XBee® SX 868

Radio Frequency (RF) Module

User Guide

Revision history—90001538

Revision	Date	Description
A	June 2017	Initial release.
B	September 2017	Updated power consumption values. Added the **LB** (LNA Bypass) command.
C	May 2018	Added note on range estimation.

Trademarks and copyright

Disclaimers

Information in this document is subject to change without notice and does not represent a commitment on the part of Digi International. Digi provides this document “as is,” without warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

Warranty

To view product warranty information, go to the following website:

www.digi.com/howtobuy/terms

Customer support

Gather support information: Before contacting Digi technical support for help, gather the following information:

- Product name and model
- Product serial number (s)
- Firmware version
- Operating system/browser (if applicable)
- Logs (from time of reported issue)
- Trace (if possible)
- Description of issue
- Steps to reproduce

Contact Digi technical support: Digi offers multiple technical support plans and service packages. Contact us at +1 952.912.3444 or visit us at www.digi.com/support.

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (XBee® SX 868 RF Module User Guide, 90001538 C) in the subject line of your email.

Contents

Advanced application features

Networking methods

Modes

Sleep modes

AT commands

Operate in API mode

Regulatory information

PCB design and manufacturing

XBee® SX 868 RF Module User Guide

The XBee SX 868 RF Module is an embedded radio frequency (RF) device that provides wireless connectivity to end-point devices in mesh networks.

The XBee SX 868 RF Module delivers up to 32 mW of Effective Isotropically Radiated Power (EIRP) and has excellent receive sensitivity, low operating current, and exceptional performance in low power modes. The module's frequency hopping technology offers advanced interference immunity, affording long range data throughput even in challenging RF environments. The XBee SX 868 RF Module uses a microprocessor that supports host communication through Serial Peripheral Interface (SPI) or universal asynchronous receiver/transmitter (UART), as well as digital, analog, and pulse width modulation (PWM) lines for interfacing with peripherals.

Applicable firmware and hardware

This manual supports the following firmware:

- 0xA00x, Europe

It supports the following hardware:

- XBee SX 868 RF Module

Technical specifications

Regulatory conformity summary

This table describes the agency approvals for the devices.

See Regulatory information for details.

Country	Approval
Europe (CE)	Yes

The following table shows the channel frequencies.

Operational frequency band[1]	Channel #	Frequency
K	0	863.15 MHz
	1	863.35 MHz
	2	863.55 MHz
	3	863.75 MHz
	4	863.95 MHz
	5	864.15 MHz
	6	864.35 MHz
	7	864.55 MHz
	8	864.75 MHz
	9	N/A[2]

Operational frequency band[1]	Channel #	Frequency
L	10	865.15 MHz
	11	865.35 MHz
	12	865.55 MHz
	13	865.75 MHz
	14	865.95 MHz
	15	866.15 MHz
	16	866.35 MHz
	17	866.55 MHz
	18	866.75 MHz
	19	866.95 MHz
	20	867.15 MHz
	21	867.35 MHz
	22	867.55 MHz
	23	867.75 MHz
	24	N/A[2]
M	25	868.15 MHz
	26	868.35 MHz
N	27	868.85 MHz
	28	869.05 MHz
Q/R[3]	29	869.85 MHz

[1] The operational frequency bands are in compliance with Table B.1 in ETSI EN 300 220-2 V3.1.1.

[2] Channels 9 and 24 are removed to comply with section 4.3.4 of ETSI EN 300 220-2 V3.1.1

[3] Band R applies when polite spectrum access is being used, i.e. more than one channel is enabled. Band Q applies when polite spectrum access is not used, i.e. when channel 29 is the only enabled channel.

Power requirements

The following table describes the power requirements for the XBee SX 868 RF Module.

Specification	Condition	Value
Supply voltage range		2.4 to 3.6 VDC
Typical supply voltage		3.3 V

Specification	Condition	Value
Receive current	VCC = 3.3 V	40 mA
	VCC = 3.3 V, LNA bypass enabled	34 mA
Transmit current	VCC = 3.3 V	55 mA @ 32 mW EIRP
	VCC = 3.3 V	45 mA @ 16 mW EIRP
	VCC = 3.3 V	40 mA @ 10 mW EIRP
	VCC = 3.3 V	35 mA @ 5 mW EIRP
	VCC = 3.3 V	32 mA @ 2 mW EIRP
Sleep current	VCC 3.3 V, temperature = 25 °C	1.8 µA

Networking and security specifications

The following table describes the networking and security specifications for the devices.

Specification	Value
Modulation	Gaussian Frequency Shift Keying (GFSK)
Spreading technology	Frequency Hopping Spread Spectrum (FHSS)
Supported network topologies (software selectable)	Peer-to-peer (master/slave relationship not required), point-to-point/point-to-multipoint, mesh
Encryption	Use EE (Security Enable) to enable encryption. Use KY (AES Encryption Key) to set the encryption key.

Performance specifications

The following table describes the performance specifications for the devices.

Note Range figure estimates are based on free-air terrain with limited sources of interference. Actual range will vary based on transmitting power, orientation of transmitter and receiver, height of transmitting antenna, height of receiving antenna, weather conditions, interference sources in the area, and terrain between receiver and transmitter, including indoor and outdoor structures such as walls, trees, buildings, hills, and mountains.

Specification	Condition	Value
Frequency range		863 to 870 MHz
RF data rate (software selectable)	Low data rate	10 kb/s
	High data rate	80 kb/s

Specification	Condition	Value
Transmit power (software selectable)	EIRP[1]	Up to 15 dBm (32 mW)
	ERP[2]	Up to 13 dBm (20 mW)
Maximum data throughput	High data rate	38.4 kb/s
Available channel frequencies	All data rates	28
Rural range line of sight[3]	Low data rate	Up to 14.5 km (9 mi)
Urban range line of sight[4]	Low data rate	Up to 2.5 km (1.5 mi)
Receiver sensitivity	Low data rate	-113 dBm
	High data rate	-106 dBm
	Low data rate, LNA bypass enabled	-100 dBm
	High data rate, LNA bypass enabled	-94 dBm
Receiver IF selectivity	Low data rate ± 200 kHz	40 dB
	Low data rate ± 400 kHz	45 dB
	High data rate ± 200 kHz	33 dB
	High data rate ± 400 kHz	40 dB
Receiver RF selectivity	Below 863 MHz and above 870 MHz	> 60 dB
UART data rate (software selectable)		1200 - 921600 baud
SPI clock rate		Up to 6 Mb/s

General specifications

The following table describes the general specifications for the devices.

Specification	Value
Dimensions	3.38 x 2.21 x 0.32 cm (1.33 x 0.87 x 0.125 in)
Weight	3 g
Restriction of Hazardous Substances (RoHS)	Compliant
Manufacturing	ISO 9001:2008 registered standards
Host interface connector	37 castellated SMT pads

1Effective Isotropically Radiated Power (EIRP) is the device's output power plus 2.1 dBi (dipole antenna gain).

2Effective radiated power (ERP) is the specification tested for regulatory compliance.

3This number was measured with the LNA bypass disabled.

4This number was measured with the LNA bypass disabled.

Specification	Value
Antenna connector options	U.FL or RF pad
Antenna impedance	50 Ω unbalanced
Maximum input RF level at antenna port	6 dBm
Operating temperature	-40°C to 85°C
Digital I/O	13 I/O lines, 5 output lines
Analog-to-digital converter (ADC)	4 10-bit analog inputs
Pulse width modulator (PWM)	2 outputs

GPIO specifications

The following table provides the electrical specifications for the GPIO pads.

GPIO electrical specification	Value
Voltage - supply	2.4 - 3.6 V
Low Schmitt switching threshold	0.3 * VCC
High Schmitt switching threshold	0.7 * VCC
Input current for logic 0	-0.1 µA
Input current for logic 1	0.1 µA
Input pull-up resistor value	40 kΩ
Input pull-down resistor value	40 kΩ
Output voltage for logic 0	0.05 * VCC
Output voltage for logic 1	0.95 * VCC
Output source/sink current	1 mA
Total output current (for GPIO pads)	20 mA

LBT and AFA specifications

The following table provides the Listen Before Talk (LBT) and Adaptive Frequency Agility (AFA) specifications.

Specification	Condition	Value
Channel spacing		200 kHz
Receiver bandwidth		150 kHz
Modulation bandwidth		< 300 kHz

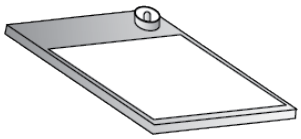
Specification	Condition	Value
LBT threshold	Low data rate	< -95 dBm
	High data rate	< -90 dBm
TX on time		< 1 second

Get started

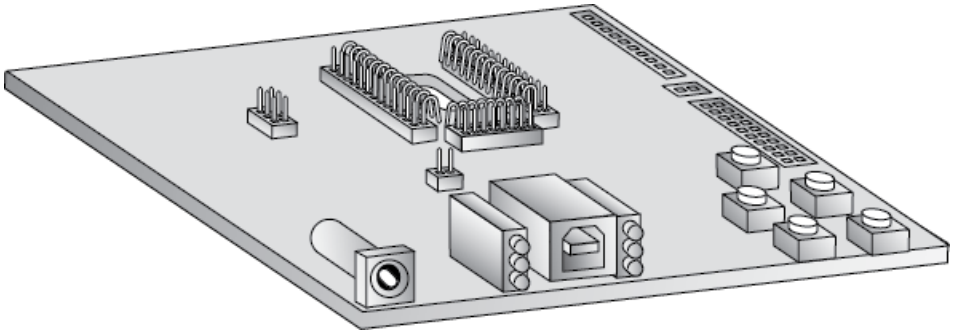
Verify kit contents

The XBee SX 868 RF Module development kit contains the following components:

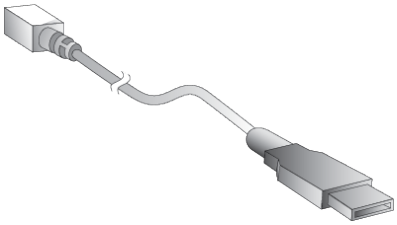
XBee U.FL module (3)



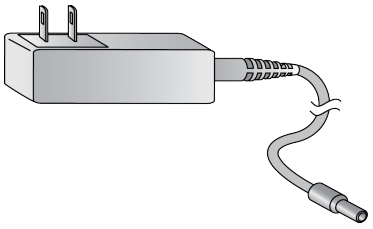
XBee development board (3)



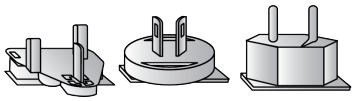
USB cable (2)



Power supply (2)



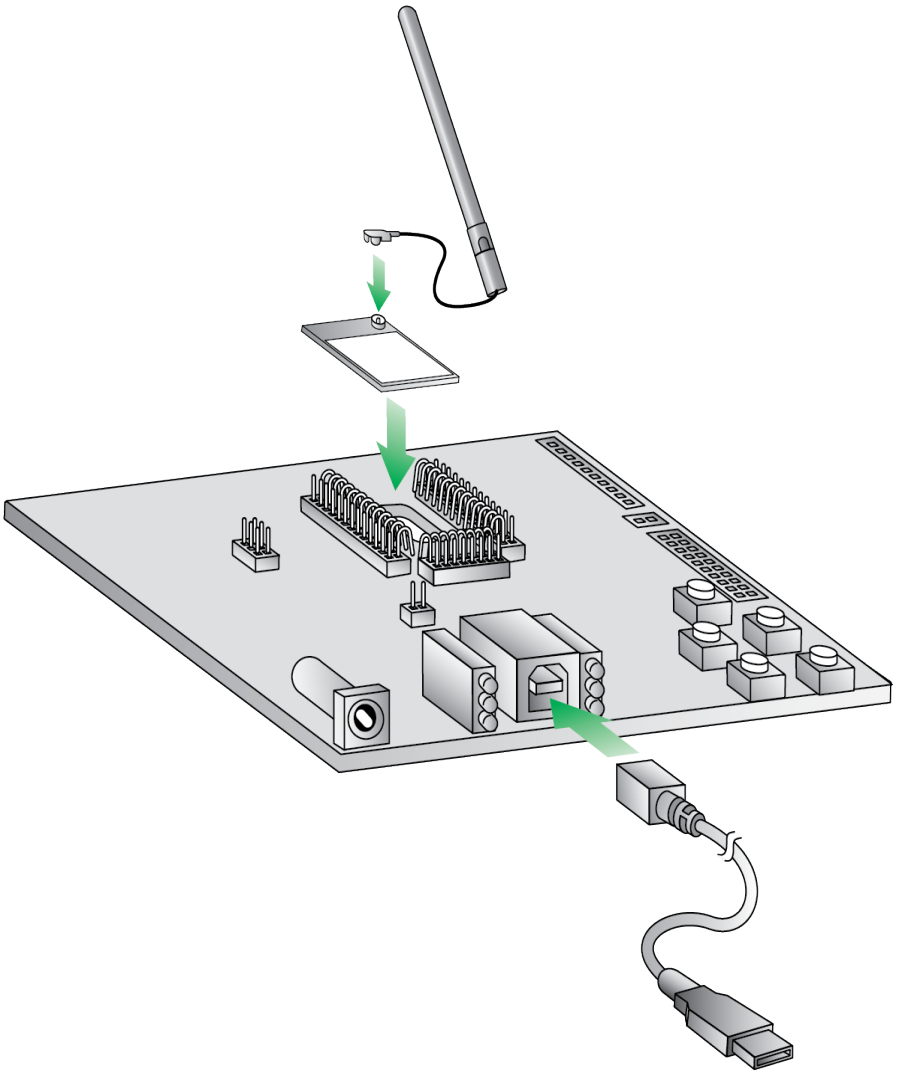
Set of power supply adapters (2)



Antenna U.FL (3)

Connect the hardware

The following illustration shows you how to assemble the hardware components of the development kit.



1. Attach the XBee SX 868 RF Modules to the development boards.
2. Attach the antennas to the devices.
3. Connect the USB cables to the development boards.

CAUTION! Before you remove a device from a development board, make sure the board is not powered by a USB cable or a battery.

Configure the device using XCTU

XBee Configuration and Test Utility (XCTU) is a multi-platform program that enables users to interact with Digi radio frequency (RF) devices through a graphical interface. The application includes built-in tools that make it easy to set up, configure, and test Digi RF devices.

For instructions on downloading and using XCTU, see the XCTU User Guide.

Click **Discover devices** and follow the instructions. XCTU should discover two XBee SX 868 RF Module modules.

Click **Add selected devices**.The devices appear in the **Radio Modules** list. You can click a module to view and configure its individual settings. For more information on these items, see AT commands.

Configure the devices for a range test

For devices to communicate with each other, you must configure them so they are in the same network. To obtain all possible data from the remote device, you must also set the local device to API mode. For more information on API mode, see Operate in API mode.

For devices to communicate with each other, you configure them so they are in the same network. You also set the local device to API mode to obtain all possible data of the remote device.

When you connect the development board to a PC for the first time, the PC automatically installs drivers, which may take a few minutes to complete.

1. Add the two devices to XCTU.
2. Select the first module and click the **Load default firmware settings** button.
3. Configure the following parameters:

   ID: 2015
   NI: LOCAL_DEVICE
   AP: API Mode Without Escapes [1]
4. Click the **Write radio settings** button.
5. Select the other module and click the **Default firmware settings** button.
6. Configure the following parameters:

   ID: 2015
   NI: REMOTE_DEVICE
   AP: Transparent Mode [0]
7. Click the **Write radio settings** button.

   After you write the radio settings for each device, their names appear in the **Radio Modules** area. The Port indicates that the LOCAL_DEVICE is in API mode.
8. Disconnect REMOTE_DEVICE from the computer, remove it from XCTU, and connect it to its own power supply.
9. Leave LOCAL_DEVICE connected to the computer. Connect LOCAL_DEVICE to its own power supply.
10. Place REMOTE_DEVICE at the testing location and connect its power supply.

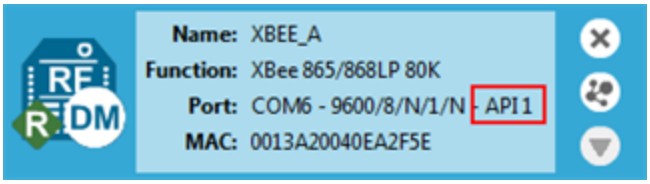
Configure remote devices

You can communicate with remote devices over the air through a corresponding local device. Configure the local device in API mode because remote commands only work in API mode. Configure remote devices in either API or Transparent mode.

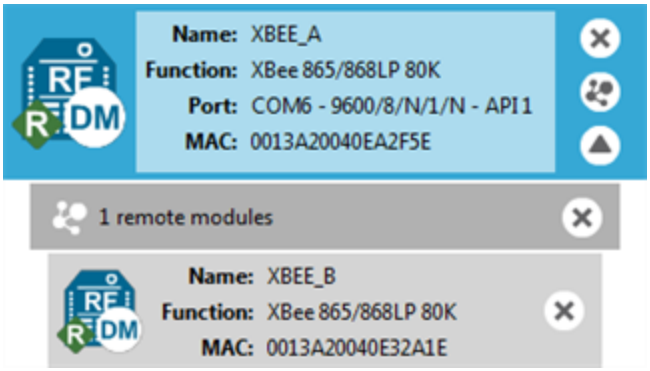
These instructions show you how to configure the LT (Associated LED Blink Time) parameter on a remote device.

1. Add two XBee devices to XCTU.
2. Configure the first device in API mode and name it **XBEE_A**.
3. Configure the second device in either API or Transparent mode, and name it **XBEE_B**.
4. Disconnect XBEE_B from your computer and remove it from XCTU.
5. Connect XBEE_B to a power supply (or laptop or portable battery).

   The **Radio Modules** area should look something like this.



6. Select **XBEE_A** and click the **Discover radio nodes in the same network** button .
7. Click **Add selected devices** in the **Discovering remote devices** dialog. The discovered remote device appears below XBEE_A.



8. Select the remote device **XBEE_B**, and configure the following parameter:

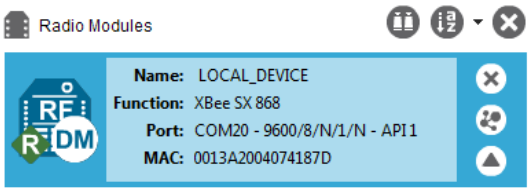
   LT: FF (hexidecimal representation for 2550 ms)

9. Click the **Write radio settings** button .

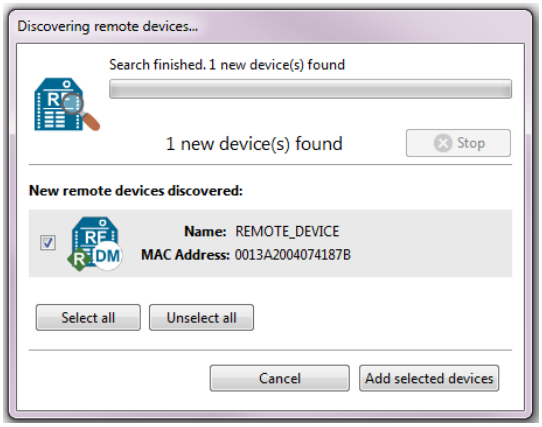
   The remote XBee device now has a different LED blink time.
10. To return to the default LED blink times, change the **LT** parameter back to 0 for XBEE_B.

Perform a range test

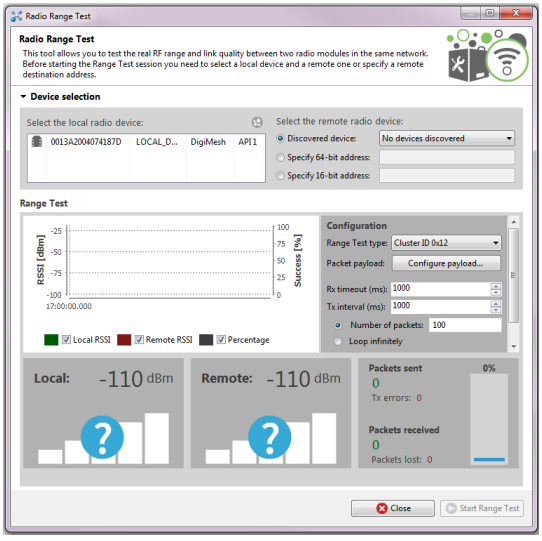
1. Go to the XCTU display for LOCAL_DEVICE.



2. Click  to discover remote devices within the same network. The **Discover remote devices** dialog appears.



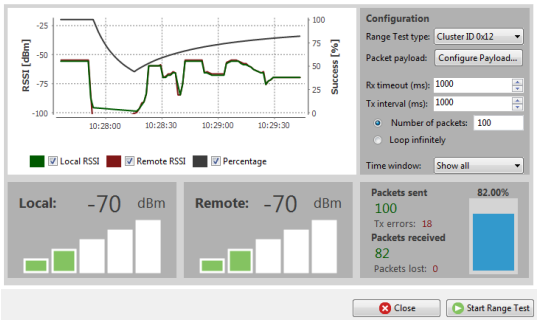
3. Click **Add selected devices**.
4. Click  and select **Range test**. The **Radio Range Test** dialog appears.



5. In the **Select the local radio device** area, select LOCAL_DEVICE. XCTU automatically selects the **Discovered device radio** button and enables the **Start Range Test** button.

6. Click **Start Range Test** to begin the range test.

   If the test is running properly, the packets sent should match the packets received. You will also see the received signal strength indicator (RSSI) update for each radio after each reception.



7. You can move the LOCAL_DEVICE and REMOTE_DEVICE farther from each other to observe the signal strength at different distances.
8. Click **Stop Range Test** when the test is complete.
9. You can test different data rates by reconfiguring the BR (RF Data Rate) parameters on both devices and starting a new range test.

XBee Network Assistant

The XBee Network Assistant is an application designed to inspect and manage RF networks created by Digi XBee devices. Features include:

- Join and inspect any nearby XBee network to get detailed information about all the nodes it contains.
- Update the configuration of all the nodes of the network, specific groups, or single devices based on configuration profiles.
- Geo-locate your network devices or place them in custom maps and get information about the connections between them.
- Export the network you are inspecting and import it later to continue working or work offline.
- Use automatic application updates to keep you up to date with the latest version of the tool.

See the *XBee Network Assistant User Guide* for more information.

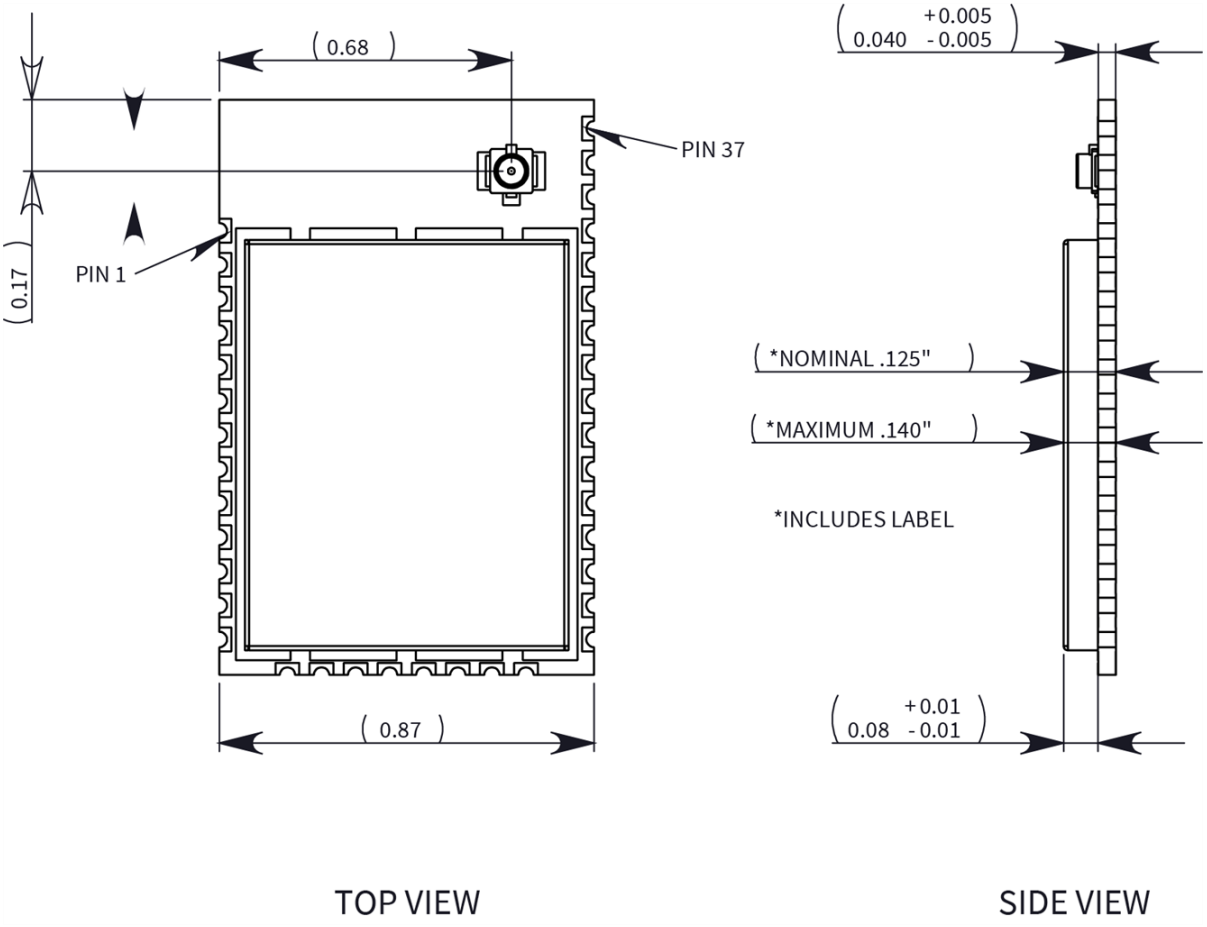
To install the XBee Network Assistant:

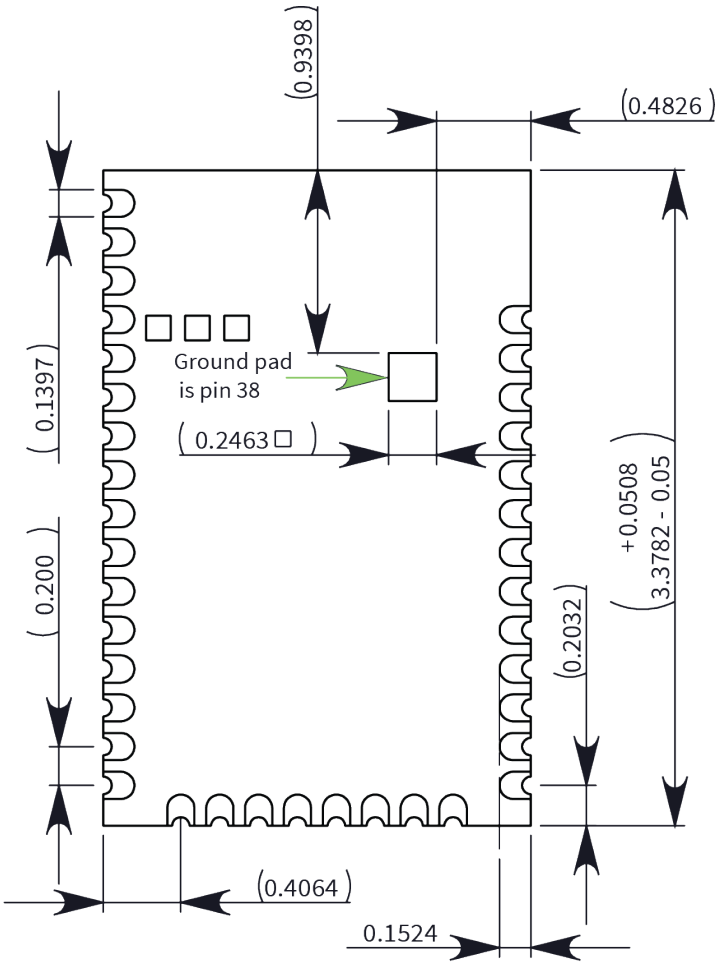
1. Navigate to digi.com/xbeenetworkassistant.
2. Click **General Diagnostics, Utilities and MIBs**.
3. Click the **XBee Network Assistant - Windows x86** link.
4. When the file finishes downloading, run the executable file and follow the steps in the XBee Network Assistant Setup Wizard.

Hardware

Mechanical drawings

The following figures show the XBee SX 868 RF Module mechanical drawings. All dimensions are in centimeters. The XBee SX 868 RF Module differs from other surface-mount XBee modules. It has an additional ground pad on the underside of the module used for heat dissipation. For more details, see PCB design and manufacturing.





Bottom view

Pin signals

The following table describes the pin signals. Low-asserted signals are distinguished with a horizontal line over the signal name.

Pin	Name	I/O	Default state	Function
1	GND	-	-	Ground
2	VCC	I	-	Power supply
3	DOUT	I/O	Output	UART data out
4	DIN/$\overline{\text{CONFIG}}$	I/O	Input	UART data in
5	DIO12	I/O	Disabled	GPIO

Pin	Name	I/O	Default state	Function
6	$\overline{\text{RESET}}$	I	-	Drive low to reset device. Do not drive pin high; pin may only be driven open drain or low. Pin has an internal 20k pullup resistor. The minimum reset pulse time is 100 ns.
7	DIO10/RSSI/PWM0	I/O	Output	GPIO / RX Signal Strength Indicator
8	DIO11/PWM1	I/O	Disabled	GPIO / Pulse Width Modulator
9	[Reserved]	-	-	Do not connect
10	DIO8/$\overline{\text{DTR}}$ /SLEEP_RQ	I/O	Input	GPIO / Pin Sleep Control line ($\overline{\text{DTR}}$ on the development board)
11	GND	-	-	Ground
12	DO19/SPI_$\overline{\text{ATTN}}$	O	Output	GPO / Serial Peripheral Interface (SPI) Attention or UART Data Present indicator
13	GND	-	-	Ground
14	DO18/SPI_CLK	I/O [1]	Input	GPO / SPI clock
15	DO17/SPI_$\overline{\text{SSEL}}$	I/O [2]	Input	GPO / SPI not select
16	DO16/SPI_MOSI	I/O [3]	Input	GPO / SPI Data In
17	DO15/SPI_MISO	O	Output	GPO/SPI Data Out Tri-stated when SPI_$\overline{\text{SSEL}}$ is high
18	[Reserved]	-	-	Do not connect
19	[Reserved]	-	-	Do not connect
20	[Reserved]	-	-	Do not connect
21	[Reserved]	-	-	Do not connect
22	GND	-	-	Ground
23	[Reserved]	-	-	Do not connect
24	DIO4	I/O	Disabled	GPIO
25	DIO7/$\overline{\text{CTS}}$	I/O	Output	GPIO / UART Clear to Send Flow Control

[1]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.
[2]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.
[3]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.

