TRW-HP9M2W RF MODULE

Hi Power 2W 900 MHz Transceiver Module

Product Description

TRW-HP9M2W is a high-power wireless radio module. With high-performance transceivers, it can reach its maximum efficiency at very low power consumption. This module is mainly used in the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device). Its frequency band is 902MHz~928 MHz, and it has a built-in PA. May flexibly adjust the output power externally. In order to keep abreast of the module in the operating temperature range, you can not only control the output and receive data but also read the RF IC's temperature through the SPI control interface.

Key Feature

- Frequency Range：902MHz~928MHz
- Modulation：(G)FSK, 4(G)FSK, (G)MSK
- Max Output Power：33dBm
- Auto frequency control (AFC)
- Automatic gain control (AGC)
- Receive sensitivity：–129dBm
- 60dB adjacent channel
- Data Rate：100 bps to 1 Mbps
- Temperature Sensor
- SPI microcontroller interface
- Operating Temperature：-30~+80℃

Application

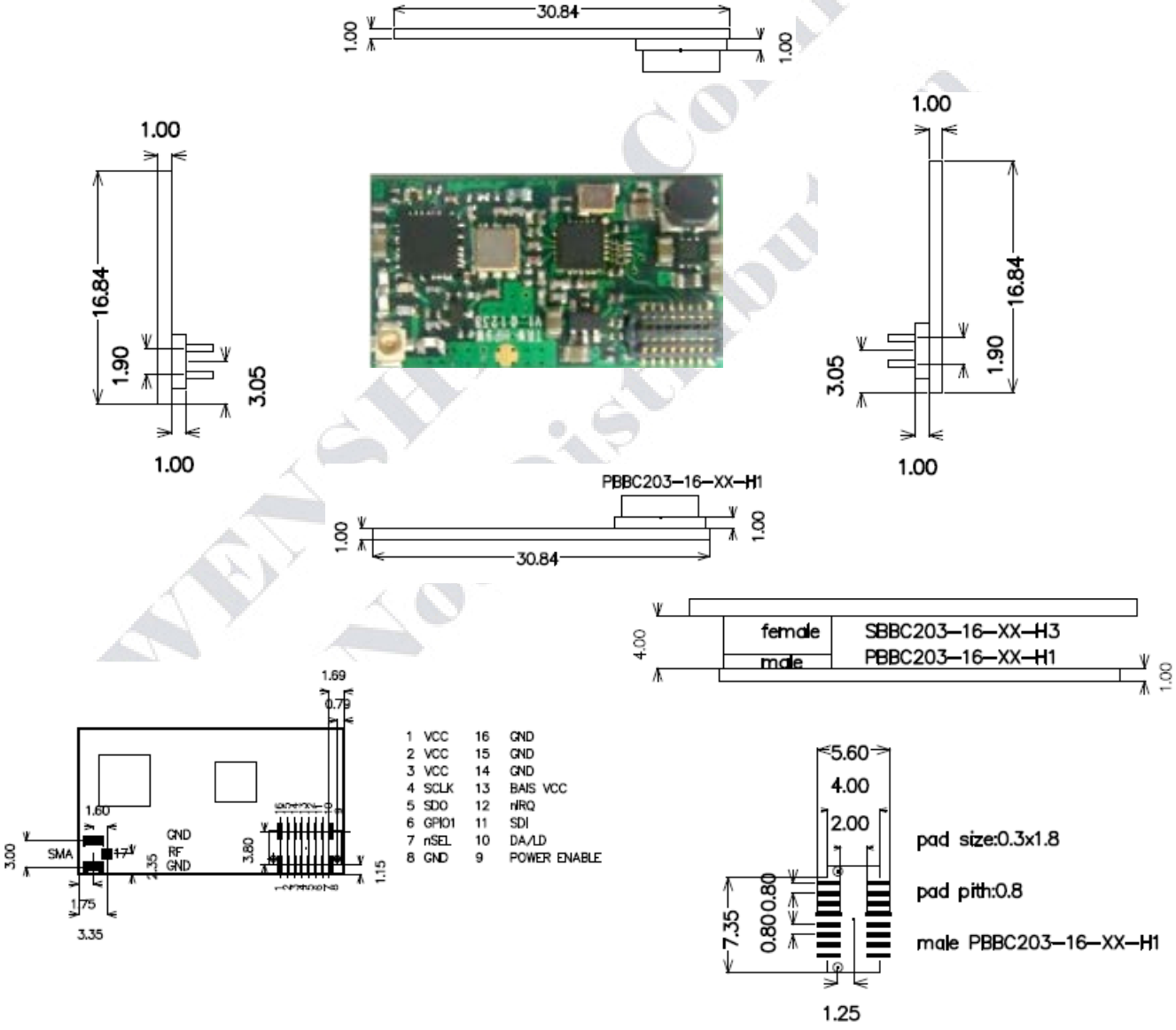
- ■ AMR
- ■ Wireless M-Bus
- ■ Wireless healthcare
- ■ Security applications
- ■ Messaging / paging
- ■ Private mobile radio
- ■ Wireless Metering
- ■ Remote control
- ■ Social Alarms

Version History

Version	Date	Changes
V1.01	Aug. 15, 2013	1st. Edition

Description

TRW-HP9M2W RF Module is high-performance, low-current transceivers covering the sub-GHz frequency bands from 902 to 928MHz. The radios are part of the TRW-HP9M2W RF Module family, which includes a complete line of transmitters, receivers, and transceivers covering a wide range of applications. All parts offer outstanding sensitivity of –129dBm while achieving extremely low active and standby current consumption. The TRW-HP9M2W offers frequency coverage in all major bands. The TRW-HP9M2W RF Module offers frequency coverage in bands not covered by TRW-HP9M2W RF Module family. Typically, these are non-standard frequencies or licensed frequency bands. The TRW-HP9M2W includes optimal phase noise, blocking, and selectivity performance for narrow band and licensed band applications, such as FCC Part90. The 60 dB adjacent channel selectivity with 12.5 KHz channel spacing ensures robust receive operation in harsh RF conditions, which is particularly important for narrow band operation. The TRW-HP9M2W RF Module offers exceptional output power of up to 33dBm with outstanding TX efficiency. The high output power and sensitivity results in an industry-leading link budget of 146 dB allowing extended ranges and highly robust communication links. FCC, ETSI, and ARIB. All devices are designed to be compliant with 802.15.4g and WMbus smart metering standards.



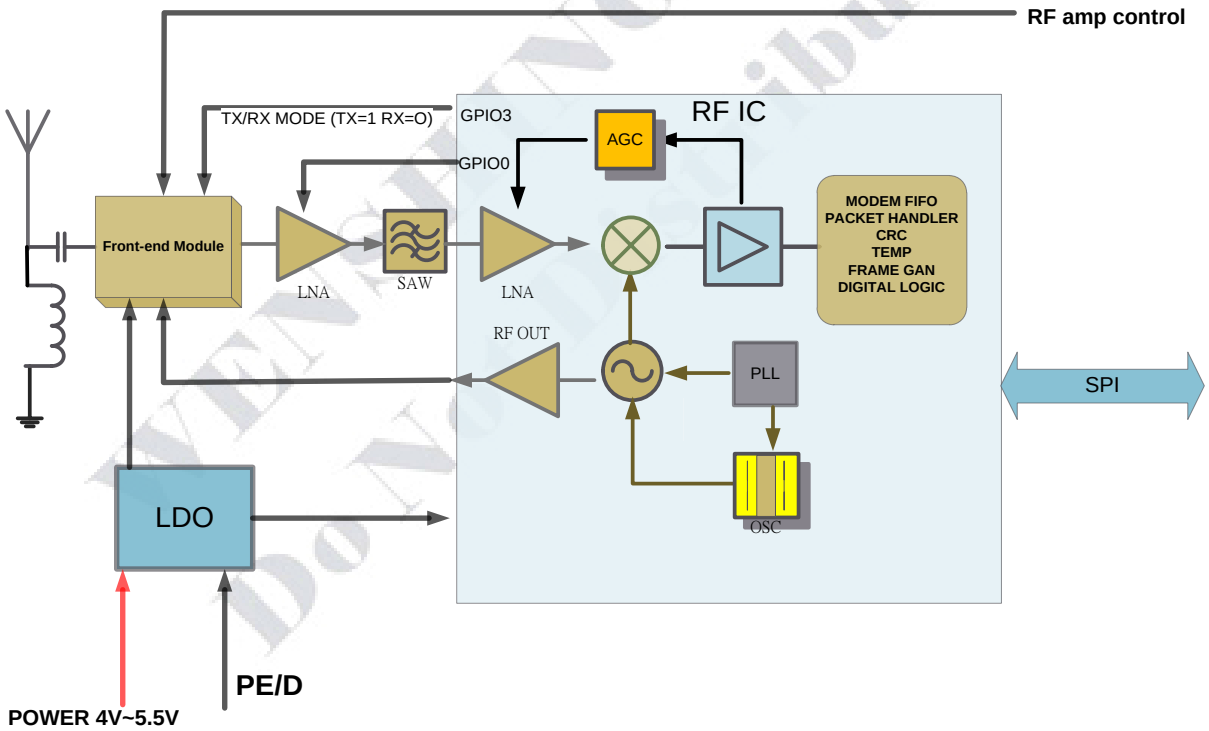
1. Specification

Conditions=925MHz VDD =VCC= +5V, VSS = 0V, TA = +25ºC

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Voltage Range	POWER			5		V
Power Down	POWER	Power Down		2		uA
RX Mode Current	Receiver	1.8KBPS		22.4		mA
RX Mode Current	Receiver	172KBPS		23.3		mA
TX Mode Current	Transmitter	27dBm(RAMP 0.75 Voltage)		730		mA
TX Mode Current	Transmitter	30dBm(RAMP 1 Voltage)		1100		mA
TX Mode Current	Transmitter	33dBm(RAMP1.8 Voltage)		2000		mA
RX Sensitivity	Receiver	1KBPs		-126		-dBm

TRW-HP9M2W RF MODULE

Function Block Diagram

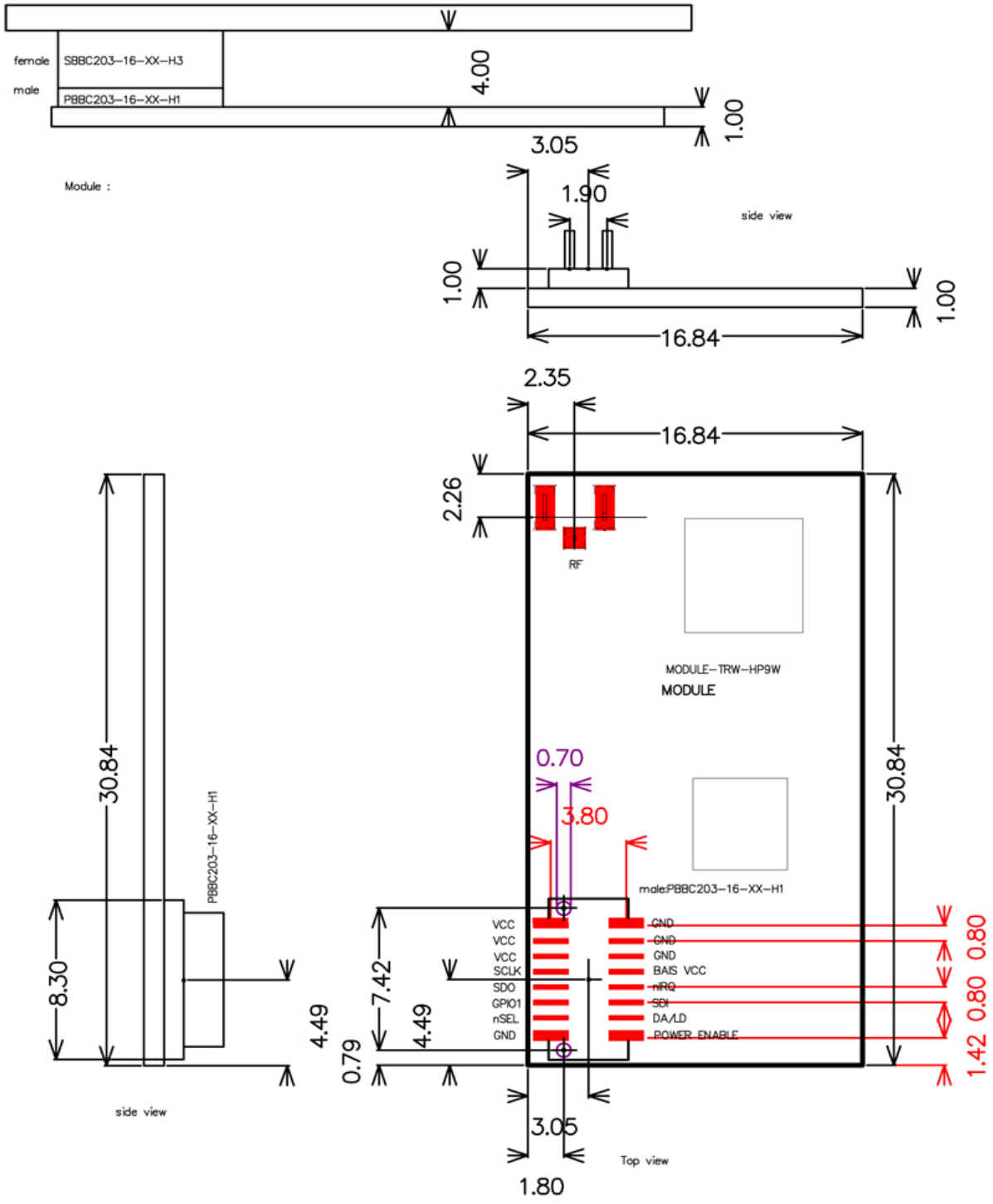


This block diagram details the whole structure of this module which allows user to adopt two channels simultaneously without adding any extra components except micro-controller. We shall have reference schematic how this module works with micro-controller in the subsequence.

