XBee®-PRO 900HP/XSC RF Modules

S3 and S3B

User Guide

Revision history—90002173

Revision	Date	Description
S	October 2016	Replaced the Programmable bootloader section with the Programmable XBee SDK section. Updated the indoor range spec. Corrected the **SP** and **ST** parameter default values.
T	May 2018	Added note on range estimation. Changed IC to ISED.
U	July 2018	Added the 0x00, 0x80 and 0x89 frames for the 900HP.
V	June 2019	Added FCC publication 996369 related information.
W	January 2020	Added IFETEL certifications.

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®-PRO 900HP/XSC RF Modules, 90002173 W) in the subject line of your email.

Contents

Operation

SPI operation

Modes

Sleep modes

Networking methods

AT commands

Operate in API mode

Frame descriptions

Advanced application features

General Purpose Flash Memory

XSC firmware

XBee-PRO XSC specifications

XBee-PRO XSC RF Module operation

Configuration and commands

Network configurations

S3B hardware certifications

Legacy S3B hardware certifications

About the XBee-PRO 900HP RF Module

The XBee-PRO 900HP RF Modules consist of firmware loaded onto XBee-PRO S3B hardware. These embedded RF devices provide wireless connectivity to end-point devices in mesh networks.

You can build networks up to 128 nodes using the XBee devices. For larger networks of up to 1,000 or more nodes, we offer RF optimization services to assist with proper network configuration.

For more information network configuration, contact Digi Technical Support.

Note The XBee-PRO 900HP RF Module is not backward compatible with the legacy XBee-PRO 900 (Part Number: XBP09-DP…) or XBee-PRO DigiMesh 900 (Part Number: XBP09-DM…) RF modules.

The XBee-PRO S3B hardware consists of:

- One Energy Micro EFM®32G230F128 microcontroller
- One Analog Devices ADF7023 radio transceiver
- One RF power amplifier
- One NXP MC9S08QE32® microcontroller, only in the programmable version of the XBee

User guide structure

This user guide contains documentation for two RF protocols: XStream Compatible (XSC) and 900HP. The XSC firmware is provided for customers who need compatibility with existing networks that need to be 9XStream compatible. Customers who do not require this compatibility should not use the XSC firmware, but rather the newer 900HP firmware.

The XSC firmware section at the back of this user guide contains documentation for the XSC firmware only. All other firmware documentation in the user guide is applicable to the 900HP firmware only. For more information about XSC firmware see the XSC firmware section.

The XBee-PRO 900HP RF Module is not backward compatible with the legacy XBee-PRO 900 (Part Number: XBP09-DP…) or XBee-PRO DigiMesh 900 (Part Number: XBP09-DM…) RF Modules.

The following table describes how to use this user guide based on the Digi part number for the module:

Digi Part Numbers	FCC ID	Hardware Platform	Pre-installed Firmware	Firmware Available	Regulatory Information
XBP09-XC…	MCQ-XBEEXSC	S3[1]	XSC	XSC	Legacy S3B hardware certifications
XBP9B-XC*T-001 (revision G and earlier) XBP9B-XC*T-002 (revision G and earlier) XBP9B-XC*T-021 (revision F and earlier) XBP9B-XC*T-022 (revision F and earlier)	MCQ-XBPS3B	S3B	XSC	XSC	Legacy S3B hardware certifications
XBP9B-XC*T-001 (revision H and later) XBP9B-XC*T-002 (revision H and later) XBP9B-XC*T-021 (revision G and later) XBP9B-XC*T-022 (revision G and later) all other part numbers beginning XBP9B-XC...	MCQ-XB900HP	S3B	XSC	XSC / 900HP	
XBP9B-D…	MCQ-XB900HP	S3B	900HP	XSC / 900HP	

[1]The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.

Technical specifications

Performance specifications

This 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	Value
Ideal RF line-of-sight range	10 kb/s: up to 9 miles (15.5 km) 200 kb/s: up to 4 miles (6.5 km) (with 2.1 dB dipole antennas)
Transmit power output	24 dBm (250 mW) (software selectable)
RF data rate (high)	200 kb/s
RF data rate (low)	10 kb/s
Serial UART interface	Complementary metal–oxide–semiconductor (CMOS) Serial universal asynchronous receiver/transmitter (UART), baud rate stability of <1%
Serial interface data rate (software selectable)	9600-230400 baud
Receiver sensitivity (typical)	-101 dBm, high data rate -110 dBm, low data rate

Power requirements

The following table describes the power requirements for the XBee-PRO 900HP RF Module.

Specification	Value
Supply voltage	2.1 to 3.6 VDC[1]
Transmit current	**PL** = 4: 215 mA typical, (290 mA max) **PL** = 3: 160 mA typical **PL** = 2: 120 mA typical **PL** = 1: 95 mA typical **PL** = 0: 60 mA typical
Idle/receive current	29 mA typical at 3.3 V (35 mA max)
Sleep current	2.5 µA (typical)

[1]Supply voltages of less than 3.0 V may reduce performance. Output power and receiver sensitivity may degrade.

General specifications

The following table describes the general specifications for the devices.

Specification	Value
Operating frequency band[1]	902 to 928 MHz (software selectable channels)
Dimensions	3.29 cm x 2.44 cm x 0.546 cm (1.297" x 0.962" x 0.215) Dimensions do not include connector/antenna or pin lengths
Weight	5 to 8 grams, depending on the antenna option
Operating temperature	-40 ºC to 85 º C (industrial)
Antenna options	Integrated wire, U. FL RF connector, reverse-polarity SMA connector
Digital I/O	Fifteen (15) I/O lines,
Analog-to-digital converter (ADC)	Four (4)10-bit analog inputs

Networking specifications

The following table provides the networking specifications for the device.