Pin	Name	I/O	Default state	Function
26	DIO9/ON/$\overline{\text{SLEEP}}$	I/O	Output	GPIO / Module Sleep Status Indicator
27	V_{REF}	-	-	Feature not supported on this device. Used on other XBee devices for analog voltage reference.
28	DIO5/ASSOC	I/O	Output	GPIO / Associate Indicator
29	DIO6/$\overline{\text{RTS}}$	I/O	Disabled	GPIO / UART Request to Send Flow Control
30	DIO3/AD3	I/O	Disabled	GPIO / Analog Input
31	DIO2/AD2	I/O	Disabled	GPIO / Analog Input
32	DIO1/AD1	I/O	Disabled	GPIO / Analog Input
33	DIO0/AD0	I/O	Input	GPIO / Analog Input / Commissioning Pushbutton
34	[Reserved]	-	-	Do not connect
35	GND	-	-	Ground
36	RF_PAD	I/O	-	RF connection for RF pad variant
37	[Reserved]	-	-	Do not connect
38	GND	-	-	Ground pad for heat transfer to host PCB. Located on the underside of the XBee module.

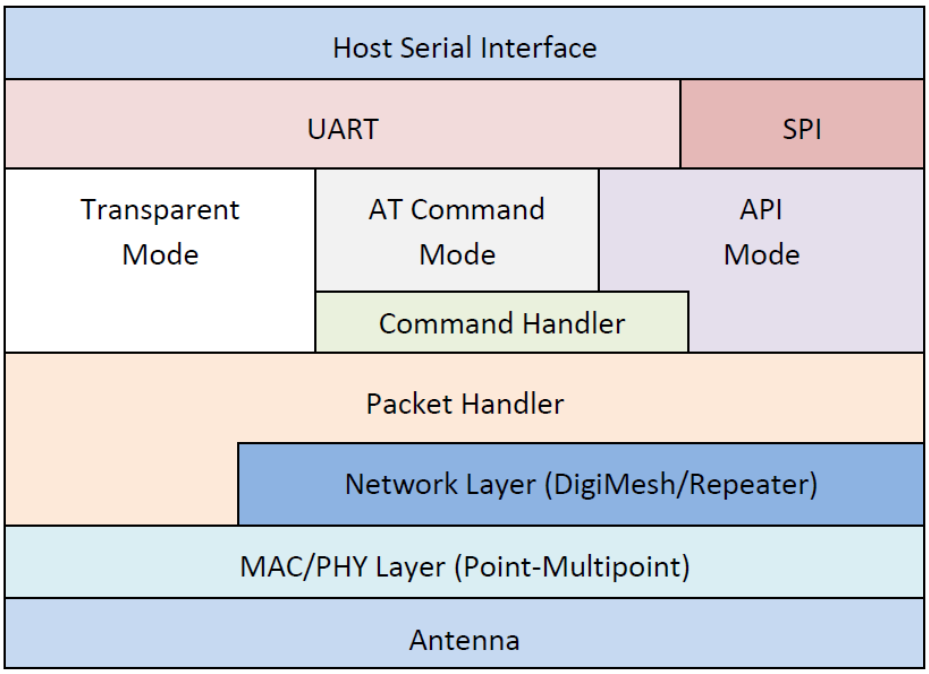
Pin connection recommendations

The only required pin connections are VCC, GND, DOUT and DIN. To support serial firmware updates, you should connect VCC, GND, DOUT, DIN, RTS, and SLEEP (DTR).

Operation

Operation

The XBee SX 868 RF Module uses a multi-layered firmware base to order the flow of data, dependent on the hardware and software configuration you choose. The following configuration block diagram shows the host serial interface as the physical starting point and the antenna as the physical endpoint for the transferred data. A block must be able to touch another block above or below it for the two interfaces to interact. For example, if the device uses SPI mode, Transparent mode is not available as shown in the following image:



The command handler code processes commands from AT Command Mode or API Mode; see API serial exchanges. The command handler also processes commands from remote devices; see Remote AT commands.

Listen Before Talk and Automatic Frequency Agility

This device implements Listen Before Talk (LBT) and Automatic Frequency Agility (AFA). The advantage of LBT with AFA is that the device bypasses the Duty Cycle requirement imposed by European standards. LBT+AFA requires that you use at least two frequencies for transmission. See Regulatory conformity summary for a complete list of channels and frequencies.

This feature provides a level of fairness to the devices in a given area. Before this device transmits, it senses a channel to determine if there is activity by taking an RSSI measurement for 5 ms. If the measurement is below the threshold, the device transmits on that channel. If there is activity, that channel is not used, and the device listens for at least 5 ms to allow transmissions to be received.

After the device transmits on a channel, it will not transmit on that channel again until the minimum TX off time has been met, which is greater than 100 ms. It is useful to have many channels in your channel mask, so transmissions are less likely to be delayed.

European requirements also state that only 100 seconds of transmission may occur over the period of an hour on 200 kHz of spectrum. This method simplifies and optimizes the calculations of spectrum use over the period of one hour. The standard states that the more channels you have, the more

transmission time you have in a one hour period. Calculate the effective duty cycle based on the number of available channels enabled as follows:

Effective Duty Cycle = (number of channels * 100)/3600.

For example, if you enabled two channels you would have an effective duty cycle of 5.6%.

The XBee SX 868 RF Module uses a sliding bucket algorithm to calculate usage over the period of 1 hour for each channel. Each bucket accumulates for 6 minutes.

This device has a maximum of 28 AFA channels to choose from, and channels can be excluded by setting the channel mask (**CM**) to reduce them. Since not all countries allow for all of these channels, the set may be dramatically smaller for some countries. For a complete list, refer to www.digi.com.

Single frequency mode band mode

When you set the channel mask to 0x20000000, the device is in a single frequency mode, and the frequency is 869.85 MHz. In this mode:

- LBT+AFA mode is disabled.
- The device assumes no duty cycle requirement (or 100% duty cycle).
- The **PL** is automatically limited to 5 mW to comply with the single frequency mode requirements.

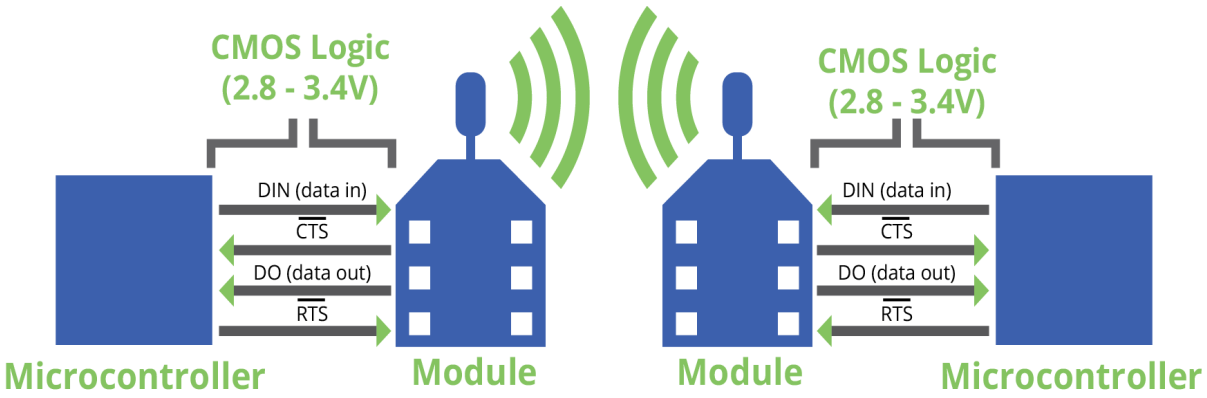
Serial communications

RF Modules interface to a host device through a serial port. Using its serial port, the device communicates with any of the following:

- Logic and voltage compatible UART
- Level translator to any serial device (for example, through an RS-232 or USB interface board)

UART data flow

Devices that have a UART interface connect directly to the pins of the XBee SX 868 RF Module as shown in the following figure. The figure shows system data flow in a UART-interfaced environment. Low-asserted signals have a horizontal line over the signal name.

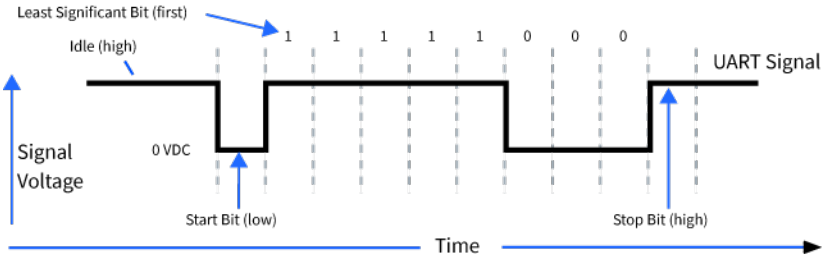


Serial data

A device sends data to the XBee SX 868 RF Module's UART through pin 4 DIN as an asynchronous serial signal. When the device is not transmitting data, the signals should idle high.

For serial communication to occur, you must configure the UART of both devices (the microcontroller and the XBee SX 868 RF Module) with compatible settings for the baud rate, parity, start bits, stop bits, and data bits.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following diagram illustrates the serial bit pattern of data passing through the device. The diagram shows UART data packet 0x1F (decimal number 31) as transmitted through the device.



SPI communications

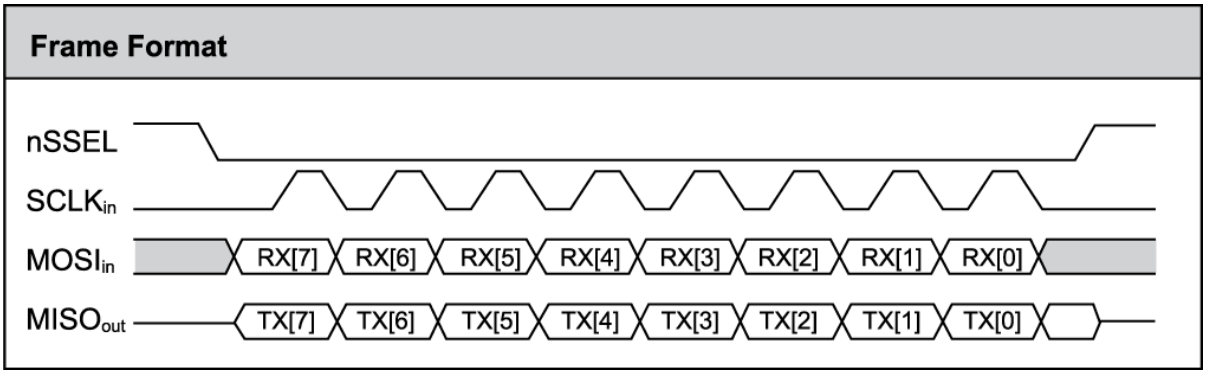
The XBee SX 868 RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.

Signal	Function
SPI_MOSI (Master Out, Slave In)	Inputs serial data from the master
SPI_MISO (Master In, Slave Out)	Outputs serial data to the master
SPI_SCLK (Serial Clock)	Clocks data transfers on MOSI and MISO
SPI_$\overline{\text{SSEL}}$ (Slave Select)	Enables serial communication with the slave
SPI_$\overline{\text{ATTN}}$ (Attention)	Alerts the master that slave has data queued to send. The XBee SX 868 RF Module asserts this pin as soon as data is available to send to the SPI master and it remains asserted until the SPI master has clocked out all available data.

In this mode:

- Data is most significant bit (MSB) first.
- Frame Format mode 0 is used. This means CPOL= 0 (idle clock is low) and CPHA = 0 (data is sampled on the clock's leading edge).
- The SPI port only supports API Mode (**AP** = **1**).

The following diagram shows the frame format mode 0 for SPI communications.



SPI operation

This section specifies how SPI is implemented on the device, what the SPI signals are, and how full duplex operations work.

SPI implementation

The XBee SX 868 RF Module operates as a SPI slave only. This means an external master provides the clock and decides when to send data. The XBee SX 868 RF Module supports an external clock rate of up to 6 Mhz (6 Mb/s).

The device transmits and receives data with the most significant bit first using SPI mode 0. This means the CPOL and CPHA are both 0. We chose Mode 0 because it is the typical default for most microcontrollers and simplifies configuring the master.

SPI signals

The XBee SX 868 RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The SPI port uses the following signals on the device:

Signal	Pin number	Applicable AT command
SPI_MOSI (Master out, Slave in)	17	**P5**
SPI_MISO (Master in, Slave out)	16	**P6**
SPI_SCLK (Serial clock)	15	**P7**
SPI_$\overline{\text{SSEL}}$ (Slave select)	14	**P8**
SPI_$\overline{\text{ATTN}}$ (Attention)	12	**P9**

Signal	Pin number	Applicable AT command
SPI_MOSI (Master out, Slave in)	33	**P5**
SPI_MISO (Master in, Slave out)	35	**P6**
SPI_CLK (Serial clock)	37	**P7**

Signal	Pin number	Applicable AT command
SPI_$\overline{\text{SSEL}}$ (Slave select)	31	**P8**
SPI_$\overline{\text{ATTN}}$ (Attention)	29	**P9**

By default, the inputs have pull-up resistors enabled. Use the **PR** command to disable the pull-up resistors. When the SPI pins are not connected but the pins are configured for SPI operation, then the device requires the pull-ups for proper UART operation.

Signal description

SPI_MOSI: When SPI_$\overline{\text{SSEL}}$ is asserted (low) and SPI_CLK is active, the device outputs the data on this line at the SPI_CLK rate. When SPI_$\overline{\text{SSEL}}$ is de-asserted (high), you should tri-state this output such that another slave device can drive the line.

SPI_MISO: The SPI master outputs data on this line at the SPI_CLK rate after it selects the desired slave. When you configure the device for SPI operations, this pin is an input.

SPI_SCLK: The SPI master outputs a low signal on this line to select the desired slave. When you configure the device for SPI operations, this pin is an input. This signal clocks data transfers on MOSI and MISO.

SPI_CLK: The SPI master outputs a low signal on this line to select the desired slave. When you configure the device for SPI operations, this pin is an input. This signal clocks data transfers on MOSI and MISO.

SPI_$\overline{\text{SSEL}}$: The SPI master outputs a clock on this pin, and the rate must not exceed the maximum allowed, 6 Mb/s. When you configure the device for SPI operations, this pin is an input. This signal enables serial communication with the slave.

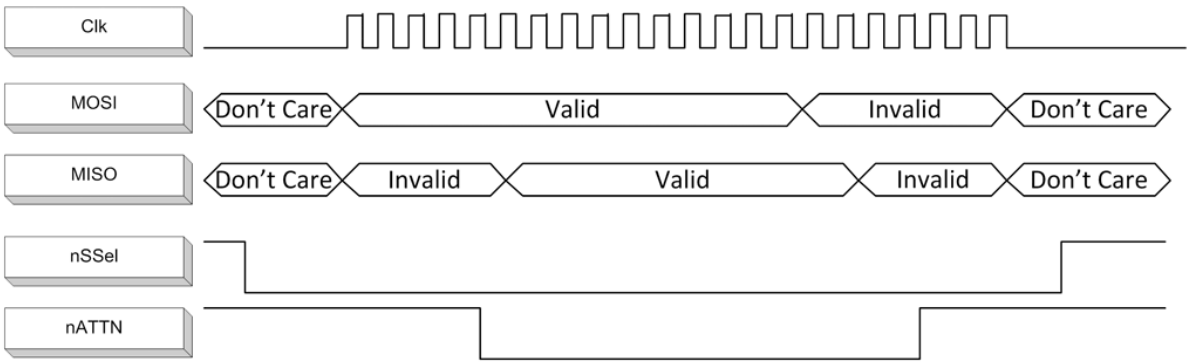
SPI_$\overline{\text{ATTN}}$: The device asserts this pin low when it has data to send to the SPI master. When you configure this pin for SPI operations, it is an output (not tri-stated). This signal alerts the master that the slave has data queued to send. The device asserts this pin as soon as data is available to send to the SPI master and it remains asserted until the SPI master has clocked out all available data.

Full duplex operation

SPI on the XBee SX 868 RF Module requires that you use API mode (without escaping) to packetize data. By design, SPI is a full duplex protocol even when data is only available in one direction. This means that when a device receives data, it also transmits and that data is normally invalid. Likewise, when the device transmits data, invalid data is probably received. To determine whether or not received data is invalid, we packetize the data with API packets.

SPI allows for valid data from the slave to begin before, at the same time, or after valid data begins from the master. When the master is sending data to the slave and the slave has valid data to send in the middle of receiving data from the master, this allows a true full duplex operation where data is valid in both directions for a period of time. Not only must the master and the slave both be able to keep up with the full duplex operation, but both sides must honor the protocol as specified.

The following diagram illustrates the SPI interface while valid data is being sent in both directions.



Low power operation

Sleep modes generally work the same on SPI as they do on UART. However, due to the addition of SPI mode, there is an option of another sleep pin, as described below.

By default, Digi configures DIO8 (SLEEP_REQUEST) as a peripheral and during pin sleep it wakes the device and puts it to sleep. This applies to both the UART and SPI serial interfaces.

If SLEEP_REQUEST is not configured as a peripheral and SPI_SSEL is configured as a peripheral, then pin sleep is controlled by SPI_SSEL rather than by SLEEP_REQUEST. Asserting SPI_SSEL (pin 15) by driving it low either wakes the device or keeps it awake. Negating SPI_SSEL by driving it high puts the device to sleep.

Using SPI_SSEL to control sleep and to indicate that the SPI master has selected a particular slave device has the advantage of requiring one less physical pin connection to implement pin sleep on SPI. It has the disadvantage of putting the device to sleep whenever the SPI master negates SPI_SSEL (meaning time is lost waiting for the device to wake), even if that was not the intent.

If the user has full control of SPI_SSEL so that it can control pin sleep, whether or not data needs to be transmitted, then sharing the pin may be a good option in order to make the SLEEP_REQUEST pin available for another purpose.

If the device is one of multiple slaves on the SPI, then the device sleeps while the SPI master talks to the other slave, but this is acceptable in most cases.

If you do not configure either pin as a peripheral, then the device stays awake, being unable to sleep in **SM**1 mode.

Configuration considerations

The configuration considerations are:

- How do you select the serial port? For example, should you use the UART or the SPI port?
- If you use the SPI port, what data format should you use in order to avoid processing invalid characters while transmitting?
- What SPI options do you need to configure?

Serial port selection

In the default configuration both the UART and SPI ports are configured for serial port operation. In this case, serial data goes out the UART until the host device asserts the SPI_SSEL signal. Thereafter all serial communications operate only on the SPI interface until a reset occurs.

If you enable only the UART, the XBee SX 868 RF Module uses only the UART, and ignores the SPI_SSEL.

If you enable only the SPI, the XBee SX 868 RF Module uses only the SPI, and ignores UART communications.

Data format

SPI only operates in API mode 1. The XBee SX 868 RF Module does not support Transparent mode or API mode 2 (which escapes control characters). This means that the AP configuration only applies to the UART, and the device ignores it while using SPI.

SPI parameters

Most host processors with SPI hardware allow you to set the bit order, clock phase and polarity. For communication with all XBee SX 868 RF Modules, the host processor must set these options as follows:

- Bit order: send MSB first
- Clock phase (CPHA): sample data on first (leading) edge
- Clock polarity (CPOL): first (leading) edge rises

All XBee SX 868 RF Modules use SPI mode 0 and MSB first. Mode 0 means that data is sampled on the leading edge and that the leading edge rises. MSB first means that bit 7 is the first bit of a byte sent over the interface.

Serial buffers

To enable the UART port, DIN and DOUT must be configured as peripherals. To enable the SPI port, SPI_MISO, SPI_MOSI, SPI_SSEL, and SPI_CLK must be enabled as peripherals. If both ports are enabled, output goes to the UART until the first input on SPI. This is the default configuration.

When input occurs on either port, that port is selected as the active port and no input or output is allowed on the other port until the next reset of the module.

If you change the configuration to configure only one port, that port is the only one enabled or used. If the parameters are written with only one port enabled, the port that is not enabled is not used even temporarily after the next reset.

If both ports are disabled on reset, the device uses the UART regardless of the incorrect configuration to ensure that at least one serial port is operational.

Serial receive buffer

When serial data enters the device through the DIN pin (or the MOSI pin), it stores the data in the serial receive buffer until the device can process it. Under certain conditions, the device may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the device such that the serial receive buffer would overflow, then it discards new data. If the UART is in use, you can avoid this by the host side honoring CTS flow control.

If the SPI is the serial port, no hardware flow control is available. It is your responsibility to ensure that the receive buffer does not overflow. One reliable strategy is to wait for a TX_STATUS response after each frame sent to ensure that the device has had time to process it.

Serial transmit buffer

When the device receives RF data, it moves the data into the serial transmit buffer and sends it out the UART or SPI port. If the serial transmit buffer becomes full and the system buffers are also full, then it drops the entire RF data packet. Whenever the device receives data faster than it can process and transmit the data out the serial port, there is a potential of dropping data.

UART flow control

You can use the RTS and CTS pins to provide RTS and/or CTS flow control. CTS flow control provides an indication to the host to stop sending serial data to the device. RTS flow control allows the host to signal the device to not send data in the serial transmit buffer out the UART. To enable RTS/CTS flow control, use the **D6** and **D7** commands.

Note Serial port flow control is not possible when using the SPI port.

CTS flow control

If you enable CTS flow control (**D7** command), when the serial receive buffer is 17 bytes away from being full, the device de-asserts CTS (sets it high) to signal to the host device to stop sending serial data. The device reasserts CTS after the serial receive buffer has 34 bytes of space. See FT (Flow Control Threshold) for the buffer size.

In either case, CTS is not re-asserted until the serial receive buffer has **FT**-17 or less bytes in use.

RTS flow control

If you send the **D6** command to enable RTS flow control, the device does not send data in the serial transmit buffer out the DOUT pin as long as RTS is de-asserted (set high). Do not de-assert RTS for long periods of time or the serial transmit buffer will fill. If the device receives an RF data packet and the serial transmit buffer does not have enough space for all of the data bytes, it discards the entire RF data packet.

The UART Data Present Indicator is a useful feature when using RTS flow control. When enabled, the DIO19 line asserts (low asserted) when UART data is queued to be transmitted from the device. For more information, see P9 (DIO19/SPI_ATTN).

If the device sends data out the UART when RTS is de-asserted (set high) the device could send up to five characters out the UART port after RTS is de-asserted.

Serial interface protocols

The XBee SX 868 RF Module supports both Transparent and Application Programming Interface (API) serial interfaces.

Transparent operating mode

When operating in Transparent mode, the devices act as a serial line replacement. The device queues up all UART data received through the DIN pin for RF transmission. When RF data is received, the device sends the data out through the serial port. Use the Command mode interface to configure the device configuration parameters.

Note Transparent operation is not provided when using SPI.

The device buffers data in the serial receive buffer and packetizes and transmits the data when it receives the following:

- No serial characters for the amount of time determined by the **RO** (Packetization Timeout) parameter. If **RO** = 0, packetization begins when the device received a character.
- Command Mode Sequence (**GT** + **CC** + **GT**). Any character buffered in the serial receive buffer before the device transmits the sequence.
- Maximum number of characters that fit in an RF packet.

API operating mode

API operating mode is an alternative to Transparent operating mode. The frame-based API extends the level to which a host application can interact with the networking capabilities of the device. When in API mode, the device contains all data entering and leaving in frames that define operations or events within the device.

The API provides alternative means of configuring devices and routing data at the host application layer. A host application can send data frames to the device that contain address and payload information instead of using Command mode to modify addresses. The device sends data frames to the application containing status packets, as well as source and payload information from received data packets.

The API operation option facilitates many operations such as:

- Transmitting data to multiple destinations without entering Command Mode
- Receive success/failure status of each transmitted RF packet
- Identify the source address of each received packet

Advanced application features

Remote configuration commands

The API firmware has provisions to send configuration commands to remote devices using the Remote Command Request API frame (see Operate in API mode). Use the API frame to send commands to a remote device to read or set command parameters.

Send a remote command

To send a remote command, populate the Remote Command Request frame with:

- 64-bit address of the remote device
- Correct command options value
- Command and parameter data (optional)

If you want a command response, set the Frame ID set to a non-zero value. Only unicasts of remote commands are supported, and remote commands cannot be broadcast.

Apply changes on remote devices

When you use remote commands to change command parameter settings on a remote device, parameter changes do not take effect until you apply the changes. For example, changing the **BD** parameter does not change the serial interface on the remote until the changes are applied. To apply changes, do one of the following:

- Set the apply changes option bit in the API frame.
- Issue an **AC** (Apply Changes) command to the remote device.
- Issue a **WR** + **FR** command to the remote device to save changes and reset the device.

Remote command responses

If the remote device receives a remote command request transmission, and the API frame ID is non-zero, the remote sends a remote command response transmission back to the device that sent the remote command. When a remote command response transmission is received, a device sends a remote command response API frame out its serial port. The remote command response indicates the status of the command (success, or reason for failure), and in the case of a command query, it includes the register value. The device that sends a remote command will not receive a remote command response frame if either of the following conditions exist:

- The destination device could not be reached.
- The frame ID in the remote command request is set to 0.

Network commissioning and diagnostics

We call the process of discovering and configuring devices in a network for operation, "network commissioning." Devices include several device discovery and configuration features. In addition to configuring devices, you must develop a strategy to place devices to ensure reliable routes. To accommodate these requirements, modules include features to aid in placing devices, configuring devices, and network diagnostics.

Configure devices

You can configure XBee devices locally through serial commands (AT or API) or remotely through remote API commands. API devices can send configuration commands to set or read the configuration settings of any device in the network.

Network link establishment and maintenance

Build aggregate routes

In many applications it is necessary for many or all of the nodes in the network to transmit data to a central aggregator node. In a new DigiMesh network the overhead of these nodes discovering routes to the aggregator node can be extensive and taxing on the network. To eliminate this overhead, use the **AG** command to automatically build routes to an aggregate node in a DigiMesh network.

Send a unicast

To send a unicast, devices configured for Transparent mode (**AP** = **0**) must set their **DH**/**DL** registers to the MAC address of the node which they need to transmit to. In networks of Transparent mode devices which transmit to an aggregator node, it is necessary to set every device's **DH**/**DL** registers to the MAC address of the aggregator node. Use the **AG** command to set the **DH**/**DL** registers of all the nodes in a DigiMesh network to that of the aggregator node.

Use the AG command

Upon deploying a DigiMesh network, send the **AG** command on the desired aggregator node to cause all nodes in the network to build routes to the aggregator node. You can use the command to automatically update the **DH**/**DL** registers to match the MAC address of the aggregator node.

The **AG** command requires a 64-bit parameter. The parameter indicates the current value of the **DH**/**DL** registers on a device which should be replaced by the 64-bit address of the node sending the **AG** broadcast. If it is not desirable to update the **DH**/**DL** of the device receiving the **AG** broadcast, you can use the invalid address of 0xFFFE. API enabled devices output an Aggregate Addressing Update frame - 0x8E if they update their **DH**/**DL** address.

All devices that receive an **AG** broadcast update their routing table information to build a route to the sending device, regardless of whether or not their **DH**/**DL** address is updated. This routing information will be used for future transmissions of DigiMesh unicasts.

Example 1: To update the **DH**/**DL** registers of all modules in the network to be equal to the MAC address of an aggregator node with a MAC address of **0x0013a2004052c507** after network deployment the following technique could be employed:

1. Deploy all devices in the network with the default **DH**/**DL** of 0xFFFF.
2. Send an **ATAGFFFF** command on the aggregator node.

Following the preceding sequence would result in all of the nodes in the network which received the **AG** broadcast to have a **DH** of **0x0013a200** and a **DL** of **0x4052c507**. These nodes would have automatically built a route to the aggregator.

Example 2: To cause all nodes in the network to build routes to an aggregator node with a MAC address of **0x0013a2004052c507** without affecting the **DH**/**DL** of any nodes in the network, send the **AGFFFE** command on the aggregator node. This sends an **AG** broadcast to all nodes in the network.

All of the nodes will update their internal routing table information to contain a route to the aggregator node. None of the nodes update their **DH**/**DL** registers, because none of the registers are set to an address of **0xFFFE**.

Node replacement

You can also use the AG command to update the routing table and **DH**/**DL** registers in the network after a device is replaced, and you can update the **DH**/**DL** registers of nodes in the network.

- To update only the routing table information without affecting the **DH**/**DL** registers, use Example 2.
- To update the **DH**/**DL** registers of the network, use the method in the following example.

Example: Use the device with serial number 0x0013a2004052c507 as a network aggregator and replace it with a device with serial number 0x0013a200f5e4d3b2. Issue the AG0013a2004052c507 command on the new module. This causes all devices with a **DH**/**DL** register setting of 0x0013a2004052c507 to update their **DH**/**DL** register setting to the MAC address of the sending device (0x0013a200f5e4d3b2).

Place devices

For a network installation to be successful, installers must be able to determine where to place individual XBee devices to establish reliable links throughout the network.

RSSI indicators

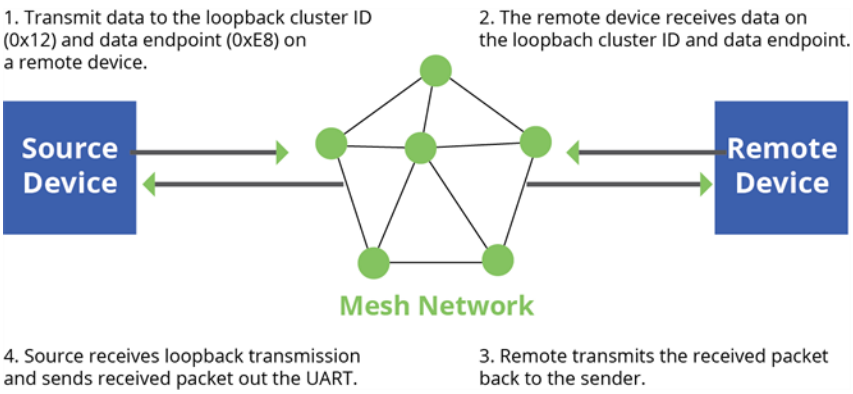
It is possible to measure the received signal strength on a device using the **DB** command. **DB** returns the RSSI value (measured in -dBm) of the last received packet. However, this number can be misleading in DigiMesh networks. The **DB** value only indicates the received signal strength of the last hop. If a transmission spans multiple hops, the **DB** value provides no indication of the overall transmission path, or the quality of the worst link; it only indicates the quality of the last link.