Pin Assignments

Pin	Name	I/O	Description
1	VCC	Input	+4 to +6 V Supply Voltage Input to Internal Regulators
2	VCC	Input	+4 to +6 V Supply Voltage Input to Internal Regulators
3	VCC	Input	+4 to +6 V Supply Voltage Input to Internal Regulators
4	SCLK	Digital Input	Serial Clock Input. digital input. This pin provides the serial data clock function for the 4-line serial data bus. Data is clocked into the RF CHIP on positive edge transitions
5	SDO	Digital Output	Provides a serial readback function of the internal control registers.
6	GPIO1	I/O	May be configured through the registers to perform various functions including: Microcontroller Clock Output, FIFO status, POR, Wake-Up timer, Low Battery Detect, AntDiversity control, etc.
7	nSEL	Digital Input	This pin provides the Select/Enable function for the 4-line serial data bus.
8	GND	GND	ground.
9	PE/D	Digital Input	digital input. PED should be = 0 Power Down 1=POWER
10	RAMP	Input	RF amp control pin (1V to 3.3V control the power. Power by the 27DBM to 33DBM, should pay attention to not exceed 3.3V Voltage)
12	SDI	Digital Input	This pin provides the serial data stream for the 4-line serial data bus
13	VDD	Input	+4 to +5.5 V Supply Voltage Input to Internal Regulators The power supply to the RF CHIP
14	GND	GND	ground.
15	GND	GND	ground.
16	GND	GND	ground.

2. Reference schematics

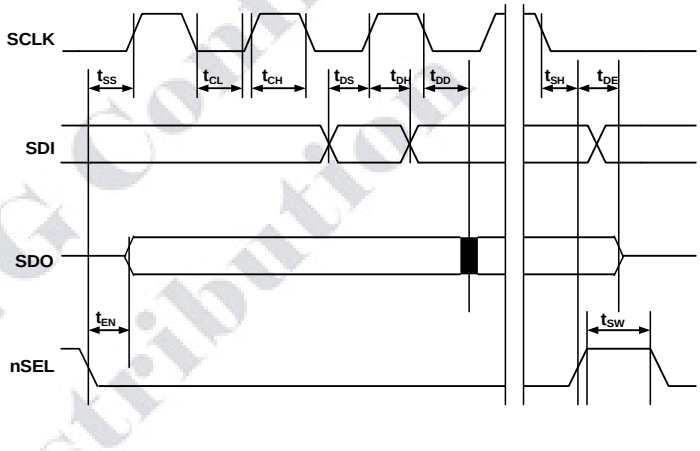


3. Controller Interface

3.1. Serial Peripheral Interface (SPI)

The TRW-HP9M2W communicates with the host MCU over a standard 4-wire serial peripheral interface (SPI): SCLK, SDI, SDO, and nSEL. The SPI interface is designed to operate at a maximum of 10 MHz. The SPI timing parameters are demonstrated in Table 8. The host MCU writes data over the SDI pin and can read data from the device on the SDO output pin. Figure 3 demonstrates an SPI write command. The nSEL pin should go low to initiate the SPI command. The first byte of SDI data will be one of the firmware commands followed by n bytes of parameter data which will be variable depending on the specific command. The rising edges of SCLK should be aligned with the center of the SDI data.

Table 8. Serial Interface Timing Parameters

Symbol	Parameter	Min (ns)	Diagram
t_{CH}	Clock high time	40	
t_{CL}	Clock low time	40	
t_{DS}	Data setup time	20	
t_{DH}	Data hold time	20	
t_{DD}	Output data delay time	20	
t_{EN}	Output enable time	20	
t_{DE}	Output disable time	50	
t_{SS}	Select setup time	20	
t_{SH}	Select hold time	50	
t_{SW}	Select high period	80	

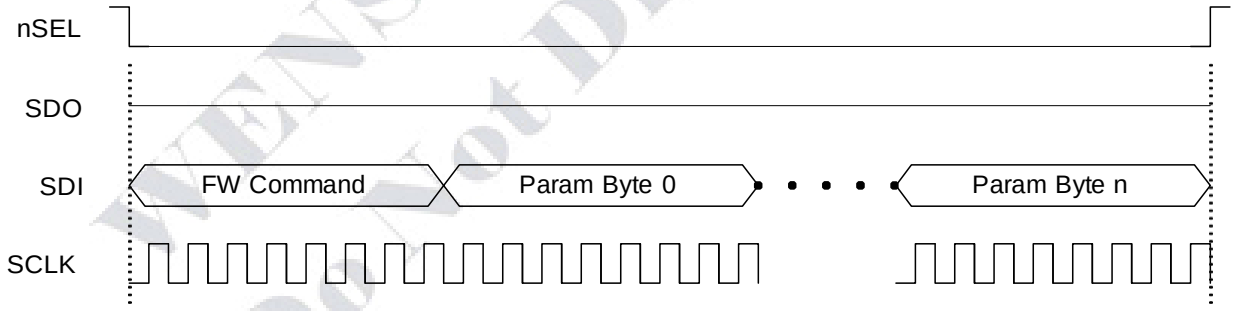


Figure 3. SPI Write Command

The TRW-HP9M2W contains an internal MCU which controls all the internal functions of the radio. For SPI read commands a typical MCU flow of checking clear-to-send (CTS) is used to make sure the internal MCU has executed the command and prepared the data to be output over the SDO pin. Figure 4 demonstrates the general flow of an SPI read command. Once the CTS value reads FFh then the read data is ready to be clocked out to the host MCU. The typical time for a valid FFh CTS reading is 20 µs. Figure 5 demonstrates the remaining read cycle after CTS is set to FFh. The internal MCU will clock out the SDO data on the negative edge so the host MCU should process the SDO data on the rising edge of SCLK.

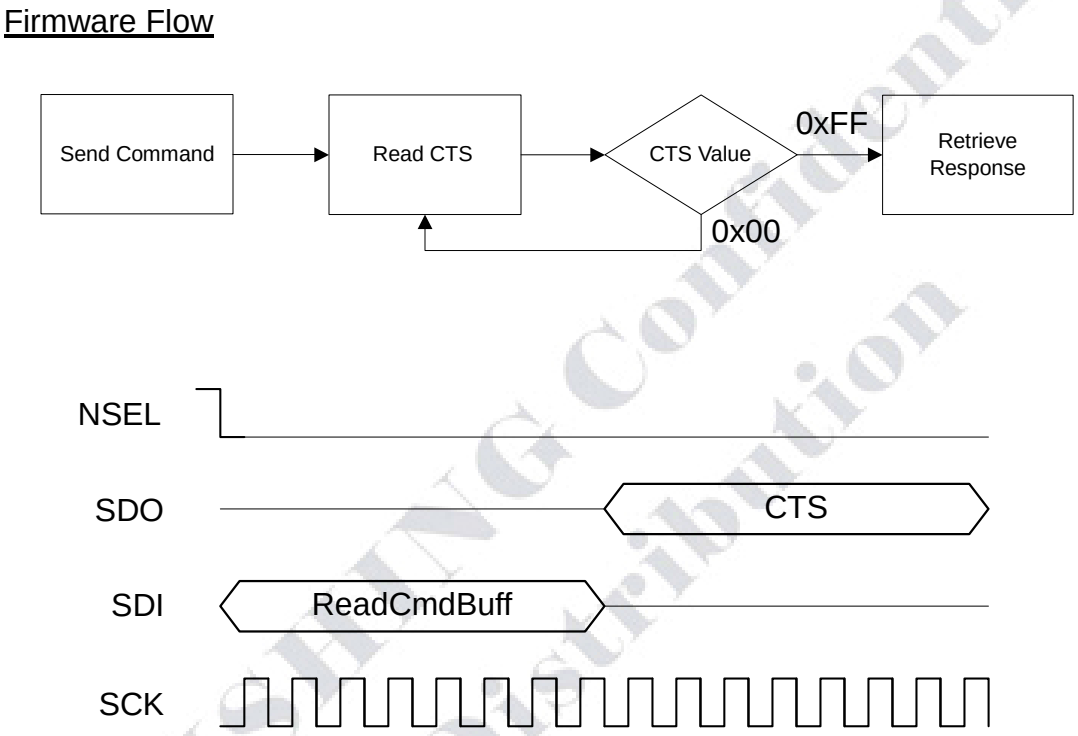


Figure 4. SPI Read Command—Check CTS Value

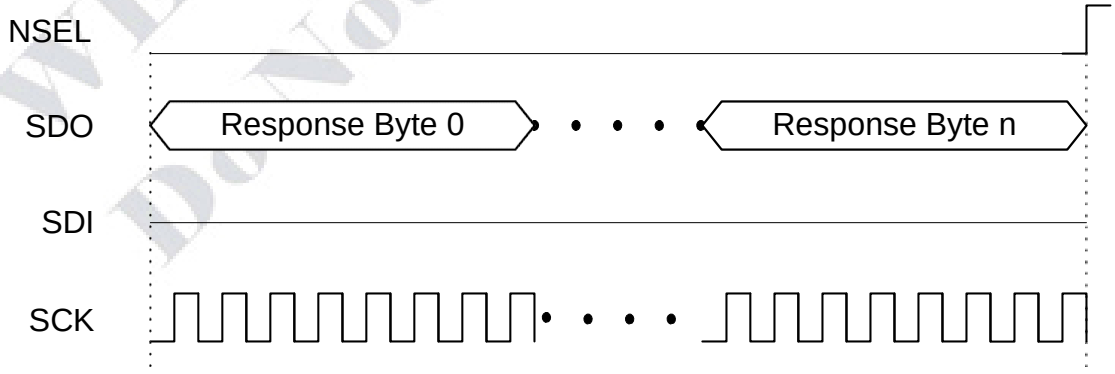


Figure 5. SPI Read Command—Clock Out Read Data

3.2. Fast Response Registers

The fast response registers are registers that can be read immediately without the requirement to monitor and check CTS. There are four fast response registers that can be programmed for a specific function. The fast response registers can be read through API commands, 0x50 for Fast Response A, 0x51 for Fast Response B, 0x53 for Fast Response C, and 0x57 for Fast Response D. The fast response registers can be configured by the "FRR_CTL_X_MODE" properties.

The fast response registers may be read in a burst fashion. After the initial 16 clock cycles, each additional eight clock cycles will clock out the contents of the next fast response register in a circular fashion. The value of the FRRs will not be updated unless NSEL is toggled.

3.3. Operating Modes and Timing

The primary states of the TRW-HP9M2W are shown in Figure 6. The shutdown state completely shuts down the radio to minimize current consumption. Standby/Sleep, SPI Active, Ready, TX Tune, and RX tune are available to optimize the current consumption and response time to RX/TX for a given application. API commands START_RX, START_TX, and CHANGE_STATE control the operating state with the exception of shutdown which is controlled by PE/D, pin Table 9 shows each of the operating modes with the time required to reach either RX or TX mode as well as the current consumption of each mode. The times in Table 9 are measured from the rising edge of nSEL until the chip is in the desired state. Note that these times are indicative of state transition timing but are not guaranteed and should only be used as a reference data point. An automatic sequencer will put the chip into RX or TX from any state. It is not necessary to manually step through the states. To simplify the diagram it is not shown but any of the lower power states can be returned to automatically after RX or TX.