Specification	Value
Supported network topologies	Mesh, point-to-point, point-to-multipoint, peer-to-peer
Number of channels, user selectable channels	64 channels available
Addressing options	Personal Area Network identifier (PAN ID), Preamble ID, and 64-bit addresses
Encryption	128 bit Advanced Encryption Standard (AES)

Regulatory conformity summary

This table describes the agency approvals for the devices.

Country	Approval
United States (FCC Part 15.247)	MCQ-XB900HP
Innovation, Science and Economic Development Canada (ISED)	1846A-XB900HP

[1]Supply voltages of less than 3.0 V may reduce performance. Output power and receiver sensitivity may degrade.

Country	Approval
Australia	RCM
Brazil	ANATEL 3727-12-1209
Singapore	License No. DA105737 (XB900HP only)
Mexico	IFETEL (XB900HP listed in Mexico IFETEL)
RoHS2	Compliant

Serial communication specifications

The XBee-PRO 900HP RF Module supports both Universal Asynchronous Receiver / Transmitter (UART) and Serial Peripheral Interface (SPI) serial connections.

UART pin assignments

UART pins	Device pin number
DOUT	2
DIN / $\overline{\text{CONFIG}}$	3
$\overline{\text{CTS}}$ / DIO7	12
$\overline{\text{RTS}}$ / DIO6	16

SPI pin assignments

SPI pins	Device pin number
SPI_SCLK / DIO18	18
SPI_$\overline{\text{SSEL}}$ / DIO17	17
SPI_MOSI / DIO16	11
SPI_MISO / DIO15	4
SPI_$\overline{\text{ATTN}}$ / DIO1	19

GPIO specifications

XBee devices have 15 General Purpose Input/Output (GPIO) ports available. The precise list depends on the device configuration as some devices use the GPIO pins for purposes such as serial communication. The following table shows the electrical specifications for the GPIO pins.

GPIO electrical specification	Value
Voltage - supply	2.1 - 3.6 V (3.0 V or higher required for optimal performance)
Low Schmitt switching threshold	0.3 x V_{DD}
High Schmitt switching threshold	0.7 x V_{DD}
Input pull-up resistor value	40 kΩ
Input pull-down resistor value	40 kΩ
Output voltage for logic 0	0.05 x V_{DD}
Output voltage for logic 1	0.95 x V_{DD}
Output source current	2 mA
Output sink current	2 mA
Total output current (for GPIO pins)	48 mA

Note For information about Mexico IFETEL, see Mexico IFETEL. Only the XBee-PRO 900HP devices listed are approved by IFETEL.

Secondary processor specifications

If the device has the programmable secondary processor, add the values from the following tables to the specifications listed in the Power requirements specifications. For more information about transmit, receive, and sleep currents, see Power requirements.

For example, if the secondary processor runs at 20 MHz and the primary processor is in receive mode, then the new current value is:

$$I_{total} = I_{r2} + I_{rx} = 14\ mA + 9\ mA = 23\ mA$$

where I_{r2} is the runtime current of the secondary processor and I_{rx} is the receive current of the primary processor.

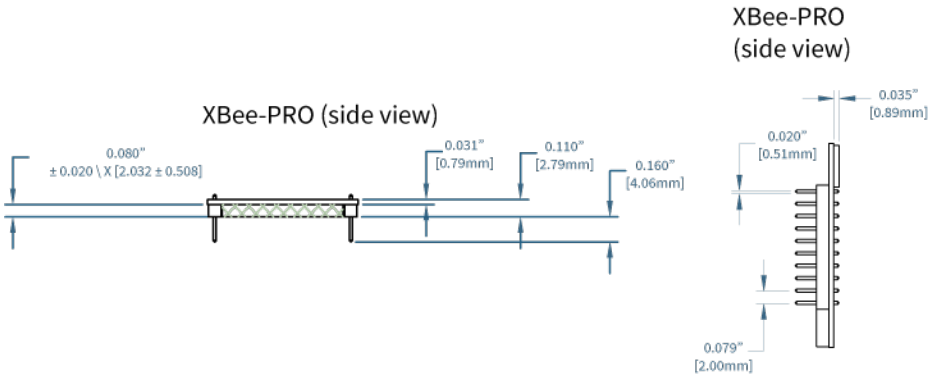
Optional secondary processor specification	Add these numbers to power requirement specifications (add to RX, TX, and sleep currents depending on mode of operation)
Runtime current for 32 k running at 20 MHz	+14 mA
Runtime current for 32 k running at 1 MHz	+1 mA
Sleep current	+0.5µ A typical
For additional specifications see the NXP datasheet and manual	MC9S08QE32
Voltage requirement for secondary processor to operate at maximum clock frequency	2.4 to 3.6 VDC

