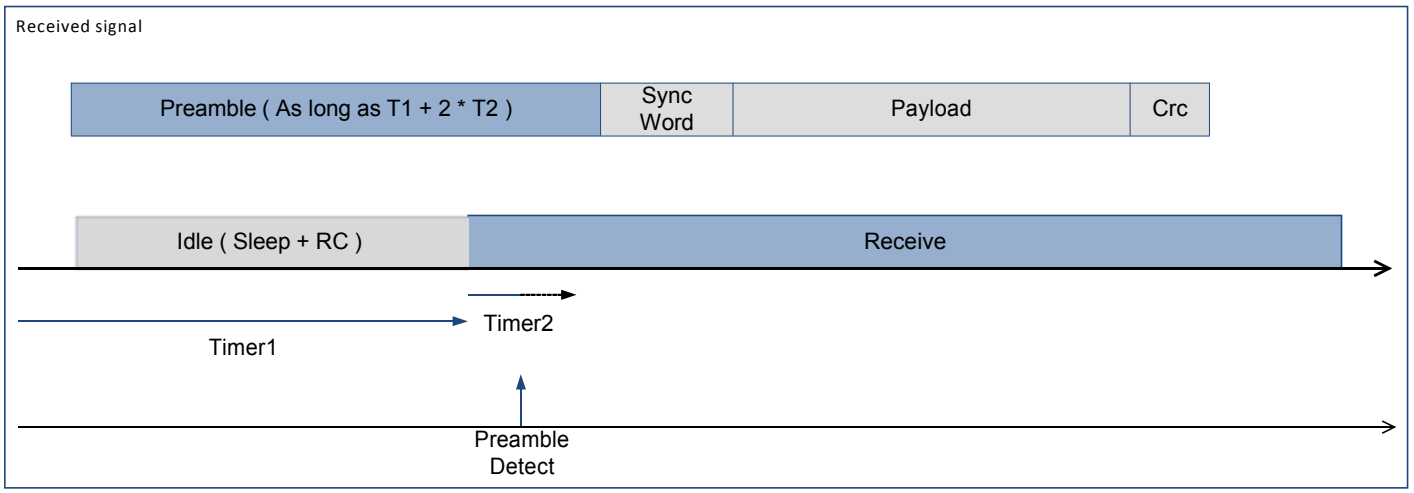



*Figure 46. Listen Mode with Preamble Received*

#### *7.3.1.2. Sequencer Configuration*

The following graph shows Listen mode - Wake on *PreambleDetect* state machine:

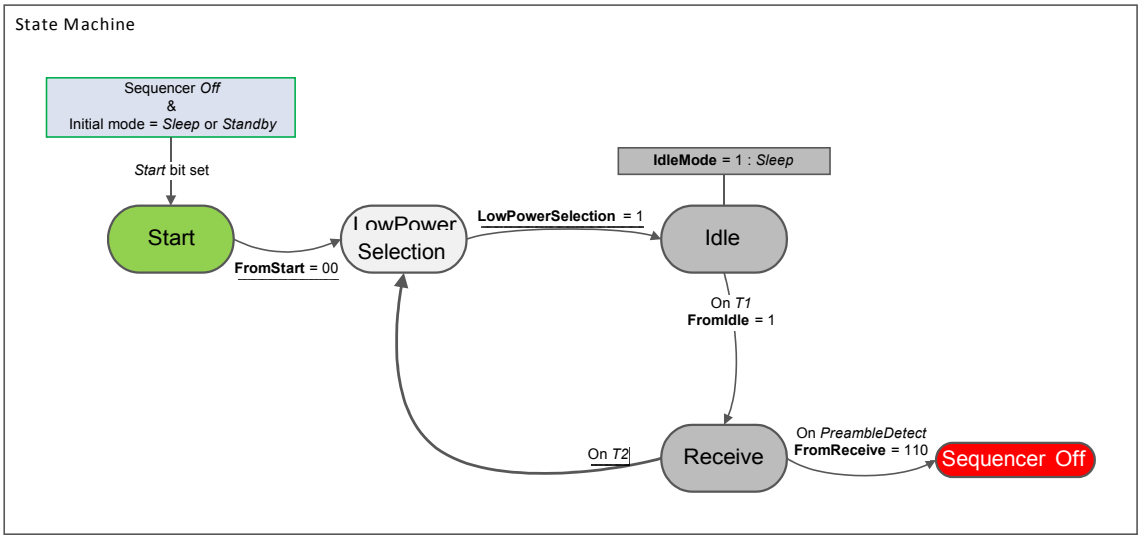



*Figure 47. Wake On PreambleDetect State Machine*

This example configuration is achieved as follows:

*Table 44 Listen Mode with PreambleDetect Condition Settings*

| Variable | Effect |
|---|---|
| IdleMode | 1: **Sleep** mode |
| FromStart | 00: To **LowPowerSelection** |
| LowPowerSelection | 1: To **Idle** state |
| FromIdle | 1: To **Receive** state on *T1* interrupt |
| FromReceive | 110: To **Sequencer Off** on *PreambleDetect* interrupt |

$T_{Timer2}$ defines the maximum duration the chip stays in Receive mode as long as no Preamble is detected. In order to optimize power consumption, Timer2 must be set just long enough for Preamble detection.