Determine the **DB** value in hardware using the RSSI/PWM device pin (pin 7). If you enable the RSSI PWM functionality (**P0** command), when the device receives data, it sets the RSSI PWM to a value based on the RSSI of the received packet (this value only indicates the quality of the last hop). You could connect this pin to an LED to indicate if the link is stable or not.

Test links in a network - loopback cluster

To measure the performance of a network, you can send unicast data through the network from one device to another to determine the success rate of several transmissions. To simplify link testing, the devices support a Loopback cluster ID (0x12) on the data endpoint (0xE8). The cluster ID on the data endpoint sends any data transmitted to it back to the sender.

The following figure demonstrates how you can use the Loopback cluster ID and data endpoint to measure the link quality in a mesh network.



The configuration steps for sending data to the loopback cluster ID depend on what mode the device is in. For details on setting the mode, see AP (API Mode). The following sections list the steps based on the device's mode.

Transparent operating mode configuration (AP = 0)

To send data to the loopback cluster ID on the data endpoint of a remote device:

1. Set the **CI** command to **0x12**.
2. Set the **SE** and **DE** commands to **0xE8** (default value).
3. Set the **DH** and **DL** commands to the address of the remote (**0** for the coordinator, or the 64-bit address of the remote).

After exiting Command mode, the device transmits any serial characters it received to the remote device, which returns those characters to the sending device.

API operating mode configuration (AP = 1 or AP = 2)

Send an Explicit Addressing Command frame - 0x11 using **0x12** as the cluster ID and **0xE8** as both the source and destination endpoint.

The remote device echoes back the data packets it receives to the sending device.

Test Link cluster

The primary difference between the Loopback cluster ID (0x12) and the Test Link cluster (0x14) is the number of hops. With the Loopback cluster you can verify that a route exists across one or more hops between any two nodes in the network.

With the Test Link cluster you can determine the signal strength between any two nodes without using intermediate nodes. If the two nodes are too far apart, you can expect 100% failure. If they are close together, you can expect 100% success.

When placing nodes, the following sequence could occur:

1. You can use the Loopback cluster to verify that a route exists between all nodes of interest; for example, between the aggregator and each of the other nodes.
2. If step 1 fails, you could perform a trace route on any pair of nodes that failed in step 1. The trace route indicates the failing link.
3. Once a weak link is identified, you can run Test Link to exercise that link to determine how strong and reliable it is.
4. Now that the problem has been isolated, you can place nodes apropriately to resolve the connectivity problems.

Device discovery

Network discovery

Use the network discovery command to discover all devices that have joined a network. Issuing the **ND** command sends a broadcast network discovery command throughout the network. All devices that receive the command send a response that includes:

- Device addressing information
- Node identifier string (see NI (Node Identifier))
- Other relevant information

You can use this command for generating a list of all module addresses in a network.

When a device receives the network discovery command, it waits a random time before sending a response. The device sets the maximum time delay on the **ND** sender with the **NT** command. The **ND** originator includes its **NT** setting in the transmission to provide a delay window for all devices in the network. Large networks may need to increase **NT** to improve network discovery reliability. The default **NT** value is .

Neighbor polling

Use the neighbor poll command to discover the modules which are immediate neighbors (within RF range) of a particular node. You can use this command to determining network topology and determining possible routes.

The device sends the command using the **FN** command. You can initiate the **FN** command locally on a node using AT command mode or by using a local AT command request frame. You can also initiate the command remotely by sending the target node an **FN** command using a remote AT command request API frame.

A node that executes an **FN** command sends a broadcast to all of its immediate neighbors. All devices that receive this broadcast send an RF packet to the node that initiated the **FN** command. In an instance where the device initiates the command remotely, it sends the responses directly to the node which sent the **FN** command to the target node. The device outputs the response packet on the initiating radio in the same format as a network discovery frame.

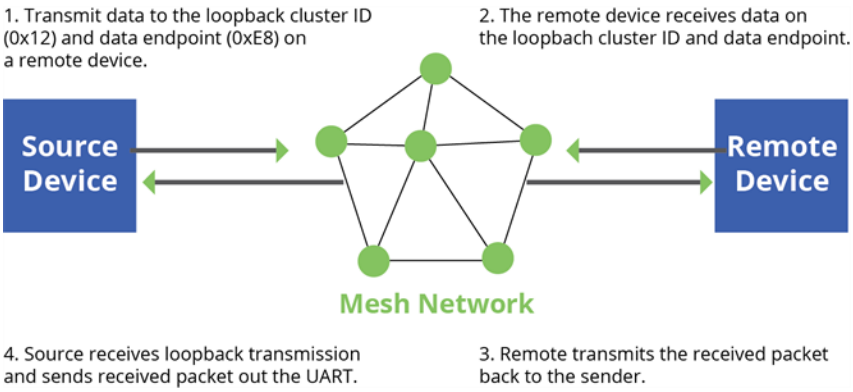
Link reliability

To install a successful mesh network, you must be able to determine where to place individual XBee devices to establish reliable links throughout the mesh network.

Network link testing

To determine the success rate of many transmissions, send unicast data through the network from one device to another to measure the performance of the mesh network.

To simplify link testing, the modules support a loopback cluster ID (0x12) on the data endpoint (0xE8). The device transmits any data sent to this cluster ID on the data endpoint back to the sender as illustrated in the following figure:



The configuration steps to send data to the loopback cluster ID depend on the AP setting.

AT configuration (AP=0)

To send data to the loopback cluster ID on the data endpoint of a remote device, set the **CI** command value to 0x12. Set the **SE** and **DE** commands set to 0xE8 (default value). Set the **DH** and **DL** commands

to the address of the remote. After exiting command mode, the source device transmits any received serial characters to the remote device, and returned to the sender.

API configuration (AP=1 or AP=2)

Send an Explicit Addressing Command API frame (0x11) using 0x12 as the cluster ID and 0xE8 as the source and destination endpoint. The remote device echoes any data packets it receives to the sender.

Link testing between adjacent devices

To test the quality of a link between two adjacent nodes in a network, use the Test Link Request Cluster ID send a number of test packets between any two nodes in a network.

Initiate a link test using an Explicit TX Request frame. Address the command frame to the Test Link Request Cluster ID (0x0014) on destination endpoint 0xE6 on the device to execute the test link. The Explicit TX Request frame contains a 12 byte payload with the following format:

Number of bytes	Field name	Description
8	Destination address	The address the device tests its link with.
2	Payload size	The size of the test packet. Use the MP command to query the maximum payload size for this device.
2	Iterations	The number of packets to send. Use a number between 1 and 4000.

After completing the transmissions of the test link packets, the executing device sends the following data packet to the requesting device's Test Link Result Cluster (0x0094) on endpoint (0xE6). If the requesting device is operating in API mode, the device outputs the following information as an API Explicit RX Indicator Frame:

Number of bytes	Field name	Description
8	Destination address	The address where the device tested its link.
2	Payload size	The size of the test packet sent to test the link.
2	Iterations	The number of packets sent.
2	Success	The number of packets successfully acknowledged.
2	Retries	The total number of MAC retries to transfer all the packets.
1	Result	0x00 - command was successful. 0x03 - invalid parameter used.
1	RR	The maximum number of MAC retries allowed.
1	maxRSSI	The strongest RSSI reading observed during the test.

Number of bytes	Field name	Description
1	minRSSI	The weakest RSSI reading observed during the test.
1	avgRSSI	The average RSSI reading observed during the test.

Example

Suppose that the link between device A (**SH**/**SL** = 0x0013a20040521234) and device B (**SH**/**SL**=0x0013a2004052abcd) is being tested by transmitting 1,000 40 byte packets. Send the following API packet to the serial interface of the device outputting the results, device C. Note that device C can be the same device as device A or B (Whitespace delineates fields and bold text is the payload portion of the packet):

7E 0020 11 01 0013A20040521234 FFFE E6 E6 0014 C105 00 00 0013A2004052ABCD 0028 03E8 EB

And the following is a possible packet returned:

7E 0027 91 0013A20040521234 FFFE E6 E6 0094 C105 00 0013A2004052ABCD 0028 03E8 03E7 0064 00 0A 50 53 52 9F

(999 out of 1000 packets successful, 100 retries used, RR=10, maxRSSI= **-** 80 dBm, minRSSI= **-** 83 dBm, avgRSSI= **-** 82 dBm)

If the result field is not equal to zero then an error occurred. Ignore the other fields in the packet. If the Success field is equal to zero then ignore the RSSI fields.

Trace routing

Determining the route a DigiMesh unicast takes to its destination is useful when setting up a network or diagnosing problems within a network. Use the Trace Route API option of Tx Request Packets to transmit routing information packets to the originator of a DigiMesh unicast by the intermediate nodes. For a description of the API frames, see API operating mode.

When a unicast is sent with the Trace Route API option enabled, the unicast is sent to its destination radios which forward the unicast to its eventual destination and transmit a Route Information (RI) packet back along the route to the unicast originator. For more information, see API operating mode.

Example:

Suppose you unicast a data packet with the trace route enabled from radio A to radio E, through radios B, C, and D. The following sequence occurs:

- After the successful MAC transmission of the data packet from A to B, A outputs an RI Packet indicating that the transmission of the data packet from A to E was successfully forwarded one hop from A to B.
- After the successful MAC transmission of the data packet from B to C, B transmits a RI Packet to A. Then, A outputs this RI packet out its serial interface.
- After the successful MAC transmission of the data packet from C to D, C transmits a RI Packet to A (through B). Then, A outputs this RI packet out its serial interface.
- After the successful MAC transmission of the data packet from D to E, D transmits an RI Packet to A (through C and B). Then, A outputs this RI packet out its serial interface.

Route Information packets are not guaranteed to arrive in the same order as the unicast packet took. It is also possible Route Information packets that are transferred on a weak route to fail before arriving at the unicast originator.

Because of the large number of Route Information packets that can be generated by a unicast with Trace Route enabled, we suggest that the Trace Route option only be used for occasional diagnostic purposes and not for normal operations.

NACK messages

Transmit Request (0x10 and 0x11) frames contain a negative-acknowledge character (NACK) API option (Bit 2 of the Transmit Options field).

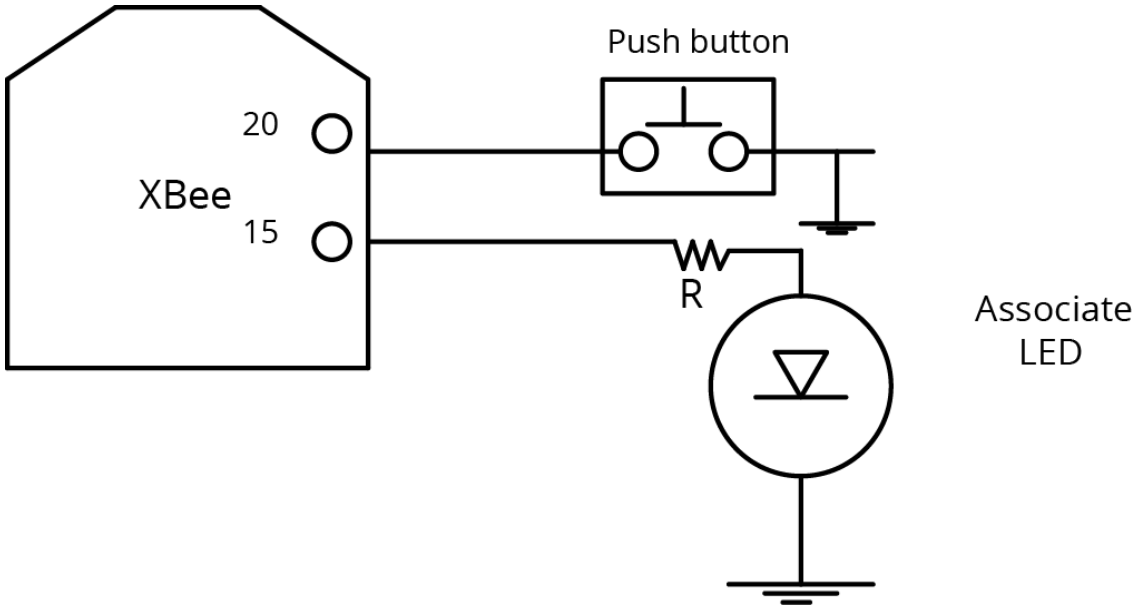
If you use this option when transmitting data, when a MAC acknowledgment failure occurs on one of the hops to the destination device, the device generates a Route Information Packet (0x8D) frame and sends it to the originator of the unicast.

This information is useful because it allows you to identify and repair marginal links.

Commissioning pushbutton and associate LED

XBee devices support a set of commissioning pushbutton and LED behaviors to aid in device deployment and commissioning. These include the commissioning push button definitions and associate LED behaviors. The following features can be supported in hardware:

TH RF Module



A pushbutton and an LED can be connected to the XBee SX 868 RF Module pins 33 and 28 (SMT), or pins 20 and 15 (TH) respectively to support the commissioning pushbutton and associate LED functionalities.

Commissioning pushbutton

The commissioning pushbutton definitions provide a variety of simple functions to help with deploying devices in a network. Enable the commissioning button functionality on pin 20 by setting the **D0** command to 1 (enabled by default).

Button Presses	Sleep configuration and sync status	Action
1	Not configured for sleep	Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission blink their Associate LED rapidly for 1 second. All API devices that receive this transmission send a Node Identification frame out their serial interface (API ID 0x95).
1	Configured for synchronous sleep	Wakes the module for 30 seconds. Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission blink their Associate LED rapidly for 1 second. All API devices that receive this transmission send a Node Identification frame out their serial interface (API ID 0x95).
1	Configured for synchronous sleep	Wakes the module for 30 seconds (or until the synchronized network goes to sleep). Queues a Node Identification broadcast transmission sent at the beginning of the next network wake cycle. All devices receiving this transmission blink their Associate LEDs rapidly for 1 second. All API devices that receive this transmission will send a Node Identification frame out their serial interface (API ID 0x95).
2	Not configured for synchronous sleep	No effect.
2	Configured for synchronous sleep	Causes a node configured with sleeping router nomination enabled (see the **SO** command in Sleep modes to immediately nominate itself as the network sleep coordinator.
4	Any	Issues an ATRE to restore module parameters to default values.

Use the **CB** command to simulate button presses in the software. Issue a **CB** command with a parameter set to the number of button presses you want executee. For example, sending **CB1** executes the actions associated with a single button press.

The node identification frame is similar to the node discovery response frame; it contains the device's address, node identifier string (**NI** command), and other relevant data. All API devices that receive the node identification frame send it out their serial interface as an API Node Identification Indicator frame (0x95).

Associate LED

The Associate pin (pin 15) provides an indication of the device's sleep status and diagnostic information. To take advantage of these indications, connect an LED to the Associate pin.

To enable the Associate LED functionality, set the **D5** command to 1; it is enabled by default. If enabled, the Associate pin is configured as an output. This section describes the behavior of the pin.

Use the **LT** command to override the blink rate of the Associate pin. If you set **LT** to 0, the device uses the default blink time: 500 ms for a sleep coordinator, 250 ms otherwise.

The following table describes the Associate LED functionality.

Sleep mode	LED status	Meaning
0	On, blinking	The device has power and is operating properly
1, 4, 5	Off	The device is in a low power mode
1, 4, 5	On, blinking	The device has power, is awake and is operating properly
7	On, solid	The network is asleep, or the device has not synchronized with the network, or has lost synchronization with the network
7, 8	On, slow blinking (500 ms blink time)	The device is acting as the network sleep coordinator and is operating properly
7, 8	On, fast blinking (250 ms blink time)	The device is properly synchronized with the network
8	Off	The device is in a low power mode
8	On, solid	The device has not synchronized or has lost synchronization with the network

Diagnostics support

The Associate pin works with the Commissioning Pushbutton to provide additional diagnostic behaviors to aid in deploying and testing a network. If you press the Commissioning Pushbutton once, the device transmits a broadcast Node Identification Indicator (0x95) frame at the beginning of the next wake cycle if the device is sleep compatible, or immediately if the device is not sleep compatible. If you enable the Associate LED functionality using the **D5** command, a device that receives this transmission blinks its Associate pin rapidly for one second.

I/O line monitoring

I/O samples

The XBee SX 868 RF Module supports both analog input and digital I/O line modes on several configurable pins.

Pin configurations

The following table provides typical parameters for the pin configuration commands (**D0** - **D9**, **P0** - **P2**).

Pin configuration commands include the following parameters:

Pin command parameter	Description
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionality
2	Analog input (A/D pins) or PWM output (PWM pins)
3	Digital input, monitored
4	Digital output, low
5	Digital output, high
7	Alternate functionality, where applicable

The following table provides the pin configurations when you set the configuration command for a particular pin.

Device pin name	Device pin number	Configuration command
DIO12	5	**P2**
PWM0 / RSSI / DIO10	7	**P0**
PWM1 / DIO11	8	**P1**
$\overline{\text{DTR}}$ / SLEEP_RQ / DIO8	10	**D8**
DIO4	24	**D4**
$\overline{\text{CTS}}$ / DIO7	25	**D7**
ON/$\overline{\text{SLEEP}}$ / DIO9	26	**D9**
ASSOC / AD5 / DIO5	15	**D5**
$\overline{\text{RTS}}$ / DIO6	29	**D6**
AD3 / DIO3	30	**D3**
AD2 / DIO2	31	**D2**
AD1 / DIO1	32	**D1**
AD0 / DIO0 / Commissioning Pushbutton	33	**D0**

Use the **PR** command to enable internal pull up/down resistors for each digital input. Use the **PD** command to determine the direction of the internal pull up/down resistor.

Queried sampling

You can use the **IS** command to query the current state of all digital input and ADC lines on the device. If no inputs are defined, the command returns with an ERROR.

Field	Name	Description
1	Sample sets	Number of sample sets in the packet. Always set to 1.
2	Digital channel mask	Indicates which digital I/O lines have sampling enabled. Each bit corresponds to one digital I/O line on the device. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 4 = DIO4 bit 5 = ASSOC/DIO5 bit 6 = RTS/DIO6 bit 7 = CTS/GPIO7 bit 8 = DTR / SLEEP_RQ / DIO8 bit 9 = ON_SLEEP / DIO9 bit 10 = RSSI/DIO10 bit 11 = PWM/DIO11 bit 12 = CD/DIO12 For example, a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital I/O.
1	Analog channel mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 4 = AD4/DIO4 bit 5 = ASSOC/AD5/DIO5
Variable	Sampled data set	If you enable any digital I/O lines, the first two bytes of the data set indicate the state of all enabled digital I/O. Only digital channels that you enable in the Digital channel mask bytes have any meaning in the sample set. If do not enable any digital I/O on the device, it omits these two bytes. Following the digital I/O data (if there is any), each enabled analog channel returns two bytes. The data starts with AIN0 and continues sequentially for each enabled analog input channel up to AIN5.

If you issue the **IS** command using a a local or remote AT Command API frame, then the device returns an AT Command Response (0x88) frame with the I/O data included in the command data portion of the packet.

Example	Sample AT response
0x01	[1 sample set]
0x0C0C	[Digital Inputs: DIO 2, 3, 10, 11 enabled]

Example	Sample AT response
0x03	[Analog Inputs: A/D 0, 1 enabled]
0x0408	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0	[Analog input: ADIO 0 = 0x3D0]
0x0124	[Analog input: ADIO 1 =0x120]

Periodic I/O sampling

Periodic sampling allows a device to take an I/O sample and transmit it to a remote device at a periodic rate. Use the **IR** command to set the periodic sample rate.

- To disable periodic sampling, set **IR** to **0**.
- For all other **IR** values, the firmware samples data when **IR** milliseconds elapse and the sample data transmits to a remote device.

The **DH** and **DL** commands determine the destination address of the I/O samples.

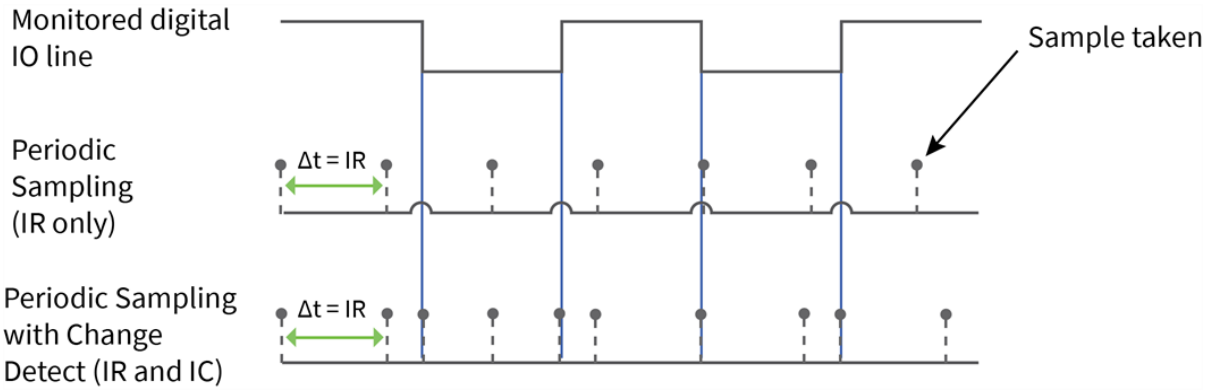
Only devices with API operating mode enabled send I/O data samples out their serial interface. Devices that are in Transparent mode (**AP** = **0**) discard the I/O data samples they receive.

A device with sleep enabled transmits periodic I/O samples at the **IR** rate until the **ST** time expires and the device can resume sleeping.

Detect digital I/O changes

You can configure devices to transmit a data sample immediately whenever a monitored digital I/O pin changes state. The **IC** command is a bitmask that you use to set which digital I/O lines to monitor for a state change. If you set one or more bits in **IC**, the device transmits an I/O sample as soon it observes a state change in one of the monitored digital I/O lines using edge detection.

The figure below shows how I/O change detection can work with periodic sampling.



Enabling edge detection forces an immediate sample of all monitored digital I/O lines if any digital I/O lines change state.

Note Use caution when combining Change Detect sampling with sleep modes. **IC** only causes a sample to be generated if the change takes place during a wake period. If the device is sleeping when the digital input transition occurs, then no change is detected and an I/O sample is not generated.

Use **IR** in conjunction with **IC** in this instance, since **IR** generates an I/O sample upon wakeup and ensures that the change is properly observed.

I/O line passing

You can configure XBee SX 868 RF Modules to perform analog and digital line passing. When a device receives an RF I/O sample data packet, you can set up the receiving device to update any enabled outputs (PWM and DIO) based on the data it receives.

Digital I/O lines are mapped in pairs; pins configured as digital input on the transmitting device affect the corresponding digital output pin on the receiving device. For example: DI5 (pin 25) can only update DO5 (pin 25).

For Analog Line Passing, the XBee SX 868 RF Module has two PWM output pins that simulate the voltage measured by the ADC lines AD0 and AD1. For example, when configured as an ADC, AD0 (pin 33) updates PWM0 (pin 7); AD1 (pin 32) updates PWM1 (pin 8).

The default setup is for outputs to not be updated. Instead, a device sends I/O sample data out the serial interface if the device is configured for API mode (**AP** = 1 or 2). You can use the **IU** command to disable sample data output.

To enable updating the outputs, set the **IA** (I/O Input Address) parameter with the address of the device that has the appropriate inputs enabled. This effectively binds the outputs to a particular device's input. This does not affect the ability of the device to receive I/O line data from other devices - only its ability to update enabled outputs. Set the **IA** parameter to 0xFFFF (broadcast address) to set up the device to accept I/O data for output changes from any device on the network.

For line passing to function, the device configured with inputs must generate sample data.

When outputs are changed from their non-active state, the device can be setup to return the output level to its non-active state. The timers are set using the **T*n*** (**D*n*** Output Timer) and **PT** (PWM Output Timeout) commands. The timers are reset every time the device receives a valid I/O sample packet with a matching **IA** address. You can adjust the **IC** (Change Detect) and **IR** (Sample Rate) parameters on the transmitting device to keep the outputs set to their active output if the system needs more time than the timers can handle.

Configuration example

As an example for a simple digital and analog link, you could set a pair of RF devices as follows:

Command	Description	Device A	Device B
SH	Serial Number High	0x0013A200	0x0013A200
SL	Serial Number Low	0x12345678	0xABCDABCD
DH	Destination High	0x0013A200	0x00000000
DL	Destination Low	0xABCDABCD	0x0000FFFF (broadcast)
IA	I/O Input Address	0x0013A200ABCDABCD	0x0013A20012345678
IR	Sample Rate	0x7D0 (2 seconds)	0 (disabled)
IC	DIO Change Detect	0 (disabled)	0x1000 (DIO3 only)
D1	DIO1/AD1	2 : ADC input	N/A

Command	Description	Device A	Device B
P1	DIO11/PWM1	N/A	2: PWM1 output
PT	PWM Output Timeout	N/A	0x1E (3 seconds)
D2	DIO2/AD2	3: Digital input	5: Digital output, HIGH
D3	DIO3/AD3	5: Digital output, HIGH	3: Digital input
T3	DIO3 Timeout	0x64 (10 seconds)	N/A

Command	Description	Device A	Device B
SH	Serial Number High	0x0013A200	0x0013A200
SL	Serial Number Low	0x12345678	0xABCDABCD
DH	Destination High	0x0013A200	0x00000000
DL	Destination Low	0xABCDABCD	0x0000FFFF (broadcast)
IA	I/O Input Address	0x0013A200ABCDABCD	0x0013A20012345678
IR	Sample Rate	0x7D0 (2 seconds)	0 (disabled)
IC	DIO Change Detect	0 (disabled)	0x8 (DIO3 only)
D1	DIO1/AD1	2 : ADC input	N/A
P1	DIO11/PWM1	N/A	2: PWM1 output
PT	PWM Output Timeout	N/A	0x1E (3 seconds)
D2	DIO2/AD2	3: Digital input	5: Digital output, HIGH
D3	DIO3/AD3	5: Digital output, HIGH	3: Digital input
T3	DIO3 Timeout	0x64 (10 seconds)	N/A

In the example, both devices have I/O Line Passing enabled with appropriate inputs and outputs configured. The **IA** parameter determines which device on the network is allowed to affect the device's outputs.

Device A takes a periodic sample of all I/O lines every two seconds and transmits it as a unicast transmission to the address defined by **DH** and **DL** (in this case, Device B). Device B does not periodically sample, instead it monitors DIO3 for a binary change. When it detects a change on that pin, it generates a sample and transmits it as a broadcast to all devices on the network.

When Device B receives a sample packet from Device A:

- DIO2 on Device B outputs the state of DIO2 from Device A.
- PWM1 outputs a duty cycle equivalent to the analog voltage read on AD1 of Device A.
- A PWM timeout has been set to three seconds; if no sample is received, PWM1 returns to 0 V after this period.

When Device A receives a sample packet from Device B:

- DIO3 on Device A outputs the state of DIO3 from Device B.
- A DIO3 timeout has been set to 10 seconds; if no sample is received, DIO3 reverts to a HIGH state after this period.

Note By default, all Digital I/O lines have internal pull-up resistors enabled with the **PR** command. This causes inputs to float high. You can use the **PD** command to change the direction of the internal pull-up/down resistors. The XBee SX 868 RF Module uses an internal reference voltage of 2.5 V for ADC lines, but you can use the **AV** command to set it to 1.25 VDC.

General Purpose Flash Memory

XBee SX 868 RF Modules provide 61 2048-byte blocks of flash memory that an application can read and write to. This memory provides a non-volatile data storage area that an application uses for many purposes. Some common uses of this data storage include:

- Storing logged sensor data
- Buffering firmware update data for a host microcontroller
- Storing and retrieving data tables needed for calculations performed by a host microcontroller

The General Purpose Memory (GPM) is also used to store a firmware update file for over-the-air firmware updates of the device itself.

Access General Purpose Flash Memory

To access the GPM of a target node locally or over-the-air, send commands to the MEMORY_ACCESS cluster ID (0x23) on the DIGI_DEVICE endpoint (0xE6) of the target node using explicit API frames. For a description of Explicit API frames, see Operate in API mode.

To issue a GPM command, format the payload of an explicit API frame as follows:

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	Specific GPM commands are described in detail in the topics that follow.
1	1	GPM_OPTIONS	Command-specific options.
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM.
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block.
6	2*	GPM_NUM_BYTES	The number of bytes in the GPM_DATA field, or in the case of a READ, the number of bytes requested.
8	varies	GPM_DATA	
* Specify multi-byte parameters with big-endian byte ordering.			

When a device sends a GPM command to another device via a unicast, the receiving device sends a unicast response back to the requesting device's source endpoint specified in the request packet. It

does not send a response for broadcast requests. If the source endpoint is set to the DIGI_DEVICE endpoint (0xE6) or Explicit API mode is enabled on the requesting device, then the requesting node outputs a GPM response as an explicit API RX indicator frame (assuming it has API mode enabled).

The format of the response is similar to the request packet:

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	This field is the same as the request field.
1	1	GPM_STATUS	Status indicating whether the command was successful.
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM.
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block.
6	2*	GPM_NUM_BYTES	The number of bytes in the GPM_DATA field.
8	varies	GPM_DATA	
* Specify multi-byte parameters with big-endian byte ordering.			

General Purpose Flash Memory commands

This section provides information about commands that interact with GPM:

PLATFORM_INFO_REQUEST (0x00)

A PLATFORM_INFO_REQUEST frame can be sent to query details of the GPM structure.

Field name	Command-specific description
GPM_CMD_ID	Should be set to PLATFORM_INFO_REQUEST (0x00).
GPM_OPTIONS	This field is unused for this command. Set to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	No data bytes should be specified for this command.

PLATFORM_INFO (0x80)

When a PLATFORM_INFO_REQUEST command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to PLATFORM_INFO (0x80).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Indicates the number of GPM blocks available.
GPM_START_INDEX	Indicates the size, in bytes, of a GPM block.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

A PLATFORM_INFO_REQUEST sent to a device with a serial number of 0x0013a200407402AC should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 00 00 0000 0000 0000 24

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 80 00 0077 0200 0000 EB

ERASE (0x01)

The ERASE command erases (writes all bits to binary 1) one or all of the GPM flash blocks. You can also use the ERASE command to erase all blocks of the GPM by setting the GPM_NUM_BYTES field to 0.