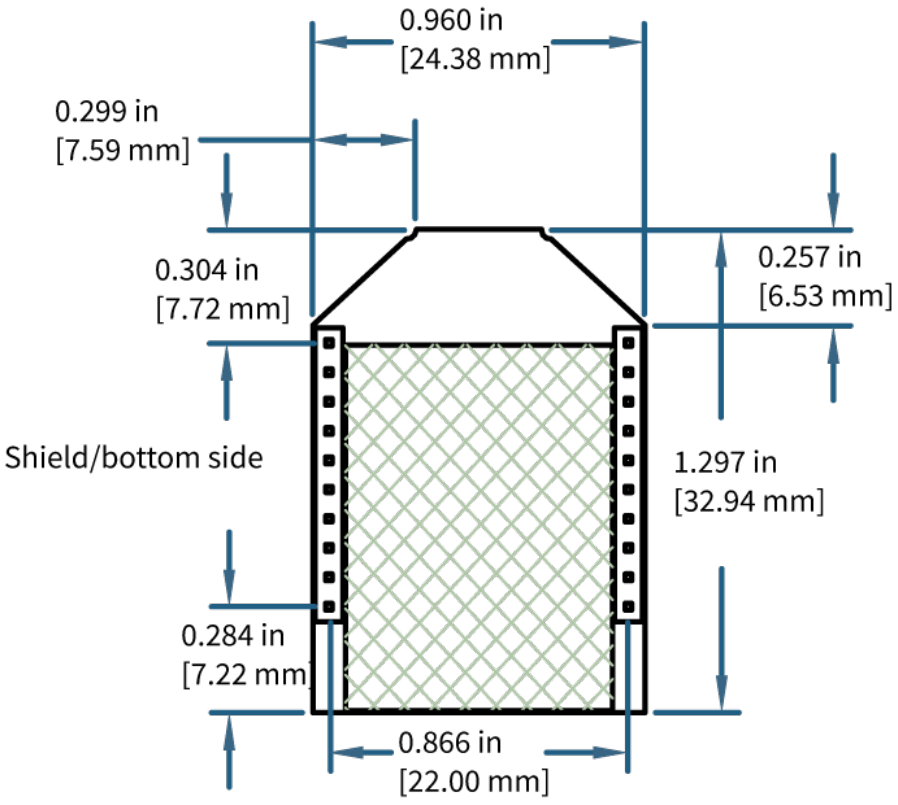
Optional secondary processor specification	Add these numbers to power requirement specifications (add to RX, TX, and sleep currents depending on mode of operation)
Minimum reset pulse for programmable variant	100 nS
Minimum reset pulse to radio	50 nS
Voltage reference (VREF) range	1.8 VDC to VCC

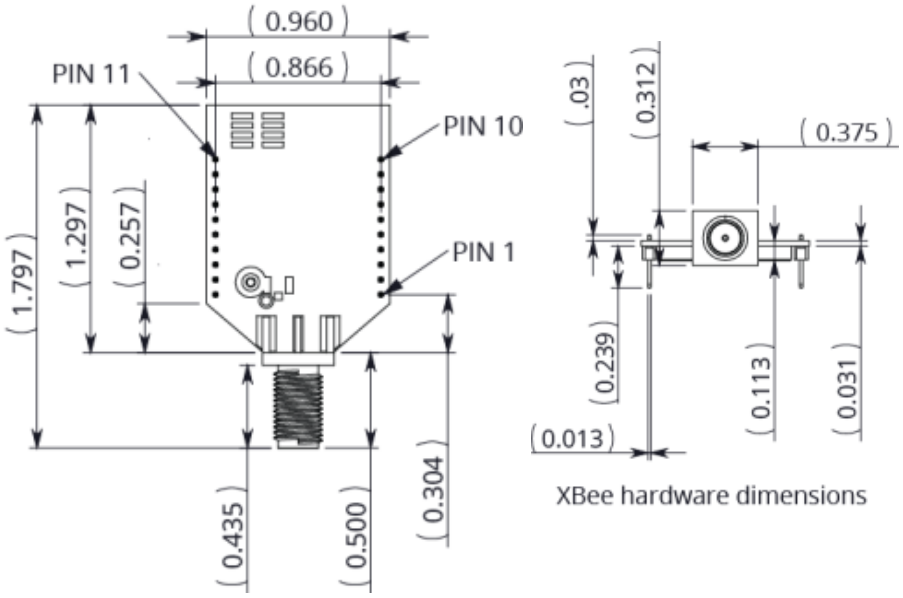
Hardware

Mechanical drawings

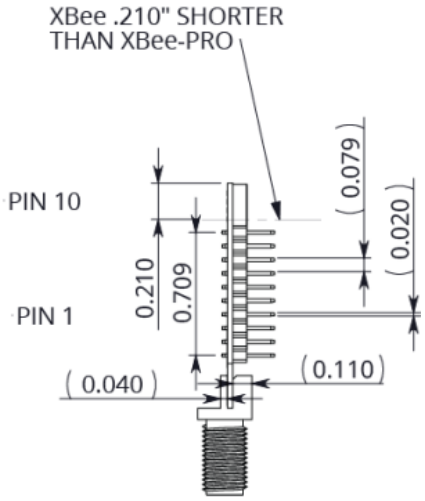
The following figures show the mechanical drawings for the XBee-PRO 900HP RF Module. The drawings do not show antenna options.







XBee hardware dimensions



XBee hardware
(side view)

Pin signals

The following table shows the pin signals and their descriptions. The table specifies signal direction with respect to the device. For more information on pin connections, see Design notes.