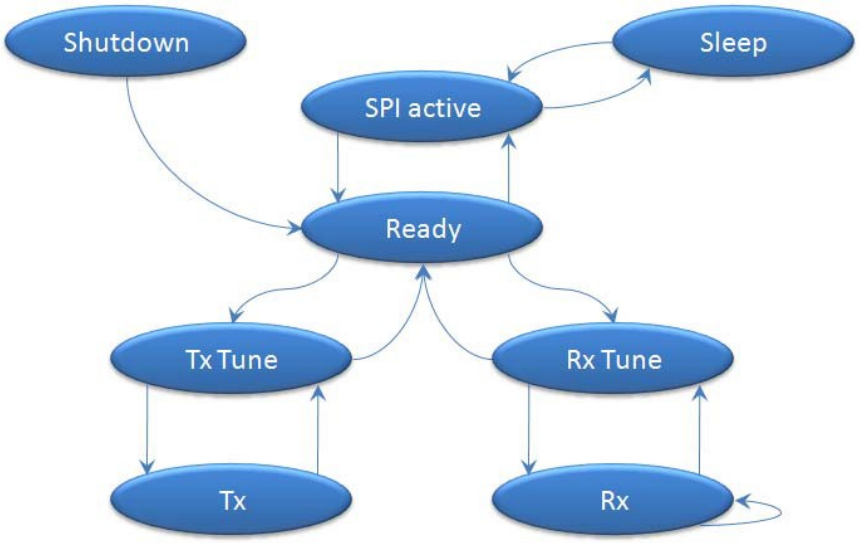


Figure 6. State Machine Diagram

Table 9. Operating State Response Time and Current Consumption*

State/Mode	Response Time to		Current in State /Mode
	TX	RX	
Shutdown State	15 ms	15 ms	30 nA
Standby State Sleep State SPI Active State Ready State TX Tune State RX Tune State	440 µs 440 µs 340 µs 126 µs 58 µs —	440 µs 440 µs 340 µs 122 µs — 74 µs	50 nA 900 nA 1.35 mA 1.8 mA 8 mA 7.2 mA
TX State	—	138 µs	18 mA @ +10 dBm
RX State	130 µs	75 µs	10 or 13 mA
***Note:** TX□RX and RX□TX state transition timing can be reduced to 70 µs if using Zero-IF mode.			

Figure 7 shows the POR timing and voltage requirements. The power consumption (battery life) depends on the duty cycle of the application or how often the part is in either Rx or Tx state. In most applications the utilization of the standby state will be most advantageous for battery life but for very low duty cycle applications shutdown will have an advantage. For the fastest timing the next state can be selected in the START_RX or START_TX API commands to minimize SPI transactions and internal MCU processing.

3.3.1. Power on Reset (POR)

A Power On Reset (POR) sequence is used to boot the device up from a fully off or shutdown state. To execute this process, VDD must ramp within 1ms and must remain applied to the device for at least 100ms. If VDD is removed, then it must stay below 0.15V for at least 10ms before being applied again. Please see Figure x and Table x for details.

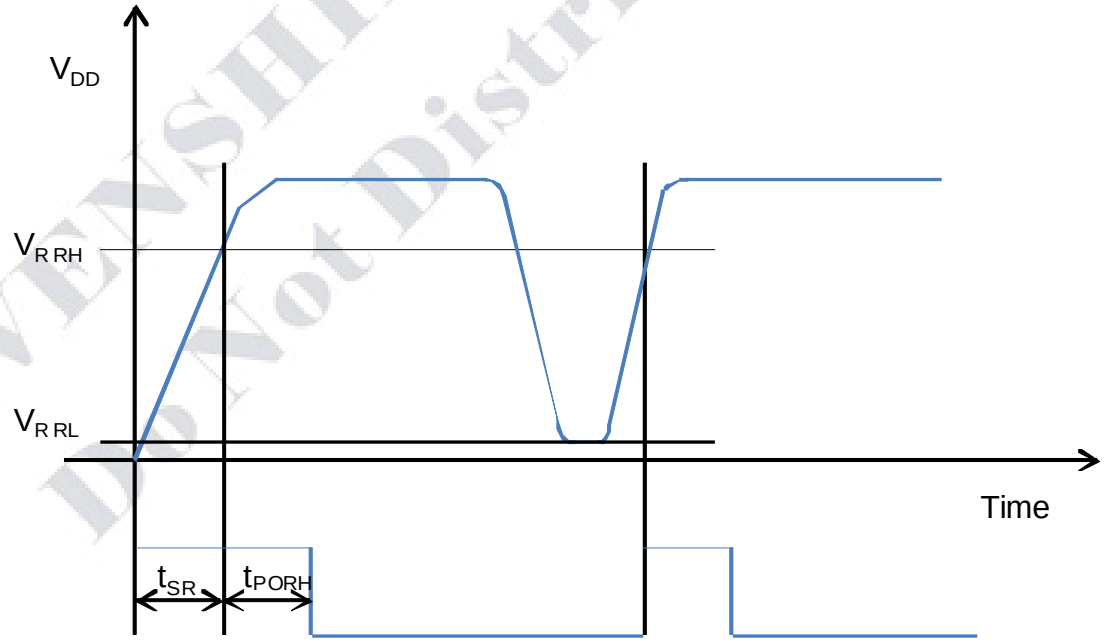


Figure 7. POR Timing Diagram

Table 10. POR Timing

Variable	Description	Min	Typ	Max	Units
t_{PORH}	High time for VDD to fully settle POR circuit	10			ms
t_{PORL}	Low time for VDD to enable POR	10			ms
V_{RRH}	Voltage for successful POR	90%*Vdd			V
V_{RRL}	Starting Voltage for successful POR	0		150	mV
t_{SR}	Slew rate of VDD for successful POR			1	ms

3.3.2. Shutdown State

The shutdown state is the lowest current consumption state of the device with nominally less than 100 nA of current consumption. The shutdown state may be entered by driving the PE/D, pin low The PE/D pin should be held high in all states except the shutdown state. In the shutdown state, the contents of the registers are lost and there is no SPI access. When coming out of the shutdown state a power on reset (POR) will be initiated along with the internal calibrations. After the POR the POWER_UP command is required to initialize the radio. The PE/D pin needs to be held low for at least 50ms before driving high again so that internal capacitors can discharge. Not holding the PE/D low for this period of time may cause the POR to be missed and the device to boot up incorrectly. If POR timing and voltage requirements cannot be met, it is highly recommended that PE/D be controlled using the host processor rather than tying it to VDD on the board.

3.3.3. Standby State

Standby state has the lowest current consumption with the exception of shutdown but has much faster response time to RX or TX mode. In most cases standby should be used as the low power state. In this state the register values are maintained with all other blocks disabled. The SPI is accessible during this mode but any SPI event, including FIFO R/W, will enable an internal boot oscillator and automatically move the part to SPI active state. After an SPI event the host will need to re-command the device back to standby through the “Change State” API command to achieve the 50 nA current consumption. If an interrupt has occurred (i.e., the nIRQ pin = 0) the interrupt registers must be read to achieve the minimum current consumption of this mode.

3.3.5. SPI Active State

In SPI active state the SPI and a boot up oscillator are enabled. After SPI transactions during either standby or sleep the device will not automatically return to these states. A “Change State” API command will be required to return to either the standby or sleep modes.

3.3.6. Ready State

Ready state is designed to give a fast transition time to TX or RX state with reasonable current consumption. In this mode the Crystal oscillator remains enabled reducing the time required to switch to TX or RX mode by eliminating the crystal start-up time.

3.3.7. TX State

The TX state may be entered from any of the state with the “Start TX” or “Change State” API commands. A built-in sequencer takes care of all the actions required to transition between states from enabling the crystal oscillator to ramping up the PA. The following sequence of events will occur automatically when going from standby to TX state.

1. Enable internal LDOs.
2. Start up crystal oscillator and wait until ready (controlled by an internal timer).
3. Enable PLL.
4. Calibrate VCO/PLL.
5. Wait until PLL settles to required transmit frequency (controlled by an internal timer).
6. Activate power amplifier and wait until power ramping is completed (controlled by an internal timer).
7. Transmit packet.

8. RF AMP began to control given voltage

Steps in this sequence may be eliminated depending on which state the chip is configured to prior to commanding to TX. By default, the VCO and PLL are calibrated every time the PLL is enabled. When the START_TX API command is utilized the next state may be defined to ensure optimal timing and turnaround.

Figure 8 shows an example of the commands and timing for the START_TX command. CTS will go high as soon as the sequencer puts the part into TX state. As the sequencer is stepping through the events listed above, CTS will be low and no new commands or property changes are allowed. If the Fast Response (FRR) or nIRQ is used to monitor the current state there will be slight delay caused by the internal hardware from when the event actually occurs to when the transition occurs on the FRR or nIRQ. The time from entering TX state to when the FRR will update is 5 µs and the time to when the nIRQ will transition is 13 µs. If a GPIO is programmed for TX state or used as control for a transmit/receive switch (TR switch) there is no delay.

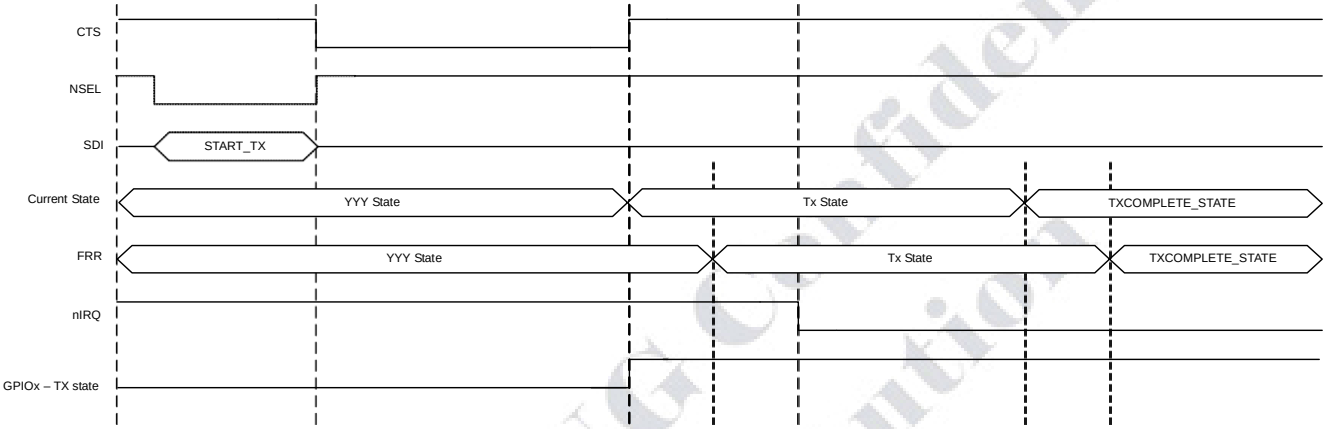


Figure 8. Start_TX Commands and Timing

3.3.8. RX State

The RX state may be entered from any of the other states by using the “Start RX” or “Change State” API command. A built-in sequencer takes care of all the actions required to transition between states. The following sequence of events will occur automatically to get the chip into RX mode when going from standby to RX state:

1. Enable the digital LDO and the analog LDOs.
2. Start up crystal oscillator and wait until ready (controlled by an internal timer).
3. Enable PLL.
4. Calibrate VCO
5. Wait until PLL settles to required receive frequency (controlled by an internal timer).
6. Enable receiver circuits: LNA, mixers, and ADC.
7. Enable receive mode in the digital modem.

Depending on the configuration of the radio, all or some of the following functions will be performed automatically by the digital modem: AGC, AFC (optional), update status registers, bit synchronization, packet handling (optional) including sync word, header check, and CRC. Similar to the TX state, the next state after RX may be defined in the “Start RX” API command. The START_RX commands and timing will be equivalent to the timing shown in Figure 8.

3.4. Application Programming Interface (API)

An application programming interface (API), which the host MCU will communicate with, is embedded inside the device. The API is divided into two sections, commands and properties. The commands are used to control the chip and retrieve its status. The properties are general configurations which will change infrequently. The API descriptions can be found in“AN625: Si446x API Descriptions”.

3.5. Interrupts

The TRW-HP9M2W is capable of generating an interrupt signal when certain events occur. The chip notifies the microcontroller that an interrupt event has occurred by setting the nIRQ output pin LOW = 0. This interrupt signal will be generated when any one (or more) of the interrupt events (corresponding to the Interrupt Status bits) occur. The nIRQ pin will remain low until the microcontroller reads the Interrupt Status Registers. The nIRQ output signal will then be reset until the next change in status is detected.

The interrupts sources are grouped into three groups: packet handler, chip status, and modem. The individual interrupts in these groups can be enabled/disabled in the interrupt property registers, 0101, 0102, and 0103. An interrupt must be enabled for it to trigger an event on the nIRQ pin. The interrupt group must be enabled as well as the individual interrupts in API property 0100.

Number	Command	Summary
0x20	GET_INT_STATUS	Returns the interrupt status—packet handler, modem, and chip
0x21	GET_PH_STATUS	Returns the packet handler status.
0x22	GET_MODEM_STATUS	Returns the modem status byte.
0x23	GET_CHIP_STATUS	Returns the chip status.