$T_{Timer1}$ + $T_{Timer2}$ defines the cycling period, i.e. time between two Preamble polling starts. In order to optimize average power consumption, Timer1 should be relatively long. However, increasing Timer1 also extends packet reception duration.

In order to insure packet detection and optimize the receiver's power consumption, the received packet Preamble should be as long as $T_{Timer1}$ + 2 x $T_{Timer2}$.

An example of DIO configuration for this mode is described in the following table:

*Table 45 Listen Mode with PreambleDetect Condition Recommended DIO Mapping*

| DIO | Value | Description |
|---|---|---|
| 0 | 01 | CrcOk |
| 1 | 00 | FifoLevel |
| 3 | 00 | FifoEmpty |
| 4 | 11 | PreambleDetect – Note: *MapPreambleDetect* bit should be set. |

### 7.3.2. Wake on SyncAddress Interrupt

In another possible scenario, the sequencer polls for a Preamble detection and then for a valid *SyncAddress* interrupt. If events occur, the sequencer is switched off and the circuit stays in Receive mode until the user switches modes. Otherwise, the receiver is switched off until the next Rx period.

#### *7.3.2.1. Timing Diagram*

Most of the sequencer running time is spent while no wanted signal is received. As shown by the timing diagram in Figure 48, the circuit wakes periodically for a short time, defined by RxTimeout. The circuit is in a Low Power mode for the rest of Timer1 + Timer2 (i.e. Timer1 + Timer2 - TrxTimeout)

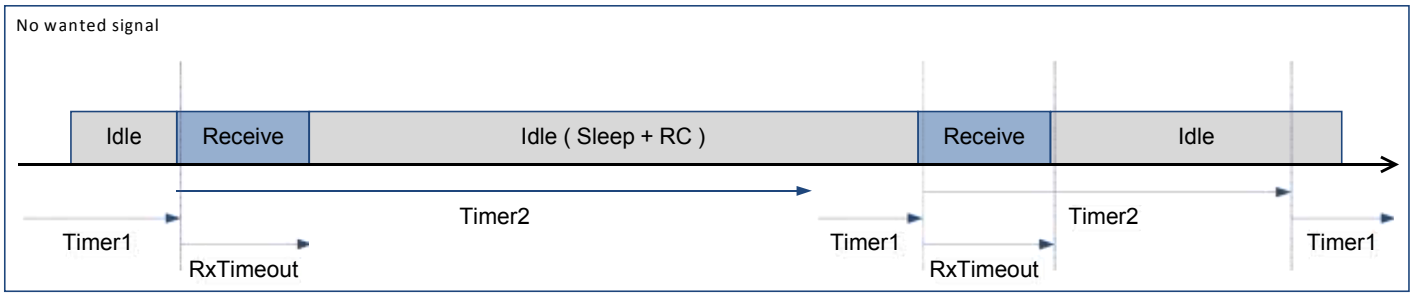



*Figure 48. Listen Mode with no SyncAddress Detected*

If a preamble is detected before *RxTimeout* timer ends, the circuit stays in Receive mode and waits for a valid *SyncAddress* detection. If none is detected by the end of Timer2, Receive mode is deactivated and the polling cycle resumes, without any user intervention.