Pin #	Name	Direction	Default state	Description
1	VCC			Power supply

Pin #	Name	Direction	Default state	Description
2	DOUT/DIO13	Both	Output	GPIO/UART data out
3	DIN/$\overline{\text{CONFIG}}$ /DIO14	Both	Input	GPIO/UART data in
4	DIO12/SPI_ MISO	Both	Output	GPIO/SPI slave out
5	$\overline{\text{RESET}}$	Input		Device reset. Drive low to reset the device. This is also an output with an open drain configuration with an internal 20 kΩ pull-up (never drive to logic high, as the device may be driving it low). The minimum pulse width is 1 mS.
6	DIO10/PWM0	Both		GPIO/RX signal strength indicator
7	DIO11/PWM1	Both		GPIO/pulse width modulator
8	Reserved		Disabled	Do not connect
9	$\overline{\text{DTR}}$/SLEEP_ RQ/DIO8	Both	Input	GPIO/pin sleep control line (DTR on the development board)
10	GND			Ground
11	DIO4/SPI_MOSI	Both		GPIO/SPI slave in
12	$\overline{\text{CTS}}$/DIO7	Both	Output	GPIO/clear-to-send flow control
13	ON_$\overline{\text{SLEEP}}$ /DIO9	Output	Output	GPIO/module status indicator
14	VREF	Input		Internally used for the programmable secondary processor. For compatibility with other XBee devices, we recommend connecting this pin to the voltage reference if you desire analog sampling. Otherwise, connect to GND.
15	Associate/DIO5	Both	Output	GPIO/associate indicator
16	$\overline{\text{RTS}}$ /DIO6	Both	Input	GPIO/request-to-send flow control
17	$\underline{\text{AD3}}$/DIO3/SPI_ SSEL	Both		GPIO/analog input/SPI slave select
18	AD2/DIO2/SPI_ CLK	Both		GPIO/analog input /SPI clock
19	$\underline{\text{AD1}}$/DIO1/SPI_ ATTN	Both		GPIO/analog input /SPI attention
20	AD0/DIO0	Both		GPIO/analog input

Design notes

The XBee modules do not require any external circuitry or specific connections for proper operation. However, there are some general design guidelines that we recommend to build and troubleshoot a robust design.

Power supply design

A poor power supply can lead to poor radio performance, especially if you do not keep the supply voltage within tolerance or if the noise is excessive. To help reduce noise, place a 1.0 µF and 47 pF capacitor as near as possible to pin 1 on the PCB. If you are using a switching regulator for the power supply, switch the frequencies above 500 kHz. Limit the power supply ripple to a maximum 50 mV peak to peak.

For designs using the programmable modules, we recommend an additional 10 µF decoupling cap near pin 1 of the device. The nearest proximity to pin 1 of the three caps should be in the following order:

1. 47 pf
2. 1 µF
3. 10 µF

Board layout

We design XBee modules to be self-sufficient and have minimal sensitivity to nearby processors, crystals or other printed circuit board (PCB) components. Keep power and ground traces thicker than signal traces and make sure that they are able to comfortably support the maximum current specifications. There are no other special PCB design considerations to integrate XBee modules, with the exception of antennas.

Antenna performance

Antenna location is important for optimal performance. The following suggestions help you achieve optimal antenna performance. Point the antenna up vertically (upright). Antennas radiate and receive the best signal perpendicular to the direction they point, so a vertical antenna's omnidirectional radiation pattern is strongest across the horizon.

Position the antennas away from metal objects whenever possible. Metal objects between the transmitter and receiver can block the radiation path or reduce the transmission distance. Objects that are often overlooked include:

- Metal poles
- Metal studs
- Structure beams
- Concrete, which is usually reinforced with metal rods

If you place the device inside a metal enclosure, use an external antenna. Common objects that have metal enclosures include:

- Vehicles
- Elevators
- Ventilation ducts
- Refrigerators

- Microwave ovens
- Batteries
- Tall electrolytic capacitors

Use the following additional guidelines for optimal antenna performance:

- Do not place XBee modules with the chip antenna inside a metal enclosure.
- Do not place any ground planes or metal objects above or below the antenna.
- For the best results, mount the device at the edge of the host PCB. Ensure that the ground, power, and signal planes are vacant immediately below the antenna section.

Recommended pin connections

The only required pin connections for two-way communication are VCC, GND, DOUT and DIN. To support serial firmware updates, you must connect VCC, GND, DOUT, DIN, RTS, and DTR.

Do not connect any pins that are not in use. Use the **PR** and **PD** commands to pull all inputs on the radio high with internal pull-up resistors. Unused outputs do not require any specific treatment.

For applications that need to ensure the lowest sleep current, never leave unconnected inputs floating. Use internal or external pull-up or pull-down resistors, or set the unused I/O lines to outputs.

You can connect other pins to external circuitry for convenience of operation including the Associate LED pin (pin 15) and the Commissioning pin (pin 20). The Associate LED pin flashes differently depending on the state of the module, and a pushbutton attached to pin 20 can enable various deployment and troubleshooting functions without you sending UART commands. For more information, see Commissioning pushbutton and associate LED.