Number	Property	Default	Summary
0x0100	INT_CTL_ENABLE	0x04	Enables interrupt groups for PH, Modem, and Chip.
0x0101	INT_CTL_PH_ENABLE	0x00	Packet handler interrupt enable property.
0x0102	INT_CTL_MODEM_ENABLE	0x00	Modem interrupt enable property.
0x0103	INT_CTL_CHIP_ENABLE	0x04	Chip interrupt enable property.

Once an interrupt event occurs and the nIRQ pin is low there are two ways to read and clear the interrupts. All of the interrupts may be read and cleared in the “GET_INT_STATUS” API command. By default all interrupts will be cleared once read. If only specific interrupts want to be read in the fastest possible method the individual interrupt groups (Packet Handler, Chip Status, Modem) may be read and cleared by the “GET_MODEM_STATUS”, “GET_PH_STATUS” (packet handler), and “GET_CHIP_STATUS” API commands.

The instantaneous status of a specific function maybe read if the specific interrupt is enabled or disabled. The status results are provided after the interrupts and can be read with the same commands as the interrupts. The status bits will give the current state of the function whether the interrupt is enabled or not.

The fast response registers can also give information about the interrupt groups but reading the fast response registers will not clear the interrupt and reset the nIRQ pin.

3.6. GPIO

Four general purpose IO pins are available to utilize in the application. The GPIO are configured by the GPIO_PIN_CFG command in address 13h. For a complete list of the GPIO options please see the API guide. GPIO pins 0 and 1 should be used for active signals such as data or clock. GPIO pins 2 and 3 have more susceptibility to generating spurious in the synthesizer than pins 0 and 1. The drive strength of the GPIOs can be adjusted with the GEN_CONFIG parameter in the GPIO_PIN_CFG command. By default the drive strength is set to minimum. The default configuration for the GPIOs and the state during SDN is shown below in Table 11.The state of the IO during shutdown is also shown inTable 11. As indicated previously in Table 6, GPIO 0 has lower drive strength than the other GPIOs.

本模組 GPIO3 是控制 RF FRONT-END MODULE MODE, 當為接收模式時需宣告為 Low, 當發設模式時需宣告為 high, GPIO0 是控制 LNA 電源則是當接收模式時需宣告為 High, 當發射模式時為 Low

Table 11.GPIOs

Pin	SDN State	POR Default
GPIO0	0	POR
GPIO1	0	CTS
GPIO2	0	POR
GPIO3	0	POR
nIRQ	resistive VDD pull-up	nIRQ
SDO	resistive VDD pull-up	SDO
SDI	High Z	SDI

4. Modulation and Hardware Configuration Options

The TRW-HP9M2W supports different modulation options and can be used in various configurations to tailor the device to any specific application or legacy system for drop in replacement. The modulation and configuration options are set in API property, MODEM_MOD_TYPE.

4.1. MODEM_MOD_TYPE

□□Summary: Modulation Type

□□Purpose:

□□This property selects between OOK, FSK, 4FSK and GFSK modulation, modulation source, and tx direct mode control.

□□The modulator must be configured for one mode through the entire packet. If portions of the packet alternate between FSK and 4FSK modes, the modem should be programmed to 4FSK mode.

□□Property: 0x2000

□□Default: 0x02

□□Fields:

□□TX_DIRECT_MODE_TYPE - default:0

0 = Direct mode operates in synchronous mode, applies to TX only.

1 = Direct mode operates in asynchronous mode, applies to TX only. GFSK is not supported.

□□TX_DIRECT_MODE_GPIO[1:0] - default:0x0

0 = TX direct mode uses gpio0 as data source, applies to TX only.

1 = TX direct mode uses gpio1 as data source, applies to TX only.

2 = TX direct mode uses gpio2 as data source, applies to TX only.

3 = TX direct mode uses gpio3 as data source, applies to TX only.

□□MOD_SOURCE[1:0] - default:0x0

0 = Modulation source is packet handler fifo

1 = Modulation source is direct mode pin

2 = Modulation source is pseudo-random generator

□□MOD_TYPE[2:0] - default:0x2

0 = CW

1 = OOK

2 = 2FSK

3 = 2GFSK

4 = 4FSK

5 = 4GFSK

□□Register View

MODEM_MOD_TYPE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX_DIRECT_MODE_TYPE	TX_DIRECT_MODE_GPIO[1:0]		MOD_SOURCE[1:0]		MOD_TYPE[2:0]		
0	0x0		0x0		0x2		

4.2. Modulation Types

The TRW-HP9M2W supports five different modulation options: Gaussian frequency shift keying (GFSK), frequency-shift keying (FSK), four-level GFSK (4GFSK), four-level FSK (4FSK), and on-off keying (OOK). Minimum shift keying (MSK) can also be created by using GFSK settings. GFSK is the recommended modulation type as it provides the best performance and cleanest modulation spectrum. The modulation type is set by the "MOD_TYPE[2:0]" registers in the "MODEM_MOD_TYPE" API property. A continuous-wave (CW) carrier may also be selected for RF evaluation purposes. The modulation source may also be selected to be a pseudo-random source for evaluation purposes.

4.3. Hardware Configuration Options

There are different receive demodulator options to optimize the performance and mutually-exclusive options for how the RX/TX data is transferred from the host MCU to the RF device.

4.3.1. Receive Demodulator Options

There are multiple demodulators integrated into the device to optimize the performance for different applications, modulation formats, and packet structures. The calculator built into WDS will choose the optimal demodulator based on the input criteria.

4.3.1.1. Synchronous Demodulator

The synchronous demodulator's internal frequency error estimator acquires the frequency error based on a 101010 preamble structure. The bit clock recovery circuit locks to the incoming data stream within four transactions of a "10" or "01" bit stream. The synchronous demodulator gives optimal performance for 2- or 4-level FSK or GFSK modulation that has a modulation index less than 2.

4.3.1.2. Asynchronous Demodulator

The asynchronous demodulator should be used OOK modulation and for FSK/GFSK/4GFSK under one or more of the following conditions:

□□Modulation index $\geq$ 2

□□Non-standard preamble (not 1010101... pattern)

When the modulation index exceeds 2, the asynchronous demodulator has better sensitivity compared to the synchronous demodulator. An internal deglitch circuit provides a glitch-free data output and a data clock signal to simplify the interface to the host. There is no requirement to perform deglitching in the host MCU. The asynchronous demodulator will typically be utilized for legacy systems and will have many performance benefits

over devices used in legacy designs. Unlike the TRW-HP9M2W solution for non-standard packet structures, there is no requirement to perform deglitching on the data in the host MCU. Glitch-free data is output from TRW-HP9M2W devices, and a sample clock for the asynchronous data can also be supplied to the host MCU; so, oversampling or bit clock recovery is not required by the host MCU. There are multiple detector options in the asynchronous demodulator block, which will be selected based upon the options entered into the WDS calculator. The asynchronous demodulator's internal frequency error estimator is able to acquire the frequency error based on any preamble structure.

4.3.2. RX/TX Data Interface With MCU

There are two different options for transferring the data from the RF device to the host MCU. FIFO mode uses the SPI interface to transfer the data, while direct mode transfers the data in real time over GPIO.

4.3.2.1. FIFO Mode

In FIFO mode, the transmit and receive data is stored in integrated FIFO register memory. The TX FIFO is accessed by writing Command 66h followed directly by the data/clk that the host wants to write into the TX FIFO. The RX FIFO is accessed by writing command 77h followed by the number of clock cycles of data the host would like to read out of the RX FIFO. The RX data will be clocked out onto the SDO pin.

In TX mode, if the packet handler is enabled, the data bytes stored in FIFO memory are “packaged” together with other fields and bytes of information to construct the final transmit packet structure. These other potential fields include the Preamble, Sync word, Header, CRC checksum, etc. The configuration of the packet structure in TX mode is determined by the Automatic Packet Handler (if enabled), in conjunction with a variety of Packet Handler properties. If the Automatic Packet Handler is disabled, the entire desired packet structure should be loaded into FIFO memory; no other fields (such as Preamble or Sync word) will be automatically added to the bytes stored in FIFO memory. For further information on the configuration of the FIFOs for a specific application or packet size, see "6. Data Handling and Packet Handler" on page 39. In RX mode, only the bytes of the received packet structure that are considered to be “data bytes” are stored in FIFO memory. Which bytes of the received packet are considered “data bytes” is determined by the Automatic Packet Handler (if enabled) in conjunction with the Packet Handler configuration. If the Automatic Packet Handler is disabled, all bytes following the Sync word are considered data bytes and are stored in FIFO memory. Thus, even if Automatic Packet Handling operation is not desired, the preamble detection threshold and Sync word still need to be programmed so that the RX Modem knows when to start filling data into the FIFO. When the FIFO is being used in RX mode, all of the received data may still be observed directly (in realtime) by properly programming a GPIO pin as the RXDATA output pin; this can be quite useful during application development. When in FIFO mode, the chip will automatically exit the TX or RX State when either the PACKET_SENT or PACKET_RX interrupt occurs. The chip will return to the IDLE state programmed in the argument of the “START TX” or “START RX” API command, TXCOMPLETE_STATE[3:0] or RXVALID_STATE[3:0]. For example, the chip may be placed into TX mode by sending the “START TX” command and by writing the 30h to the TXCOMPLETE_STATE[3:0] argument. The chip will transmit all of the contents of the FIFO, and the ipksent interrupt will occur. When this event occurs, the chip will return to the ready state as defined by TXCOMPLETE_STATE[3:0] = 30h.

4.3.2.2. Direct Mode

For legacy systems that perform packet handling within the host MCU or other baseband chip, it may not be desirable to use the FIFO. For this scenario, a Direct mode is provided, which bypasses the FIFOs entirely. In TX Direct mode, the TX modulation data is applied to an input pin of the chip and processed in “real time” (i.e., not stored in a register for transmission at a later time). Any of the GPIOs may be configured for use as the TX Data input function. Furthermore, an additional pin may be required for a TX Clock output function if GFSK modulation is desired (only the TX Data input pin is required for FSK). To achieve direct mode, the GPIO must be configured in the “GPIO_PIN_CFG” API command as well as the “MODEM_MOD_TYPE” API property. For GFSK, “TX_DIRECT_MODE_TYPE” must be set to Synchronous. For 2FSK or OOK, the type can be set to asynchronous or synchronous. The MOD_SOURCE[1:0] should be set to 01h for are all direct mode configurations. In RX Direct mode, the RX Data and RX Clock can be programmed for direct (real-time) output to GPIO pins. The microcontroller may then process the RX data without using the FIFO or packet handler functions of the RFIC.

4.4. Preamble Length

The preamble length requirement is only relevant if using the synchronous demodulator. If the asynchronous demodulator is being used, then there is no requirement for a conventional 101010 pattern.

The preamble detection threshold determines the number of valid preamble bits the radio must receive to qualify a

valid preamble. The preamble threshold should be adjusted depending on the nature of the application. The required preamble length threshold depends on when receive mode is entered in relation to the start of the transmitted packet and the length of the transmit preamble. With a shorter than recommended preamble detection threshold, the probability of false detection is directly related to how long the receiver operates on noise before the transmit preamble is received. False detection on noise may cause the actual packet to be missed. The preamble detection threshold may be adjusted in the modem calculator by modifying the “PM detection threshold” in the “RX parameters tab” in the radio control panel. For most applications with a preamble length longer than 32 bits, the default value of 20 is recommended for the preamble detection threshold. A shorter Preamble Detection Threshold may be chosen if occasional false detections may be tolerated. When antenna diversity is enabled, a 20- bit preamble detection threshold is recommended. When the receiver is synchronously enabled just before the start of the packet, a shorter preamble detection threshold may be used. Table 12 demonstrates the recommended preamble detection threshold and preamble length for various modes.

Table 12. Recommended Preamble Length

Mode	AFC	Antenna Diversity	Preamble Type	Recommended Preamble Length	Recommended Preamble Detection Threshold
(G)FSK	Disabled	Disabled	Standard	4 Bytes	20 bits
(G)FSK	Enabled	Disabled	Standard	5 Bytes	20 bits
(G)FSK	Disabled	Disabled	Non-standard	2 Bytes	0 bits
(G)FSK	Enabled		Non-standard	Not Supported	
(G)FSK	Disabled	Enabled	Standard	7 Bytes	24 bits
(G)FSK	Enabled	Enabled	Standard	8 Bytes	24 bits
4(G)FSK	Disabled	Disabled	Standard	40 symbols	16 symbols
4(G)FSK	Enabled	Disabled	Standard	48 symbols	16 symbols
4(G)FSK			Non-standard	Not Supported	
OOK	Disabled	Disabled	Standard	4 Bytes	20 bits
OOK	Disabled	Disabled	Non-standard	2 Bytes	0 bits
OOK	Enabled			Not Supported	

Notes:

1. The recommended preamble length and preamble detection thresholds listed above are to achieve 0% PER. They may be shortened when occasional packet errors are tolerable.
2. All recommended preamble lengths and detection thresholds include AGC and BCR settling times.
3. “Standard” preamble type should be set for an alternating data sequence at the max data rate (…10101010…)
4. “Non-standard” preamble type can be set for any preamble type including …10101010...
5. When preamble detection threshold = 0, sync word needs to be 3 Bytes to avoid false syncs. When only a 2 Byte sync word is available the sync word detection can be extended by including the last preamble Byte into the RX sync word setting.