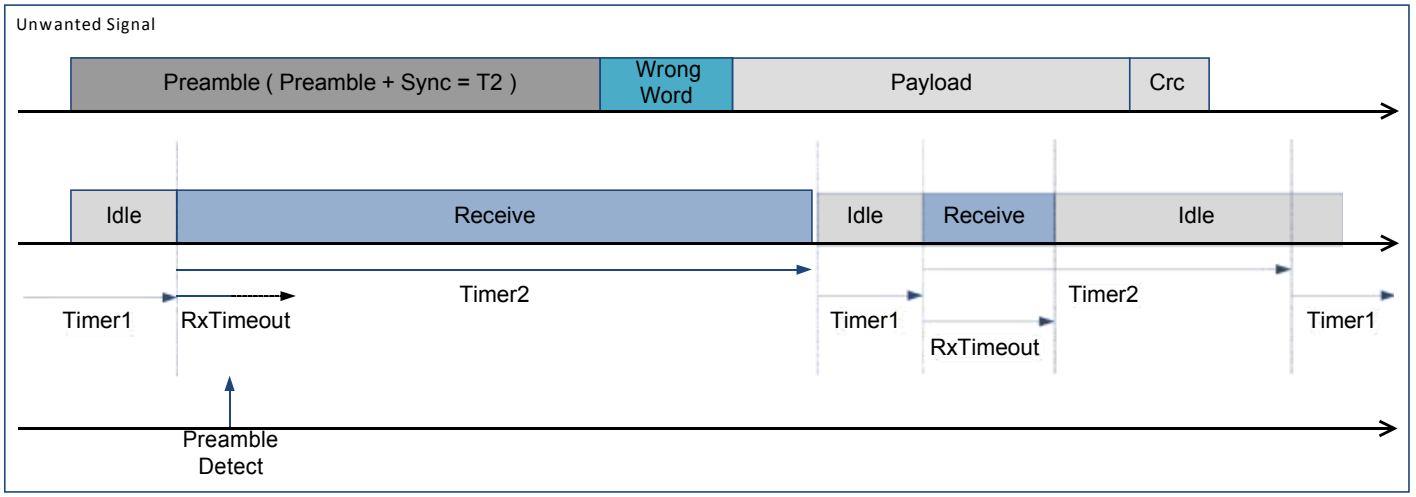



*Figure 49. Listen Mode with Preamble Received and no SyncAddress*

But if a valid Sync Word is detected, a *SyncAddress* interrupt is fired, the Sequencer is switched off and the circuit stays in Receive mode as long as the user doesn't switch modes.

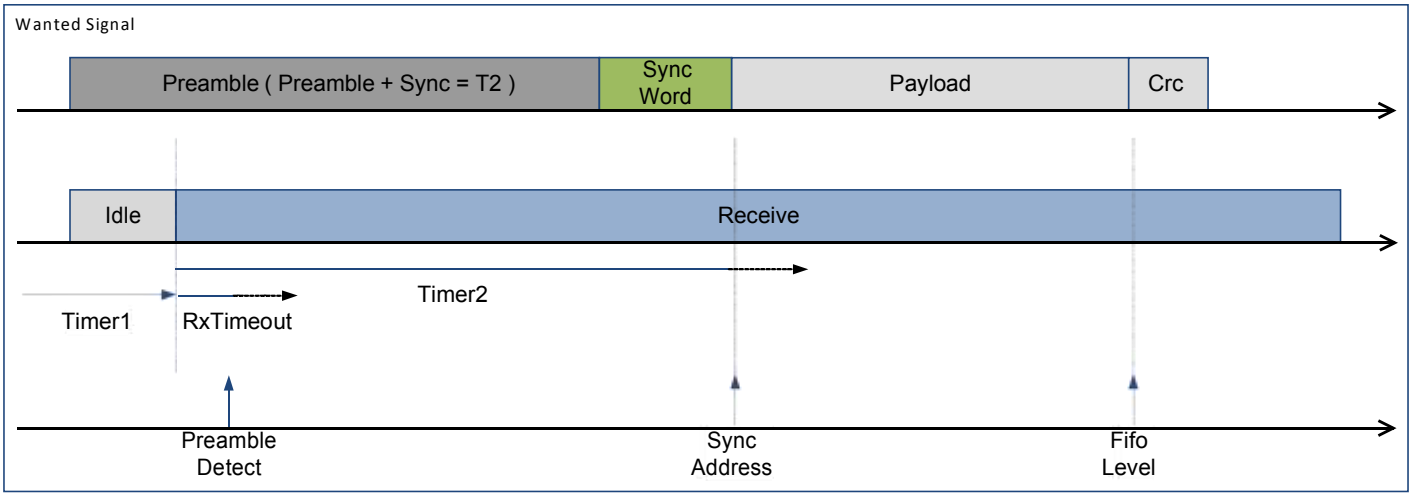



*Figure 50. Listen Mode with Preamble Received & Valid SyncAddress*

### *7.3.2.2. Sequencer Configuration*

The following graph shows Listen mode - Wake on SyncAddress state machine:

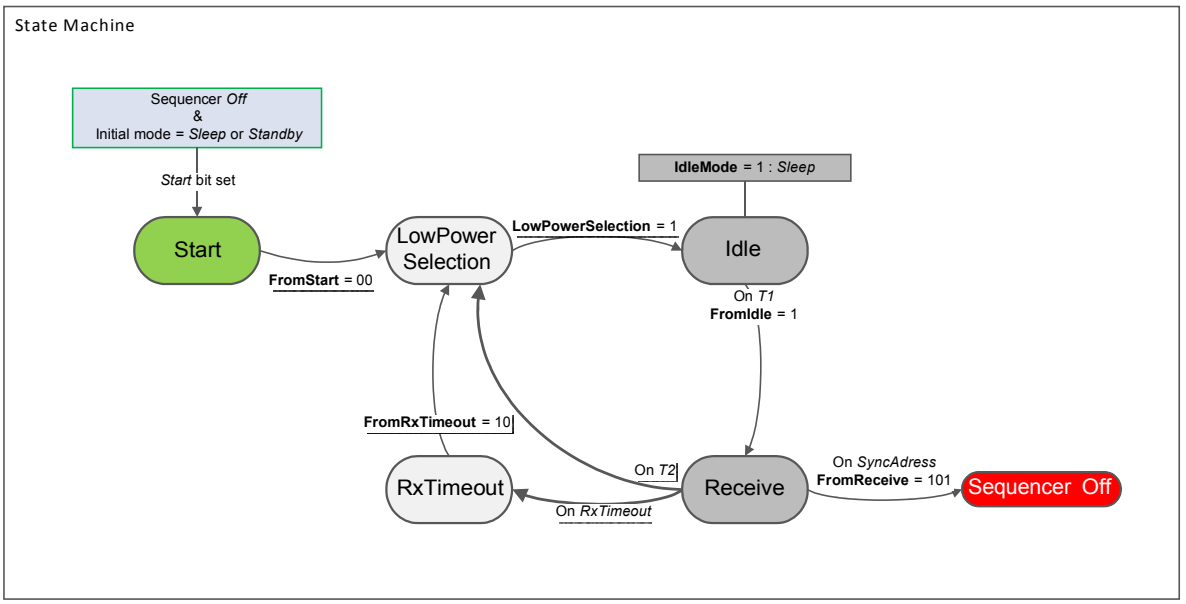



*Figure 51. Wake On SyncAddress State Machine*

This example configuration is achieved as follows:

*Table 46 Listen Mode with SyncAddress Condition Settings*

| Variable | Effect |
|---|---|
| IdleMode | 1: **Sleep** mode |
| FromStart | 00: To **LowPowerSelection** |
| LowPowerSelection | 1: To **Idle** state |
| FromIdle | 1: To **Receive** state on *T1* interrupt |
| FromReceive | 101: To **Sequencer off** on *SyncAddress* interrupt |
| FromRxTimeout | 10: To **LowPowerSelection** |

$T_{TimeoutRxPreamble}$ should be set to just long enough to catch a preamble (depends on *PreambleDetectSize* and *BitRate*).

$T_{Timer1}$ should be set to 64 µs (shortest possible duration).

$T_{Timer2}$ is set so that $T_{Timer1} + T_{Timer2}$ defines the time between two start of reception.

In order to insure packet detection and optimize the receiver power consumption, the received packet Preamble should be defined so that $T_{Preamble} = T_{Timer2} - T_{SyncAddress}$ with $T_{SyncAddress} = (SyncSize + 1)*8/BitRate$.

An example of DIO configuration for this mode is described in the following table:

*Table 47 Listen Mode with PreambleDetect Condition Recommended DIO Mapping*

| DIO | Value | Description |
|---|---|---|
| 0 | 01 | CrcOk |
| 1 | 00 | FifoLevel |
| 2 | 11 | SyncAddress |
| 3 | 00 | FifoEmpty |
| 4 | 11 | PreambleDetect – Note: *MapPreambleDetect* bit should be set. |

## 7.4. Top Sequencer: Beacon Mode

In this mode, a repetitive message is transmitted periodically. If the Payload being sent is always identical, and *PayloadLength* is smaller than the FIFO size, the use of the *BeaconOn* bit in *RegPacketConfig2* together with the Sequencer permit to achieve periodic beacon without any user intervention.

### 7.4.1. Timing diagram

In this mode, the Radio is switched to Transmit mode every $T_{Timer1}$ + $T_{Timer2}$ and back to Idle mode after *PacketSent*, as shown in the diagram below. The Sequencer insures minimal time is spent in Transmit mode, and therefore power consumption is optimized.