Field name	Command-specific description
GPM_CMD_ID	Should be set to ERASE (0x01).
GPM_OPTIONS	There are currently no options defined for the ERASE command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be erased. When erasing all GPM blocks, this field is ignored (set to 0).
GPM_START_INDEX	The ERASE command only works on complete GPM blocks. The command cannot be used to erase part of a GPM block. For this reason GPM_START_INDEX is unused (set to 0).
GPM_NUM_BYTES	Setting GPM_NUM_BYTES to 0 has a special meaning. It indicates that every flash block in the GPM should be erased (not just the one specified with GPM_BLOCK_NUM). In all other cases, the GPM_NUM_BYTES field should be set to the GPM flash block size.
GPM_DATA	No data bytes are specified for this command.

ERASE_RESPONSE (0x81)

When an ERASE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to ERASE_RESPONSE (0x81).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

To erase flash block 42 of a target radio with serial number of 0x0013a200407402ac format an ERASE packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 01 00 002A 0000 0800 31

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 81 00 002A 0000 0000 39

WRITE (0x02) and ERASE_THEN_WRITE (0x03)

The WRITE command writes the specified bytes to the GPM location specified. Before writing bytes to a GPM block it is important that the bytes have been erased previously. The ERASE_THEN_WRITE command performs an ERASE of the entire GPM block specified with the GPM_BLOCK_NUM field prior to doing a WRITE.

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE (0x02) or ERASE_THEN_WRITE (0x03).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be written.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be written.
GPM_NUM_BYTES	Set to the number of bytes specified in the GPM_DATA field. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. The maximum payload size can be queried with the **NP** command.
GPM_DATA	The data to be written.

WRITE _RESPONSE (0x82) and ERASE_THEN_WRITE_RESPONSE (0x83)

When a WRITE or ERASE_THEN_WRITE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE_RESPONSE (0x82) or ERASE_THEN_WRITE_RESPONSE (0x83)
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time
GPM_BLOCK_NUM	Matches the parameter passed in the request frame
GPM_START_INDEX	Matches the parameter passed in the request frame
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0
GPM_DATA	No data bytes are specified for these commands

Example

To write 15 bytes of incrementing data to flash block 22 of a target radio with serial number of 0x0013a200407402ac a WRITE packet should be formatted as follows (spaces added to delineate fields):

7E 002B 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 02 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C5

Assuming all transmissions were successful and that flash block 22 was previously erased, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 82 00 0016 0000 0000 4C

READ (0x04)

You can use the READ command to read the specified number of bytes from the GPM location specified. Data can be queried from only one GPM block per command.

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ (0x04).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be read.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be read.

Field name	Command-specific description
GPM_NUM_BYTES	Set to the number of data bytes to be read. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. You can query the maximum payload size with the **NP** AT command.
GPM_DATA	No data bytes should be specified for this command.

READ_RESPONSE (0x84)

When a READ command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ_RESPONSE (0x84).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field.
GPM_DATA	The bytes read from the GPM block specified.

Example

To read 15 bytes of previously written data from flash block 22 of a target radio with serial number of 0x0013a200407402ac a READ packet should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 04 00 0016 0000 000F 3B

Assuming all transmissions were successful and that flash block 22 was previously written with incrementing data, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 0029 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 84 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C3

FIRMWARE_VERIFY (0x05) and FIRMWARE_VERIFY_AND_INSTALL(0x06)

Use the FIRMWARE_VERIFY and FIRMWARE_VERIFY_AND_INSTALL commands when remotely updating firmware on a device. For more inoformation about firmware updates. These commands check if the GPM contains a valid over-the-air update file. For the FIRMWARE_VERIFY_AND_INSTALL command, if the GPM contains a valid firmware image then the device resets and begins using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY (0x05) or FIRMWARE_ VERIFY_AND_INSTALL (0x06)
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

FIRMWARE_VERIFY_RESPONSE (0x85)

When a FIRMWARE_VERIFY command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_RESPONSE (0x85)
GPM_STATUS	A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. A 0 in the least significant bit indicates the GPM does contain a valid firmware image. All other bits are reserved at this time.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86)

When a FIRMWARE_VERIFY_AND_INSTALL command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame only if the GPM memory does not contain a valid image. If the image is valid, the device resets and begins using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_AND_INSTALL_RESPONSE (0x86).
GPM_STATUS	A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. All other bits are reserved at this time.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.

Field name	Command-specific description
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command.

Example

To verify a firmware image previously loaded into the GPM on a target device with serial number 0x0013a200407402ac, format a FIRMWARE_VERIFY packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 05 00 0000 0000 0000 1F

Assuming all transmissions were successful and that the firmware image previously loaded into the GPM is valid, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 85 00 0000 0000 0000 5F

Work with flash memory

When working with the General Purpose Memory, observe the following limitations:

- Flash memory write operations are only capable of changing binary 1s to binary 0s. Only the erase operation can change binary 0s to binary 1s. For this reason, you should erase a flash block before performing a write operation.
- When performing an erase operation, you must erase the entire flash memory block—you cannot erase parts of a flash memory block.
- Flash memory has a limited lifetime. The flash memory on which the GPM is based is rated at 20,000 erase cycles before failure. Take care to ensure that the frequency of erase/write operations allows for the desired product lifetime. Digi's warranty does not cover products that have exceeded the allowed number of erase cycles.
- Over-the-air firmware upgrades erase the entire GPM. Any user data stored in the GPM will be lost during an over-the-air upgrade.

Over-the-air firmware updates

There are two methods of updating the firmware on the device. You can update the firmware locally with XCTU using the device's serial port interface. You can also update firmware using the device's RF interface (over-the-air updating.)

The over-the-air firmware update method provided is a robust and versatile technique that you can tailor to many different networks and applications. OTA updates are reliable and minimize disruption of normal network operations.

In the following sections, we refer to the node that will be updated as the target node. We refer to the node providing the update information as the source node. In most applications the source node is locally attached to a computer running update software.

There are three phases of the over-the-air update process:

1. Distribute the new application
2. Verify the new application
3. Install the application

Distribute the new application

The first phase of performing an over-the-air update on a device is transferring the new firmware file to the target node. Load the new firmware image in the target node's GPM prior to installation. XBee SX 868 RF Modules use an encrypted binary (.ebin) file for both serial and over-the-air firmware updates. These firmware files are available on the Digi Support website and via XCTU.

Send the contents of the .ebin file to the target device using general purpose memory WRITE commands. Erase the entire GPM prior to beginning an upload of an .ebin file. The contents of the .ebin file should be stored in order in the appropriate GPM memory blocks. The number of bytes that are sent in an individual GPM WRITE frame is flexible and can be catered to the user application.

Example

The example firmware version has an .ebin file of 55,141 bytes in length. Based on network traffic, we determine that sending a 128 byte packet every 30 seconds minimizes network disruption. For this reason, you would divide and address the .ebin as follows:

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
0	0	128	0 to 127
0	128	128	128 to 255
0	256	128	256 to 383
0	384	128	384 to 511
1	0	128	512 to 639
1	128	128	640 to 767
-	-	-	-
-	-	-	-
-	-	-	-
107	0		54784 to 54911
107	128		54912 to 55039
107	256	101	55040 to 55140

Verify the new application

For an uploaded application to function correctly, every single byte from the .ebin file must be properly transferred to the GPM. To guarantee that this is the case, GPM VERIFY functions exist to ensure that all bytes are properly in place. The FIRMWARE_VERIFY function reports whether or not the uploaded data is valid. The FIRMWARE_VERIFY_AND_INSTALL command reports if the uploaded data is invalid. If the data is valid, it begins installing the application. No installation takes place on invalid data.

Install the application

When the entire .ebin file is uploaded to the GPM of the target node, you can issue a FIRMWARE_VERIFY_AND_INSTALL command. Once the target receives the command it verifies the .ebin file loaded in the GPM. If it is valid, then the device installs the new firmware. This installation process can take up to eight seconds. During the installation the device is unresponsive to both serial and RF communication. To complete the installation, the target module resets. AT parameter settings which have not been written to flash using the **WR** command will be lost.

Important considerations

The firmware upgrade process requires that the device resets itself. Write all parameters with the **WR** command before performing a firmware update. Packet routing information is also lost after a reset. Route discoveries are necessary for DigiMesh unicasts involving the updated node as a source, destination, or intermediate node.

Because explicit API Tx frames can be addressed to a local node (accessible via the SPI or UART) or a remote node (accessible over the RF port) the same process can be used to update firmware on a device in either case.

Software libraries

One way to communicate with the XBee SX 868 RF Module is by using a software library. The libraries available for use with the XBee SX 868 RF Module include:

- XBee Java library
- XBee Python library

The XBee Java Library is a Java API. The package includes the XBee library, its source code and a collection of samples that help you develop Java applications to communicate with your XBee devices.

The XBee Python Library is a Python API that dramatically reduces the time to market of XBee projects developed in Python and facilitates the development of these types of applications, making it an easy process.

Networking methods

This section explains the basic layers and the three networking methods available on the XBee SX 868 RF Modules, building from the simplest to the most complex.

Directed Broadcast/Repeater mode

In this broadcast mode, the exact transmission method is determined by the data rate of your device.

- In the 10k version, set the network in a repeater mode, where there is no route discovery. The transmission is sent out to the network and each device repeats the message to its neighboring devices. There is no route discovery in this method.
- In the 80k version of the device, the transmission is directed to a specific media access control (MAC) address, using a route discovered by a router device. In both methods, all transmissions are broadcast messages, not unicast messages.

Point to Point/Multipoint mode

In this mode, there is a permanent link between two endpoints. Switched point-to-point topologies are the basic model of conventional telephony. The value of a permanent point-to-point network is unimpeded communications between the two endpoints. The value of an on-demand point-to-point connection is proportional to the number of potential pairs of subscribers.

Permanent (dedicated)

One of the variations of point-to-point topology is a point-to-point communications channel that appears, to the user, to be permanently associated with the two endpoints. Within many switched telecommunications systems, it is possible to establish a permanent circuit. One example might be a telephone in the lobby of a public building that is programmed to ring only the number of a telephone dispatcher. “Nailing down” a switched connection saves the cost of running a physical circuit between the two points. The resources in such a connection can be released when it is no longer needed.

Switched

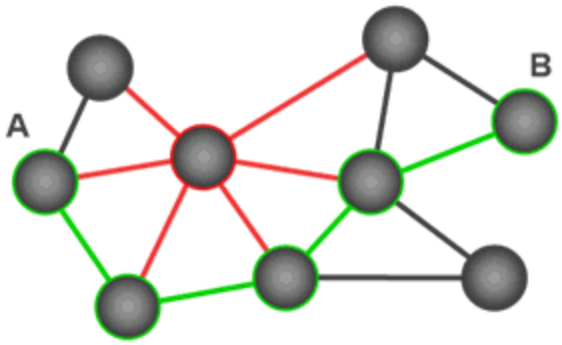
Using circuit-switching or packet-switching technologies, you can set up a point-to-point circuit dynamically and dropped when no longer needed.

DigiMesh networking

A mesh network is a topology in which each node in the network is connected to other nodes around it. Each node cooperates in transmitting information. Mesh networking provides these important benefits:

- **Routing**. With this technique, the message is propagated along a path by hopping from node to node until it reaches its final destination.
- **Ad-hoc network creation**. This is an automated process that creates an entire network of nodes on the fly, without any human intervention.
- **Self-healing**. This process automatically figures out if one or more nodes on the network is missing and reconfigures the network to repair any broken routes.
- **Peer-to-peer architecture**. No hierarchy and no parent-child relationships are needed.
- **Quiet protocol**. Routing overhead will be reduced by using a reactive protocol similar to AODV.
- **Route discovery**. Rather than maintaining a network map, routes will be discovered and created only when needed.
- **Selective acknowledgments**. Only the destination node will reply to route requests.

- **Reliable delivery**. Reliable delivery of data is accomplished by means of acknowledgments.
- **Sleep modes.** Low power sleep modes with synchronized wake are supported with variable sleep and wake times.



With mesh networking, the distance between two nodes does not matter as long as there are enough nodes in between to pass the message along. When one node wants to communicate with another, the network automatically calculates the best path.

A mesh network is also reliable and offers redundancy. For example, If a node can no longer operate because it has been removed from the network or because a barrier blocks its ability to communicate, the rest of the nodes can still communicate with each other, either directly or through intermediate nodes.

Note Mesh networks use more bandwidth for administration and therefore have less available for payloads.

DigiMesh feature set

DigiMesh contains the following features:

- **Self-healing**
  Any node may enter or leave the network at any time without causing the network as a whole to fail.
- **Peer-to-peer architecture**
  No hierarchy and no parent-child relationships are needed.
- **Quiet protocol**
  Routing overhead will be reduced by using a reactive protocol similar to AODV.
- **Route discovery**
  Rather than maintaining a network map, routes will be discovered and created only when needed.
- **Selective acknowledgments**
  Only the destination node will reply to route requests.
- **Reliable delivery**
  Reliable delivery of data is accomplished by means of acknowledgments.
- **Sleep modes**
  Low power sleep modes with synchronized wake are supported with variable sleep and wake times.

Networking concepts

This section provides information on configuring DigiMesh devices and network identifiers.

Device Configuration

You can configure DigiMesh devices to act as routers or end devices with the **CE** command. By default, all devices in a DigiMesh network act as routers. Devices configured as routers actively relay network unicast and broadcast traffic.

Network ID

DigiMesh networks are defined with a unique network identifier. Set the identifier using the **ID** command. For devices to communicate they must be configured with the same network identifier. The **ID** parameter allows multiple DigiMesh networks to co-exist on the same physical channel.

Data transmission and routing

This section provides information on data transmission, routing, throughput, and transmission timeouts.

Unicast addressing

When devices transmit using DigiMesh unicast, the network uses retries and acknowledgments (ACKs) for reliable data delivery. In a retry and acknowledgment scheme, for every data packet that a device sends, the receiving device must send an acknowledgment back to the transmitting device to let the sender know that the data packet arrived at the receiver. If the transmitting device does not receive an acknowledgment then it re-sends the packet. It sends the packet a finite number of times before the system times out.

The **MR** (Mesh Network Retries) parameter determines the number of mesh network retries. The sender device transmits RF data packets up to **MR** + 1 times across the network route, and the receiver transmits ACKs when it receives the packet. If the sender does not receive a network ACK within the time it takes for a packet to traverse the network twice, the sender retransmits the packet.

To send unicast messages while in Transparent operating mode, set the **DH** and **DL** on the transmitting device to match the corresponding **SH** and **SL** parameter values on the receiving device.

Broadcast addressing

All of the routers in a network receive and repeat broadcast transmissions. Broadcast transmissions do not use ACKs, so the sending device sends the broadcast multiple times. By default, the sending device sends a broadcast transmission four times. The transmissions become automatic retries without acknowledgments. This results in all nodes repeating the transmission four times as well.

In order to avoid RF packet collisions, the network inserts a random delay before each router relays the broadcast message. You can change this random delay time with the **NN** parameter.

Sending frequent broadcast transmissions can quickly reduce the available network bandwidth. Use broadcast transmissions sparingly.

The broadcast address is a 64 bit address with the lowest 16 bits set to 1. The upper bits are set to 0. To send a broadcast transmission:

- Set **DH** to 0.
- Set **DL** to 0xFFFF.

In API operating mode, this sets the destination address to 0x000000000000FFFF.

Routing

Devices within a mesh network determine reliable routes using a routing algorithm and table. The routing algorithm uses a reactive method derived from Ad-hoc On-demand Distance Vector (AODV). The device uses an associative routing table to map a destination node address with its next hop. By sending a message to the next hop address, the message reaches its destination or is forwarded to an intermediate router which routes the message on to its destination.

The device broadcasts a message with a broadcast address to all neighbors. All routers receiving the message rebroadcast the message **MT**+1 times and eventually the message reaches all corners of the network. Packet tracking prevents a node from resending a broadcast message more than **MT**+1 times.

Route discovery

Route discovery is a process that occurs when:

1. The source node does not have a route to the requested destination.
2. A route fails. This happens when the source node uses up its network retries without receiving an ACK.

Route discovery begins by the source node broadcasting a route request (RREQ). We call any router that receives the RREQ and is not the ultimate destination, an intermediate node.

Intermediate nodes may either drop or forward a RREQ, depending on whether the new RREQ has a better route back to the source node. If so, the node saves, updates and broadcasts the RREQ.

When the ultimate destination receives the RREQ, it unicasts a route reply (RREP) back to the source node along the path of the RREQ. It does this regardless of route quality and regardless of how many times it has seen an RREQ before.

This allows the source node to receive multiple route replies. The source node selects the route with the best round trip route quality, which it uses for the queued packet and for subsequent packets with the same destination address.

DigiMesh throughput

Throughput in a DigiMesh network can vary due to a number of variables, including:

- The number of hops.
- If you enable or disable encryption.
- Sleeping end devices.
- Failures and route discoveries.

The following table shows the results of this test with various configurations. All measurements were acquired by streaming 10,000 bytes of data in Transparent mode from the transmitter to the receiver at a baud rate of 115,200 b/s.

Configuration	Data throughput
Point to Point, encryption disabled	34.63 kb/s
Point to Point, encryption enabled	34.48 kb/s
Mesh unicast, one hop, encryption disabled	27.54 kb/s
Mesh unicast, one hop, encryption enabled	27.3 kb/s
Mesh unicast, three hops, encryption disabled	9.55 kb/s
Mesh unicast, three hops, encryption enabled	9.38 kb/s

The following table shows the latency (in ms) for 10000 Bytes at 115200 b/s serial data rate.

Configuration	Latency
Point to Point, encryption disabled	69.8 ms
Point to Point, encryption enabled	69.99 ms
Mesh, encryption disabled	94.31 ms
Mesh, encryption enabled	95.06 ms

Note We made the data throughput measurements by setting the serial interface rate to 115200 b/s, and measuring the time taken to send 10000 bytes from source to destination. During the test, no route discoveries or failures occurred.

Transmission timeouts

When a device in API operating mode receives a Transmit Request (0x10, 0x11) frame, or a device in Transparent operating mode meets the packetization requirements (**RO**, **RB**), the time required to route the data to its destination depends on:

- A number of configured parameters.
- Whether the transmission is a unicast or a broadcast.
- If the route to the destination address is known.

Timeouts or timing information is provided for the following transmission types:

- Broadcast transmission
- Unicast transmission on a known route
- Unicast transmission on an unknown route
- Unicast transmission on a broken route

Note The timeouts in this documentation are theoretical timeouts and are not precisely accurate. Your application should pad the calculated maximum timeouts by a few hundred milliseconds. When you use API operating mode, use Transmit Status frame - 0x8B as the primary method to determine if a transmission is complete.

Unicast one hop time

unicastOneHopTime is a building block of many of the following calculations. It represents the amount of time it takes to send a unicast transmission between two adjacent nodes. The amount of time depends on the parameter.

Transmit a broadcast

All of the routers in a network must relay a broadcast transmission.

The maximum delay occurs when the sender and receiver are on the opposite ends of the network.

The **NH** and **%H** parameters define the maximum broadcast delay as follows:

BroadcastTxTime = NH * NN * %8

Unless **BH** < **NH**, in which case the formula is:

BroadcastTxTime = BH * NN * %8

Transmit a unicast with a known route

When a device knows a route to a destination node, the transmission time is largely a function of the number of hops and retries. The timeout associated with a unicast assumes that the maximum number of hops is necessary, as specified by the **NH** command.

You can estimate the timeout in the following manner:

knownRouteUnicastTime=2*NH*MR*unicastOneHopTime

Transmit a unicast with an unknown route

If the transmitting device does not know the route to the destination, it begins by sending a route discovery. If the route discovery is successful, then the transmitting device transmits data. You can estimate the timeout associated with the entire operation as follows:

unknownRouteUnicastTime=BroadcastTxTime+
(**NH***unicastOneHopTime)+knownRouteUnicastTime

Transmit a unicast with a broken route

If the route to a destination node changes after route discovery completes, a node begins by attempting to send the data along the previous route. After it fails, it initiates route discovery and, when the route discovery finishes, transmits the data along the new route. You can estimate the timeout associated with the entire operation as follows:

brokenRouteUnicastTime=BroadcastTxTime+(NH*unicastOneHopTime)+
(2*knownRouteUnicastTime)

Modes

The XBee SX 868 RF Module is in Receive Mode when it is not transmitting data. The device shifts into the other modes of operation under the following conditions:

- Transmit mode (Serial data in the serial receive buffer is ready to be packetized)
- Sleep mode
- Command Mode (Command mode sequence is issued (not available when using the SPI port))

Transmit mode

When the device receives serial data and is ready to packetize it, the device attempts to transmit the serial data. The destination address determines which node(s) will receive and send the data.

In the following diagram, route discovery applies only to DigiMesh transmissions. Once route discovery establishes a route, the device transmits the data. If route discovery fails to establish a route, the device discards the packet.

For more information, see Data transmission and routing.

Receive mode

This is the default mode for the XBee SX 868 RF Module. The device is in Receive mode when it is not transmitting data. If a destination node receives a valid RF packet, the destination node transfers the data to its serial transmit buffer.

Command mode

Command mode is a state in which the firmware interprets incoming characters as commands. It allows you to modify the device's configuration using parameters you can set using AT commands. When you want to read or set any parameter of the XBee SX 868 RF Module using this mode, you have to send an AT command. Every AT command starts with the letters **AT** followed by the two characters that identify the command and then by some optional configuration values.

The operating modes of the XBee SX 868 RF Module are controlled by the AP (API Mode) setting, but Command mode is always available as a mode the device can enter while configured for any of the operating modes.

Command mode is available on the UART interface for all operating modes. You cannot use the SPI interface to enter Command mode.

Enter Command mode

To get a device to switch into Command mode, you must issue the following sequence: **+++** within one second. There must be at least one second preceding and following the **+++** sequence. Both the command character (**CC**) and the silence before and after the sequence (**GT**) are configurable. When the entrance criteria are met the device responds with **OK\r** on UART signifying that it has entered Command mode successfully and is ready to start processing AT commands.

If configured to operate in Transparent operating mode, when entering Command mode the XBee SX 868 RF Module knows to stop sending data and start accepting commands locally.

Note Do not press **Return** or **Enter** after typing **+++** because it interrupts the guard time silence and prevents you from entering Command mode.

When the device is in Command mode, it listens for user input and is able to receive AT commands on the UART. If **CT** time (default is 10 seconds) passes without any user input, the device drops out of Command mode and returns to the previous operating mode. You can force the device to leave Command mode by sending CN (Exit Command Mode).

You can customize the command character, the guard times and the timeout in the device's configuration settings. For more information, see CC (Command Sequence Character), CT (Command Mode Timeout) and GT (Guard Times).

Troubleshooting

Failure to enter Command mode is often due to baud rate mismatch. Ensure that the baud rate of the connection matches the baud rate of the device. By default, BD (Baud Rate) = **3** (9600 b/s).

There are two alternative ways to enter Command mode:

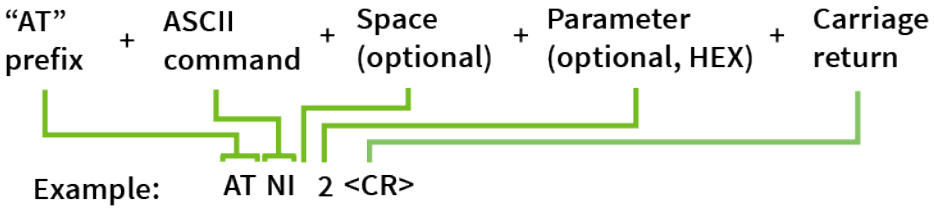
- A serial break for six seconds enters Command mode. You can issue the "break" command from a serial console, it is often a button or menu item.
- Asserting DIN (serial break) upon power up or reset enters Command mode. XCTU guides you through a reset and automatically issues the break when needed.

Both of these methods temporarily set the device's baud rate to 9600 and return an **OK** on the UART to indicate that Command mode is active. When Command mode exits, the device returns to normal operation at the baud rate that **BD** is set to.

Send AT commands

Once the device enters Command mode, use the syntax in the following figure to send AT commands. Every AT command starts with the letters **AT**, which stands for "attention." The AT is followed by two characters that indicate which command is being issued, then by some optional configuration values.

To read a parameter value stored in the device's register, omit the parameter field.



The preceding example changes NI (Node Identifier) to **My XBee**.

Multiple AT commands

You can send multiple AT commands at a time when they are separated by a comma in Command mode; for example, **ATNIMy XBee,AC<cr>**.

The preceding example changes the **NI (Node Identifier)** to **My XBee** and makes the setting active through AC (Apply Changes).

Parameter format

Refer to the list of AT commands for the format of individual AT command parameters. Valid formats for hexidecimal values include with or without a leading **0x** for example **FFFF** or **0xFFFF**.

Response to AT commands

When using AT commands to set parameters the XBee SX 868 RF Module responds with **OK<cr>** if successful and **ERROR<cr>** if not.

For devices with a file system:

ATAP1<cr>

OK<cr>

When reading parameters, the device returns the current parameter value instead of an **OK** message.

ATAP<cr>

1<cr>

Apply command changes

Any changes you make to the configuration command registers using AT commands do not take effect until you apply the changes. For example, if you send the **BD** command to change the baud rate, the actual baud rate does not change until you apply the changes. To apply changes:

1. Send AC (Apply Changes).
2. Send WR (Write).
   or:
3. Exit Command mode.

Make command changes permanent

Send a WR (Write) command to save the changes. **WR** writes parameter values to non-volatile memory so that parameter modifications persist through subsequent resets.

Send as RE (Restore Defaults) to wipe settings saved using **WR** back to their factory defaults.

Note You still have to use **WR** to save the changes enacted with **RE**.

Exit Command mode

1. Send CN (Exit Command Mode) followed by a carriage return.
   or:
2. If the device does not receive any valid AT commands within the time specified by CT (Command Mode Timeout), it returns to Transparent or API mode. The default Command mode timeout is 10 seconds.

For an example of programming the device using AT Commands and descriptions of each configurable parameter, see AT commands.

Sleep mode

Sleep modes allow the device to enter states of low power consumption when not in use. The XBee SX 868 RF Module supports both pin sleep (Sleep mode entered on pin transition) and cyclic sleep (device sleeps for a fixed time).

Sleep modes allow the device to enter states of low power consumption when not in use. XBee devices support both pin sleep, where the device enters sleep mode upon pin transition, and cyclic sleep, where the device sleeps for a fixed time. For more information, see Sleep modes.

Force UART operation

Condition

You configure a device with only the SPI enabled and no SPI master is available to access the SPI slave port

Solution

Use the following steps to recover the device to UART operation:

1. Hold the DIN/CONFIG low at reset time.
2. DIN/CONFIG forces a default configuration on the UART at 9600 baud and brings up the device in Command Mode on the UART port.

   You can send the appropriate commands to the device to configure it for UART operation.
3. If you write these parameters to the device, the module comes up with the UART enabled on the next reset.

Sleep modes

About sleep modes

A number of low-power modes exist to enable devices to operate for extended periods of time on battery power. Use the **SM** command to enable these sleep modes. The sleep modes are characterized as either:

- Asynchronous (**SM** = 1, 4, 5).
- Synchronous (**SM** = 7, 8).

Asynchronous modes

- Do not use asynchronous sleep modes in a synchronous sleeping network, and vice versa.
- Use the asynchronous sleep modes to control the sleep state on a device by device basis.
- Do not use devices operating in asynchronous sleep mode to route data.
- We strongly encourage you to set asynchronous sleeping devices as end-devices using the **CE** command. This prevents the node from attempting to route data.

Synchronous modes

Synchronous sleep makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time.

This forms a cyclic sleeping network.

- A device acting as a sleep coordinator sends a special RF packet called a sync message to synchronize nodes.
- To make a device in the network a coordinator, a node uses several resolution criteria.
- The sleep coordinator sends one sync message at the beginning of each wake period. The coordinator sends the sync message as a broadcast and every node in the network repeats it.
- You can change the sleep and wake times for the entire network by locally changing the settings on an individual device. The network uses the most recently set sleep settings.

Normal mode

Set **SM** to 0 to enter Normal mode.

Normal mode is the default sleep mode. If a device is in this mode, it does not sleep and is always awake.

Use mains-power for devices in Normal mode.

A device in Normal mode synchronizes to a sleeping network, but does not observe synchronization data routing rules; it routes data at any time, regardless of the network's wake state.

When synchronized, a device in Normal mode relays sync messages that sleep-compatible nodes generate, but does not generate sync messages itself.

Once a device in Normal mode synchronizes with a sleeping network, you can put it into a sleep-compatible sleep mode at any time.

Asynchronous pin sleep mode

Set **SM** to 1 to enter asynchronous pin sleep mode.

Pin sleep allows the device to sleep and wake according to the state of the $\overline{\text{SLEEP}}$_RQ pin (pin 10).

When you assert SLEEP_RQ (high), the device finishes any transmit or receive operations and enters a low-power state.

When you de-assert SLEEP_RQ (low), the device wakes from pin sleep.

Asynchronous cyclic sleep mode

Set **SM** to 4 to enter asynchronous cyclic sleep mode.

Cyclic sleep allows the device to sleep for a specific time and wake for a short time to poll.

If the device receives serial or RF data while awake, it extends the time before it returns to sleep by the specific amount the **ST** command provides. Otherwise, it enters sleep mode immediately.

The $\overline{\text{ON\_SLEEP}}$ line asserts (high) when the device wakes, and is de-asserted (low) when the device sleeps.

If you use the **D7** command to enable hardware flow control, the $\overline{\text{CTS}}$ pin asserts (low) when the device wakes and can receive serial data, and de-asserts (high) when the device sleeps.

If the sleeping node does not receive a poll response from the indirect messaging sleep coordinator, then it stays awake waiting for that poll for an extended period of time waiting for that poll response. This time is about 8 seconds if BR (RF Data Rate) is **0** and RR (Unicast Mac Retries) is **0x0A**.

Therefore, to save battery life it is important to configure an indirect messaging sleep coordinator (CE (Node Messaging Options) **1**) within range of the sleeping node. Also the sleeping node (**CE 6**) needs to point to the indirect messaging sleep coordinator with **DH**/**DL** set to the **SH**/**SL** of the indirect messaging sleep coordinator.