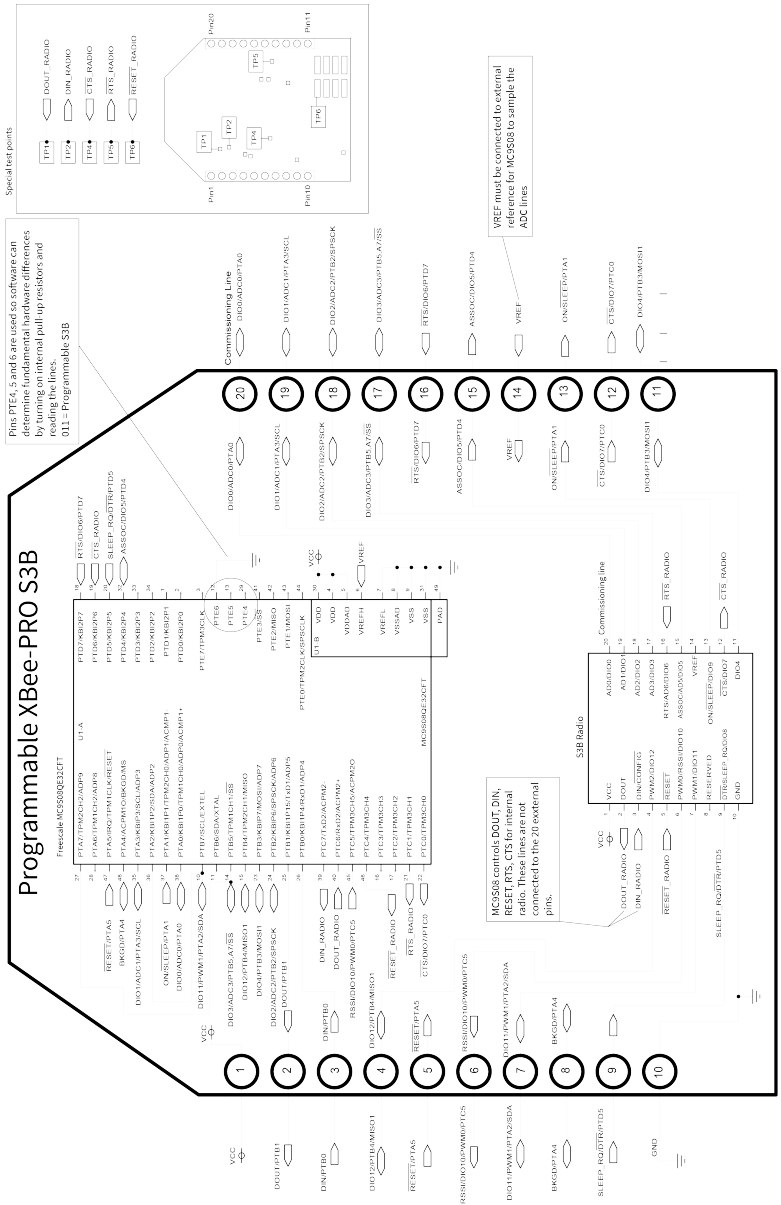
Only the programmable versions of these devices use the VREF pin (pin 14). For compatibility with other XBee modules, we recommend connecting this pin to a voltage reference if you want to enable analog sampling. Otherwise, connect to GND.

Module operation for the programmable variant

The modules with the programmable option have a secondary processor with 32k of flash and 2k of RAM. This allows module integrators to put custom code on the XBee module to fit their own unique needs. The DIN, DOUT, RTS, CTS, and RESET lines are intercepted by the secondary processor to allow it to be in control of the data transmitted and received. All other lines are in parallel and can be controlled by either the internal microcontroller or the MC9SO8QE micro; see the block diagram in Operation for details. The internal microcontroller by default has control of certain lines. The internal microcontroller can release these lines by sending the proper command(s) to disable the desired DIO line(s). For more information about commands, see AT commands.

For the secondary processor to sample with ADCs, the XBee 14 (VREF) must be connected to a reference voltage.

Digi provides a bootloader that can take care of programming the processor over-the-air or through the serial interface. This means that over-the-air updates can be supported through an XMODEM protocol. The processor can also be programmed and debugged through a one wire interface BKGD (Pin 8).


Programmable XBee-PRO S3B
Freescale MC9S08QE32CFT
Pins PTE4, 5 and 6 are used so software can determine fundamental hardware differences by turning on internal pull up resistors and reading the lines.
011 = Programmable S3B
Special test points
TP1
TP2
TP3
TP4
TP5
TP6
DOUT_RADIO
DIN_RADIO
CTS_RADIO
RTS_RADIO
RESET_RADIO
Commissioning Line
MC9S08 controls DOUT, DIN, RESET, RTS, CTS for internal radio. These lines are not connected to the 20 external pins.
S3B Radio
VREF must be connected to external reference for MC9S08 to sample the ADC lines

Programmable XBee SDK

The XBee Programmable module is equipped with a NXP MC9S08QE32 application processor. This application processor comes with a supplied bootloader. To interface your application code running on this processor to the XBee Programmable module's supplied bootloader, use the Programmable XBee SDK.

To use the SDK, you must also download CodeWarrior. The download links are:

- CodeWarrior IDE: http://ftp1.digi.com/support/sampleapplications/40003004_B.exe
- Programmable XBee SDK: http://ftp1.digi.com/support/sampleapplications/40003003_D.exe

If these revisions change, search for the part number on Digi's website. For example, search for **40003003**.

Install the IDE first, and then install the SDK.

The documentation for the Programmable XBee SDK is built into the SDK, so the Getting Started guide appears when you open CodeWarrior.

Configure the XBee-PRO 900HP RF Module

Software libraries

One way to communicate with the XBee-PRO 900HP RF Module is by using a software library. The libraries available for use with the XBee-PRO 900HP 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.

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 the connected XBee-PRO 900HP RF Modules using the provided settings.

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.

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-PRO 900HP 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

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.

XBee Multi Programmer

The XBee Multi Programmer is a combination of hardware and software that enables partners and distributors to program multiple Digi Radio frequency (RF) devices simultaneously. It provides a fast and easy way to prepare devices for distribution or large networks deployment.

The XBee Multi Programmer board is an enclosed hardware component that allows you to program up to six RF modules thanks to its six external XBee sockets. The XBee Multi Programmer application communicates with the boards and allows you to set up and execute programming sessions. Some of the features include:

- Each XBee Multi Programmer board allows you to program up to six devices simultaneously. Connect more boards to increase the programming concurrency.
- Different board variants cover all the XBee form factors to program almost any Digi RF device.