5. Internal Functional Blocks

The following sections provide an overview to the key internal blocks and features.

5.1. RX Chain

The internal low-noise amplifier (LNA) is designed to be a wide-band LNA that can be matched with three external discrete components to cover any common range of frequencies in the sub-GHz band. The LNA has extremely low noise to suppress the noise of the following stages and achieve optimal sensitivity; so, no external gain or front-end modules are necessary. The LNA has gain control, which is controlled by the internal automatic gain control (AGC)

algorithm. The LNA is followed by an I-Q mixer, filter, programmable gain amplifier (PGA), and ADC. The I-Q mixers downconvert the signal to an intermediate frequency. The PGA then boosts the gain to be within dynamic range of the ADC. The ADC rejects out-of-band blockers and converts the signal to the digital domain where filtering, demodulation, and processing is performed. Peak detectors are integrated at the output of the LNA and PGA for use in the AGC algorithm.

The RX and TX pins maybe directly tied externally for output powers less than +17 dBm, see the direct-tie reference designs on the Silicon Labs web site for more details.

5.1.1. RX Chain Architecture

It is possible to operate the RX chain in different architecture configurations: fixed-IF, zero-IF, scaled-IF, and modulated IF. There are trade-offs between the architectures in terms of sensitivity, selectivity, and image rejection. Fixed-IF is the default configuration and is recommended for most applications. With 35 dB native image rejection and autonomous image calibration to achieve 55 dB, the fixed-IF solution gives the best performance for most applications. Fixed-IF obtains the best sensitivity, but it has the effect of degraded selectivity at the image frequency. An autonomous image rejection calibration is included in TRW-HP9M2W devices and described in more detail in "5.2.3. Image Rejection and Calibration" on page 31. For fixed-IF and zero-IF, the sensitivity is degraded for data rates less than 100 kbps or bandwidths less than 200 kHz. The reduction in sensitivity is caused by increased flicker noise as dc is approached. The benefit of zero-IF is that there is no image frequency; so, there is no degradation in the selectivity curve, but it has the worst sensitivity. Scaled-IF is a trade-off between fixed-IF and zero-IF. In the scaled-IF architecture, the image frequency is placed or hidden in the adjacent channel where it only slightly degrades the typical adjacent channel selectivity. The scaled-IF approach has better sensitivity than zero-IF but still some degradation in selectivity due to the image. In scaled-IF mode, the image frequency is directly proportional to the channel bandwidth selected. Figure 9 demonstrates the trade-off in sensitivity between the different architecture options.

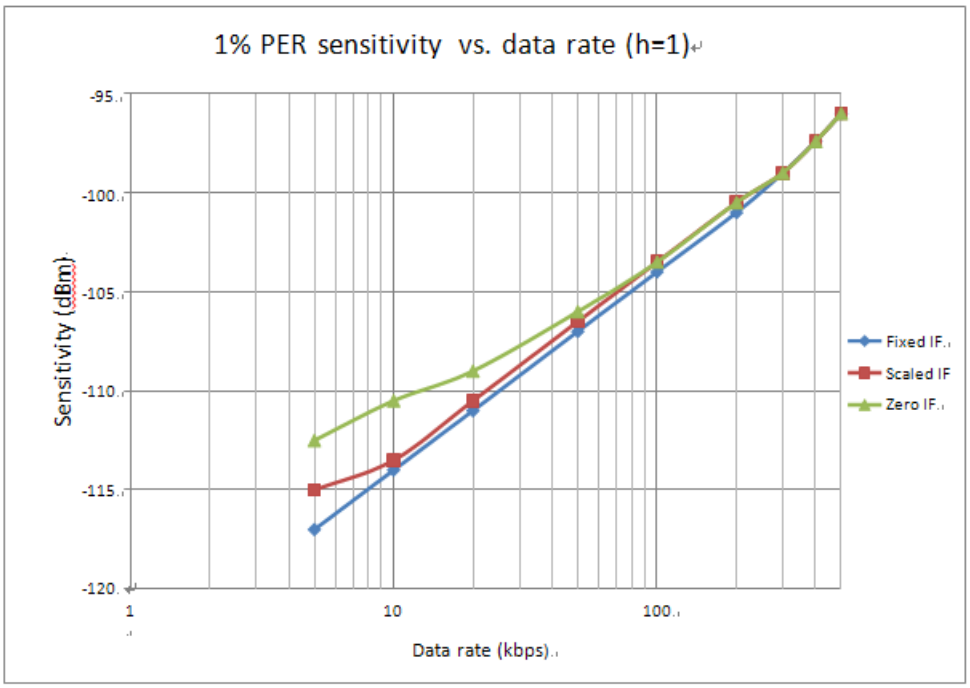


Figure 9. RX Architecture vs. Data Rat

5.2. RX Modem

Using high-performance ADCs allows channel filtering, image rejection, and demodulation to be performed in the digital domain, which allows for flexibility in optimizing the device for particular applications. The digital modem performs the following functions:

- □□Channel selection filter
- □□TX modulation
- □□RX demodulation
- □□Automatic Gain Control (AGC)
- □□Preamble detection
- □□Invalid preamble detection
- □□Radio signal strength indicator (RSSI)
- □□Automatic frequency compensation (AFC)
- □□Image Rejection Calibration
- □□Packet handling including EZMAC® features
- □□Cyclic redundancy check (CRC)

The digital channel filter and demodulator are optimized for ultra-low-power consumption and are highly configurable. Supported modulation types are GFSK, FSK, 4GFSK, 4FSK, GMSK, and OOK. The channel filter can be configured to support bandwidths ranging from 850 down to 1.1 kHz. A large variety of data rates are supported ranging from 100 bps up to 1 Mbps. The configurable preamble detector is used with the synchronous demodulator to improve the reliability of the sync-word detection. Preamble detection can be skipped using only sync detection, which is a valuable feature of the asynchronous demodulator when very short preambles are used in protocols, such as MBus. The received signal strength indicator (RSSI) provides a measure of the signal strength received on the tuned channel. The resolution of the RSSI is 0.5 dB. This high-resolution RSSI enables accurate channel power measurements for clear channel assessment (CCA), carrier sense (CS), and listen before talk (LBT) functionality. A comprehensive programmable packet handler including key features of Silicon Labs' EZMAC is integrated to create a variety of communication topologies ranging from peer-to-peer networks to mesh networks. The extensive programmability of the packet header allows for advanced packet filtering, which, in turn enables a mix of broadcast, group, and point-to-point communication. A wireless communication channel can be corrupted by noise and interference, so it is important to know if the received data is free of errors. A cyclic redundancy check (CRC) is used to detect the presence of erroneous bits in each packet. A CRC is computed and appended at the end of each transmitted packet and verified by the receiver to confirm that no errors have occurred. The packet handler and CRC can significantly reduce the load on the system microcontroller allowing for a simpler and cheaper microcontroller. The digital modem includes the TX modulator, which converts the TX data bits into the corresponding stream of digital modulation values to be summed with the fractional input to the sigma-delta modulator. This modulation approach results in highly accurate resolution of the frequency deviation. A Gaussian filter is implemented to support GFSK and 4GFSK, considerably reducing the energy in adjacent channels. The default bandwidth-time product (BT) is 0.5 for all programmed data rates, but it may be adjusted to other values.

5.2.1. Automatic Gain Control (AGC)

The AGC algorithm is implemented digitally using an advanced control loop optimized for fast response time. The AGC occurs within a single bit or in less than 2 µs. Peak detectors at the output of the LNA and PGA allow for optimal adjustment of the LNA gain and PGA gain to optimize IM3, selectivity, and sensitivity performance.

5.2.2. Auto Frequency Correction (AFC)

Frequency mistuning caused by crystal inaccuracies can be compensated for by enabling the digital automatic frequency control (AFC) in receive mode. There are two types of integrated frequency compensation: modem frequency compensation, and AFC by adjusting the PLL frequency. With AFC disabled, the modem compensation can correct for frequency offsets up to ±0.25 times the IF bandwidth. When the AFC is enabled, the received signal will be centered in the pass-band of the IF filter, providing optimal sensitivity and selectivity over a wider range of frequency offsets up to ±0.35 times the IF bandwidth. When AFC is enabled, the preamble length needs to be long enough to settle the AFC. As shown in Table 12 on page 28, an additional byte of preamble is typically required to settle the AFC.

5.2.3. Image Rejection and Calibration

Since the receiver utilizes a low-IF architecture, the selectivity will be affected by the image frequency. The IF frequency is 468.75 kHz (Fxtal/64), and the image frequency will be at 937.5 kHz below the RF frequency. The native image rejection of the TRW-HP9M2W family is 35 dB. Image rejection calibration is available in the TRW-HP9M2W to improve the image rejection to more than 55 dB. The calibration is initiated with the IRCAL API command. The calibration uses an internal signal source, so no external signal generator is required. The initial calibration takes
250 ms, and periodic re-calibration takes 100 ms. Re-calibration should be initiated when the temperature has changed more than 30 °C.

5.2.4. Received Signal Strength Indicator

The received signal strength indicator (RSSI) is an estimate of the signal strength in the channel to which the receiver is tuned. The RSSI measurement is done after the channel filter, so it is only a measurement of the desired or undesired in-band signal power. There are two different methods for reading the RSSI value and several different options for configuring the RSSI value that is returned. The fastest method for reading the RSSI is to configure one of the four fast response registers (FRR) to return a latched RSSI value. The latched RSSI value is measured once per packet and is latched at a configurable amount of time after RX mode is entered. The fast response registers can be read in 16 SPI clock cycles with no requirement to wait for CTS. The RSSI value may also be read out of the GET_MODEM_STATUS command. In this command, both the current RSSI and the latched RSSI are available. The current RSSI value represents the signal strength at the instant in time the GET_MODEM_STATUS command is processed and may be read multiple times per packet. Reading the RSSI in the GET_MODEM_STATUS command takes longer than reading the RSSI out of the fast response register. After the initial command, it will take 33 µs for CTS to be set and then the four or five bytes of SPI clock cycles to read out the respective current or latched RSSI values.

The RSSI configuration options are set in the MODEM_RSSI_CONTROL API property. The latched RSSI value may be latched and stored based on the following events: preamble detection, sync detection, or a configurable number of bit times measured after the start of RX mode (minimum of 4 bit times). The requirement for four bit times is determined by the processing delay and settling through the modem and digital channel filter. In MODEM_RSSI_CONTROL, the RSSI may be defined to update every bit period or to be averaged and updated every four bit periods. If RSSI averaging over four bits is enabled, the latched RSSI value will be delayed to a minimum of 7 bits after the start of RX mode to allow for the averaging. The latched RSSI values are cleared when entering RX mode so they may be read after the packet is received or after dropping back to standby mode. If the RSSI value has been cleared by the start of RX but not latched yet, a value of 0 will be returned if it is attempted to be read.

The RSSI value read by the API could be translated to dBm by the following linear equation:

RSSI (in dBm) = (RSSI_value /2) – RSSIcal

RSSIcal in the above formula depends on the matching network, modem settings, and external LNA gain (if present). The RSSIcal value can be obtained by a simple calibration with a signal generator connected at the antenna input. Without external LNA, the value of RSSIcal is around 130 ±30.

During packet reception, it may be useful to detect whether a secondary interfering signal (desired or undesired) arrives. To detect this event, a feature for RSSI jump detection is available. If the RSSI level changes by a programmable amount during the reception of a packet, an interrupt or GPIO can be configured to notify the host. The level of RSSI increase or decrease (jump) is programmable through the MODEM_RSSI_JUMP_THRESH API property. If an RSSI jump is detected, the modem may be programmed to automatically reset so that it may lock onto the new stronger signal. The chip may also be configured to automatically reset the receiver upon jump detection in order to acquire the new signal. The configuration and options for RSSI jump detection are programmed in the MODEM_RSSI_CONTROL2 API property. By default, RSSI jump detection is not enabled.

The RSSI values and curves may be offset by the MODEM_RSSI_COMP API property. The default value of 7'h32 corresponds to no RSSI offset. Setting a value less than 7'h32 corresponds to a negative offset, and a value higher than 7'h32 corresponds to a positive offset. The offset value is in 1 dB steps. For example, setting a value of 7'h3A corresponds to a positive offset of 8 dB.