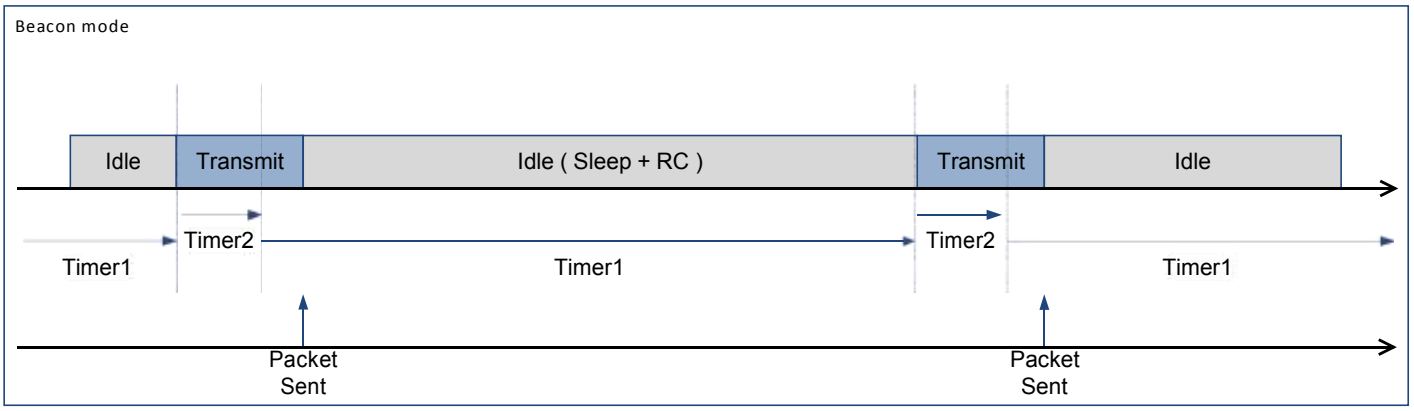



*Figure 52. Beacon Mode Timing Diagram*

### 7.4.2. Sequencer Configuration

The Beacon mode state machine is presented in the following graph. It is noticeable that the sequencer enters an infinite loop and can only be stopped by setting *SequencerStop* bit in *RegSeqConfig1*.

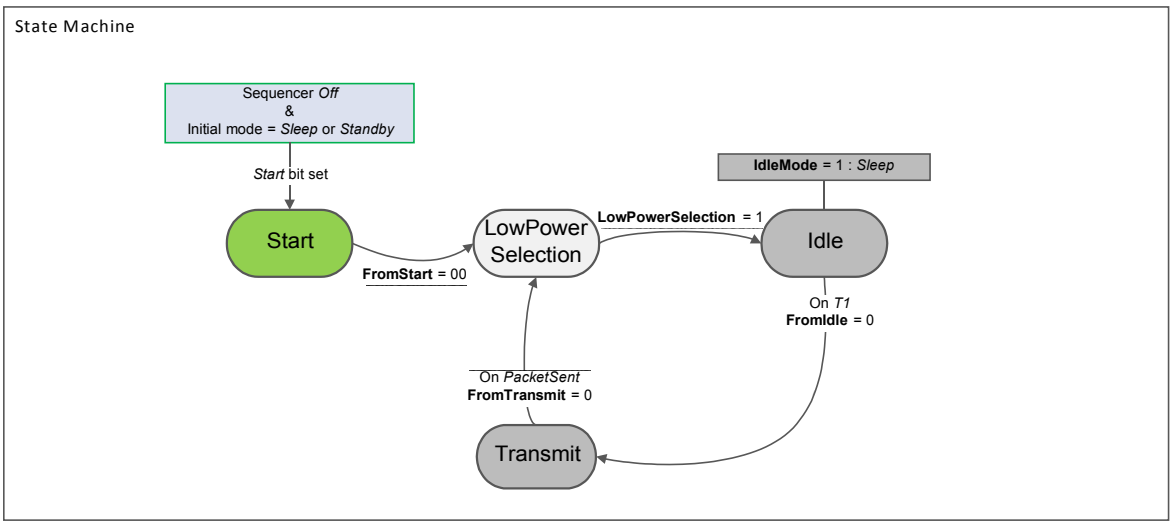



*Figure 53. Beacon Mode State Machine*

This example is achieved by programming the Sequencer as follows:

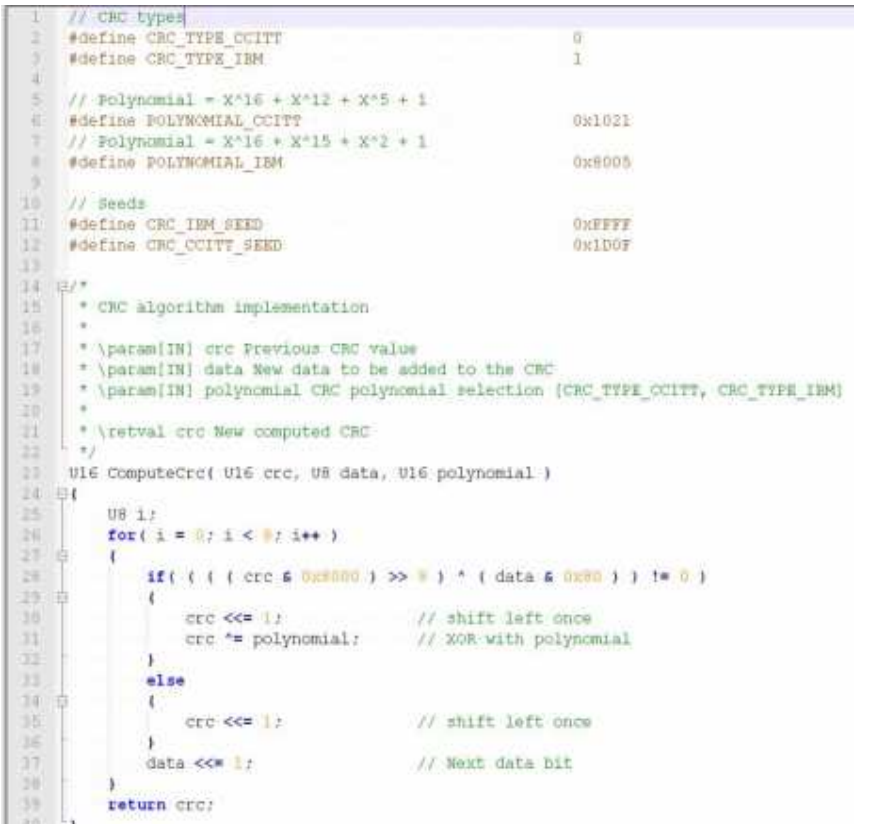
*Table 48 Beacon Mode Settings*

| Variable | Effect |
|---|---|
| IdleMode | 1: **Sleep** mode |
| FromStart | 00: To **LowPowerSelection** |
| LowPowerSelection | 1: To **Idle** state |
| FromIdle | 0: To **Transmit** state on *T1* interrupt |
| FromTransmit | 0: To **LowPowerSelection** on *PacketSent* interrupt |

$T_{Timer1} + T_{Timer2}$ define the time between the start of two transmissions.

## 7.5. Example CRC Calculation

The following routine(s) may be implemented to mimic the CRC calculation of the RFM95W/96W/98W:



```
// CRC types
#define CRC_TYPE_CCITT                          0
#define CRC_TYPE_IBM                            1

// Polynomial = X^16 + X^12 + X^5 + 1
#define POLYNOMIAL_CCITT                        0x1021
// Polynomial = X^16 + X^15 + X^2 + 1
#define POLYNOMIAL_IBM                          0x8005

// Seeds
#define CRC_IBM_SEED                            0xFFFF
#define CRC_CCITT_SEED                          0x1D0F

/*
 * CRC algorithm implementation
 *
 * \param[IN] crc Previous CRC value
 * \param[IN] data New data to be added to the CRC
 * \param[IN] polynomial CRC polynomial selection (CRC_TYPE_CCITT, CRC_TYPE_IBM)
 *
 * \retval crc New computed CRC
 */
U16 ComputeCrc( U16 crc, U8 data, U16 polynomial )
{
    U8 i;
    for( i = 0; i < 8; i++ )
    {
        if( ( ( ( crc & 0x8000 ) >> 8 ) ^ ( data & 0x80 ) ) != 0 )
        {
            crc <<= 1;              // shift left once
            crc ^= polynomial;      // XOR with polynomial
        }
        else
        {
            crc <<= 1;              // shift left once
        }
        data <<= 1;                 // Next data bit
    }
    return crc;
}
```

```
/*
 * CRC algorithm implementation
 *
 * \param[IN] buffer Array containing the data
 * \param[IN] bufferLength Buffer length
 * \param[IN] crcType Selects the CRC polynomial(CRC_TYPE_CCITT, CRC_TYPE_IBM)
 *
 * \retval crc Buffer computed CRC
 */
U16 RadioPacketComputeCrc( U8 *buffer, U8 bufferLength, U8 crcType )
{
    U8 i;
    U16 crc;
    U16 polynomial;

    polynomial = ( crcType == CRC_TYPE_IBM ) ? POLYNOMIAL_IBM : POLYNOMIAL_CCITT;
    crc = ( crcType == CRC_TYPE_IBM ) ? CRC_IBM_SEED : CRC_CCITT_SEED;

    for( i = 0; i < bufferLength; i++ )
    {
        crc = ComputeCrc( crc, buffer[i], polynomial );
    }

    if( crcType == CRC_TYPE_IBM )
    {
        return crc;
    }
    else
    {
        return ( U16 )( ~crc );
    }
}
```