When SO (Sleep Options) bit 8 is set (**0x100**), the device stays awake for the maximum of **ST** and the minimum wake time. When **BR** is **0** and **RR** is **0x0A**, the minimum wake time is about 8 seconds, which may be more than **ST**.

Asynchronous cyclic sleep with pin wake up mode

Set **SM** to 5 to enter Asynchronous cyclic sleep with pin wake up mode.

This mode is a slight variation on (**SM** = 4) that allows the device to wake prematurely by asserting the SLEEP_RQ pin (pin 9). In (**SM** =5), the device wakes after the sleep period expires, or if a high-to-low transition occurs on the SLEEP_RQ pin.

Synchronous sleep support mode

Set **SM** to 7 to enter synchronous sleep support mode.

A device in synchronous sleep support mode synchronizes itself with a sleeping network but will not itself sleep. At any time, the device responds to new devices that are attempting to join the sleeping network with a sync message. A sleep support device only transmits normal data when the other devices in the sleeping network are awake. You can use sleep support devices as preferred sleep coordinator devices and as aids in adding new devices to a sleeping network.

Synchronous cyclic sleep mode

Set **SM** to 8 to enter synchronous cyclic sleep mode.

A device in synchronous cyclic sleep mode sleeps for a programmed time, wakes in unison with other nodes, exchanges data and sync messages, and then returns to sleep. While asleep, it cannot receive RF messages or receive data (including commands) from the UART port.

Generally, the network's sleep coordinator specifies the sleep and wake times based on its **SP** and **ST** settings. The device only uses these parameters at startup until the device synchronizes with the network.

When a device has synchronized with the network, you can query its sleep and wake times with the **OS** and **OW** commands respectively.

If **D9** = 1 (ON_SLEEP enabled) on a cyclic sleep node, the ON_SLEEP line asserts when the device is awake and de-asserts when the device is asleep.

If **D7** = 1, the device de-asserts CTS while asleep.

A newly-powered, unsynchronized, sleeping device polls for a synchronized message and then sleeps for the period that the **SP** command specifies, repeating this cycle until it synchronizes by receiving a sync message. Once it receives a sync message, the device synchronizes itself with the network.

Note Configure all nodes in a synchronous sleep network to operate in either synchronous sleep support mode or synchronous cyclic sleep mode. asynchronous sleeping nodes are not compatible with synchronous sleeping nodes.

Wake timer

In asynchronous cyclic sleep mode (**SM** = **4** or **SM** = **5**), if a device receives serial or RF data, it starts a sleep timer (time until sleep). Any data received serially or by RF link resets the timer. Use ST (Wake Time) to set the timer duration. While the device is awake, it sends regular poll requests to its parent to check for buffered data. If the RF data rate is 80 kb/s (**BR** = **1**), the poll occurs every 100 ms. Otherwise, (**BR** = **0**), the poll occurs every 300 ms. The device returns to sleep when the sleep timer expires.

Indirect messaging and polling

To enable reliable communication with sleeping devices, you can use the **CE** (Routing/Messaging Mode) command to enable indirect messaging and polling.

Indirect messaging

Indirect messaging is a communication mode designed for communicating with asynchronous sleeping devices. A device can enable indirect messaging by making itself an indirect messaging coordinator with the **CE** command. An indirect messaging coordinator does not immediately transmit a P2MP unicast when it is received over the serial port. Instead the device holds onto the data until it is requested via a poll. On receiving a poll, the indirect messaging coordinator sends a queued data packet (if available) to the requestor.

Because it is possible for a polling device to be eliminated, a mechanism is in place to purge unrequested data packets. If the coordinator holds an indirect data packet for an indirect messaging poller for more than 2.5 times its **SP** value, then the packet is purged. We suggest setting the **SP** of the coordinator to the same value as the highest **SP** time that exists among the pollers in the network. If the coordinator is in API mode, a TxStatus message is generated for a purged data packet with a status of 0x75 (**INDIRECT_MESSAGE_UNREQUESTED**).

An indirect messaging coordinator queues up as many data packets as it has buffers available. After the coordinator uses all of its available buffers, it holds transmission requests unprocessed on the serial input queue. After the serial input queue is full, the device de-asserts CTS (if hardware flow control is enabled). After receiving a poll or purging data from the indirect messaging queue the buffers become available again.

Indirect messaging only functions with P2MP unicast messages. Indirect messaging has no effect on P2MP broadcasts, directed broadcasts, repeater packets, or DigiMesh packets. These messages are sent immediately when received over the serial port and are not put on the indirect messaging queue.

Polling

Polling is the automatic process by which a node can request data from an indirect messaging coordinator. To enable polling on a device, configure it as an end device with the **CE** command. When you enable polling, the device sends a poll request a minimum of once every polling interval, where the polling interval is 100 ms for the 80 Kb/s data rate and it is 300 ms for the 10 Kb/s data rate. When the device sends normal data to the destination specified by the **DH**/**DL** of end device module, the data also functions as a poll.

When a polling device is also an asynchronous sleeping device, that device sends a poll shortly after waking from sleep. After that first poll is sent, the device sends polls in the normal manner described previously until it returns to sleep.

Sleeping routers

The Sleeping Router feature of DigiMesh makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time. This forms a cyclic sleeping network.

Devices synchronize by receiving a special RF packet called a sync message sent by a device acting as a sleep coordinator. A device in the network becomes a sleep coordinator through a process called nomination. The sleep coordinator sends one sync message at the beginning of each wake period. The device sends a sync message as a broadcast that is repeated by every device in the network. To change the sleep and wake times for the entire network, change the settings on an individual node locally. The network uses the most recently set sleep settings.

For more information, see Become a sleep coordinator.

Sleep coordinator sleep modes in the DigiMesh network

In a synchronized sleeping network, one node acts as the sleep coordinator. During normal operations, at the beginning of a wake cycle the sleep coordinator sends a sync message as a broadcast to all nodes in the network. This message contains synchronization information and the wake and sleep times for the current cycle. All cyclic sleep nodes that receive a sync message remain awake for the wake time and then sleep for the specified sleep period.

The sleep coordinator sends one sync message at the beginning of each cycle with the current wake and sleep times. All router nodes that receive this sync message relay the message to the rest of the network. If the sleep coordinator does not hear a rebroadcast of the sync message by one of its immediate neighbors, then it re-sends the message one additional time.

If you change the **SP** or **ST** parameters, the network does not apply the new settings until the beginning of the next wake time. For more information, see Change sleep parameters.

A sleeping router network is robust enough that an individual node can go several cycles without receiving a sync message, due to RF interference, for example. As a node misses sync messages, the time available for transmitting messages during the wake time reduces to maintain synchronization accuracy. By default, a device reduces its active sleep time progressively as it misses sync messages.

Synchronization messages

A sleep coordinator regularly sends sync messages to keep the network in sync. Unsynchronized nodes also send messages requesting sync information.

Sleep compatible nodes use Deployment mode when they first power up and the sync message has not been relayed. A sleep coordinator in Deployment mode rapidly sends sync messages until it receives a relay of one of those messages. Deployment mode:

- Allows you to effectively deploy a network.
- Allows a sleep coordinator that resets to rapidly re-synchronize with the rest of the network.

If a node exits deployment mode and then receives a sync message from a sleep coordinator that is in Deployment mode, it rejects the sync message and sends a corrective sync to the sleep coordinator.

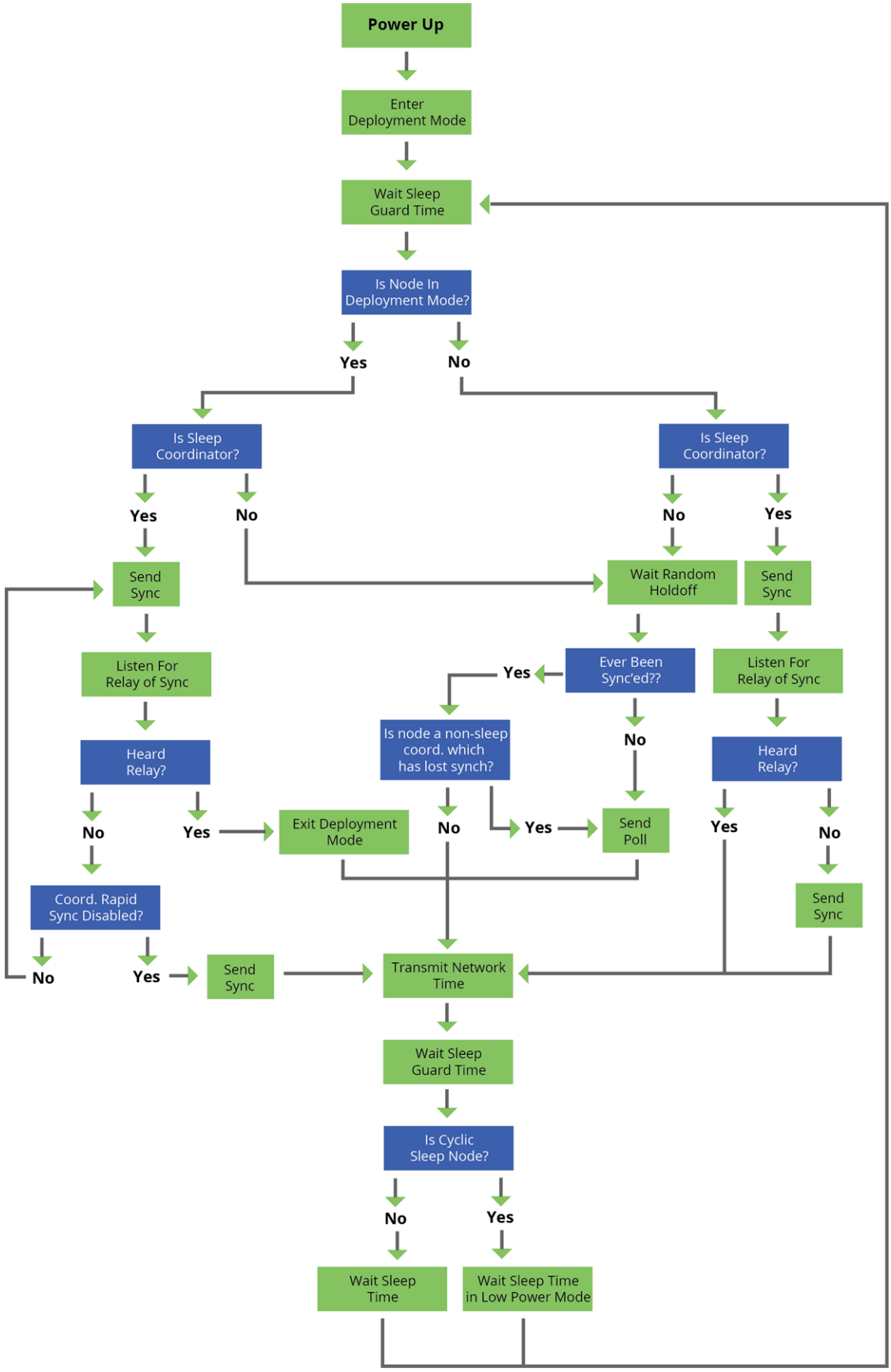
Use the **SO** (sleep options) command to disable deployment mode. This option is enabled by default.

A sleep coordinator that is not in deployment mode sends a sync message at the beginning of the wake cycle. The sleep coordinator listens for a neighboring node to relay the sync. If it does not hear the relay, the sleep coordinator sends the sync one additional time.

A node that is not a sleep coordinator and has never been synchronized sends a message requesting sync information at the beginning of its wake cycle. Synchronized nodes which receive one of these messages respond with a synchronization packet.

If you use the **SO** command to configure nodes as non-coordinators, and if the non-coordinators go six or more sleep cycles without hearing a sync, they send a message requesting sync at the beginning of their wake period.

The following diagram illustrates the synchronization behavior of sleep compatible devices.


Power Up
Enter Deployment Mode
Wait Sleep Guard Time
Is Node In Deployment Mode?
Yes
No
Is Sleep Coordinator?
Yes
No
Send Sync
Listen For Relay of Sync
Heard Relay?
No
Yes
Coord. Rapid Sync Disabled?
No
Yes
Send Sync
Exit Deployment Mode
Is Sleep Coordinator?
No
Yes
Wait Random Holdoff
Ever Been Sync'ed??
Yes
No
Is node a non-sleep coord. which has lost synch?
No
Yes
Send Poll
Send Sync
Listen For Relay of Sync
Heard Relay?
Yes
No
Send Sync
Transmit Network Time
Wait Sleep Guard Time
Is Cyclic Sleep Node?
No
Yes
Wait Sleep Time
Wait Sleep Time in Low Power Mode

Become a sleep coordinator

In DigiMesh networks, a device can become a sleep coordinator in one of four ways:

- Define a preferred sleep coordinator
- A potential sleep coordinator misses three or more sync messages
- Press the Commissioning Pushbutton twice on a potential sleep coordinator
- Change the sleep timing values on a potential sleep coordinator

Preferred sleep coordinator option

You can specify that a node always act as a sleep coordinator. To do this, set the preferred sleep coordinator bit (bit 0) in the **SO** command to 1.

A node with the sleep coordinator bit set always sends a sync message at the beginning of a wake cycle. To avoid network congestion and synchronization conflicts, do not set this bit on more than one node in the network.

Although it is not necessary to specify a preferred sleep coordinator, doing so improves network performance.

A node that is centrally located in the network can serve as a good sleep coordinator, because it minimizes the number of hops a sync message takes to get across the network.

A sleep support node and/or a node that is mains powered is a good candidate to be a sleep coordinator.

CAUTION! Use the preferred sleep coordinator bit with caution. The advantages of using the option become weaknesses if you use it on a node that is not in the proper position or configuration.

You can also use the preferred sleep coordinator option when you set up a network for the first time. When you start a network, you can configure a node as a sleep coordinator so it will begin sending sleep messages. After you set up the network, disable the preferred sleep coordinator bit.

Resolution criteria and selection option

There is an optional selection process with resolution criteria that occurs on a node if it loses contact with the network sleep coordinator. By default, this process is disabled. Use the **SO** command to enable this process. This process occurs automatically if a node loses contact with the previous sleep coordinator.

If you enable the process on any sleep compatible node, it is eligible to become the sleep coordinator for the network.

A sleep compatible node may become a sleep coordinator if it:

- Misses three or more sync messages.
- Is not configured as a non-coordinator (presumably because the sleep coordinator has been disabled).

Depending on the platform and other configurable options, such a node eventually uses the selection process after a number of sleep cycles without a sync.

A node that uses the selection process begins acting as the new network sleep coordinator.

It is possible for multiple nodes to declare themselves as the sleep coordinator. If this occurs, the firmware uses the following resolution criteria to identify the sleep coordinator from among the nodes using the selection process:

1. Newer sleep parameters: the network considers a node using newer sleep parameters (**SP** and **ST**) as higher priority to a node using older sleep parameters. See Change sleep parameters.
2. Preferred sleep coordinator: a node acting as a preferred sleep coordinator is higher priority to other nodes.
3. Sleep support node: sleep support nodes are higher priority to cyclic sleep nodes. You can modify this behavior using the **SO** parameter.
4. Serial number: If the previous factors do not resolve the priority, the network considers the node with the higher serial number to be higher priority.

Commissioning Pushbutton option

If you enable the Commissioning Pushbutton functionality, you can immediately select a device as a sleep coordinator by pressing the Commissioning Pushbutton twice or by issuing the **CB2** command. The device you select in this manner is still subject to the resolution criteria process.

Only sleep coordinator nodes honor Commissioning Pushbutton nomination requests. A node configured as a non-sleep coordinator ignores commissioning button nomination requests.

Change sleep parameters

Any sleep compatible node in the network that does not have the non-coordinator sleep option set can make changes to the network's sleep and wake times. If you change a node's **SP** or **ST** to values different from those that the network is using, the node becomes the sleep coordinator. The node begins sending sync messages with the new sleep parameters at the beginning of the next wake cycle.

- For normal operations, a device uses the sleep and wake parameters it gets from the sleep sync message, not the ones specified in its **SP** and **ST** parameters. It does not update the **SP** and **ST** parameters with the values of the sync message. Use the **OS** and **OW** commands to query the operating network sleep and wake times currently being used by the node.
- Changing network parameters can cause a node to become a sleep coordinator and change the sleep settings of the network. The following commands can cause this to occur: **NH**, **NN**, **NQ**, and **MR**.

For most applications, we recommend configuring the **NH**, **NN**, and **MR** network parameters during initial deployment only.

Sleep guard times

To compensate for variations in the timekeeping hardware of the various devices in a sleeping router network, the network allocates sleep guard times at the beginning and end of the wake period. The size of the sleep guard time varies based on the sleep and wake times you select and the number of sleep cycles that elapse since receiving the last sync message. The sleep guard time guarantees that a destination module will be awake when the source device sends a transmission. As a node misses more and more consecutive sync messages, the sleep guard time increases in duration and decreases the available transmission time.

Auto-early wake-up sleep option

Similar to the sleep guard time, the auto early wake-up option decreases the sleep period based on the number of sync messages a node misses. This option comes at the expense of battery life.

Use the **SO** command to disable auto-early wake-up sleep. This option is enabled by default.

Select sleep parameters

Choosing proper sleep parameters is vital to creating a robust sleep-enabled network with a desirable battery life. To select sleep parameters that will be good for most applications, follow these steps:

1. Choose **NH**.

   Based on the placement of the nodes in your network, select the appropriate values for the **NH** (Network Hops) parameter .

   We optimize the default values of **NH** to work for the majority of deployments. In most cases, we suggest that you do not modify these parameters from their default values. Decreasing these parameters for small networks can improve battery life, but take care to not make the values too small.

2. Calculate the Sync Message Propagation Time (SMPT).

   This is the maximum amount of time it takes for a sleep synchronization message to propagate to every node in the network. You can estimate this number with the following formula:
   SMPT = **NH***(**MT**+1)*18 ms.

3. Select the duty cycle you want.
4. Choose the sleep period and wake time.

   The wake time must be long enough to transmit the desired data as well as the sync message. The **ST** parameter automatically adjusts upwards to its minimum value when you change other AT commands that affect it (**SP**, and **NH**).

   Use a value larger than this minimum. If a device misses successive sync messages, it reduces its available transmit time to compensate for possible clock drift. Budget a large enough **ST** time to allow for the device to miss a few sync messages and still have time for normal data transmissions.

Start a sleeping synchronous network

By default, all new nodes operate in normal (non-sleep) mode. To start a synchronous sleeping network, follow these steps:

1. Set **SO** to 1 to enable the preferred sleep coordinator option on one of the nodes.
2. Set its **SM** to a synchronous sleep compatible mode (7 or 8) with its **SP** and **ST** set to a quick cycle time. The purpose of a quick cycle time is to allow the network to send commands quickly through the network during commissioning.
3. Power on the new nodes within range of the sleep coordinator. The nodes quickly receive a sync message and synchronize themselves to the short cycle **SP** and **ST** set on the sleep coordinator.
4. Configure the new nodes to the sleep mode you want, either cyclic sleeping modes or sleep support modes.
5. Set the **SP** and **ST** values on the sleep coordinator to the values you want for the network.
6. Wait a sleep cycle for the sleeping nodes to sync themselves to the new **SP** and **ST** values.

7. Disable the preferred sleep coordinator option bit on the sleep coordinator unless you want a preferred sleep coordinator.
8. Deploy the nodes to their positions.

Alternatively, prior to deploying the network you can use the **WR** command to set up nodes with their sleep settings pre-configured and written to flash. If this is the case, you can use the Commissioning Pushbutton and associate LED to aid in deployment:

1. If you are going to use a preferred sleep coordinator in the network, deploy it first.
2. If there will not be a preferred sleep coordinator, select a node for deployment, power it on and press the Commissioning Pushbutton twice. This causes the node to begin emitting sync messages.
3. Verify that the first node is emitting sync messages by watching its associate LED. A slow blink indicates that the node is acting as a sleep coordinator.
4. Power on nodes in range of the sleep coordinator or other nodes that have synchronized with the network. If the synchronized node is asleep, you can wake it by pressing the Commissioning Pushbutton once.
5. Wait a sleep cycle for the new node to sync itself.
6. Verify that the node syncs with the network. The associate LED blinks when the device is awake and synchronized.
7. Continue this process until you deploy all of the nodes.

Add a new node to an existing network

To add a new node to the network, the node must receive a sync message from a node already in the network. On power-up, an unsynchronized, sleep compatible node periodically sends a broadcast requesting a sync message and then sleeps for its **SP** period. Any node in the network that receives this message responds with a sync. Because the network can be asleep for extended periods of time, and cannot respond to requests for sync messages, there are methods you can use to sync a new node while the network is asleep.

1. Power the new node on within range of a sleep support node. Sleep support nodes are always awake and able to respond to sync requests promptly.
2. You can wake a sleeping cyclic sleep node in the network using the Commissioning Pushbutton. Place the new node in range of the existing cyclic sleep node. Wake the existing node by holding down the Commissioning Pushbutton for two seconds, or until the node wakes. The existing node stays awake for 30 seconds and responds to sync requests while it is awake.

If you do not use one of these two methods, you must wait for the network to wake up before adding the new node.

Place the new node in range of the network with a sleep/wake cycle that is shorter than the wake period of the network.

The new node periodically sends sync requests until the network wakes up and it receives a sync message.

Change sleep parameters

To change the sleep and wake cycle of the network, select any sleep coordinator capable node in the network and change the **SP** and/or **ST** of the node to values different than those the network currently uses.

- If you use a preferred sleep coordinator or if you know which node acts as the sleep coordinator, we suggest that you use this node to make changes to network settings.
- If you do not know the network sleep coordinator, you can use any node that does not have the non-sleep coordinator sleep option bit set. For details on the bit, see SO (Sleep Options).

When you make changes to a node's sleep parameters, that node becomes the network's sleep coordinator unless it has the non-sleep coordinator option selected. It sends a sync message with the new sleep settings to the entire network at the beginning of the next wake cycle. The network immediately begins using the new sleep parameters after it sends this sync.

Changing sleep parameters increases the chances that nodes will lose sync. If a node does not receive the sync message with the new sleep settings, it continues to operate on its old settings. To minimize the risk of a node losing sync and to facilitate the re-syncing of a node that does lose sync, take the following precautions:

1. Whenever possible, avoid changing sleep parameters.
2. Enable the missed sync early wake up sleep option in the **SO** command. This option is enabled by default. This command tells a node to wake up progressively earlier based on the number of cycles it goes without receiving a sync. This increases the probability that the un-synced node will be awake when the network wakes up and sends the sync message.

Note Using this sleep option increases reliability but may decrease battery life. Nodes using this sleep option that miss sync messages increase their wake time and decrease their sleep time during cycles where they miss the sync message. This increases power consumption.

When you are changing between two sets of sleep settings, choose settings so that the wake periods of the two sleep settings occur at the same time. In other words, try to satisfy the following equation:

$$(SP_1 + ST_1) = N * (SP_2 + ST_2)$$

where SP_1/ST_1 and SP_2/ST_2 are the desired sleep settings and N is an integer.

Rejoin nodes that lose sync

DigiMesh networks get their robustness from routing redundancies which may be available. We recommend architecting the network with redundant mesh nodes to increase robustness.

If a scenario exists where the only route connecting a subnet to the rest of the network depends on a single node, and that node fails or the wireless link fails due to changing environmental conditions (a catastrophic failure condition), then multiple subnets may arise using the same wake and sleep intervals. When this occurs the first task is to repair, replace, and strengthen the weak link with new and/or redundant devices to fix the problem and prevent it from occurring in the future.

When you use the default DigiMesh sleep parameters, separated subnets do not drift out of phase with each other. Subnets can drift out of phase with each other if you configure the network in one of the following ways:

- If you disable the non-sleep coordinator bit in the **SO** command on multiple devices in the network, they are eligible for the network to nominate them as a sleep coordinator. For more details, see SO (Sleep Options).
- If the devices in the network do not use the auto early wake-up sleep option.

If a network has multiple subnets that drift out of phase with each other, get the subnets back in phase with the following steps:

1. Place a sleep support node in range of both subnets.
2. Select a node in the subnet that you want the other subnet to sync with.
3. Use this node to slightly change the sleep cycle settings of the network, for example, increment **ST**.
4. Wait for the subnet's next wake cycle. During this cycle, the node you select to change the sleep cycle parameters sends the new settings to the entire subnet it is in range of, including the sleep support node that is in range of the other subnet.
5. Wait for the out of sync subnet to wake up and send a sync. When the sleep support node receives this sync, it rejects it and sends a sync to the subnet with the new sleep settings.
6. The subnets will now be in sync. You can remove the sleep support node.
7. You can also change the sleep cycle settings back to the previous settings.

If you only need to replace a few nodes, you can use this method:

1. Reset the out of sync node and set its sleep mode to Synchronous Cyclic Sleep mode (**SM** = 8).
2. Set up a short sleep cycle.
3. Place the node in range of a sleep support node or wake a sleeping node with the Commissioning Pushbutton.
4. The out of sync node receives a sync from the node that is synchronized to the network. It then syncs to the network sleep settings.

Diagnostics

The following diagnostics are useful in applications that manage a sleeping router network:

Query sleep cycle

Use the **OS** and **OW** commands to query the current operational sleep and wake times that a device uses.

Sleep status

Use the **SS** command to query useful information regarding the sleep status of the device. Use this command to query if the node is currently acting as a network sleep coordinator.

Missed sync messages command

Use the **MS** command to query the number of cycles that elapsed since the device received a sync message.

Sleep status API messages

When you use the **SO** command to enable this option, a device that is in API operating mode outputs modem status frames immediately after it wakes up and prior to going to sleep.

AT commands

Special commands

The following commands are special commands.

AC (Apply Changes)

Immediately applies new settings without exiting Command mode.

Parameter range

N/A

Default

N/A

FR (Software Reset)

Resets the device. The device responds immediately with an **OK** and performs a reset 100 ms later. If you issue **FR** while the device is in Command Mode, the reset effectively exits Command mode.

Parameter range

N/A

Default

N/A

RE (Restore Defaults)

Restore device parameters to factory defaults.

Parameter range

N/A

Default

N/A

WR (Write)

Writes parameter values to non-volatile memory so that parameter modifications persist through subsequent resets.

Note Once you issue a **WR** command, do not send any additional characters to the device until after you receive the **OK** response.

Parameter range

N/A

Default

N/A

MAC/PHY commands

The following AT commands are MAC/PHY commands.

CM (Channel Mask)

CM allows you to selectively enable or disable channels used for RF communication. This is useful to avoid using frequencies that experience unacceptable levels of RF interference, or to operate two networks of radios on separate frequencies.

This mask limits the channels where the device transmits. See Technical specifications for the list of frequencies. Channel 0 is bit 0. You must enable at least two channels, except when using only a single frequency of 869.85 MHz. When you use this mode (use 0x20000000), LBT+AFA is disabled and the power level is automatically limited to 5 dBm.

This command is a bitfield.

The **CM** command does not limit receive channels. If two devices have mutually exclusive values for **CM** (for example 0x0000FF00 and 0x000000FF), then communication is possible because both devices still listen on all possible channels, while limiting the transmission channels to those specified in the **CM** command.

The **CM** channel mask no longer supports channels 9 and 24 for transmissions due to regulatory requirements. However, the device can receive transmissions on channels 9 and 24, which then will transmit on any other valid channel in the channel mask (this is to maintain compatibility with the existing S8 product).

Parameter range

3 - 0x3EFFFDFF [bitfield]

Default

Europe: 0x3EFFFDFF (channels 0 - 29, 863.15 - 869.85 MHz), excluding channels 9 and 24

Europe (single frequency mode): 0x20000000 (channel 29, 869.85 MHz)

HP (Preamble ID)

The preamble ID for which the device communicates. Only devices with matching preamble IDs can communicate with each other. Different preamble IDs minimize interference between multiple sets of devices operating in the same vicinity. When receiving a packet, the device checks this before the network ID, as it is encoded in the preamble, and the network ID is encoded in the MAC header.

Parameter range

0 - 9

Default

0

ID (Network ID)

Set or read the user network identifier.

Devices must have the same network identifier to communicate with each other.

When receiving a packet, the device check this after the preamble ID. If you are using Original equipment manufacturer (OEM) network IDs, **0xFFFF** uses the factory value.

Parameter range

0 - 0x7FFF

Default

0x7FFF

MT (Broadcast Multi-Transmits)

Set or read the number of additional MAC-level broadcast transmissions. All broadcast packets are transmitted **MT**+1 times to ensure they are received.

Parameter range

0 - 5

Default

3

BR (RF Data Rate)

Sets and reads the device's RF data rate (the rate at which the device transmits and receives RF data over-the-air).

DigiMesh and synchronized sleep are not supported when **BR** = **0**. All devices on the network must have the same **BR** value set in order to communicate. **BR** directly affects the range of the device. The higher the RF data rate, the lower the receive sensitivity.

Parameter range

0 - 1

Parameter	RF data rate	Receiver sensitivity
0	10 kb/s	-113 dBm
1	80 kb/s	-106 dBm

Default

1

PL (TX Power Level)

Sets or displays the power level at which the device transmits conducted power. Power levels are approximate.

Parameter range

0 - 4

Setting	Power level*
0	2 mW EIRP
1	5 mW EIRP

Setting	Power level*
2	10 mW EIRP
3	16 mW EIRP
4	32 mW EIRP
* EIRP stands for Effective Isotropically Radiated Power, which is the device's output power plus 2.1 dBi (dipole antenna gain).	

Default

4

RR (Unicast Mac Retries)

Set or read the maximum number of MAC level packet delivery attempts for unicasts. If **RR** is non-zero, the sent unicast packets request an acknowledgment from the recipient. Unicast packets can be retransmitted up to **RR** times if the transmitting device does not receive a successful acknowledgment.

Parameter range

0 - 0xF

Default

0x0A (decimal 10)

ED (Energy Detect)

Starts an energy detect scan. This command accepts an argument to specify the time in milliseconds to scan all channels. The device loops through all the available channels until the time elapses. It returns the maximal energy on each channel, a comma follows each value, and the list ends with a carriage return. The values returned reflect the energy level that **ED** detects in -dBm units.

Parameter range

0 - 0xFF

Default

0x10

LB (LNA Bypass)

Sets or reads the LNA bypass enable of the device. If the LNA bypass is enabled, the RX current draw improves by 6 mA while the RX sensitivity degrades by approximately 12 dB.