Clear channel assessment (CCA) or RSSI threshold detection is also available. An RSSI threshold may be set in the MODEM_RSSI_THRESH API property. If the RSSI value is above this threshold, an interrupt or GPIO may notify the host. Both the latched version and asynchronous version of this threshold are available on any of the GPIOs. Automatic fast hopping based on RSSI is available. See "5.3.1.2. Automatic RX Hopping and Hop Table".

5.3. Synthesizer

synthesizer has many advantages; it provides flexibility in choosing data

rate, deviation, channel frequency, and channel spacing. The transmit modulation is applied directly to the loop in the digital domain through the fractional divider, which results in very precise accuracy and control over the transmit deviation. The frequency resolution in the 922–928 MHz band is 28.6 Hz with more resolution in the other bands. The nominal reference frequency to the PLL is 30 MHz, but any XTAL frequency from 25 to 32 MHz may be used. The modem configuration calculator in WDS will automatically account for the XTAL frequency being used. The PLL utilizes a differential LC VCO with integrated on-chip inductors. The output of the VCO is followed by a configurable divider, which will divide the signal down to the desired output frequency band.

5.3.1. Synthesizer Frequency Control

The frequency is set by changing the integer and fractional settings to the synthesizer. The WDS calculator will automatically provide these settings, but the synthesizer equation is shown below for convenience. The APIs for setting the frequency are FREQ_CONTROL_INTE, FREQ_CONTROL_FRAC2, FREQ_CONTROL_FRAC1, and FREQ_CONTROL_FRAC0.**5.3.1.1. EZ Frequency Programming**

In applications that utilize multiple frequencies or channels, it may not be desirable to write four API registers each time a frequency change is required. EZ frequency programming is provided so that only a single register write (channel number) is required to change frequency. A base frequency is first set by first programming the integer and fractional components of the synthesizer. This base frequency will correspond to channel 0. Next, a channel step size is programmed into the FREQ_CONTROL_CHANNEL_STEP_SIZE_1 and FREQ_CONTROL_CHANNEL_STEP_SIZE_0 API registers. The resulting frequency will be:

$$\text{RF Frequency} = \text{Base Frequency} + \text{Channel} \times \text{Stepsize}$$

The second argument of the START_RX or START_TX is CHANNEL, which sets the channel number for EZ frequency programming. For example, if the channel step size is set to 1 MHz, the base frequency is set to 900 MHz with the INTE and FRAC API registers, and a CHANNEL number of 5 is programmed during the START_TX command, the resulting frequency will be 905 MHz. If no CHANNEL argument is written as part of the START_RX/TX command, it will default to the previous value. The initial value of CHANNEL is 0; so, if no CHANNEL value is written, it will result in the programmed base frequency.

5.3.1.2. Automatic RX Hopping and Hop Table

The transceiver supports an automatic hopping feature that can be fully configured through the API. This is intended for RX hopping where the device has to hop from channel to channel and look for packets. Once the device is put into the RX state, it automatically starts hopping through the hop table if the feature is enabled.

The hop table can hold up to 64 entries and is maintained in firmware. Each entry is a channel number; so, the hop table can hold up to 64 channels. The number of entries in the table is set by RX HOP TABLE_SIZE API. The specified channels correspond to the EZ frequency programming method for programming the frequency. The receiver starts at the base channel and hops in sequence from the top of the hop table to the bottom. The table will wrap around to the base channel once it reaches the end of the table. An entry of 0xFF in the table indicates that the entry should be skipped. The device will hop to the next non 0xFF entry.

There are three conditions that can be used to determine whether to continue hopping or to stay on a particular channel. These conditions are:

- □□RSSI threshold
- □□Preamble timeout (invalid preamble pattern)
- □□Sync word timeout (invalid or no sync word detected after preamble)

These conditions can be used individually, or they can be enabled all together by configuring the RX_HOP_CONTROL API. However, the firmware will make a decision on whether or not to hop based on the first condition that is met.

The RSSI that is monitored is the current RSSI value. This is compared to the threshold, and, if it is above the threshold value, it will stay on the channel. If the RSSI is below the threshold, it will continue hopping. There is no averaging of RSSI done during the automatic hopping from channel to channel. Since the preamble timeout and the sync word timeout are features that require packet handling, the RSSI threshold is the only condition that can be used if the user is in “direct” or “RAW” mode where packet handling features are not used.

Note that the RSSI threshold is not an absolute RSSI value; instead, it is a relative value and should be verified on the bench to find an optimal threshold for the application.

The turnaround time from RX to RX on a different channel using this method is 115 µs. The time spent in receive mode will be determined by the configuration of the hop conditions. Manual RX hopping will have the fastest turn-around time but will require more overhead and management by the host MCU.

The following are example steps for using Auto Hop:

1. Set the base frequency (inte + frac) and channel step size.
2. Define the number of entries in the hop table (RX_HOP_TABLE_SIZE).
3. Write the channels to the hop table (RX_HOP_TABLE_ENTRY_n)
4. Configure the hop condition and enable auto hopping- RSSI, preamble, or sync (RX_HOP_CONTROL).
5. Set preamble and sync parameters if enabled.
6. Program the RSSI threshold property in the modem using “MODEM_RSSI_THRESH”.
7. Set the preamble threshold using “PREAMBLE_CONFIG_STD_1”.
8. Program the preamble timeout property using “PREAMBLE_CONFIG_STD_2”.
9. Set the sync detection parameters if enabled.
10. If needed, use “GPIO_PIN_CFG” to configure a GPIO to toggle on hop and hop table wrap.
11. Use the “START_RX” API with channel number set to the first valid entry in the hop table (i.e., the first non 0xFF entry).
12. Device should now be in auto hop mode.

5.3.1.3. Manual RX Hopping

The RX_HOP command provides the fastest method for hopping from RX to RX but it requires more overhead and management by the host MCU. Using the RX_HOP command, the turn-around time is 75 µs. The timing is faster with this method than Start_RX or RX hopping because one of the calculations required for the synthesizer calibrations is offloaded to the host and must be calculated/stored by the host, VCO_CNT0. For information about using fast manual hopping, contact customer support.

5.4. Transmitter (TX)

The TRW-HP9M2W contains an integrated +20 dBm transmitter or power amplifier that is capable of transmitting from
–20 to +20 dBm. The output power steps are less than 0.25 dB within 6 dB of max power but become larger and more non-linear close to minimum output power. The Si4464/63 PA is designed to provide the highest efficiency and lowest current consumption possible. The Si4461 PA is capable of transmitting from –40 to +16 dBm. The Si4461 PA can be optimized for either optimum current consumption (Class E) or for fine output power steps and performance over voltage and temperature (switched-current). Switched-current matching will have fine output power steps and more constant output power over VDD, but it will have higher current consumption than the class-E matching. The class E will have the most efficient current consumption, but it will have more coarse output power steps and variation across VDD. The Si4460 is designed to supply +10 dBm output power for less than 20 mA for applications that require operation from a single coin cell battery. The Si4460 can also operate with either class-E or switched current matching and output up to +13 dBm Tx power. All PA options are single-ended to allow for easy antenna matching and low BOM cost. Automatic ramp-up and ramp-down is automatically performed to reduce unwanted spectral spreading.

Chip’s TXRAMP pin is disabled by default to save current in cases where on-chip PA will be able to drive the antenna. In cases where on-chip PA will drive the external PA, and the external PA needs a ramping signal,

TXRAMP is the signal to use. To enable TXRAMP, set the API Property PA_MODE[7] = 1. TXRAMP will start to ramp up, and ramp down at the SAME time as the internal on-chip PA ramps up/down.

The ramping speed is programmed by TC[3:0] in the PA_RAMP_EX API property, which has the following characteristics:

TC	Ramp Time (µs)
0.0	2.0
1.0	2.1
2.0	2.2
3.0	2.4
4.0	2.6
5.0	2.8
6.0	3.1
7.0	3.4
8.0	3.7
9.0	4.1
10.0	4.5
11.0	5.0
12.0	6.0
13.0	8.0
14.0	10.0
15.0	20.0

The ramping profile is close to a linear ramping profile with smoothed out corner when approaching Vhi and Vlo. The TXRAMP pin can source up to 1 mA without voltage drooping. The TXRAMP pin's sinking capability is equivalent to a 10 k□pull-down resistor.

Vhi = 3 V when Vdd > 3.3 V. When Vdd < 3.3 V, the Vhi will be closely following the Vdd, and ramping time will be smaller also.

Vlo = 0 V when NO current needed to be sunk into TXRAMP pin. If 10uA need to be sunk into the chip, Vlo will be 10 µA x 10k = 100 mV.

Number	Command	Summary
0x2200	PA_MODE	Sets PA type.
0x2201	PA_PWR_LVL	Adjust TX power in fine steps.
0x2202	PA_BIAS_CLKDUTY	Adjust TX power in coarse steps and optimizes for different match configurations.
0x2203	PA_TC	Changes the ramp up/down time of the PA.

5.4.1. TRW-HP9M2W:+20dBm PA

The +20 dBm configuration utilizes a class-E matching configuration. Typical performance for the 900 MHz band for output power steps, voltage, and temperature are shown in Figures 10–12. The output power is changed in 128 steps through PA_PWR_LVL API. For detailed matching values, BOM, and performance at other frequencies, refer to the PA Matching application note.

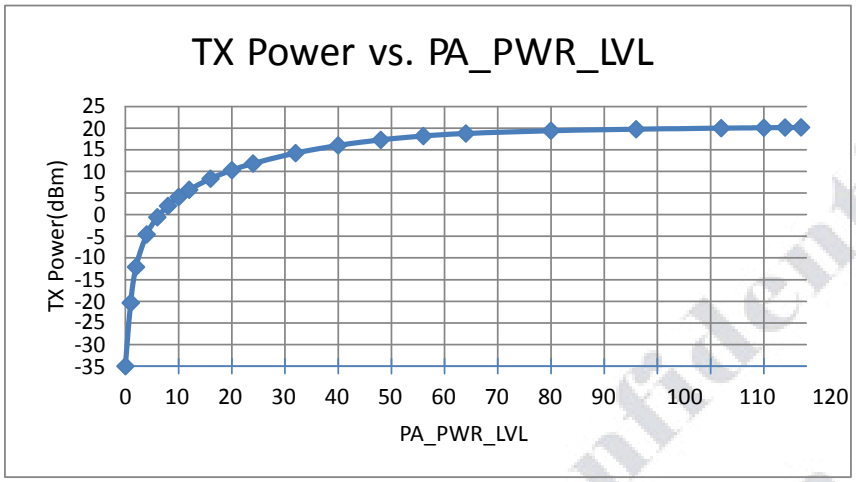


Figure 10. +20 dBm TX Power vs. PA_PWR_LVL

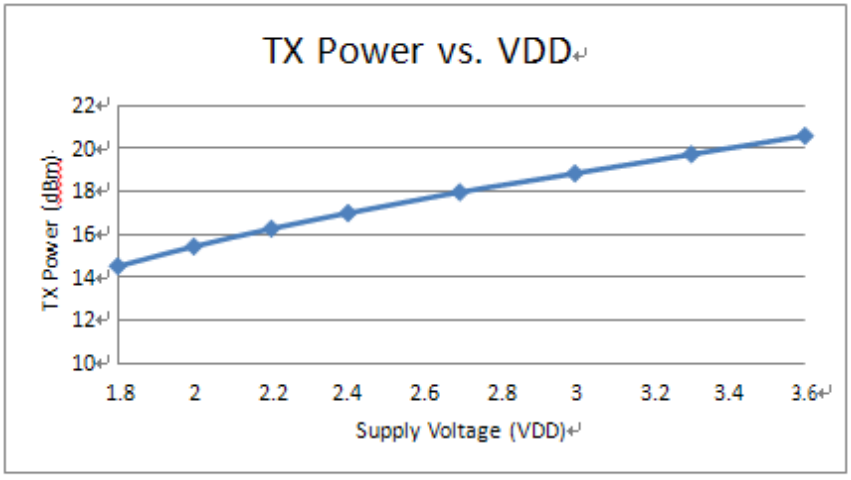


Figure 11. +20 dBm TX Power vs. VDD

Figure 11. +20 dBm TX Power vs. VDD

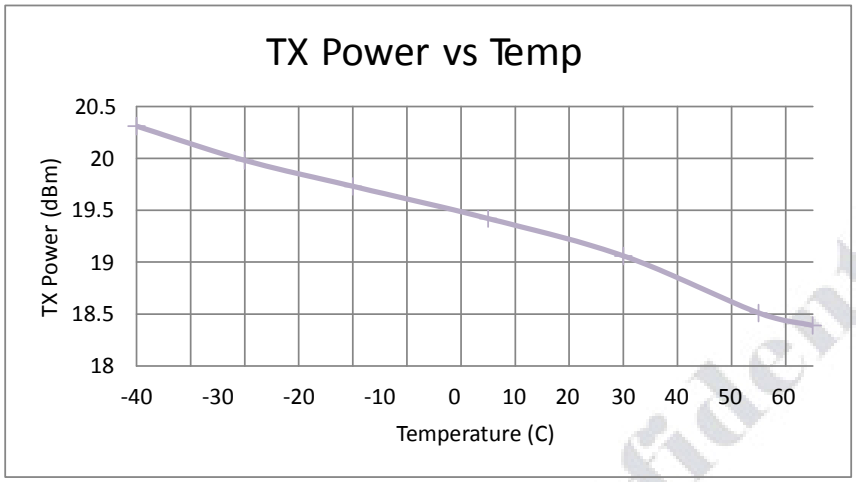


Figure 12. +20 dBm TX Power vs. Temp

5.4.2. Si4461 +16 dBm PA

The Si4461 PA can utilize different matches to optimize the performance for 16, 14, 13 dBm, or a lower power. A class-E match is recommended for 16 dBm to maximize the efficiency and battery life. For 13 and 14 dBm, a switched current match is recommended to provide optimal performance over VDD and temperature variation. Typical performance for the 900 MHz band for output power steps, voltage, and temperature are shown in Figures 13 and 14. The output power is changed in 128 steps through the PA_PWR_LVL API. For detailed matching values, BOM, and performance at other frequencies, refer to "AN627: Si4460/61 Low-Power PA Matching.