*Figure 54. Example CRC Code*

## 7.6. Example Temperature Reading

The following routine(s) may be implemented to read the temperature and calibrate the sensor:

Temperature.c

```
/*!
 * Reads the raw temperature
 * \retval temperature New raw temperature reading in 2's complement format
 */
S8 RadioGetRawTemp( void )
{
    S8 temp = 0;
    U8 regValue = 0;

    regValue = RadioRead( 0x3C );

    // 2's complements conversion
    temp = regValue & 0x7F;
    if( ( regValue & 0x80 ) == 0x80 )
    {
        temp *= -1;
    }
    return temp;
}

/*!
 * Computes the temperature compensation factor
 * \param [IN] actualTemp Actual temperature measured by an external device
 * \retval compensationFactor Computed compensation factor
 */
S8 RadioCalibrateTemp( S8 actualTemp )
{
    return actualTemp - RadioGetRawTemp( );
}

/*!
 * Gets the actual compensated temperature
 * \param [IN] compensationFactor Return value of the calibration function
 * \retval New compensated temperature value
 */
S8 RadioGetTemp( S8 compensationFactor )
{
    return RadioGetRawTemp( ) + compensationFactor;
}

/*!
 * Usage example
 */
void main( void )
{
    S8 temp;
    S8 actualTemp = 0;
    S8 compensationFactor = 0;

    // Ask user for the temperature during calibration
    actualTemp = AskUserTemperature( );
    compensationFactor = RadioCalibrateTemp( actualTemp );

    while( True )
    {
        temp = RadioGetTemp( compensationFactor );
    }
}
```

*Figure 55. Example Temperature Reading*

## 7.7. Reference Design

Please contact your representative for evaluation tools, reference designs and design assistance. Note that all schematics shown in this section are partial schematics, maybe not listing ALL required components.