Range

0 - 1

Parameter	Mode
0	LNA Bypass Disabled
1	LNA Bypass Enabled

Default

0

Diagnostic commands

The following AT commands are diagnostic commands. Diagnostic commands are typically volatile and will not persist across a power cycle.

BC (Bytes Transmitted)

The number of RF bytes transmitted. The firmware counts every byte of every packet, including MAC/PHY headers and trailers. The purpose of this count is to estimate battery life by tracking time spent performing transmissions.

This number rolls over to **0** from **0xFFFF**.

You can reset the counter to any unsigned 16-bit value by appending a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFF

Default

0

DB (Last Packet RSSI)

Reports the RSSI in -dBm of the last received RF data packet. **DB** returns a hexadecimal value for the -dBm measurement.

For example, if **DB** returns 0x60, then the RSSI of the last packet received was -96 dBm.

DB only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link.

Parameter range

0 - 0xFF [read-only]

Default

0

ER (Received Error Count)

This count increments when a device receives a packet that contains integrity errors of some sort. When the number reaches 0xFFFF, the firmware does not count further events.

To reset the counter to any 16-bit unsigned value, append a hexadecimal parameter to the **ER** command.

Parameter range

0 - 0xFFFF

Default

0

GD (Good Packets Received)

This count increments when a device receives a good frame with a valid MAC header on the RF interface. Once the number reaches 0xFFFF, it does not count further events.

To reset the counter to any 16-bit unsigned value, append a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFF

Default

0

EA (MAC ACK Timeouts)

This count increments whenever a MAC ACK timeout occurs on a MAC-level unicast. When the number reaches **0xFFFF**, the firmware does not count further events.

To reset the counter to any 16-bit value, append a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFF

Default

0

TR (Transmission Errors)

This count increments whenever a MAC transmission attempt exhausts all MAC retries without ever receiving a MAC acknowledgment message from the destination node. Once the number reaches **0xFFFF**, it does not count further events.

To reset the counter to any 16-bit value, append a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFF

Default

0

UA (MAC Unicast Transmission Count)

This count increments whenever a MAC unicast transmission occurs that requests an ACK. Once the number reaches 0xFFFF, it does not count further events.

You can reset the counter to any 16-bit unsigned value by appending a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFF

Default

0

%H (MAC Unicast One Hop Time)

The MAC unicast one hop time timeout in milliseconds. If you change the MAC parameters it can change this value.

Parameter range

[read-only]

Default

0x267

0xF3

%8 (MAC Broadcast One Hop Time)

The MAC broadcast one hop time timeout in milliseconds. If you change MAC parameters, it can change this value.

Parameter range

[read-only]

Default

0x204

Network commands

The following commands are network commands.

CE (Node Messaging Options)

The routing and messaging mode bit field of the device.

A routing device repeats broadcasts. Indirect Messaging Coordinators do not transmit point-to-multipoint unicasts until an end device requests them. Setting a device as an end device causes it to regularly send polls to its Indirect Messaging Coordinator. Nodes can also be configured to route, or not route, multi-hop packets.

Bit	Description
Bit 0	Indirect Messaging Coordinator enable. All point-to-multipoint unicasts will be held until requested by a polling end device.
Bit 1	Disable routing on this node. When set, this node will not propagate broadcasts or become an intermediate node in a DigiMesh route. This node will not function as a repeater.

Bit	Description
Bit 2	Indirect Messaging Polling enable. Periodically send requests for messages held by the node's coordinator.

Note Bit 0 and Bit 2 cannot be set at the same time.

Parameter range

0 - 6

Default

0

BH (Broadcast Hops)

The number of hops for broadcast data transmissions.

Set the value to **0** for the maximum number of hops.

If you set **BH** greater than **NH**, the device uses the value of **NH**.

Parameter range

0 - 0x20

Default

0

NH (Network Hops)

The maximum number of hops expected to be seen in a network route. This value does not limit the number of hops allowed, but it is used to calculate timeouts waiting for network acknowledgments.

Parameter range

1 - 0x20

Default

7

NN (Network Delay Slots)

Set or read the maximum random number of network delay slots before rebroadcasting a network packet.

Parameter range

1 - 5

Default

3

MR (Mesh Unicast Retries)

Set or read the maximum number of network packet delivery attempts. If **MR** is non-zero, the packets a device sends request a network acknowledgment, and can be resent up to **MR**+1 times if the device does not receive an acknowledgment.

We recommend that you set this value to **1**.

If you set this parameter to **0**, it disables network ACKs. Initially, the device can find routes, but a route will never be repaired if it fails.

Parameter range

0 - 7

Default

1

Addressing commands

The following AT commands are addressing commands.

SH (Serial Number High)

Displays the upper 32 bits of the unique IEEE 64-bit extended address assigned to the XBee in the factory.

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

SL (Serial Number Low)

Displays the lower 32 bits of the unique IEEE 64-bit RF extended address assigned to the XBee in the factory.

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

DH (Destination Address High)

Set or read the upper 32 bits of the 64-bit destination address. When you combine **DH** with **DL**, it defines the destination address that the device uses for transmissions in Transparent mode. **0x000000000000FFFF** is the broadcast address.

Parameter range

0 - 0xFFFFFFFF

Default

0

DL (Destination Address Low)

Set or display the lower 32 bits of the 64-bit destination address. When you combine **DH** with **DL**, it defines the destination address that the device uses for transmissions in Transparent mode.

Parameter range

0 - 0xFFFFFFFF

Default

0x0000FFFF

TO (Transmit Options)

The bitfield that configures the transmit options for Transparent mode.

The device's transmit options. The device uses these options for all transmissions. You can override these options using the TxOptions field in the API TxRequest frames.

Sets or displays transmit options for all serial transmissions. **TO** options can be overridden packet-by-packet using the **TxOptions** field of an API **TxRequest** frame.

Parameter range

0 - 0xFF

Bit	Meaning	Description
6,7	Delivery method	b'00 = <invalid option> b'01 = Point-multipoint b'10 = Repeater mode (directed broadcast of packets) b'11 = DigiMesh (not available when **BR** = **0**)
5	Reserved	<set this bit to 0>
4	Reserved	<set this bit to 0>
3	Trace Route	Enable a Trace Route on all DigiMesh API packets
2	NACK	Enable a NACK messages on all DigiMesh API packets
1	Disable RD	Disable Route Discovery on all DigiMesh unicasts
0	Disable ACK	Disable acknowledgments on all unicasts

One of the following hexadecimal values:

Value	Description
0x40	Point-to-point/multipoint, ACK enabled
0x41	Point-to-point/mulitpoint, ACK disabled

Value	Description
0x80	Repeater/Directed broadcast, ACK enabled
0x81	Repeater/Directed broadcast, ACK disabled

Example 1: Set **TO** to **0x80** to send all transmissions using repeater mode.

Example 2: Set **TO** to **0xC1** to send transmissions using DigiMesh, with network acknowledgments disabled.

- Bits 6 and 7 cannot be set to DigiMesh when **BR** = **0**, 10k.
- Bits 1, 2, and 3 cannot be set when **BR** = **0**, 10k.

When you set **BR** to **0** the **TO** option has the DigiMesh and Repeater mode disabled automatically.

Default

0x40

0x40 When **BR** = **0**, 10k

0xC0 When **BR** = **1**, 80k

NI (Node Identifier)

Stores the node identifier string for a device, which is a user-defined name or description of the device. This can be up to 20 ASCII characters.

- The command automatically ends when the maximum bytes for the string have been entered.
- To set **NI**to its default value of one ASCII space, type **ATNI**followed by a space and press **Enter**.

Use the **ND** (Network Discovery) command with this string as an argument to easily identify devices on the network.

The **DN** command also uses this identifier.

Parameter range

A string of case-sensitive ASCII printable characters from 1 to 20 bytes in length. The string cannot start with the space character. A carriage return or a comma automatically ends the command.

Default

One ASCII space character (0x20)

NT (Node Discover Timeout)

Sets the amount of time a base node waits for responses from other nodes when using the **ND** (Node Discover) or **DN** (Discover Node) commands. The value randomizes the responses to alleviate network congestion.

Parameter range

0x20 - 0x2EE0 (x 100 ms)

Default

0x82 (13 seconds)

NO (Node Discovery Options)

Set or read the network discovery options value for the **ND** (Network Discovery) command on a particular device. The options bit field value changes the behavior of the **ND** command and what optional values the local device returns when it receives an **ND** command or API Node Identification Indicator (0x95) frame.

These options also apply to FN (Find Neighbors) command responses.

Parameter range

0x0 - 0x7 (bit field)

Option	Description
0x01	Append the **DD** (Digi Device Identifier) value to **ND** or **FN** responses or API node identification frames.
0x02	Local device returns own **ND** response frame when a **ND** or **FN** is issued.
0x04	Append the RSSI of the last hop to **ND**, **FN** responses or API node identification frames.
Option	**Description**
0x01	Append the **DD** (Digi Device Identifier) value to **ND** responses or API node identification frames.
0x02	Local device sends **ND** response frame when the **ND** is issued.

Default

0x0

CI (Cluster ID)

The application layer cluster ID value. The device uses this value as the cluster ID for all data transmissions.

Parameter range

0 - 0xFFFF

Default

0x11

DE (Destination Endpoint)

Sets or displays the application layer destination ID value. The value is used as the destination endpoint for all data transmissions. The default value (0xE8) is the Digi data endpoint.

Parameter range

0 - 0xFF

Default

0xE8

SE (Source Endpoint)

Sets or displays the application layer source endpoint value. The value is used as the source endpoint for all data transmissions. The default value (0xE8) is the Digi data endpoint.

This command only affects outgoing transmissions in transparent mode (**AP** = **0**).

0xE8 is the Digi data endpoint used for outgoing data transmissions.

0xE6 is the Digi device object endpoint used for configuration and commands.

Parameter range

0 - 0xFF

Default

0xE8

Addressing discovery/configuration commands

AG (Aggregator Support)

The **AG** command sends a broadcast through the network that has the following effects on nodes that receive the broadcast:

- The receiving node establishes a DigiMesh route back to the originating node, if there is space in the routing table.
- The **DH** and **DL** of the receiving node update to the address of the originating node if the **AG** parameter matches the current **DH**/**DL** of the receiving node.
- API-enabled devices with updated **DH** and **DL** send an Aggregate Addressing Update frame (0x8E) out the serial port.

Parameter range

Any 64-bit address

Default

N/A

DN (Discover Node)

Resolves an **NI** (Node identifier) string to a physical address (case sensitive).

The following events occur after **DN** discovers the destination node:

When **DN** is sent in Command mode:

1. The device sets **DL** and **DH** to the extended (64-bit) address of the device with the matching **NI** string.
2. The receiving device returns **OK** (or **ERROR**).
3. The device exits Command mode to allow for immediate communication. If an **ERROR** is received, the device does not exit Command mode.

When **DN** is sent as an API frame, the receiving device returns 0xFFFE followed by its 64-bit extended addresses in an API Command Response frame.

Parameter range

20-byte ASCII string

Default

N/A

ND (Network Discover)

Discovers and reports all of the devices it finds on a network. If you send **ND** through a local API frame, each network node returns a separate AT Command Response frame - 0x88 or Remote Command Response frame - 0x97 frame, respectively.

The command reports the following information after a jittered time delay.

SH<CR> (4 bytes)

SL<CR> (4 bytes)

DB<CR> (Contains the detected signal strength of the response in negative dBm units)

NI <CR> (variable, 0-20 bytes plus 0x00 character)

DEVICE_TYPE<CR> (1 byte: **0** = Coordinator, **1** = Router, **2** = End Device)

STATUS<CR> (1 byte: reserved)

PROFILE_ID<CR> (2 bytes)

MANUFACTURER_ID<CR> (2 bytes)

DIGI DEVICE TYPE<CR> (4 bytes. Optionally included based on **NO** settings.)

RSSI OF LAST HOP<CR> (1 byte. Optionally included based on **NO** settings.)

After (**NT** * 100) milliseconds, the command ends by returning a <CR>.

If you send **ND** through a AT Command frame - 0x08 frame, each network node returns a separate AT Command Response frame - 0x88 or Remote Command Response frame - 0x97 frame, respectively. The data consists of the bytes listed above without the carriage return delimiters. The **NI** string ends in a **0x00** null character.

Broadcast an **ND** command to the network. If the command includes an optional node identifier string parameter, only those devices with a matching **NI** string respond without a random offset delay. If the command does not include a node identifier string parameter, all devices respond with a random offset delay.

The NT (Node Discover Timeout) setting determines the range of the random offset delay. The NO (Node Discovery Options) setting sets options for the Node Discovery. For more information about options that affect the behavior of **ND** see the description of **NO**.

WARNING! If the **NT** setting is small relative to the number of devices on the network, responses may be lost due to channel congestion. Regardless of the **NT** setting, because the random offset only mitigates transmission collisions, getting responses from all devices in the network is not guaranteed.

Parameter range

N/A

Default

N/A

FN (Find Neighbors)

Discovers and reports all devices found within immediate (1 hop) RF range. **FN** reports the following information for each device it discovers:

MY<CR> (always 0xFFFE)

SH<CR>

SL<CR>

NI<CR> (Variable length)

PARENT_NETWORK ADDRESS<CR> (2 Bytes) (always 0xFFFE)

DEVICE_TYPE<CR> (1 Byte: 0 = Coordinator, 1 = Router, 2 = End Device)

STATUS<CR> (1 Byte: Reserved)

PROFILE_ID<CR> (2 Bytes)

MANUFACTURER_ID<CR> (2 Bytes)

DIGI DEVICE TYPE<CR> (4 Bytes. Optionally included based on **NO** settings.)

RSSI OF LAST HOP<CR> (1 Byte. Optionally included based on **NO** settings.)

<CR>

If you send the **FN** command in Command mode, after (**NT***100) ms + overhead time, the command ends by returning a carriage return, represented by <CR>.

If you send the **FN** command through a local API frame, each response returns as a separate Local or Remote AT Command Response API packet, respectively. The data consists of the bytes in the previous list without the carriage return delimiters. The **NI** string ends in a 0x00 null character.

Parameter range

N/A

Default

N/A

Diagnostic - addressing commands

The following AT command is a Diagnostic - addressing command.

N? (Network Discovery Timeout)

The maximum response time, in milliseconds, for **ND** (Network Discovery) responses and **DN** (Discover Node) responses. The timeout is the sum of **NT** (Network Discovery Back-off Time) and the network propagation time.

Parameter range

[read-only]

Default

N/A

Security commands

The following AT commands are security commands.

EE (Security Enable)

Enables or disables 128-bit Advanced Encryption Standard (AES) encryption.

Set this command parameter the same on all devices in a network.

Parameter range

0 - 1

Parameter	Description
0	Encryption Disabled
1	Encryption Enabled

Default

0

KY (AES Encryption Key)

Sets the 16-byte network security key value that the device uses for encryption and decryption.

This command is write-only. If you attempt to read **KY**, the device returns an **OK** status.

Set this command parameter the same on all devices in a network.

The value passes in as hex characters when you set it from AT command mode, and as binary bytes when you set it in API mode.

Parameter range

128-bit value

Default

N/A

Serial interfacing commands

BD (Interface Data Rate)

Sets and reads the serial interface data rate (baud rate) between the device and the host. The baud rate is the rate that the host sends serial data to the device.

When you make an update to the interface data rate, the change does not take effect until the host issues the **CN** command and the device returns the **OK** response.

The **BD** parameter does not affect the RF data rate. If you set the interface data rate higher than the RF data rate, you may need to implement a flow control configuration.

Non-standard interface data rates

The firmware interprets any value within 0x4B0 - 0x2580 and 0x4B00 - 0x1C9468 as an actual baud rate. When the host sends a value above 0x4B0, the firmware stores the closest interface data rate represented by the number in the **BD** register. For example, to set a rate of 19200 b/s, send the following command line: **ATBD4B00**.

Note When using XCTU, you can only set and read non-standard interface data rates using the XCTU Serial Console tool. You cannot access non-standard rates through the configuration section of XCTU.

When you send the **BD** command with a non-standard interface data rate, the UART adjusts to accommodate the interface rate you request. In most cases, the clock resolution causes the stored **BD** parameter to vary from the sent parameter. Sending **ATBD** without an associated parameter value returns the value actually stored in the device's **BD** register.

The following table provides the parameters sent versus the parameters stored.

BD parameter sent (HEX)	Interface data rate (b/s)	BD parameter stored (HEX)
0	1200	0
4	19,200	4
7	115,200	7
1C200	115,200	1B207

Parameter ranges

0 - 8 (standard rates)

0x100 to 0x1C9468 (non-standard rates)

Parameter	Configuration (b/s)
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200
8	230400

Default

3

NB (Parity)

Set or read the serial parity settings for UART communications.

Parameter range

0x00 - 0x02

Parameter	Description
0x00	No parity
0x01	Even parity
0x02	Odd parity

Parameter	Description
0	No parity
1	Even parity
2	Odd parity

Default

0x00

SB (Stop Bits)

Sets or displays the number of stop bits for UART communications.

Parameter range

0 - 1

Parameter	Configuration
0	One stop bit
1	Two stop bits

Default

0

RO (Packetization Timeout)

Set or read the number of character times of inter-character silence required before transmission begins when operating in Transparent mode.

Set **RO** to **0** to transmit characters as they arrive instead of buffering them into one RF packet.

Parameter range

0 - 0xFF (x character times)

Default

3

FT (Flow Control Threshold)

Set or display the flow control threshold.

The device de-asserts CTS and/or send XOFF when **FT** bytes are in the UART receive buffer. It re-asserts CTS when less than **FT**-16 bytes are in the UART receive buffer.

Parameter range

0x1A - 0x166 bytes

Default

0x11D

AP (API Mode)

Sets or reads the UART API mode.

Parameter range

0 - 2

The following settings are allowed:

Parameter	Description
0	Transparent mode, API mode is off. All UART input and output is raw data and the device uses the **RO** parameter to delineate packets.
1	API Mode Without Escapes. The device packetizes all UART input and output data in API format, without escape sequences.
2	API Mode With Escapes. The device is in API mode and inserts escaped sequences to allow for control characters. The device passes XON, XOFF, Escape, and the 0x7E delimiter as data.

Default

0

AO (API Options)

The API data frame output format for RF packets received.

Parameter range

0, 1

Parameter	Description
0	API Rx Indicator - 0x90, this is for standard data frames.
1	API Explicit Rx Indicator - 0x91, this is for Explicit Addressing data frames.

Default

0

I/O settings commands

The following AT commands are I/O settings commands.

CB (Commissioning Pushbutton)

Use **CB** to simulate commissioning pushbutton presses in software.

Set the parameter value to the number of button presses that you want to simulate. For example, send **CB1** to perform the action of pressing the Commissioning Pushbutton once.

See The Commissioning Pushbutton.

See Commissioning pushbutton.

Parameter range

0 - 4

Default

N/A

D0 (DIO0/AD0)

Sets or displays the DIO0/AD0 configuration (pin 33).

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	Commissioning Pushbutton
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

D1 (DIO1/AD1)

Sets or displays the DIO1/AD1 configuration (pin 32).

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	N/A

Parameter	Description
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D2 (DIO2/AD2)

Sets or displays the DIO2/AD2 configuration (pin 31).

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	N/A
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D3 (DIO3/AD3)

Sets or displays the DIO3/AD3 configuration (pin 30).

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	N/A
2	ADC
3	Digital input

Parameter	Description
4	Digital output, low
5	Digital output, high

Default

0

D4 (DIO4)

Sets or displays the DIO4 configuration (pin 24).

Parameter range

0, 3 - 5

Parameter	Description
0	Disabled
1	N/A
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D5 (DIO5/ASSOCIATED_INDICATOR)

Sets or displays the DIO5/AD5/ASSOCIATED_INDICATOR configuration (pin 28).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	Associate LED indicator
2	N/A
3	Digital input
4	Digital output, default low
5	Digital output, default high

Default

1

D6 (DIO6/RTS)

Sets or displays the DIO6/$\overline{\text{RTS}}$ configuration (pin 29).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	$\overline{\text{RTS}}$ flow control
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D7 (DIO7/CTS)

Parameter range

0, 1, 3 - 7

Parameter	Description
0	Disabled
1	$\overline{\text{CTS}}$ flow control
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high
6	RS-485 Tx enable, low Tx (0 V on transmit, high when idle)
7	RS-485 Tx enable high, high Tx (high on transmit, 0 V when idle)

Default

0x1

D8 (DIO8/DTR/SLEEP_REQUEST)

Sets or displays the DIO8/SLEEP_REQUEST configuration (pin 10).

The XBee SX 868 RF Module does not support sleep. The SLEEP_REQUEST option is provided for compatibility purposes and does not affect the device.

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	Sleep request
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

D9 (DIO9/ON_SLEEP)

Sets or displays the DIO9/ON_$\overline{\text{SLEEP}}$ configuration (pin 26).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	ON/$\overline{\text{SLEEP}}$ output
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

P0 (DIO10/RSSI/PWM0 Configuration)

Sets or displays the PWM0/RSSI/DIO10 configuration (pin 7).

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	RSSI PWM0 output
2	PWM0 output
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

P1 (DIO11/PWM1 Configuration)

Sets or displays the DIO11/PWM1 configuration (pin 8).

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	32.768 kHz clock output
2	PWM1 output
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P2 (DIO12 Configuration)

Sets or displays the DIO12 configuration (pin 5).

Parameter range

0, 3 - 5

Parameter	Description
0	Disabled
1	N/A
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P3 (DOUT)

Sets or displays the DOUT configuration (pin 3).

Parameter range

0, 1

Parameter	Description
0	Unmonitored digital input
1	Data out for UART

Parameter	Description
0	Disabled
1	UART DOUT enabled

Default

1

P4 (DIN/CONFIG)

Sets or displays the DIN/$\overline{\text{CONFIG}}$ configuration (pin 4).

Parameter range

Parameter	Description
0	Disabled
1	UART DIN/$\overline{\text{CONFIG}}$ enabled

Default

1

P5 (DIO15/SPI_MISO Configuration)

Sets or displays the DIO15/SPI_MISO configuration (pin 17).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_MISO
2	N/A
3	N/A
4	Digital output low
5	Digital output high

Parameter	Description
0	Disabled
1	SPI_MISO
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Default

1

P6 (SPI_MOSI Configuration)

Sets or displays the DIO16/SPI_MOSI configuration (pin 16).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_MOSI
2	N/A
3	N/A

Parameter	Description
4	Digital output low
5	Digital output, high

Parameter	Description
0	Disabled
1	SPI_MOSI
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Default

1

P7 (DIO17/SPI_SSEL)

Sets or displays the DIO17/SPI_$\overline{\text{SSEL}}$ configuration (pin 15).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{SSEL}}$
2	N/A
3	N/A
4	Digital output low
5	Digital output, high

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{SSEL}}$
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Default

1

P8 (DIO18/SPI_SCLK)

Sets or displays the DIO18/SPI_SCLK configuration (pin 14).
Sets or displays the DIO18/SPI_CLK configuration (pin 14).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_SCLK
2	N/A
e	N/A
4	Digital output low
5	Digital output high

Parameter	Description
0	Disabled
1	SPI_CLK
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Parameter	Description
0	Disabled
1	N/A
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Default

1

P9 (DIO19/SPI_ATTN)

Sets or displays the DIO19/SPI_$\overline{\text{ATTN}}$ configuration (pin 12).

Parameter range

0, 1, 4 - 6

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{ATTN}}$
2	N/A
3	N/A
4	Digital output low
5	Digital output high
6	UART data present indicator

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{ATTN}}$
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high
6	UART data present indicator

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{ATTN}}$
2	N/A
3	N/A
4	N/A
5	N/A
6	PTI_DATA

Parameter	Description
0	Disabled
1	N/A
2	N/A
3	N/A
4	Digital output, low
5	Digital output, high

Default

1

PD (Pull Up/Down Direction)

The resistor pull direction bit field (1 = pull-up, 0 = pull-down) for corresponding I/O lines that are set by the **PR** command.

Parameter range

0x0 - 0xFFFFF

Default

0xFFFFF

PR (Pull-up/Down Resistor Enable)

The bit field that configures the internal pull-up resistor status for the I/O lines.

- If you set a **PR** bit to 1, it enables the pull-up/down resistor
- If you set a **PR** bit to 0, it specifies no internal pull-up/down resistor.

PR and **PD** only affect lines that are configured as digital inputs or disabled.

The following table defines the bit-field map for **PR** and **PD** commands.

The bit field that configures internal pull-up/down resistors status for I/O lines. If you set a **PR** bit to 1, it enables the internal pull-up/down resistor, 0 specifies no internal pull-up/down. The following table defines the bit-field map for both the **PR** and **PD** commands.

Bit	I/O line	Module pin
0	DIO4/AD4	26
1	DIO3/AD3	24
2	DIO2/AD2	22
3	DIO1/AD1	20
4	DIO0/AD0	18

Bit	I/O line	Module pin
5	DIO6/$\overline{\text{RTS}}$	41
6	DIO8/$\overline{\text{DTR}}$/SLEEP_REQUEST	30
7	DIO14/DIN/$\overline{\text{CONFIG}}$	45
8	DIO5/ASSOCIATE	28
9	DIO9/On/$\overline{\text{SLEEP}}$	32
10	DIO12	40
11	DIO10/RSSI/PWM0	36
12	DIO11/PWM1	38
13	DIO7/$\overline{\text{CTS}}$	39
14	DIO13/DOUT	43
15	DIO15/SPI_MISO	35
16	DIO16/SPI_MOSI	33
17	DIO17/SPI_$\overline{\text{SSEL}}$	31
18	DIO18/SPI_SCLK	37
19	DIO19/SPI_$\overline{\text{ATTN}}$	29

Parameter range

0 - 0xFFFFF (bit field)

Default

0xFFFFF

If **PR** is set to 0x41F, and **PD** is set to 0xFF8, then DIO4, DIO3, and DIO2 have a pull-down resistor enabled, while DIO1, DIO0, and DIO12 have a pull-up resistor enabled. These pins are only affected if they are configured as digital inputs.

M0 (PWM0 Duty Cycle)

The duty cycle of the PWM0 line (pin 7).

Use the **P0** command to configure the line as a PWM output.

Parameter range

0 - 0x3FF

Default

0

M1 (PWM1 Duty Cycle)

The duty cycle of the PWM1 line (pin 8).

Use the **P1** command to configure the line as a PWM output.

Parameter range

0 - 0x3FF

Default

0

LT (Associated LED Blink Time)

Set or read the Associate LED blink time. If you use the **D5** command to enable the Associate LED functionality (DIO5/Associate pin), this value determines the on and off blink times for the LED when the device has joined the network.

If **LT** = **0**, the device uses the default blink rate: 500 ms for a sleep coordinator, 250 ms for all other nodes.

Set or read the Associate LED blink time. If you use the **D5** command to enable the Associate LED functionality (DIO5/Associate pin), this value determines the on and off blink times for the LED. The XBee SX 868 RF Module does not use network authentication or synchronized sleep support, so the Associate LED steadily blinks regardless of the current network status.

Parameter range

0x14 - 0xFF (x 10 ms)

0, 0x14 - 0xFF (x 10 ms)

Default

0

RP (RSSI PWM Timer)

The PWM timer expiration in 0.1 seconds. **RP** sets the duration of pulse width modulation (PWM) signal output on the RSSI pin.

When **RP** = **0xFF**, the output is always on.

Parameter range

0 - 0xFF (x 100 ms)

Default

0x28 (four seconds)

I/O sampling commands

The following AT commands configure I/O sampling parameters.

AV (Analog Voltage Reference)

The analog voltage reference used for A/D sampling.

Parameter range

0 - 2

0, 1

Parameter	Description
0	1.25 V reference
1	2.5 V reference

Default

0

IC (DIO Change Detection)

Set or read the digital I/O pins to monitor for changes in the I/O state.

IC works with the individual pin configuration commands (**D0** - **D9**, **P0** - **P2**) . If you enable a pin as a digital I/O, you can use the **IC** command to force an immediate I/O sample transmission when the DIO state changes. IC is a bitmask that you can use to enable or disable edge detection on individual channels.

Set unused bits to 0.

Bit	I/O line
0	DIO0
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	DIO5
6	DIO6
7	DIO7
8	DIO8
9	DIO9
10	DIO10
11	DIO11
12	DIO12

Bit	I/O line	Module pin
0	DIO0	18
1	DIO1	20
2	DIO2	22
3	DIO3	24

Bit	I/O line	Module pin
4	DIO4	26
5	DIO5	28
6	DIO6	41
7	DIO7	39
8	DIO8	30
9	DIO9	32
10	DIO10	36
11	DIO11	38
12	DIO12	40

Parameter range

0 - 0xFFFF (bit field)

Default

0

IF (Sleep Sample Rate)

Set or read the number of sleep cycles that must elapse between periodic I/O samples. This allows the firmware to take I/O samples only during some wake cycles. During those cycles, the firmware takes I/O samples at the rate specified by **IR**.

Parameter range

1 - 0xFF

Default

1

t b

IR (I/O Sample Rate)

Set or read the I/O sample rate to enable periodic sampling.

If you set the I/O sample rate to greater than **0**, the device samples all enabled digital I/O and analog inputs at a specified interval. Samples are sent to the address specified by the **DH** and **DL** commands.

To enable periodic sampling, set **IR** to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin. The sample rate is measured in milliseconds.

WARNING! If you set **IR** to 1 or 2, the device will not keep up and many samples will be lost.

Set or read the I/O sample rate to enable periodic sampling.

When set, this parameter causes the device to sample all enabled digital I/O and analog inputs at a specified interval. Samples will be sent to the address specified by the **DH** and **DL** commands. The target device must be operating in API mode in order to output the received sample data.

To enable periodic sampling, set **IR** to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin (**D0** – **D9**, **P0** – **P9**).

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

TP (Temperature)

The current module temperature in degrees Celsius in 8-bit two's compliment format. For example 0x1A = 26 ℃, and 0xF6 = -10 ℃.

Parameter range

0 - 0xFF [read-only]

Default

N/A

IS (Force Sample)

Forces a read of all enabled digital and analog input lines.

Parameter range

N/A

Default

N/A

%V (Voltage Supply Monitoring)

Displays the supply voltage of the device in mV units.

Parameter range

This is a read-only parameter

0 - 0xFFFF [read-only]

Default

N/A

I/O line passing commands

The following AT commands are I/O line passing commands.

I/O Line Passing allows the digital and analog inputs of a remote device to affect the corresponding outputs of the local device.