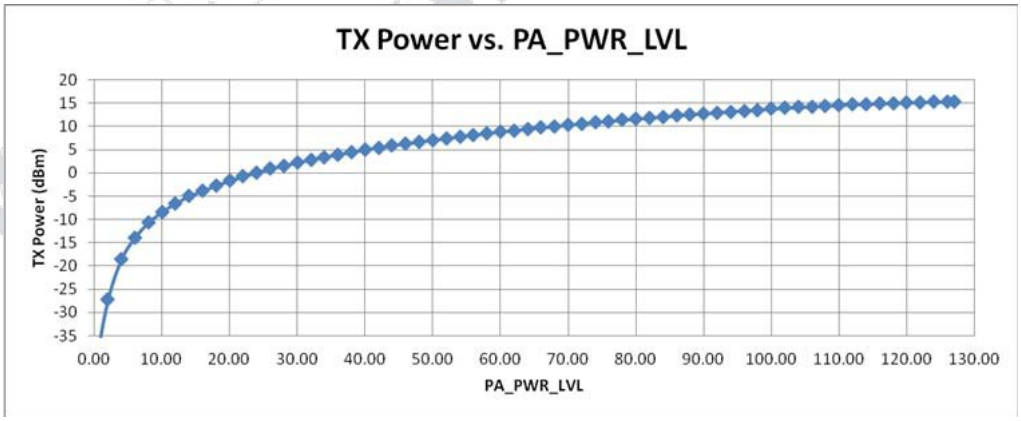


Figure 13. +13 dBm TX Power vs. PA_PWR_LVL

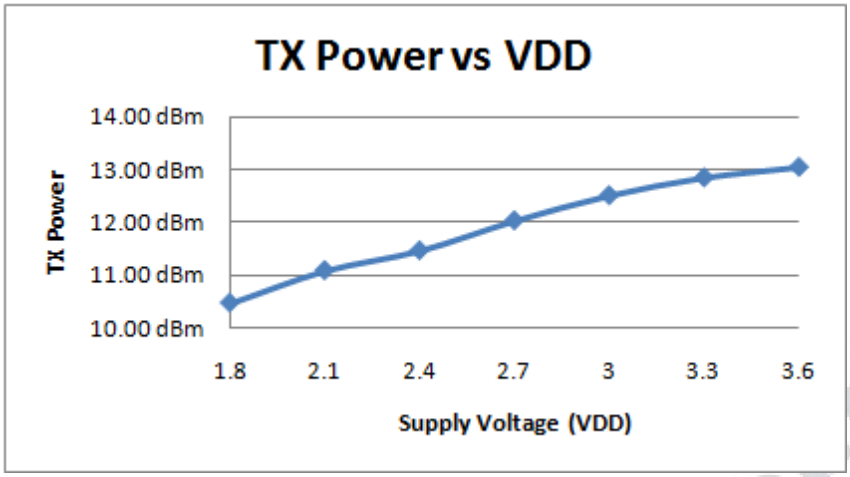


Figure 14. +13 dBm TX Power vs. Supply Voltage (VDD)

5.5. Crystal Oscillator

The TRW-HP9M2W includes an integrated crystal oscillator with a fast start-up time of less than 250 µs. The design is differential with the required crystal load capacitance integrated on-chip to minimize the number of external components. By default, all that is required off-chip is the crystal. The default crystal is 30 MHz, but the circuit is designed to handle any XTAL from 25 to 32 MHz. If a crystal different than 30 MHz is used, the POWER_UP API boot command must be modified. The WDS calculator crystal frequency field must also be changed to reflect the frequency being used. The crystal load capacitance can be digitally programmed to accommodate crystals with various load capacitance requirements and to adjust the frequency of the crystal oscillator. The tuning of the crystal load capacitance is programmed through the GLOBAL_XO_TUNE API property. The total internal capacitance is
11 pF and is adjustable in 127 steps (70 fF/step). The crystal frequency adjustment can be used to compensate for crystal production tolerances. The frequency offset characteristics of the capacitor bank are demonstrated in Figure 15.

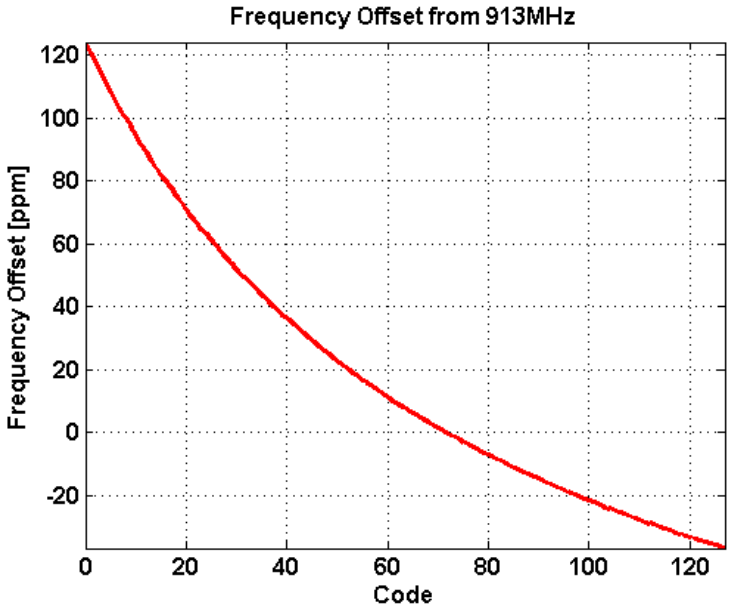


Figure 15. Capacitor Bank Frequency Offset Characteristics

Utilizing the on-chip temperature sensor and suitable control software, the temperature dependency of the crystal can be canceled.

A TCXO or external signal source can easily be used in place of a conventional XTAL and should be connected to the XIN pin. The incoming clock signal is recommended to have a peak-to-peak swing in the range of 600 mV to 1.4 V and ac-coupled to the XIN pin. If the peak-to-peak swing of the TCXO exceeds 1.4 V peak-to-peak, then dc coupling to the XIN pin should be used. The maximum allowed swing on XIN is 1.8 V peak-to-peak.

The XO capacitor bank should be set to 0 whenever an external drive is used on the XIN pin. In addition, the POWER_UP command should be invoked with the TCXO option whenever external drive is used.

6. Data Handling and Packet Handler

6.1. RX and TX FIFOs

Two 64-byte FIFOs are integrated into the chip, one for RX and one for TX, as shown in Figure 16. Writing to command Register 66h loads data into the TX FIFO, and reading from command Register 77h reads data from the RX FIFO. The TX FIFO has a threshold for when the FIFO is almost empty, which is set by the “TX_FIFO_EMPTY” property. An interrupt event occurs when the data in the TX FIFO reaches the almost empty threshold. If more data is not loaded into the FIFO, the chip automatically exits the TX state after the PACKET_SENT interrupt occurs. The RX FIFO has one programmable threshold, which is programmed by setting the “RX_FIFO_FULL” property. When the incoming RX data crosses the Almost Full Threshold, an interrupt will be generated to the microcontroller via the nIRQ pin. The microcontroller will then need to read the data from the RX FIFO. The RX Almost Full Threshold indication implies that the host can read at least the threshold number of bytes from the RX FIFO at that time. Both the TX and RX FIFOs may be cleared or reset with the “FIFO_RESET” command.

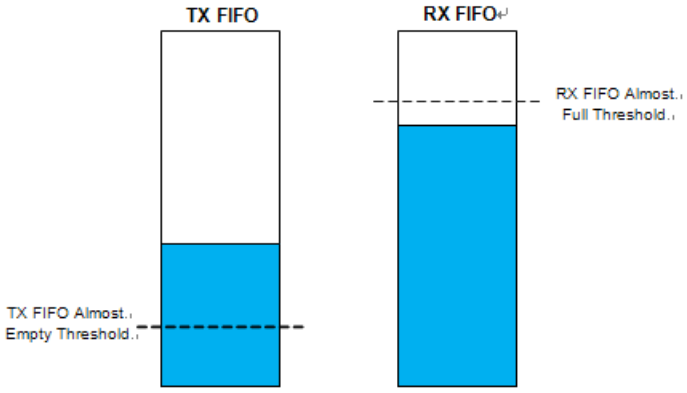


Figure 16. TX and RX FIFOs

6.2. Packet Handler

When using the FIFOs, automatic packet handling may be enabled for TX mode, RX mode, or both. The usual fields for network communication, such as preamble, synchronization word, headers, packet length, and CRC, can be configured to be automatically added to the data payload. The fields needed for packet generation normally change infrequently and can therefore be stored in registers. Automatically adding these fields to the data payload in TX mode and automatically checking them in RX mode greatly reduces the amount of communication between the microcontroller and TRW-HP9M2W. It also greatly reduces the required computational power of the microcontroller. The general packet structure is shown in Figure 17. Any or all of the fields can be enabled and checked by the internal packet handler.

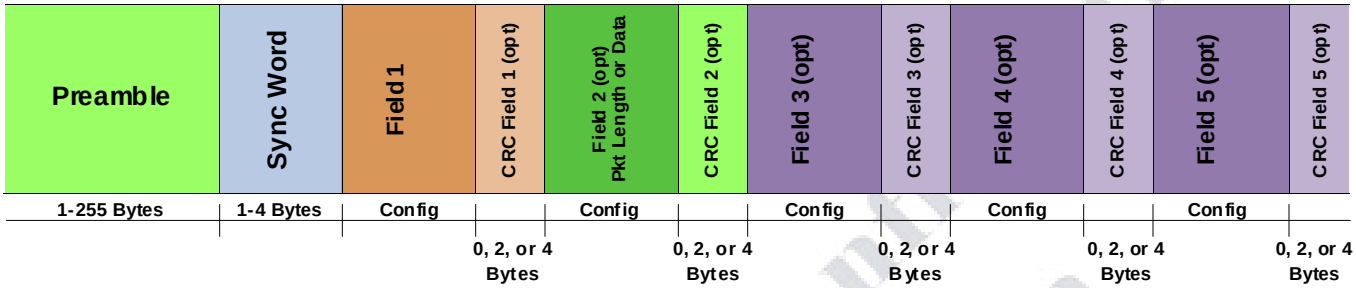


Figure 17. Packet Handler Structure

The fields are highly programmable and can be used to check any kind of pattern in a packet structure. The general functions of the packet handler include the following:

- □□Detection/validation of Preamble quality in RX mode (PREAMBLE_VALID signal)
- □□Detection of Sync word in RX mode (SYNC_OK signal)
- □□Detection of valid packets in RX mode (PKT_VALID signal)
- □□Detection of CRC errors in RX mode (CRC_ERR signal)
- □□Data de-whitening and/or Manchester decoding (if enabled) in RX mode
- □□Match/Header checking in RX mode
- □□Storage of Data Field bytes into FIFO memory in RX mode
- □□Construction of Preamble field in TX mode
- □□Construction of Sync field in TX mode
- □□Construction of Data Field from FIFO memory in TX mode
- □□Construction of CRC field (if enabled) in TX mode
- □□Data whitening and/or Manchester encoding (if enabled) in TX mode

For details on how to configure the packet handler, see “AN626: Packet Handler Operation for TRW-HP9M2W RFICs”.