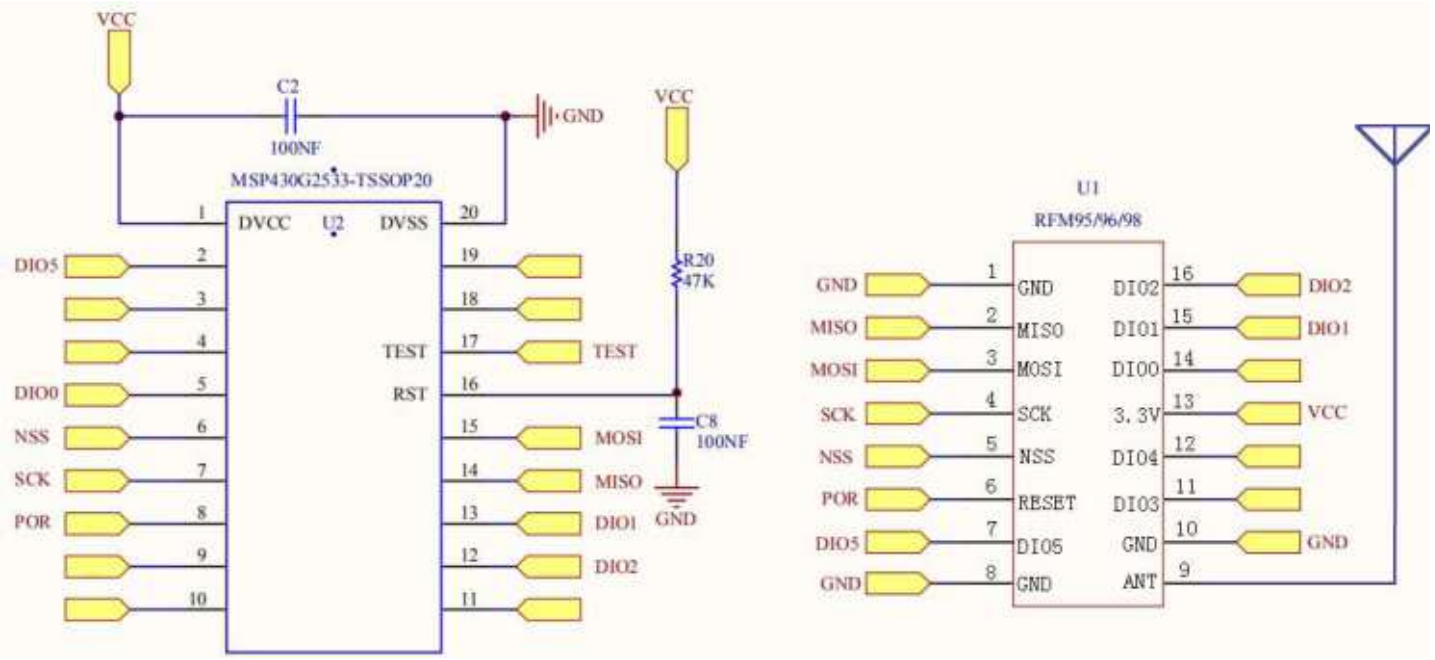



*Figure 56:Reference Schematic*

# 8. Packaging Information

## 8.1. Package Outline Drawing

8.2. The RFM95W/96W/98W Package Outline Drawing is available as shown in Figure 57

UNIT：mm

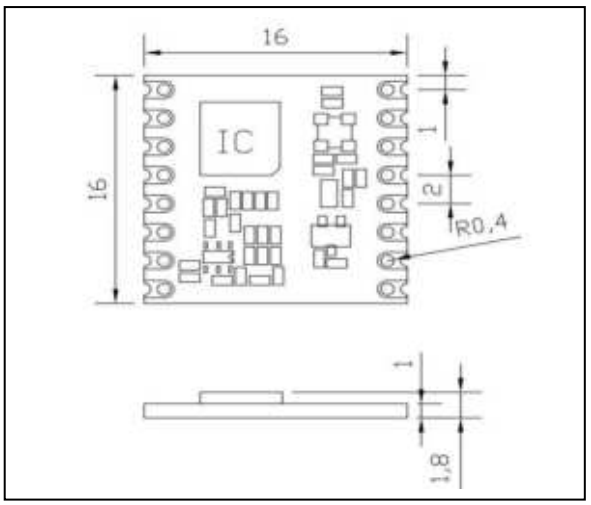
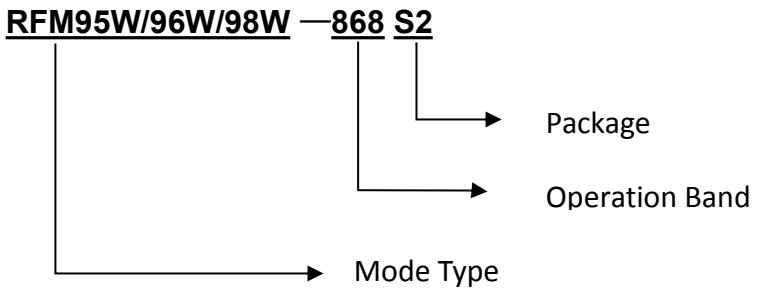



*Figure 57:Package Outline Drawing*

## 8.2. Ordering Information




**P/N: RFM95W-868S2**
**RFM95W module at 868MHz band, SMD Package**
**P/N: RFM95W-915S2**
**RFM95W module at 915MHz band, SMD Package**
**P/N: RFM96W-433S2**
**RFM96W module at 433MHz band, SMD Package**
**P/N: RFM96W-470S2**
**RFM96W module at 470MHz band, SMD Package**
**P/N: RFM98W-169S2**
**RFM98W module at 169MHz band, SMD Package**
**P/N: RFM98W-433S2**
**RFM98W module at 433MHz band, SMD Package**
**P/N: RFM98W-470S2**
**RFM98W module at 470MHz band, SMD Package**

# 9. Contact Information

**HOPE MICROELECTRONICS CO.,LTD**
Add: 2/F, Building 3, Pingshan Minqi Park, Lishan Road, XiLi Town, Nanshan District, Shenzhen, Guangdong, China
Tel: +86-755-82973805
Fax: +86-755-82973550
Email: sales@hoperf.com
Website: http://www.hoperf.com
http://www.hoperf.cn

## Revision History

| Version | Date | Description |
|---|---|---|
| V1.0 | 10/2013 | Created. |
| V2.0 | 07/2019 | Remove RFM97 information, modify some LoRa descriptions, change page format, update reference application SCH, add revision history. |

This document may contain preliminary information and is subject to change by Hope Microelectronics without notice. Hope Microelectronics assumes no responsibility or liability for any use of the information contained herein. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Hope Microelectronics or third parties. The products described in this document are not intended for use in implantation or other direct life support applications where malfunction may result in the direct physical harm or injury to persons.
NO WARRANTIES OF ANY KIND, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MECHANTABILITY OR FITNESS FOR A ARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.

©2018, HOPE MICROELECTRONICS CO.,LTD. All rights reserved.