You can perform Digital Line Passing on any of the Digital I/O lines. Digital Inputs directly map to Digital Outputs of each digital pin.

Analog Line Passing can be performed only on the first two ADC lines:

- ADC0 corresponds with PWM0
- ADC1 corresponds with PWM1

IU (I/O Output Enable)

Enable or disable I/O data received to be sent out UART/SPI using an API frame when **AP** = 1 or 2 and when I/O line passing is enabled.

Enable or disable the serial output of received I/O sample data when I/O line passing is enabled. **IU** only affects the device's behavior when **IA** is set to a non-default value.

When **IU** is enabled, any received I/O sample data is sent out the UART/SPI interface using an API frame. Sample data is only generated if the local device is operating in API mode (**AP** = 1 or 2).

Parameter range

0 – 0xFFFF [read-only]

0 - 1

Parameter	Description
0	Disabled
1	Enabled

Default

1

N/A

IA (I/O Input Address)

The source address of the device to which outputs are bound. Setting all bytes to 0xFF disables I/O line passing. Setting **IA** to 0xFFFF allows any I/O packet addressed to this device (including broadcasts) to change the outputs.

The source address of the device to which outputs are bound. If an I/O sample is received from the address specified, any pin that is configured as a digital output or PWM changes its state to match that of the I/O sample.

Set **IA** to 0xFFFFFFFFFFFFFFFF to disable I/O line passing.

Set **IA** to 0xFFFF to allow any I/O packet addressed to this device (including broadcasts) to change the outputs.

Parameter range

0 - 0xFFFF FFFF FFFF FFFF

Default

0xFFFFFFFFFFFFFFFF (I/O line passing disabled)

T0 (D0 Timeout)

Specifies how long pin D0 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T1 (D1 Output Timeout)

Specifies how long pin D1 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T2 (D2 Output Timeout)

Specifies how long pin D2 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T3 (D3 Output Timeout)

Specifies how long pin D3 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T4 (D4 Output Timeout)

Specifies how long pin D4 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T5 (D5 Output Timeout)

Specifies how long pin D5 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T6 (D6 Output Timeout)

Specifies how long pin D6 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T7 (D7 Output Timeout)

Specifies how long pin D7 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T8 (D8 Timeout)

Specifies how long pin D8 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T9 (D9 Timeout)

Specifies how long pin D9 holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q0 (P0 Timeout)

Specifies how long pin **P0** holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q1 (P1 Timeout)

Specifies how long pin P1 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q2 (P2 Timeout)

Specifies how long pin P2 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q3 (P3 Timeout)

Specifies how long pin P3 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q4 (P4 Timeout)

Specifies how long pin P4 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

PT (PWM Output Timeout)

Specifies how long both PWM outputs (**P0**, **P1**) output a given PWM signal before it reverts to zero. If set to 0, there is no timeout. This timeout only affects these pins when they are configured as PWM output.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0xFF

Sleep commands

The following AT commands are sleep commands.

SM (Sleep Mode)

Sets or displays the sleep mode of the device.

Parameter range

0, 1, 4, 5, 7, 8

Parameter	Description
0	Normal - always awake.
1	Pin sleep. In this mode, the sleep/wake state of the module is controlled by the SLEEP_REQUEST line.
4	Asynchronous Cyclic Sleep. In this mode, the device periodically sleeps and wakes based on the **SP** and **ST** commands.
5	Asynchronous cyclic sleep with pin wake-up. In this mode, the device behaves similar to asynchronous cyclic sleep, but the device also terminates a sleep period when it detects a falling edge of the SLEEP_REQUEST line.
7	Sleep Support
8	Synchronized Cyclic Sleep

Default

0

SO (Sleep Options)

Set or read the sleep options bit field of a device. This command is a bitmask.

You cannot set bit 0 and bit 1 at the same time.

Parameter range

0 - 0x13E

For synchronous sleep devices, the following sleep bit field options are defined:

Bit	Option
0	Preferred sleep coordinator; setting this bit causes a sleep compatible device to always act as sleep coordinator
1	Non-sleep coordinator; setting this bit causes a device to never act as a sleep coordinator
2	Enable API sleep status messages
3	Disable early wake-up for missed syncs
4	Enable node type equality (disables seniority based on device type)
5	Disable coordinator rapid sync deployment mode
8	Always wake for **ST** time.

Default

0x2 (non-sleep coordinator)

SN (Number of Sleep Periods)

Set or read the number of sleep periods value. This command controls the number of sleep periods that must elapse between assertions of the ON_SLEEP line during the wake time of Asynchronous Cyclic Sleep.

During cycles when ON_SLEEP is de-asserted, the device wakes up and checks for any serial or RF data. If it receives any such data, then it asserts the ON_SLEEP line and the device wakes up fully. Otherwise, the device returns to sleep after checking.

This command does not work with synchronous sleep devices.

Parameter range

1 - 0xFFFF

Default

1

SP (Sleep Period)

Sets or displays the device's sleep time. This command defines the amount of time the device sleeps per cycle.

For a node operating as an Indirect Messaging Coordinator, this command defines the amount of time that it will hold an indirect message for an end device. The coordinator will hold the message for (2.5 * **SP**).

Parameter range

0x1 - 0x15F900 (x 10 ms)

Default

0x190 (4 seconds)

ST (Wake Time)

Sets or displays the wake time of the device.

For devices in asynchronous sleep, **ST** defines the amount of time that a device stays awake after it receives RF or serial data.

For devices in synchronous sleep, **ST** defines the amount of time that a device stays awake when operating in cyclic sleep mode. The command adjusts the value upwards automatically if it is too small to function properly based on other settings.

Parameter range

0x1 - 0x36EE80 (x 1 ms)

Default

0x1F40 (8 seconds)

WH (Wake Host)

Sets or displays the wake host timer value.

If you set **WH** to a non-zero value, this timer specifies a time in milliseconds that the device delays after waking from sleep before sending data out the UART or transmitting an I/O sample. If the device receives serial characters, the **WH** timer stops immediately.

When in synchronous sleep, the device shortens its sleep period by the **WH** value to ensure it is prepared to communicate when the network wakes up. When in this sleep mode, the device always stays awake for the **WH** time plus the amount of time it takes to transmit a one-hop unicast to another node.

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

Diagnostic - sleep status/timing commands

The following AT commands are Diagnostic sleep status/timing commands.

SS (Sleep Status)

Queries a number of Boolean values that describe the device's status.

Bit	Description
0	This bit is true when the network is in its wake state.
1	This bit is true if the node currently acts as a network sleep coordinator.
2	This bit is true if the node ever receives a valid sync message after it powers on.
3	This bit is true if the node receives a sync message in the current wake cycle.
4	This bit is true if you alter the sleep settings on the device so that the node nominates itself and sends a sync message with the new settings at the beginning of the next wake cycle.
5	This bit is true if you request that the node nominate itself as the sleep coordinator using the Commissioning Pushbutton or the **CB**2 command.
6	This bit is true if the node is currently in deployment mode.
All other bits	Reserved. Ignore all non-documented bits.

Parameter range

[read-only]

Default

0x40

OS (Operating Sleep Time)

Reads the current network sleep time that the device is synchronized to, in units of 10 milliseconds. If the device has not been synchronized, then **OS** returns the value of **SP**.

If the device synchronizes with a sleeping router network, **OS** may differ from **SP**.

Parameter range

[read-only]

Default

0x190

OW (Operating Wake Time)

Reads the current network wake time that a device is synchronized to, in 1 ms units.

If the device has not been synchronized, then **OW** returns the value of **ST**.

If the device synchronizes with a sleeping router network, **OW** may differ from **ST**.

Parameter range

[read-only]

Default

0x1F40

MS (Missed Sync Messages)

Reads the number of sleep or wake cycles since the device received a sync message.

Parameter range

[read-only]

Default

0

SQ (Missed Sleep Sync Count)

Counts the number of sleep cycles in which the device does not receive a sleep sync.

Set the value to 0 to reset this value.

When the value reaches 0xFFFF it does not increment anymore.

Parameter range

0 - 0xFFFF

Default

0

Command mode options

The following commands are Command mode option commands.

CC (Command Sequence Character)

Sets or displays the character the device uses between guard times of the Command mode sequence. The Command mode sequence causes the device to enter Command mode (from Idle mode).

Note We recommend using the a value within the rage of 0x20 - 0x7F as those are ASCII characters.

Parameter range

0 - 0xFF

Default

0x2B (the ASCII plus character: **+**)

CT (Command Mode Timeout)

Sets or displays the Command mode timeout parameter. If a device does not receive any valid commands within this time period, it returns to Idle mode from Command mode.

Parameter range

2 - 0x1770 (x 100 ms)

Default

0x64 (10 seconds)

CN (Exit Command Mode)

Immediately exits Command Mode and applies pending changes.

Parameter range

N/A

Default

N/A

GT (Guard Times)

Set the required period of silence before and after the command sequence characters of the Command mode sequence (**GT** + **CC** + **GT**). The period of silence prevents inadvertently entering Command mode.

Parameter range

0x2 - 0x95C (x 1 ms)

Default

0x3E8 (one second)

Firmware commands

The following AT commands are firmware commands.

VL (Version Long)

Shows detailed version information including the application build date and time.

Parameter range

[read-only]

Default

N/A

VR (Firmware Version)

Reads the firmware version on a device.

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in firmware

HV (Hardware Version)

Display the hardware version number of the device.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in firmware

HS (Hardware Series)

Read the device's hardware series number.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in the firmware

DD (Device Type Identifier)

Stores the Digi device type identifier value. Use this value to differentiate between multiple XBee devices.

If you change **DD**, RE (Restore Defaults) will not restore defaults. The only way to get **DD** back to default values is to explicitly set it to defaults.

Parameter range

0 - 0xFFFFFFFF

Default

0x110000

NP (Maximum Packet Payload Bytes)

Reads the maximum number of RF payload bytes that you can send in a transmission.

Parameter range

0 - 0xFFFF (bytes) [read-only]

Default

0x100

CK (Configuration CRC)

Displays the cyclic redundancy check (CRC) of the current AT command configuration settings.

This command allows you to detect an unexpected configuration change on a device.

Parameter range

N/A

Default

N/A

Operate in API mode

API mode overview

As an alternative to Transparent operating mode, you can use API operating mode. API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol. The API specifies how commands, command responses and device status messages are sent and received from the device using the serial interface or the SPI interface.

We may add new frame types to future versions of firmware, so build the ability to filter out additional API frames with unknown frame types into your software interface.

Use the AP command to set the operation mode

Use AP (API Mode) to specify the operation mode:

AP command setting	Description
AP = 0	Transparent operating mode, UART serial line replacement with API modes disabled. This is the default option.
AP = 1	API operation.
AP = 2	API operation with escaped characters (only possible on UART).

The API data frame structure differs depending on what mode you choose.

API frame format

An API frame consists of the following:

- Start delimeter
- Length
- Frame data
- Checksum

API operation (AP parameter = 1)

This is the recommended API mode for most applications. The following table shows the data frame structure when you enable this mode:

Frame fields	Byte	Description
Start delimiter	1	0x7E
Length	2 - 3	Most Significant Byte, Least Significant Byte
Frame data	4 - number (n)	API-specific structure
Checksum	n + 1	1 byte

Any data received prior to the start delimiter is silently discarded. If the frame is not received correctly or if the checksum fails, the XBee replies with a radio status frame indicating the nature of the failure.

API operation with escaped characters (AP parameter = 2)

Setting API to 2 allows escaped control characters in the API frame. Due to its increased complexity, we only recommend this API mode in specific circumstances. API 2 may help improve reliability if the serial interface to the device is unstable or malformed frames are frequently being generated.

When operating in API 2, if an unescaped 0x7E byte is observed, it is treated as the start of a new API frame and all data received prior to this delimiter is silently discarded. For more information on using this API mode, see the Escaped Characters and API Mode 2 in the Digi Knowledge base.

API escaped operating mode works similarly to API mode. The only difference is that when working in API escaped mode, the software must escape any payload bytes that match API frame specific data, such as the start-of-frame byte (0x7E). The following table shows the structure of an API frame with escaped characters:

Frame fields	Byte	Description	
Start delimiter	1	0x7E	
Length	2 - 3	Most Significant Byte, Least Significant Byte	Characters escaped if needed
Frame data	4 - n	API-specific structure	
Checksum	n + 1	1 byte	

Start delimiter field

This field indicates the beginning of a frame. It is always 0x7E. This allows the device to easily detect a new incoming frame.

Escaped characters in API frames

If operating in API mode with escaped characters (**AP** parameter = 2), when sending or receiving a serial data frame, specific data values must be escaped (flagged) so they do not interfere with the data frame sequencing. To escape an interfering data byte, insert 0x7D and follow it with the byte to be escaped (XORed with 0x20).

The following data bytes need to be escaped:

- 0x7E: start delimiter
- 0x7D: escape character
- 0x11: XON
- 0x13: XOFF

To escape a character:

1. Insert 0x7D (escape character).
2. Append it with the byte you want to escape, XORed with 0x20.

In API mode with escaped characters, the length field does not include any escape characters in the frame and the firmware calculates the checksum with non-escaped data.

Example: escape an API frame

To express the following API non-escaped frame in API operating mode with escaped characters:

Start delimiter	Length	Frame type	Frame Data	Checksum
			Data	
7E	00 0F	17	01 00 13 A2 00 40 AD 14 2E FF FE 02 4E 49	6D

You must escape the 0x13 byte:

1. Insert a 0x7D.
2. XOR byte 0x13 with 0x20: 13 ⊕ 20 = 33

The following figure shows the resulting frame. Note that the length and checksum are the same as the non-escaped frame.

Start delimiter	Length	Frame type	Frame Data	Checksum
			Data	
7E	00 0F	17	01 00 7D 33 A2 00 40 AD 14 2E FF FE 02 4E 49	6D

The length field has a two-byte value that specifies the number of bytes in the frame data field. It does not include the checksum field.

Length field

The length field is a two-byte value that specifies the number of bytes contained in the frame data field. It does not include the checksum field.

Frame data

This field contains the information that a device receives or will transmit. The structure of frame data depends on the purpose of the API frame:

Start delimiter	Length		Frame data							Checksum
			Frame type	Data						
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Data							Single byte

- **Frame type** is the API frame type identifier. It determines the type of API frame and indicates how the Data field organizes the information.
- **Data** contains the data itself. This information and its order depend on the what type of frame that the Frame type field defines.

Multi-byte values are sent big-endian.

Calculate and verify checksums

To calculate the checksum of an API frame:

1. Add all bytes of the packet, except the start delimiter 0x7E and the length (the second and third bytes).
2. Keep only the lowest 8 bits from the result.
3. Subtract this quantity from 0xFF.

To verify the checksum of an API frame:

1. Add all bytes including the checksum; do not include the delimiter and length.
2. If the checksum is correct, the last two digits on the far right of the sum equal 0xFF.

Example

Consider the following sample data packet: **7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8**+

Byte(s)	Description
7E	Start delimiter
00 0A	Length bytes
01	API identifier
01	API frame ID
50 01	Destination address low
00	Option byte
48 65 6C 6C 6F	Data packet
B8	Checksum

To calculate the check sum you add all bytes of the packet, excluding the frame delimiter **7E** and the length (the second and third bytes):

7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8

Add these hex bytes:

01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F = 247

Now take the result of 0x247 and keep only the lowest 8 bits which in this example is 0xC4 (the two far right digits). Subtract 0x47 from 0xFF and you get 0x3B (0xFF - 0xC4 = 0x3B). 0x3B is the checksum for this data packet.

If an API data packet is composed with an incorrect checksum, the XBee SX 868 RF Module will consider the packet invalid and will ignore the data.

To verify the check sum of an API packet add all bytes including the checksum (do not include the delimiter and length) and if correct, the last two far right digits of the sum will equal FF.

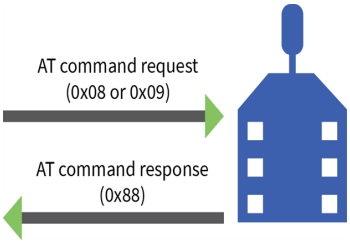
01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F + B8 = 2FF

API serial exchanges

You can use the Frame ID field to assign an identifier to each outgoing API frame. This Frame ID, if non-zero, can correlate between the outgoing frames and the associated responses.

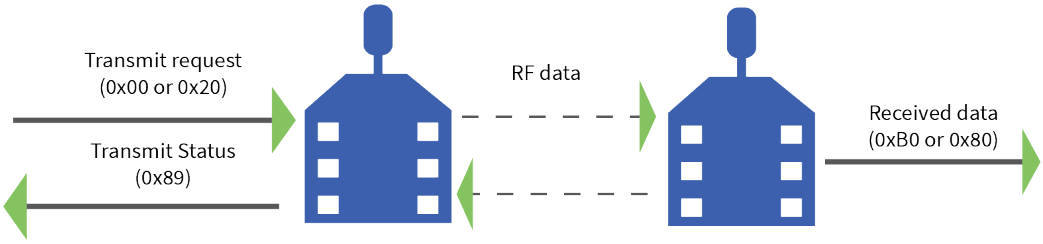
AT command frames

The following image shows the API frame exchange that takes place at the interface when sending an AT command request to read or set an XBee parameter. To disable the response, set the frame ID to 0 in the request.



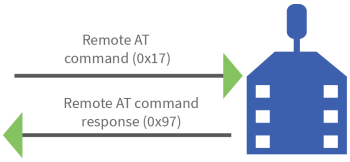
Transmit and receive RF data

The following image shows the API exchanges that take place at the serial interface when sending RF data to another device. The transmit status frame is always sent at the end of a data transmission unless the frame ID is set to 0 in the TX request. If the packet cannot be delivered to the destination, the transmit status frame indicates the cause of failure. The received data frame type (standard 0x90, or explicit 0x91) is set by the **AP** command.



Remote AT commands

The following image shows the API frame exchanges that take place at the serial interface when sending a remote AT command. A remote command response frame is not sent out the serial interface if the remote device does not receive the remote command.



Frame descriptions

The following sections describe the API frames.

AT Command frame - 0x08

Description

Use this frame to query or set device parameters on the local device. This API command applies changes after running the command. You can query parameter values by sending the 0x08 AT Command frame with no parameter value field (the two-byte AT command is immediately followed by the frame checksum).

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x08
Frame ID	4	Identifies the UART data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
AT command	5-6	Command name: two ASCII characters that identify the AT command.
Parameter value	7-n	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the register.

Example

The following example illustrates an AT Command frame when you query the device's **NH** parameter value.

The following example illustrates an AT Command frame where the baud rate (**BD**) for the device is set to 1200.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x04
Frame type	3	0x08

Frame data fields	Offset	Example
Frame ID	4	0x52 (R)
AT command	5	0x4E (N)
	6	0x48 (H)
Parameter value (optional)		
Checksum	7	0x0F

AT Command - Queue Parameter Value frame - 0x09

Description

This frame allows you to query or set device parameters. In contrast to the AT Command (0x08) frame, this frame queues new parameter values and does not apply them until you issue either:

- The **AT** Command (0x08) frame (for API type)
- The **AC** command

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x09
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
AT command	5-6	Command name: two ASCII characters that identify the AT command.
Parameter value	7-n	If present, indicates the requested parameter value to set the given register. If no characters are present, queries the register.

Example

The following example sends a command to change the baud rate (**BD**) to 115200 baud, but does not apply the changes immediately. The device continues to operate at the previous baud rate until you apply the changes.

Note In this example, you could send the parameter as a zero-padded 2-byte or 4-byte value.

Frame data fields		
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05
Frame type	3	0x09
Frame ID	4	0x01

Frame data fields		
AT command	5	0x42 (B)
	6	0x44 (D)
Parameter value (**BD**7 = 115200 baud)		0x07
Checksum	8	0x68

Transmit Request frame - 0x10

Description

This frame causes the device to send payload data as an RF packet to a specific destination.

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF** .
- For unicast transmissions, set the 64 bit address field to the address of the desired destination node.
- Set the reserved field to **0xFFFE**.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x10
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
64-bit destination address	5-12	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF
Reserved	13-14	Set to 0xFFFE.
Broadcast radius	15	Sets the maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius is set to the maximum hops value.
Transmit options	16	See the following Transmit Options table.
RF data	17-n	Up to **NP** bytes per packet. Sent to the destination device.

Transmit Options bit field

Bit	Meaning	Description
0	Disable ACK	Disable acknowledgments on all unicasts.
1	Disable RD	Disable Route Discovery.
2	NACK	Enable unicast NACK messages.
3	Trace route	Enable unicast trace route messages.

Bit	Meaning	Description
6,7	Delivery method	b'00 = <invalid option> b'01 - Point-multipoint b'10 = Repeater mode (directed broadcast) b'11 = DigiMesh (not available on the 10k product)

Example

The example shows how to send a transmission to a device if you disable escaping (**AP** = 1), with destination address 0x0013A200 400A0127, and payload "TxData0A".

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x16
Frame type	3	0x10
Frame ID	4	0x01
64-bit destination address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x40
	10	0x0A
	11	0x01
	LSB 12	0x27
Reserved	MSB 13	0xFF
	LSB 14	0xFE
Broadcast radius	15	0x00
Options	16	0x40[1]

[1]Whenever the Options field is set to **0**, it is ignored, and the options are determined by the **TO** command.

Frame data fields	Offset	Example
RF data	17	0x54
	18	0x78
	19	0x44
	20	0x61
	21	0x74
	22	0x61
	23	0x30
	24	0x41
Checksum	25	0x13

If you enable escaping (**AP** = 2), the frame should look like:

0x7E 0x00 0x16 0x10 0x01 0x00 0x7D 0x33 0xA2 0x00 0x40 0x0A 0x01 0x27 0xFF 0xFE 0x00 0x00 0x54 0x78 0x44 0x61 0x74 0x61 0x30 0x41 0x7D 0x33

The device calculates the checksum (on all non-escaped bytes) as [0xFF - (sum of all bytes from API frame type through data payload)].

Explicit Addressing Command frame - 0x11

Description

This frame is similar to Transmit Request (0x10), but it also requires you to specify the application-layer addressing fields: endpoints, cluster ID, and profile ID.

This frame causes the device to send payload data as an RF packet to a specific destination, using specific source and destination endpoints, cluster ID, and profile ID.

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF** .
- For unicast transmissions, set the 64 bit address field to the address of the desired destination node.
- For all other transmissions, setting the 16-bit address to the correct 16-bit address helps improve performance when transmitting to multiple destinations. If you do not know a 16-bit address, set this field to 0xFFFE (unknown). If successful, the Transmit Status frame (0x8B) indicates the discovered 16-bit address.
- Set the reserved field to **0xFFFE**.

You can set the broadcast radius from 0 up to **NH** to 0xFF. If the broadcast radius exceeds the value of **NH**, the devices uses the value of **NH** as the radius. This parameter is only used for broadcast transmissions.

You can read the maximum number of payload bytes with the **NP** command.

Format

The following table provides the contents of the frame. For details on the frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x11
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
64-bit destination address	5-12	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF
Reserved	13-14	Set to 0xFFFE.
Source endpoint	15	Source endpoint for the transmission.
Destination endpoint	16	Destination endpoint for the transmission.
Cluster ID	17-18	The Cluster ID that the host uses in the transmission.
Profile ID	19-20	The Profile ID that the host uses in the transmission.

Frame data fields	Offset	Description
Broadcast radius	21	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius set to the network maximum hops value.
Transmission options	22	Bitfield: bits 6,7: b'01 - Point-to-Multipoint b'10 - Repeater mode (directed broadcast) b'11 - DigiMesh (not available on 10k product) All other bits must be set to 0.
Data payload	23-n	

Example

The following example sends a data transmission to a device with:

- 64-bit address: 0x0013A200 01238400
- Source endpoint: 0xE8
- Destination endpoint: 0xE8
- Cluster ID: 0x11
- Profile ID: 0xC105
- Payload: TxData

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x1A
Frame type	3	0x11
Frame ID	4	0x01
64-bit destination address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x01
	10	0x23
	11	0x84
	LSB12	0x00

Frame data fields	Offset	Example
Reserved	MSB 13	0xFF
	LSB 14	0xFE
Source endpoint	15	0xA0
Destination endpoint	16	0xA1
Cluster ID	17	0x15
	18	0x54
Profile ID	19	0xC1
	20	0x05
Broadcast radius	21	0x00
Transmit options	22	0x00
Data payload	23	0x54
	24	0x78
	25	0x44
	26	0x61
	27	0x74
	28	0x61
Checksum	29	0xDD

Remote AT Command Request frame - 0x17

Description

Used to query or set device parameters on a remote device. For parameter changes on the remote device to take effect, you must apply changes, either by setting the Apply Changes options bit, or by sending an **AC** command to the remote.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x17
Frame ID	4	Identifies the UART data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
64-bit destination address	5-12	MSB first, LSB last. Set to the 64-bit address of the destination device.
Reserved	13-14	
Remote command options	15	0x02 = Apply changes on remote. If you do not set this, you must send an **AC** command on the remote device for changes to take effect. Set all other bits to 0.
AT command	16-17	Command name: two ASCII characters that identify the command.
Command parameter	18-n	If present, indicates the parameter value you request for a given register. If no characters are present, it queries the register.

Example

The following example sends a remote command:

In this example, the 64-bit address of the remote device is 0x0013A200 40401122.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x10
Frame type	3	0x17

Frame data fields	Offset	Example
Frame ID	4	0x01
64-bit destination address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x40
	10	0x40
	11	0x11
	LSB 12	0x22
Reserved	13	0xFF
	14	0xFE
Remote command options	15	0x02 (apply changes)
AT command	16	0x42 (B)
	17	0x48 (H)
Command parameter	18	0x01
Checksum	19	0xF5

AT Command Response frame - 0x88

Description

A device sends this frame in response to an AT Command (0x08 or 0x09) frame. Some commands send back multiple frames; for example, the **ND** command.

Format

Frame data fields	Offset	Description
Frame type	3	0x88
Frame ID	4	Identifies the serial port data frame being reported. If Frame ID = 0 in**AT**Command Mode, the device does not give an**AT**Command Response.
AT command	5-6	Command name: two ASCII characters that identify the command.
Command status	7	0 = OK 1 = ERROR 2 = Invalid command 3 = Invalid parameter The most significant nibble is a bitfield as follows: 0x40 = The RSSI field is invalid and should be ignored. Software prior to version 8x60 did not include RSSI information. 0x80 = Response is a remote command.
Command data		The register data in binary format. If the host sets the register, the device does not return this field.

Example

If you change the **BD** parameter on a local device with a frame ID of 0x01, and the parameter is valid, the user receives the following response.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05
Frame type	3	0x88
Frame ID	4	0x01

Frame data fields	Offset	Example
AT command	5	0x42 (B)
	6	0x44 (D)
Command status	7	0x00
Command data		(No command data implies the parameter was set rather than queried)
Checksum	8	0xF0

Modem Status frame - 0x8A

Description

Devices send the status messages in this frame in response to specific conditions.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x8A
Status	4	0x00 = Hardware reset 0x01 = Watchdog timer reset 0x0B = Network woke up 0x0C = Network went to sleep

Example

When a device powers up, it returns the following API frame.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
LSB 2	LSB 2	0x02
Frame type	3	0x8A
Status	4	0x00
Checksum	5	0x75

Transmit Status frame - 0x8B

Description

When a Transmit Request (0x10, 0x11) completes, the device sends a Transmit Status message out of the serial interface. This message indicates if the Transmit Request was successful or if it failed.

Note Broadcast transmissions are not acknowledged and always return a status of 0x00, even if the delivery failed.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x8B
Frame ID	4	Identifies the serial interface data frame being reported. If Frame ID = 0 in the associated request frame, no response frame is delivered.
Reserved	5-6	Set to 0xFFFE.
Transmit retry count	7	The number of application transmission retries that occur.
Delivery status	8	0x00 = Success 0x01 = MAC ACK failure 0x02 = Collision avoidance failure 0x02 = LBT Failure 0x03 = No Spectrum Available 0x21 = Network ACK failure 0x25 = Route not found 0x31 = Internal resource error 0x32 = Internal error 0x74 = Payload too large 0x75 = Indirect message requested
Discovery status	9	0x00 = No discovery overhead 0x02 = Route discovery

Example

In the following example, the destination device reports a successful unicast data transmission. The outgoing Transmit Request that this response frame uses Frame ID of 0x47.

Frame Fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x07
Frame type	3	0x8B
Frame ID	4	0x47
Reserved	5	0xFF
	6	0xFE
Transmit retry count	7	0x00
Delivery status	8	0x00
Discovery status	9	0x02
Checksum	10	0x2E

Route Information Packet frame - 0x8D

Description

If you enable NACK or the Trace Route option on a DigiMesh unicast transmission, a device can output this frame for the transmission.

Format

Frame data fields	Offset	Description
Frame type	3	0x8D
Source event	4	0x11 = NACK 0x12 = Trace route
Length	5	The number of bytes that follow, excluding the checksum. If the length increases, new items have been added to the end of the list for future revisions.
Timestamp	6-9	MSB first, LSB last. System timer value on the node generating the Route Information Packet.
ACK timeout count	10	The number of MAC ACK timeouts that occur.
TX blocked count	11	The number of times the transmission was blocked due to reception in progress.
Reserved	12	Reserved, set to 0s.
Destination address	13-20	MSB first, LSB last. The address of the final destination node of this network-level transmission.
Source address	21-28	MSB first, LSB last. Address of the source node of this network-level transmission.
Responder address	29-36	MSB first, LSB last. Address of the node that generates this Route Information packet after it sends (or attempts to send) the packet to the next hop (the Receiver node).
Receiver address	37-44	MSB first, LSB last. Address of the node that the device sends (or attempts to send) the data packet.

Example

The following example represents a possible Route Information Packet. A device receives the packet when it performs a trace route on a transmission from one device (serial number 0x0013A200 4052AAAA) to another (serial number 0x0013A200 4052DDDD).

This particular frame indicates that the network successfully forwards the transmission from one device (serial number 0x0013A200 4052BBBB) to another device (serial number 0x0013A200 4052CCCC).