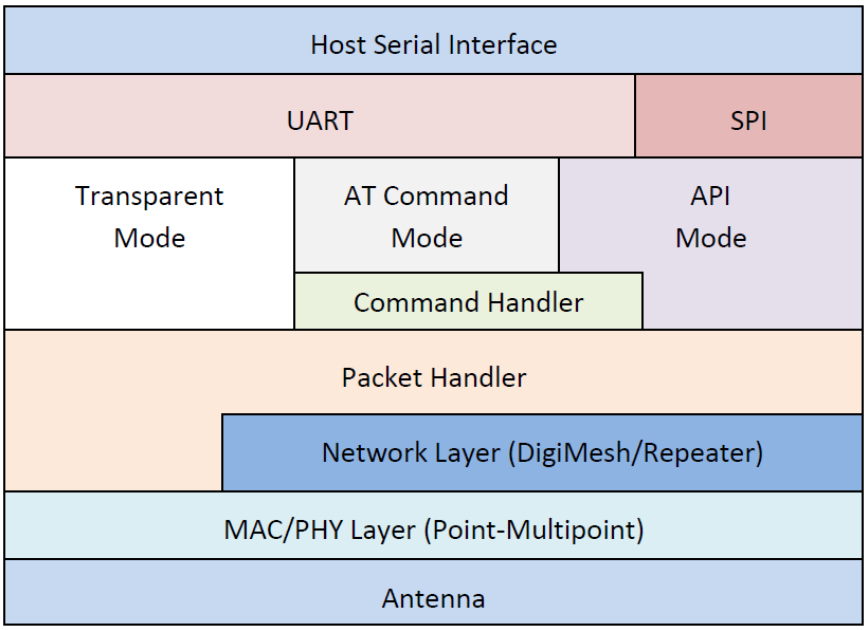
Download the XBee Multi Programmer application from: digi.com/support/productdetail?pid=5641

See the *XBee Multi Programmer User Guide* for more information.

Operation

Basic operational design

The XBee-PRO 900HP RF ModuleRF Module uses a multi-layered firmware base to order the flow of data, dependent on the hardware and software configuration that you choose. The following graphic shows a configuration block diagram, with the host serial interface as the physical starting point and the antenna as the physical endpoint for the transferred data. As long as one block can touch another block, the two interfaces can interact. For example, if the device is using SPI mode, Transparent Mode is not available.



The command handler is the code that processes commands from AT Command Mode or Application Programming Interface (API) Mode (see AT commands). The command handler also processes commands from remote radios (see Remote AT commands).

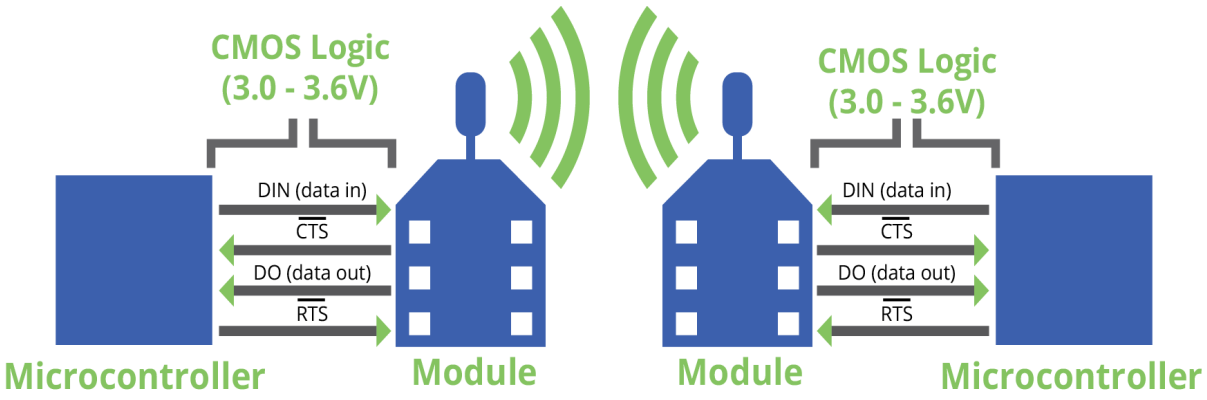
Serial interface

The XBee-PRO 900HP RF Module interfaces to a host device through a serial port. The device can communicate through its serial port with the following:

- Logic and voltage compatible universal asynchronous receiver/transmitter (UART).
- Level translator to any serial device, for example, through an RS-232 or USB interface board.
- SPI, as described in SPI communications.

UART data flow

Devices that have a UART interface connect directly to the pins of the XBee-PRO 900HP 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.

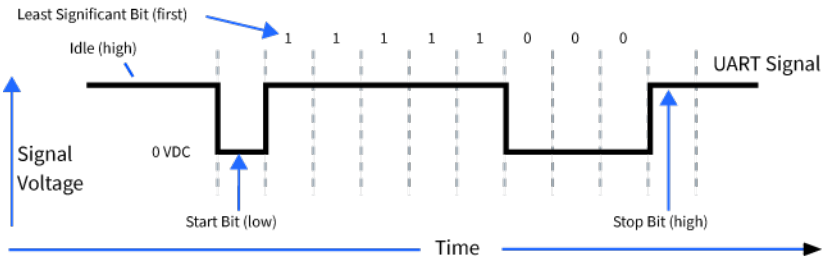


Serial data

A device sends data to the XBee-PRO 900HP RF Module's UART through pin 3 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-PRO 900HP 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.



You can configure the UART baud rate, parity, and stop bits settings on the device with the **BD**, **NB**, and **SB** commands respectively. For more information, see Serial interfacing commands.

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?

Select the serial port

Both the UART and SPI ports are configured for serial port operation by default.