7. RX Modem Configuration

The TRW-HP9M2W can easily be configured for different data rate, deviation, frequency, etc. by using the WDS settings calculator, which generates an initialization file for use by the host MCU.

8. Auxiliary Blocks

8.1. Wake-up Timer and 32 kHz Clock Source

The chip contains an integrated wake-up timer that can be used to periodically wake the chip from sleep mode. The wake-up timer runs from either the internal 32 kHz RC Oscillator, or from an external 32 kHz XTAL.

The wake-up timer can be configured to run when in sleep mode. If WUT_EN = 1 in the GLOBAL_WUT_CONFIG property, prior to entering sleep mode, the wake-up timer will count for a time specified defined by the GLOBAL_WUT_R and GLOBAL_WUT_M properties. At the expiration of this period, an interrupt will be generated on the nIRQ pin if this interrupt is enabled in the INT_CTL_CHIP_ENABLE property. The microcontroller will then need to verify the interrupt by reading the chip interrupt status either via GET_INT_STATUS or a fast response register. The formula for calculating the Wake-Up Period is as follows:

$$WUT = WUT\_M \times \frac{4}{32{,}768} \times 2^{WUT\_R} \text{ ms}$$

The RC oscillator frequency will change with temperature; so, a periodic recalibration is required. The RC oscillator is automatically calibrated during the POWER_UP command and exits from the Shutdown state. To enable the recalibration feature, CAL_EN must be set in the GLOBAL_WUT_CONFIG property, and the desired calibration period should be selected via WUT_CAL_PERIOD[2:0] in the same API property. During the calibration, the 32 kHz RC oscillator frequency is compared to the 30 MHz XTAL and then adjusted accordingly. The calibration needs to start the 30 MHz XTAL, which increases the average current consumption; so, a longer CAL_PERIOD results in a lower average current consumption. The 32 kHz XTAL accuracy is comprised of both the XTAL parameters and the internal circuit. The XTAL accuracy can be defined as the XTAL initial error + XTAL aging + XTAL temperature drift + detuning from the internal oscillator circuit. The error caused by the internal circuit is typically less than 10 ppm.

API Properties	Description	Requirements/Notes
GLOBAL_WUT_CONFIG	GLOBAL WUT configuration	WUT_EN—Enable/disable wake up timer. WUT_LBD_EN—Enable/disable low battery detect measurement on WUT interval. WUT_LDC_EN: 0 = Disable low duty cycle operation. 1 = RX LDC operation treated as wake up START_RX WUT state is used 2 = TX LDC operation treated as wakeup START_TX WUT state is used CAL_EN—Enable calibration of the 32 kHz RC oscillator WUT_CAL_PERIOD[2:0]—Sets calibration period.
GLOBAL_WUT_M_15_8	Sets HW WUT_M[15:8]	WUT_M—Parameter to set the actual wakeup time. See equation above.
GLOBAL_ WUT_M_7_0	Sets HW WUT_M[7:0]	WUT_M—Parameter to set the actual wakeup time. See equation above.
GLOBAL_WUT_R	Sets WUT_R[4:0] Sets WUT_SLEEP to choose WUT state	WUT_R—Parameter to set the actual wakeup time. See equation above. WUT_SLEEP: 0 = Go to ready state after WUT 1 = Go to sleep state after WUT
GLOBAL_WUT_LDC	Sets FW internal WUT_LDC	WUT_LDC—Parameter to set the actual wakeup time. See equation in "8.2. Low Duty Cycle Mode (Auto RX Wake-Up)" on page 43.

Table 15. WUT Specific Commands and Properties

8.2. Low Duty Cycle Mode (Auto RX Wake-Up)

Command/Property	Description	Requirements/Notes
WUT Interrupt Enable		
INT_CTL_ENABLE	Interrupt enable property	CHIP_INT_STATUS_EN—Enables chip status interrupt.
INT_CTL_CHIP_ENABLE	Chip interrupt enable property	WUT_EN—Enables WUT interrupt.
32 kHz Clock Source Selection		
GLOBAL_CLK_CFG	Clock configuration options	CLK_32K_SEL[2:0]—Configuring the source of WUT.
WUT Interrupt Output		
GPIO_PIN_CFG	Host can enable interrupt on WUT expire	GPIOx_MODE[5:0] = 14 and NIRQ_MODE[5:0] = 39.
RX/TX Operation		
START_RX/TX	START RX/TX when wake up timer expire	START = 1.

Table 16. WUT Related API Commands and Properties

The low duty cycle (LDC) mode is implemented to automatically wake-up the receiver to check if a valid signal is available or to enable the transmitter to send a packet. It allows low average current polling operation by the TRW-HP9M2W for which the wake-up timer (WUT) is used. RX and TX LDC operation must be set via the GLOBAL_WUT_CONFIG property when setting up the WUT. The LDC wake-up period is determined by the following formula:

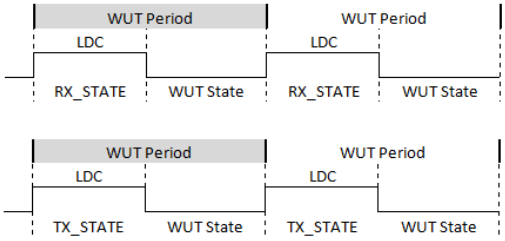


Figure 18. RX and TX LDC Sequences

The basic operation of RX LDC mode is shown in Figure 19. The receiver periodically wakes itself up to work on RX_STATE during LDC mode duration. If a valid preamble is not detected, a receive error is detected, or an entire packet is not received, the receiver returns to the WUT state (i.e., ready or sleep) at the end of LDC mode duration and remains in that mode until the beginning of the next wake-up period. If a valid preamble or sync word is detected, the receiver delays the LDC mode duration to receive the entire packet. If a packet is not received during two LDC mode durations, the receiver returns to the WUT state at the last LDC mode duration until the beginning of the next wake-up period.

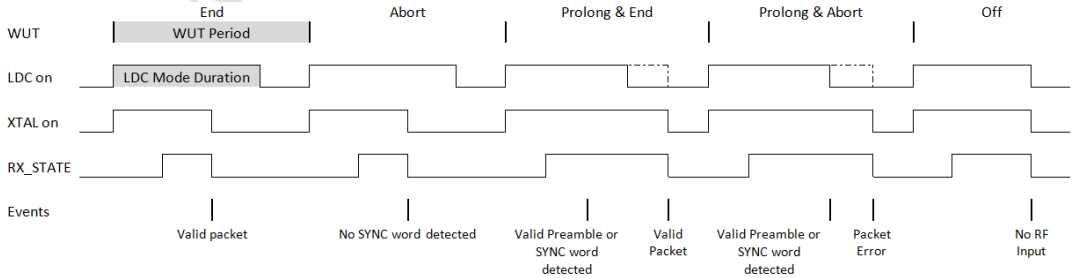


Figure 19. Low Duty Cycle Mode for RX

In TX LDC mode, the transmitter periodically wakes itself up to transmit a packet that is in the data buffer. If a

packet has been transmitted, nIRQ goes low if the option is set in the INT_CTL_ENABLE property. After transmitting, the transmitter immediately returns to the WUT state and stays there until the next wake-up time expires.

8.3. Temperature, Battery Voltage, and Auxiliary ADC

The TRW-HP9M2W family contains an integrated auxiliary ADC for measuring internal battery voltage, an internal temperature sensor, or an external component over a GPIO. The ADC utilizes a SAR architecture and achieves
11-bit resolution. The Effective Number of Bits (ENOB) is 9 bits. When measuring external components, the input voltage range is 1 V, and the conversion rate is between 300 Hz to 2.44 kHz. The ADC value is read by first sending the GET_ADC_READING command and enabling the inputs that are desired to be read: GPIO, battery, or temp. The temperature sensor accuracy at 25 °C is typically ±2 °C.

Command Stream

GET_ADC_READING Command	7	6	5	4	3	2	1	0
CMD	0							
ADC_EN	0	0	0	TEMPERATURE_EN	BATTERY_VOLTAGE_EN	ADC_GPIO_EN	ADC_GPIO_PIN[1:0]	
ADC_CFG				UDTIME[3:0]	GPIO_ATT [3:0]			

Reply Stream

GET_ADC_READING Reply	7	6	5	4	3	2	1	0
CTS	CTS[7:0]							
GPIO_ADC	GPIO_ADC[15:8]							
GPIO_ADC	GPIO_ADC[7:0]							
BATTERY_ADC	BATTERY_ADC[15:8]							
BATTERY_ADC	BATTERY_ADC[7:0]							
TEMP_ADC	TEMP_ADC[15:8]							
TEMP_ADC	TEMP_ADC[7:0]							
RESERVED	Reserved							
RESERVED	Reserved							

Parameters

□□TEMPERATURE_EN

0 = Do not perform ADC conversion of temperature. This will read 0 value in reply TEMPERATURE.

1 = Perform ADC conversion of temperature. This results in
TEMP_ADC. Temp (°C) = TEMP_ADC[15:0] x 568/2560 – 297

□□BATTERY_VOLTAGE_EN

0 = Don't do ADC conversion of battery voltage, will read 0 value in reply BATTERY_ADC

1 = Do ADC conversion of battery voltage, results in BATTERY_ADC. Vbatt = 3*BATTERY_ADC/1280

□□ADC_GPIO_EN

0 = Don't do ADC conversion on GPIO, will read 0 value in reply

1 = Do ADC conversion of GPIO, results in GPIO_ADC. Vgpio = GPIO_ADC/GPIO_ADC_DIV where GPIO_ADC_DIV is defined by GPIO_ATT selection.

□□ADC_GPIO_PIN[1:0] - Select GPIOx pin. The pin must be set as input.

0 = Measure voltage of GPIO0

1 = Measure voltage of GPIO1

2 = Measure voltage of GPIO2

3 = Measure voltage of GPIO3

□□UDTIME[7:4] - ADC conversion Time = SYS_CLK / 12 / 2^(UDTIME + 1). Defaults to 0xC if ADC_CFG is 0.

Selecting shorter conversion times will result in lower ADC resolution and longer times will result in higher ADC resolution.

GPIO_ATT[3:0] - Sets attenuation of gpio input voltage when vgpio measured. Defaults to 0xC if ADC_CFG is 0.

0x0 = ADC range 0 to 0.8V. GPIO_ADC_DIV = 2560

0x4 = ADC range 0 to 1.6V. GPIO_ADC_DIV = 1280

□□ □□ 0x8 = ADC range 0 to 2.4V. GPIO_ADC_DIV = 853.33

0x9 = ADC range 0 to 3.6V. GPIO_ADC_DIV = 426.66

0xC = ADC range 0 to 3.2V. GPIO_ADC_DIV = 640

□□ Response

□□GPIO_ADC[15:0] - ADC value of voltage on GPIO

□□BATTERY_ADC[15:0] - ADC value of battery voltage

□□TEMP_ADC[15:0] - ADC value of temperature sensor voltage

□□RESERVED[7:0] - RESERVED FOR FUTURE USE

□□RESERVED[7:0] - RESERVED FOR FUTURE USE

8.4. Low Battery Detector

The low battery detector (LBD) is enabled and utilized as part of the wake-up-timer (WUT). The LBD function is not available unless the WUT is enabled, but the host MCU can manually check the battery voltage anytime with the auxiliary ADC. The LBD function is enabled in the GLOBAL_WUT_CONFIG API property. The battery voltage will be compared against the threshold each time the WUT expires. The threshold for the LBD function is set in GLOBAL_LOW_BATT_THRESH. The threshold steps are in increments of 50 mV, ranging from a minimum of 1.5 V up to 3.05 V. The accuracy of the LBD is ±3%. The LBD notification can be configured as an interrupt on the nIRQ pin or enabled as a direct function on one of the GPIOs.

8.5. Antenna Diversity

To mitigate the problem of frequency-selective fading due to multipath propagation, some transceiver systems use a scheme known as antenna diversity. In this scheme, two antennas are used. Each time the transceiver enters RX mode the receive signal strength from each antenna is evaluated. This evaluation process takes place during the preamble portion of the packet. The antenna with the strongest received signal is then used for the remainder of that RX packet. The same antenna will also be used for the next corresponding TX packet. This chip fully supports antenna diversity with an integrated antenna diversity control algorithm. The required signals needed to control an external SPDT RF switch (such as a PIN diode or GaAs switch) are available on the GPIOx pins. The operation of these GPIO signals is programmable to allow for different antenna diversity architectures and configurations. The antdiv[2:0] bits are found in the MODEM_ANT_DIV_CONTROL API property descriptions and enable the antenna diversity mode. The GPIO pins are capable of sourcing up to 5 mA of current; so, it may be used directly to forward-bias a PIN diode if desired. The antenna diversity algorithm will automatically toggle back and forth between the antennas until the packet starts to arrive. The recommended preamble length for optimal antenna selection is 8 bytes.