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x2A
Frame type	3	0x8D
Source event	4	0x12
Length	5	
Timestamp	MSB 6	0x9C
	7	0x93
	8	0x81
	LSB 9	0x7F
ACK timeout count	10	0x00
TX blocked count	11	0x00
Reserved	12	0x00
Destination address	MSB 13	0x00
	14	0x13
	15	0xA2
	16	0x00
	17	0x40
	18	0x52
	19	0xAA
	LSB 20	0xAA
Source address	MSB 21	0x00
	22	0x13
	23	0xA2
	24	0x00
	25	0x40
	26	0x52
	27	0xDD
	LSB 28	0xDD

Frame data fields	Offset	Example
Responder address	MSB 29	0x00
	30	0x13
	31	0xA2
	32	0x00
	33	0x40
	34	0x52
	35	0xBB
	LSB 36	0xBB
Receiver address	MSB 37	0x00
	38	0x13
	39	0xA2
	40	0x00
	41	0x40
	42	0x52
	43	0xCC
	LSB 44	0xCC
Checksum	45	0xCE

Aggregate Addressing Update frame - 0x8E

Description

The device sends out an Aggregate Addressing Update frame on the serial interface of an API-enabled node when an address update frame (generated by the **AG** command being issued on a node in the network) causes the node to update its **DH** and **DL** registers.

Format

Frame data fields	Offset	Description
Frame type	3	0x8E
Format ID	4	Byte reserved to indicate the format of additional packet information which may be added in future firmware revisions. In the current firmware revision, this field returns 0x00.
New address	5-12	MSB first, LSB last. Address to which **DH** and **DL** are being set.
Old address	13-20	Address to which **DH** and **DL** were previously set.

Example

In the following example, a device with destination address (**DH**/**DL**) of 0x0013A200 4052AAAA updates its destination address to 0x0013A200 4052BBBB.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x12
Frame type	3	0x8E
Format ID	4	0x00

Frame data fields	Offset	Example
New address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x40
	10	0x52
	11	0xBB
	LSB 12	0xBB
Old address	13	0x00
	14	0x13
	15	0xA2
	16	0x00
	17	0x40
	18	0x52
	19	0xAA
	20	0xAA
Checksum	21	0x2E

Receive Packet frame - 0x90

Description

When a device configured with a standard API Rx Indicator (**AO** = **0**) receives an RF data packet, it sends it out the serial interface using this message type.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x90
64-bit source address	4-11	The sender's 64-bit address. MSB first, LSB last.
Reserved	12-13	Reserved.
Receive options	14	Bit field: 0x00 = Packet acknowledged 0x01 = Packet was a broadcast packet 0x40 = point-multipoint 0x80 = Repeater Mode (Directed Broadcast) 0xC0 = DigiMesh (Not available on the 10k data rate) Option values can be combined, for example: 0x40 and 0x1 will show as 0x41 Ignore all other bits.
Received data	15-n	The RF data the device receives.

Example

In the following example, a device with a 64-bit address of 0x0013A200 40522BAA sends a unicast data transmission to a remote device with payload RxData. If **AO**=0 on the receiving device, it sends the following frame out its serial interface.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x12
Frame type	3	0x90

Frame data fields	Offset	Example
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	12	0xFF
	13	0xFE
Receive options	14	0x01
Received data	15	0x52
	16	0x78
	17	0x44
	18	0x61
	19	0x74
	20	0x61
Checksum	21	0x11

Explicit Rx Indicator frame - 0x91

Description

When a device configured with explicit API Rx Indicator (**AO** = **1**) receives an RF packet, it sends it out the serial interface using this message type.

Note If a Transmit Request frame - 0x10 is sent to a device with **AO** = **1**, the receiving device receives a 0x91 frame with the Source endpoint (SE), Destination endpoint (DE), and Cluster ID (CI) that were set on the transmitting device in Transparent mode, and not the default values.

The Cluster ID and endpoints must be used to identify the type of transaction that occurred.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x91
64-bit source address	4-11	MSB first, LSB last. The sender's 64-bit address.
Reserved	12-13	Reserved.
Source endpoint	14	Endpoint of the source that initiates transmission.
Destination endpoint	15	Endpoint of the destination where the message is addressed.
Cluster ID	16-17	The Cluster ID where the frame is addressed.
Profile ID	18-19	The Profile ID where the fame is addressed.
Receive options	20	Bit field: 0x00 = Packet acknowledged 0x01 = Packet was a broadcast packet 0x40 = point-multipoint 0x80 = Repeater Mode (Directed Broadcast) 0xC0 = DigiMesh (Not available on the 10k data rate) Option values can be combined, for example: 0x40 and 0x1 will show as 0x41 Ignore all other bits.
Received data	21-n	Received RF data.

Example

In the following example, a device with a 64-bit address of 0x0013A200 40522BAA sends a broadcast data transmission to a remote device with payload RxData.

If a device sends the transmission:

- With source and destination endpoints of 0xE0
- Cluster ID = 0x2211
- Profile ID = 0xC105

If **AO** = **1** on the receiving device, it sends the following frame out its serial interface.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x18
Frame type	3	0x91
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	12	0xFF
	13	0xFE
Source endpoint	14	0xE0
Destination endpoint	15	0xE0
Cluster ID	16	0x22
	17	0x11
Profile ID	18	0xC1
	19	0x05
Receive options	20	0x02

Frame data fields	Offset	Example
Received data	21	0x52
	22	0x78
	23	0x44
	24	0x61
	25	0x74
	26	0x61
Checksum	27	0x56

Data Sample Rx Indicator frame - 0x92

Description

When you enable periodic I/O sampling or digital I/O change detection on a remote device, the UART of the device that receives the sample data sends this frame out.

Format

Frame data fields	Offset	Description
Frame type	3	0x92
64-bit source address	4-11	The sender's 64-bit address.
Reserved	12-13	Reserved.
Receive options	14	Bit field: 0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet Ignore all other bits
Number of samples	15	The number of sample sets included in the payload. Always set to 1.
Digital channel mask	16-17	Bitmask field that indicates which digital I/O lines on the remote have sampling enabled, if any.
Analog channel mask	18	Bitmask field that indicates which analog I/O lines on the remote have sampling enabled, if any.
Digital samples (if included)	19-20	If the sample set includes any digital I/O lines (Digital channel mask > 0), these two bytes contain samples for all enabled digital I/O lines. DIO lines that do not have sampling enabled return 0. Bits in these two bytes map the same as they do in the Digital channel mask field.

Frame data fields	Offset	Description
Analog sample	21-n	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from ADO/DIO0 to AD3/DIO3.

Example

In the following example, the device receives an I/O sample from a device with a 64-bit serial number of 0x0013A20040522BAA.

The configuration of the transmitting device takes a digital sample of a number of digital I/O lines and an analog sample of AD1. It reads the digital lines to be 0x0014 and the analog sample value is 0x0225.

The complete example frame is:

7E00 1492 0013 A200 4052 2BAA FFFE 0101 001C 0200 1402 25F9

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x14
Frame-specific data		
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	MSB 12	0xfffe
	LSB 13	0x84
Receive options	14	0x01
Number of samples	15	0x01
Digital channel mask	16	0x00
	17	0x1C
Analog channel mask	18	0x02

Frame fields	Offset	Example
Digital samples (if included)	19	0x00
	20	0x14
Analog sample	21	0x02
	22	0x25
Checksum	23	0xF5

Node Identification Indicator frame - 0x95

Description

A device receives this frame when:

- it transmits a node identification message to identify itself
- **AO** = **0**

The data portion of this frame is similar to a network discovery response. For more information, see ND (Network Discover).

If you press the commissioning push button on a remote router device with 64-bit address 0x0013a200407402ac and default **NI** string, the device receives the following node identification indicator:
0x7e 0025 9500 13a2 0040 7402 acff fec2 fffe 0013 a200 4074 02ac 2000 fffe 0101 c105 101e 000c 0000 2e33

Format

Frame data fields	Offset	Description
Frame type	3	0x95
64-bit source address	4-11	MSB first, LSB last. The sender's 64-bit address.
Reserved	12-13	Reserved.
Receive options	14	Bit field: 0x01 = Packet acknowledged 0x02 = Packet was a broadcast packet 0x40 = Point-multipoint packet 0x80 = Directed broadcast packet 0xC0 = DigiMesh packet Ignore all other bits
Reserved	15-16	Reserved.
64-bit remote address	17-24	MSB first, LSB last. Indicates the 64-bit address of the remote device that transmitted the Node Identification Indicator frame.
NI string	25-26	Node identifier string on the remote device. The NI string is terminated with a NULL byte (0x00).
Reserved	27-28	Reserved.

Frame data fields	Offset	Description
Device type	29	0 = Coordinator 1 = Normal Mode 2 = End Device For more options, see NO (Node Discovery Options).
Source event	30	1 = Frame sent by node identification pushbutton event.
Digi Profile ID	31-32	Set to the Digi application profile ID.
Digi Manufacturer ID	33-34	Set to the Digi Manufacturer ID.
Digi DD value (optional)	35-38	Reports the **DD** value of the responding device. Use the **NO** command to enable this field.
RSSI (optional)	39	Received signal strength indicator. Use the **NO** command to enable this field.

Example

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x25
Frame type	3	0x95
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x74
	10	0x02
	LSB 11	0xAC
Reserved	12	0xFF
	13	0xFE
Receive options	14	0xC2

Frame data fields	Offset	Example
Reserved	15	0xFF
	16	0xFE
64-bit remote address	MSB 17	0x00
	18	0x13
	19	0xA2
	20	0x00
	21	0x40
	22	0x74
	23	0x02
	LSB 24	0xAC
NI string	25	0x20
	26	0x00
Reserved	27	0xFF
	28	0xFE
Device type	29	0x01
Source event	30	0x01
Digi Profile ID	31	0xC1
	32	0x05
Digi Manufacturer ID	33	0x10
	34	0x1E
Digi DD value (optional)	35	0x00
	36	0x0C
	37	0x00
	38	0x00
RSSI (optional)	39	0x2E
Checksum	40	0x33

Remote Command Response frame - 0x97

Description

If a device receives this frame in response to a Remote Command Request (0x17) frame, the device sends an AT Command Response (0x97) frame out the serial interface.

Some commands, such as the **ND** command, may send back multiple frames. For details on the behavior of **ND**, see ND (Network Discover).

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Frame data fields	Offset	Description
Frame type	3	0x97
Frame ID	4	This is the same value passed in to the request. If Frame ID = 0 in the associated request frame the device does not deliver a response frame.
64-bit source (remote) address	5-12	The address of the remote device returning this response.
Reserved	13-14	Reserved.
AT commands	15-16	The name of the command.
Command status	17	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter 4 = Remote command transmission failed
Command data	18-n	The value of the requested register.

Example

If a device sends a remote command to a remote device with 64-bit address 0x0013A200 40522BAA to query the **SL** command, and if the frame ID = 0x55, the response would look like the following example.

Frame data fields	Offset	Example
Start delimiter	0	0x7E

Frame data fields	Offset	Example
Length	MSB 1	0x00
	LSB 2	0x13
Frame type	3	0x97
Frame ID	4	0x55
64-bit source (remote) address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x40
	10	0x52
	11	0x2B
	LSB 12	0xAA
Reserved	13	0xFF
	14	0xFE
AT commands	15	0x53
	16	0x4C
Command status	17	0x00
Command data	18	0x40
	19	0x52
	20	0x2B
	21	0xAA
Checksum	22	0xF4

Regulatory information

Europe (CE)

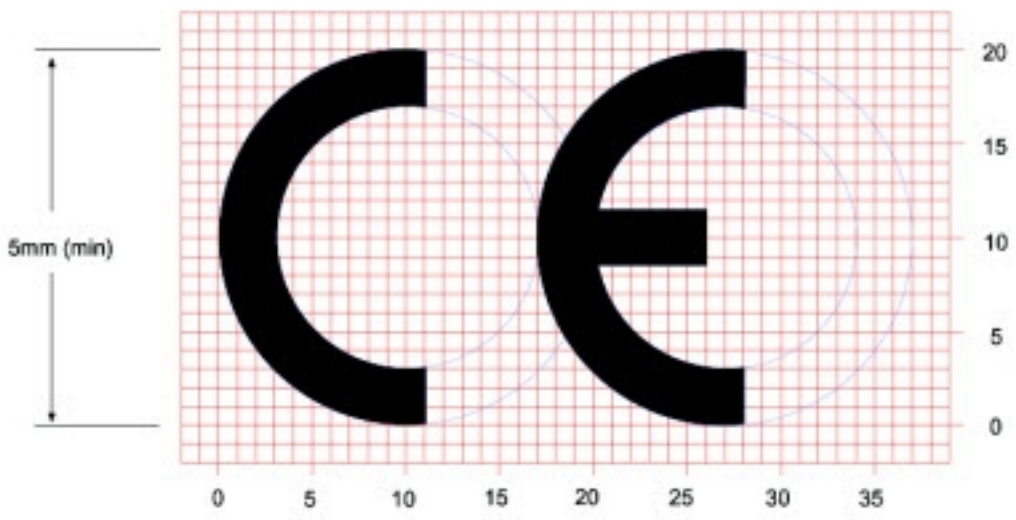
The XBee SX 868 RF Module has been tested for use in several European countries. For a complete list, refer to www.digi.com/resources/certifications.

If XBee SX 868 RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product with articles 3.1a and 3.1b of the Radio Equipment Directive. A Declaration of Conformity must be issued for each of these standards and kept on file as described in the Radio Equipment Directive.

Furthermore, the manufacturer must maintain a copy of the XBee SX 868 RF Module user guide documentation and ensure the final product does not exceed the specified power ratings, antenna specifications, and/or installation requirements as specified in the user guide.

OEM labeling requirements

The “CE” marking must be affixed to a visible location on the OEM product. The following figure shows CE labeling requirements.



The CE mark shall consist of the initials “CE” taking the following form:

- If the CE marking is reduced or enlarged, the proportions given in the above graduated drawing must be respected.
- The CE marking must have a height of at least 5 mm except where this is not possible on account of the nature of the apparatus.
- The CE marking must be affixed visibly, legibly, and indelibly.

Important note

Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. Refer to the radio regulatory agency in the desired countries of operation for more information.

Declarations of conformity

Digi has issued Declarations of Conformity for the XBee RF Modules concerning emissions, EMC, and safety. For more information, see www.digi.com/resources/certifications.

Antennas

The following antennas have been tested and approved for use with the XBee SX 868 RF Module:

Dipole (2.1 dBi), Digi PN A08-HABUF-P5I*

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

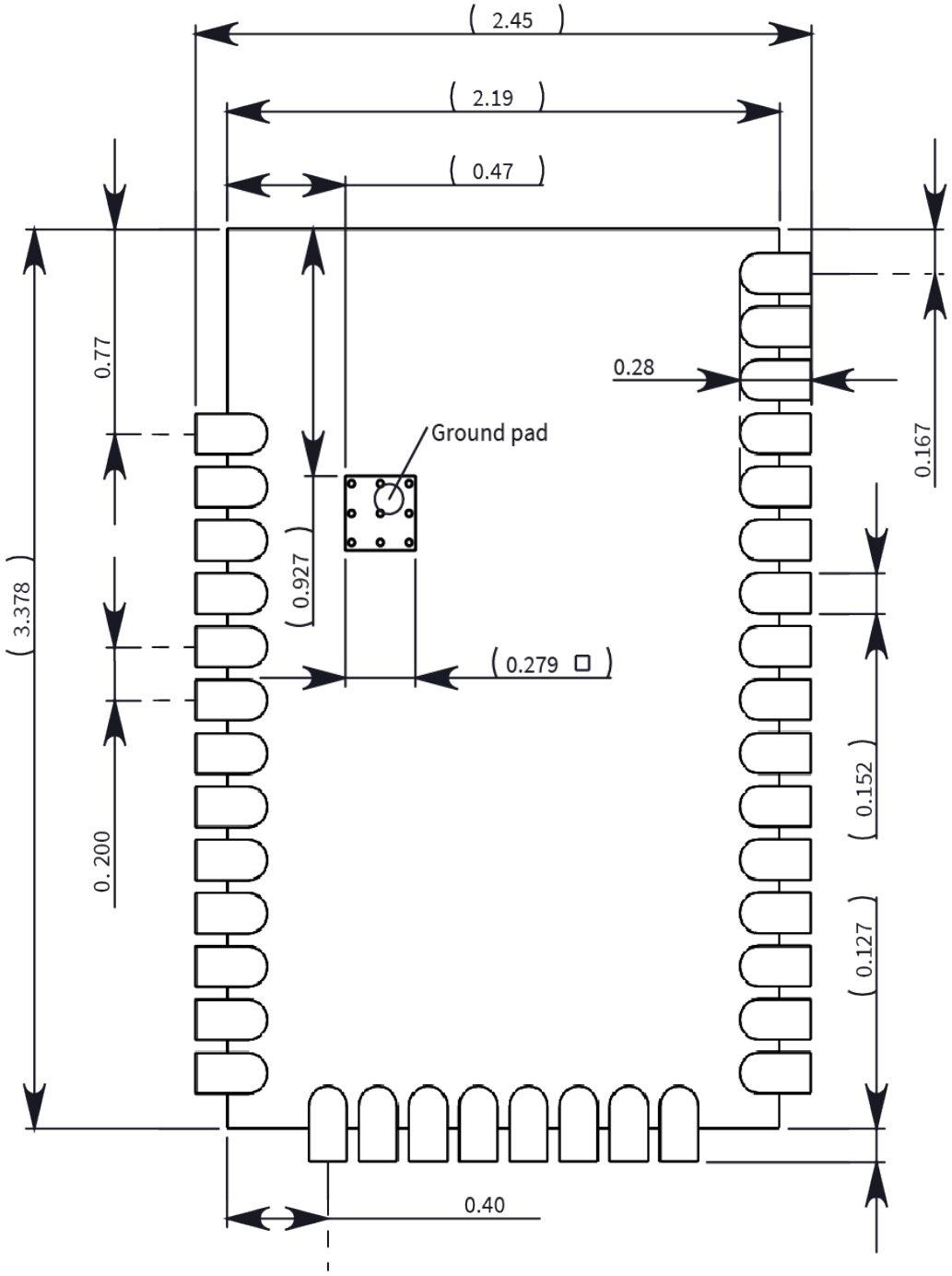
PCB design and manufacturing

The XBee SX 868 RF Module is designed for surface-mount on the OEM PCB. It has castellated pads to allow for easy solder attach inspection. The pads are all located on the edge of the module, so there are no hidden solder joints on these modules.

Recommended footprint and keepout

We designed the XBee SX 868 RF Module for surface-mounting on the OEM printed circuit board (PCB). It has castellated pads around the edges and one ground pad on the bottom. Mechanical drawings includes a detailed mechanical drawing.

We recommend that you use the following PCB footprint for surface-mounting. Dimensions are in centimeters.



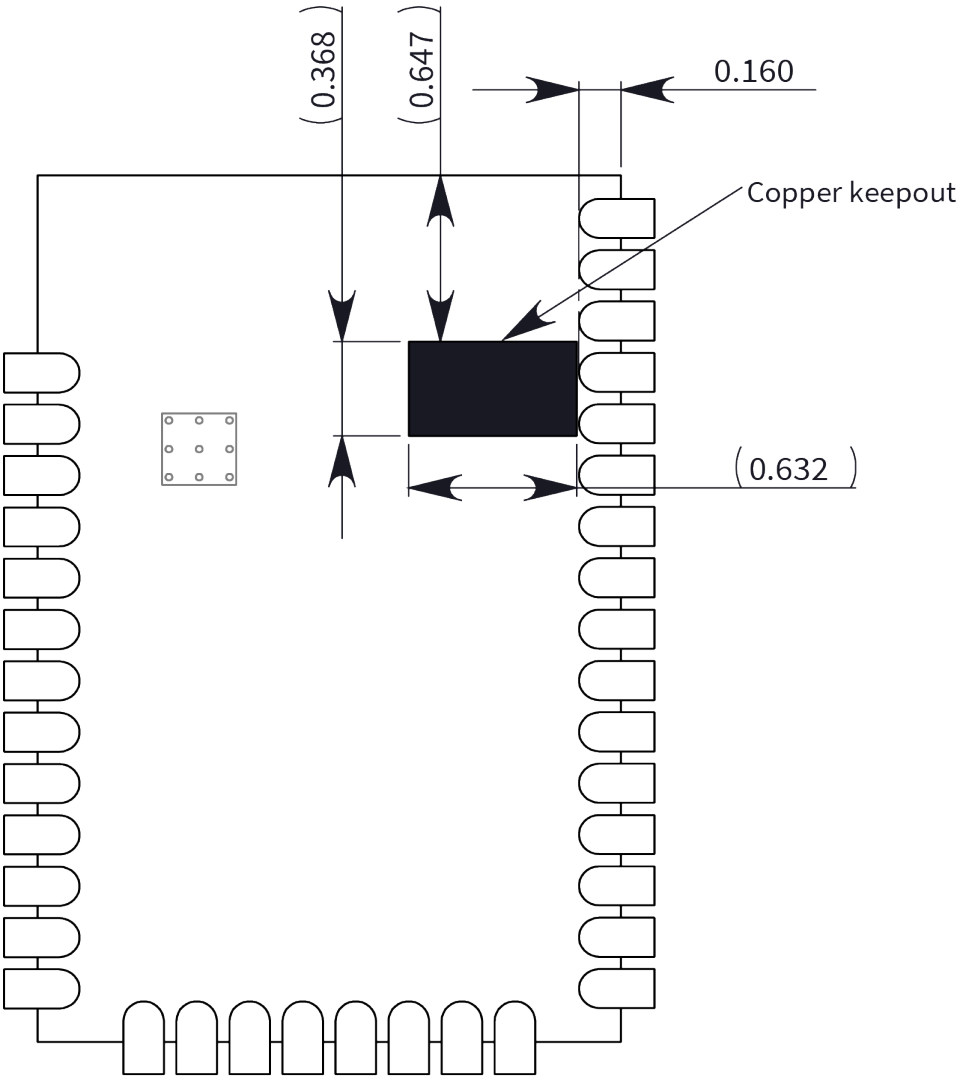
The recommended footprint includes an additional ground pad that you must solder to the corresponding pad on the device. This ground pad transfers heat generated during transmit mode away from the device's power amplifier. The pad must connect through vias to a ground plane on the host PCB. Connecting to planes on multiple layers will further improve the heat transfer performance and we recommend doing this for applications that will be in transmit mode for sustained periods. We recommend using nine 0.030 cm diameter vias in the pad as shown. Plug vias with epoxy or solder mask them on the opposite side to prevent solder paste from leaking through the holes during reflow. Do not mask over the ground pad.

Note The ground pad is unique to the XBee/XBee-PRO XTC and SX modules. This footprint is not compatible with other SMT XBees.

Although the underside of the device is mostly coated with solder mask, we recommend that you leave the copper layer directly below the device open to avoid unintended contacts. Most importantly, copper or vias must not interfere with the three exposed RF test points on the bottom of the device shown in the following keepout drawing. Observe the copper keepout on all layers of the host PCB, to avoid the possibility of capacitive coupling that could impact RF performance.

Match the solder footprint to the copper pads, but you may need to adjust it depending on the specific needs of assembly and product standards. We recommend a stencil thickness of 0.15 mm (0.005 in). Place the component last and set the placement speed to the slowest setting.

The following drawing show the SMT footprint, with the required copper keepout (all layers). Dimensions are in centimeters.



Design notes

The following guidelines help to ensure a robust design.

Host board design

A good power supply design is critical for proper device operation. If the supply voltage is not kept within tolerance, or is excessively noisy, it may degrade device performance and reliability. To help reduce noise, we recommend placing both a 1 uF and 100 pF capacitor as near to VCC as possible. If you use a switching regulator, we recommend switching frequencies above 500 kHz and you should limit power supply ripple to a maximum 50 mV peak to peak.

As with all PCB designs, make power and ground traces thicker than signal traces and make them able to comfortably support the maximum current specifications. Ground planes are preferable.

Improve antenna performance

The choice of antenna and antenna location is important for optimal performance. In general, antenna elements radiate perpendicular to the direction they point. Thus a vertical antenna, such as a dipole, emit across the horizon.

Metal objects near the antenna cause parasitic coupling and detuning, preventing the antenna from radiating efficiently. Metal objects between the transmitter and receiver can also block the radiation path or reduce the transmission distance, so position external antennas away from them as much as possible. Some objects that are often overlooked are:

- Metal poles
- Metal studs or beams in structures
- Concrete (reinforced with metal rods)
- Metal enclosures
- Vehicles
- Elevators
- Ventilation ducts
- Large appliances
- Batteries
- Tall electrolytic capacitors

RF pad version

The RF pad is a soldered antenna connection. The RF signal travels from pin on the module to the antenna through a single ended RF transmission line on the PCB. This line should have a controlled impedance of 50 Ω.

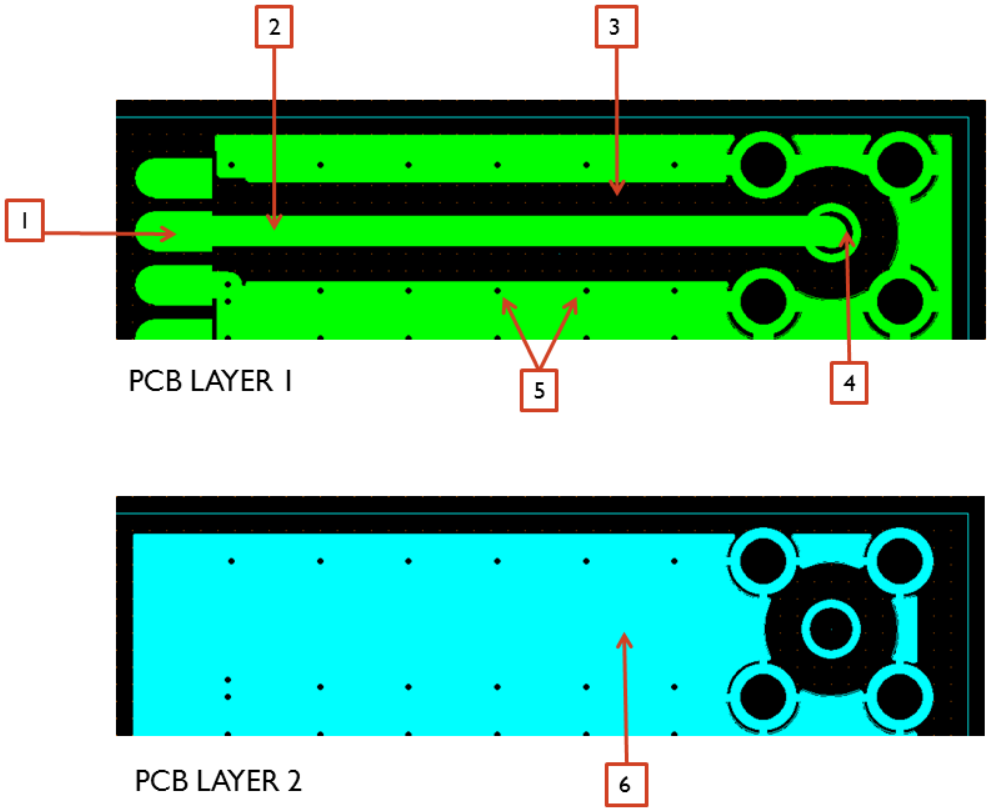
For the transmission line, we recommend either a microstrip or coplanar waveguide trace on the PCB. We provide a microstrip example below, because it is simpler to design and generally requires less area on the host PCB than coplanar waveguide.

We do not recommend using a stripline RF trace because that requires routing the RF trace to an inner PCB layer, and via transitions can introduce matching and performance problems.

The following figure shows a layout example of a microstrip connecting an RF pad module to a through-hole RPSMA RF connector.

- The top two layers of the PCB have a controlled thickness dielectric material in between. The second layer has a ground plane which runs underneath the entire RF pad area. This ground plane is a distance *d*, the thickness of the dielectric, below the top layer.
- The top layer has an RF trace running from pin 36 of the device to the RF pin of the RPSMA connector. The RF trace's width determines the impedance of the transmission line with relation to the ground plane. Many online tools can estimate this value, although you should consult the PCB manufacturer for the exact width. Assuming *d* = 0.025 in, and that the dielectric has a relative permittivity of 4.4, the width in this example will be approximately 0.045 in for a 50 Ω trace. This trace width is a good fit with the module footprint's 0.060 in pad width.

We do not recommend using a trace wider than the pad width, and using a very narrow trace can cause unwanted RF loss. You can minimize the length of the trace by placing the RPSMA jack close to the module. All of the grounds on the jack and the module are connected to the ground planes directly or through closely placed vias. Space any ground fill on the top layer at least twice the distance *d* (in this case, at least 0.050 in) from the microstrip to minimize their interaction.



Number	Description
1	XBee pin 36
2	50 Ω microstrip trace
3	Back off ground fill at least twice the distance between layers 1 and 2
4	RF connector
5	Stitch vias near the edges of the ground plane
6	Pour a solid ground plane under the RF trace on the reference layer

Implementing these design suggestions helps ensure that the RF pad device performs to specifications.

Recommended solder reflow cycle

The following table provides the recommended solder reflow cycle. The table shows the temperature setting and the time to reach the temperature; it does not show the cooling cycle.

Time (seconds)	Temperature (degrees C)
30	65
60	100
90	135
120	160
150	195
180	240
210	260

The maximum temperature should not exceed 260 °C.

The SX device will reflow during this cycle, and therefore must not be reflowed upside down. Take care not to jar the device while the solder is molten, as this can remove components under the shield from their required locations.

The device has a Moisture Sensitivity Level (MSL) of 3. When using this product, consider the relative requirements in accordance with standard IPC/JEDEC J-STD-020.

In addition, note the following conditions:

a. Calculated shelf life in sealed bag: 12 months at < 40 °C and < 90% relative humidity (RH).
b. Environmental condition during the production: 30 °C /60% RH according to IPC/JEDEC J-STD-033C, paragraphs 5 through 7.
c. The time between the opening of the sealed bag and the start of the reflow process cannot exceed 168 hours if condition b) is met.
d. Baking is required if conditions b) or c) are not met.
e. Baking is required if the humidity indicator inside the bag indicates a RH of 10% more.
f. If baking is required, bake modules in trays stacked no more than 10 high for 4-6 hours at 125 °C.

Flux and cleaning

We recommend that you use a “no clean” solder paste in assembling these devices. This eliminates the clean step and ensures that you do not leave unwanted residual flux under the device where it is difficult to remove. In addition:

- Cleaning with liquids can result in liquid remaining under the device or in the gap between the device and the host PCB. This can lead to unintended connections between pads.
- The residual moisture and flux residue under the device are not easily seen during an inspection process.

Rework

Once you mount the device, do not perform rework on the SX device (for example, removing it from the host PCB).

CAUTION! Any modification to the device voids the warranty coverage and certifications.