- If both interfaces are configured, serial data goes out the UART until the SPI_SSEL signal is asserted. After that, all serial communications operate on the SPI interface.
- If you enable only the UART, then the device only uses the UART and ignores SPI_SSEL. If you only enable the SPI, then the device uses only the SPI.
- If you do not enable either serial port, the device does not support serial operations and all communications must occur over-the-air. The device discards all data that would normally go to the serial port.

Force UART operation

If you configure a device with only the SPI enabled and no SPI master is available to access the SPI slave port, you can recover the device to UART operation by holding DIN / CONFIG low at reset time. 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 then send the appropriate commands to the device to configure it for UART operation. If you write those parameters, the device comes up with the UART enabled on the next reset.

Select the SPI port

To force SPI mode, hold DOUT/DIO13 (pin 2) low while resetting the device until SPI_ATTN asserts. This causes the device to disable the UART and go straight into SPI communication mode. Once configuration is complete, the device queues a modem status frame to the SPI port, which causes the SPI_ATTN line to assert. The host can use this to determine that the SPI port is configured properly. This method forces the configuration to provide full SPI support for the following parameters:

- **D1** (This parameter will only be changed if it is at a default of zero when the method is invoked.)
- **D2**
- **D3**
- **D4**
- **P2**

As long as the host does not issue a **WR** command, these configuration values revert to previous values after a power-on reset. If the host issues a **WR** command while in SPI mode, these same parameters are written to flash. After a reset, parameters that were forced and then written to flash become the mode of operation.

If the UART is disabled and the SPI is enabled in the written configuration, then the device comes up in SPI mode without forcing it by holding DOUT low. If both the UART and the SPI are enabled at the time of reset, then output goes to the UART until the host sends the first input. If that first input comes on the SPI port, then all subsequent output goes to the SPI port and the UART is disabled. If the first input comes on the UART, then all subsequent output goes to the UART and the SPI is disabled.

When the master asserts the slave select (SPI_SSEL) signal, SPI transmit data is driven to the output pin SPI_MISO, and SPI data is received from the input pin SPI_MOSI. The SPI_SSEL pin has to be asserted to enable the transmit serializer to drive data to the output signal SPI_MISO. A rising edge on SPI_SSEL causes the SPI_MISO line to be tri-stated such that another slave device can drive it, if so desired.

If the output buffer is empty, the SPI serializer transmits the last valid bit repeatedly, which may be either high or low. Otherwise, the device formats all output in API mode 1 format, as described in Operate in API mode. The attached host is expected to ignore all data that is not part of a formatted API frame.

Serial port selection

To enable the UART port, configure DIN and DOUT (**P3** and **P4** parameters) as peripherals. To enable the SPI port, enable SPI_MISO, SPI_MOSI, SPI_SSEL , and SPI_CLK (**P5** through **P9**) as peripherals. If you enable both ports then output goes to the UART until the first input on SPI.

When both the UART and SPI ports are enabled on power-up, all serial data goes out the UART. As soon as 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 device reset.

If you change the configuration so that only one port is configured, then that port is the only one enabled or used. If the parameters are written with only one port enabled, then 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 in spite of the wrong configuration so 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.

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 $\overline{\text{RTS}}$ flow control. When enabled, the DIO1 line asserts (low asserted) when UART data is queued to be transmitted from the module. For more information, see D1 (DIO1/AD1).

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

SPI operation

SPI communications

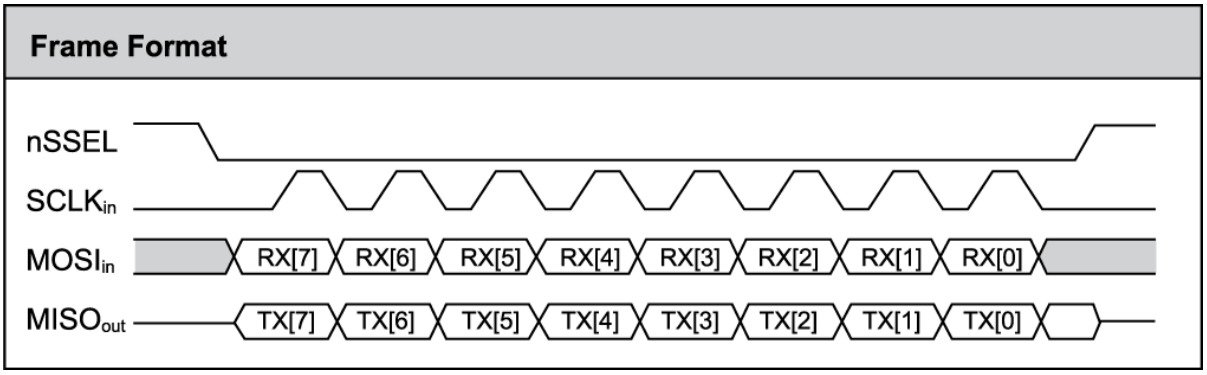
The XBee-PRO 900HP RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.

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-PRO 900HP 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:

- SPI clock rates up to 3.5 MHz are possible.
- 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 implementation

The XBee-PRO 900HP RF Module operates as a SPI slave only. This means an external master provides the clock and decides when to send data. The XBee-PRO 900HP RF Module supports an external clock rate of up to 3.5 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 specification for SPI includes the four signals: SPI_MISO, SPI_MOSI, SPI_CLK, and SPI_SSEL. Using only these four signals, the master cannot know when the slave needs to send and the SPI slave cannot transmit unless enabled by the master. For this reason, the SPI_ATTN signal is available. This allows the device to alert the SPI master that it has data to send. In turn, the SPI master asserts SPI_SSEL and starts SPI_CLK unless these signals are already asserted and active respectively. This allows the XBee-PRO 900HP RF Module to send data to the master.

The following table names the SPI signals and specifies their pinouts. It also describes the operation of each pin.

Signal name	Pin number	Applicable AT command	Description
SPI_MISO (Master In, Slave out)	4	**P2**	When SPI_SSEL is asserted (low) and SPI_CLK is active, the device outputs the data on this line at the SPI_CLK rate. When SPI_SSEL is de-asserted (high), this output should be tri-stated such that another slave device can drive the line.
SPI_MOSI (Master out, Slave in)	11	**D4**	The SPI master outputs data on this line at the SPI_CLK rate after it selects the desired slave. When the device is configured for SPI operations, this pin is an input.
SPI_SSEL (Slave Select) (Master out, Slave in)	17	**D3**	The SPI master outputs a low signal on this line to select the desired slave. When the device is configured for SPI operations, this pin is an input.
SPI_CLK (Clock) (Master out, Slave in)	18	**D2**	The SPI master outputs a clock on this pin, and the rate must not exceed the maximum allowed, 3.5 Mb/s. When you configure the device for SPI operations, this pin is an input.
SPI_ATTN (Attention) (Master in, Slave out)	19	**D1**	The device asserts this pin low when it has data to send to the SPI master. When this pin is configured for SPI operations, it is an output (not tri-stated).

Note By default, the inputs have pull-up resistors enabled. See PR (Pull-up/Down Resistor Enable) to disable the pull-up resistors. When the SPI pins are not connected but the pins are configured for SPI operation, the pull-ups are required for proper UART operation.

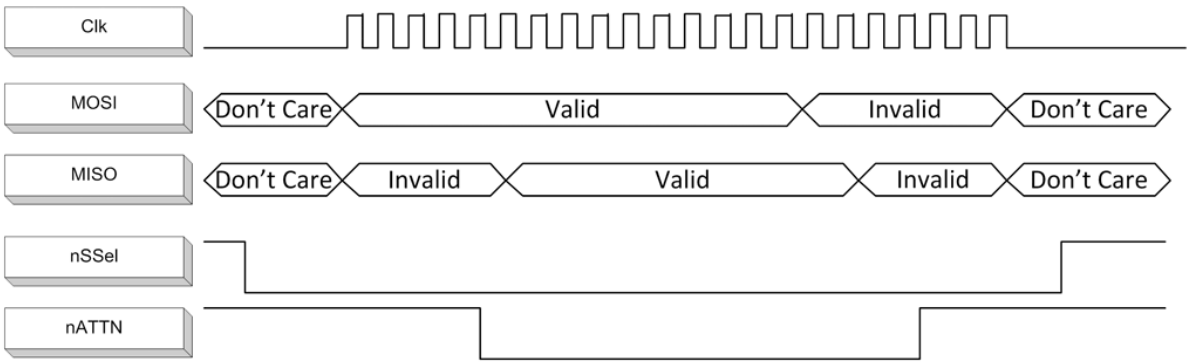
Full duplex operation

SPI on the XBee-PRO 900HP 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 17) 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.

SPI and API mode

The SPI only operates in API mode 1. The SPI does not support Transparent mode or API mode 2 (with escaped characters). This means that the **AP** configuration only applies to the UART interface and is ignored while using the 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-PRO 900HP 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-PRO 900HP 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.

Modes

Serial modes

The firmware operates in several different modes. Two top-level modes establish how the device communicates with other devices through its serial interface: Transparent operating mode and API operating mode. Use the **AP** command to choose Serial mode. XBee-PRO 900HP RF Modules use Transparent operation as the default serial mode.

The following modes describe how the serial port sends and receives data.

Transparent operating mode

Devices operate in this mode by default. The device acts as a serial line replacement when it is in Transparent operating mode. The device queues all UART data it receives through the DIN pin for RF transmission. When a device receives RF data, it sends the data out through the DOUT pin. You can set the configuration parameters using Command mode.

Note Transparent operating mode is not available when using the SPI interface; see SPI operation.

The device buffers data in the serial receive buffer until one of the following causes the data to be packetized and transmitted:

- The device receives no serial characters for the amount of time determined by the **RO** (Packetization Timeout) parameter. If **RO** = 0, packetization begins when a character is received.
- The device receives the Command Mode Sequence (**GT** + **CC** + **GT**). Any character buffered in the serial receive buffer before the sequence is transmitted.
- The device receives the maximum number of characters that fits in an RF packet (100 bytes). See NP (Maximum Packet Payload Bytes).

API operating mode

Application programming interface (API) operating mode is an alternative to Transparent mode. It is helpful in managing larger networks and is more appropriate for performing tasks such as collecting data from multiple locations or controlling multiple devices remotely. API mode is a frame-based protocol that allows you to direct data on a packet basis. It can be particularly useful in large networks where you need control over the operation of the radio network or when you need to know which node a data packet is from. The device communicates UART or SPI data in packets, also known as API frames. This mode allows for structured communications with serial devices.

The application programming interface (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.

For more information, see API mode overview.

Comparing Transparent and API modes

The XBee-PRO 900HP RF Module can use its serial connection in two ways: Transparent mode or API operating mode. You can use a mixture of devices running API mode and transparent mode in a network.

The following table compares the advantages of transparent and API modes of operation:

Feature	Description
Transparent mode features	
Simple interface	All received serial data is transmitted unless the device is in Command mode
Easy to support	It is easier for an application to support Transparent operation and Command mode
API mode features	
Easy to manage data transmissions to multiple destinations	Transmitting RF data to multiple remote devices only requires the application to change the address in the API frame. This process is much faster than in Transparent mode where the application must enter Command mode, change the address, exit Command mode, and then transmit data.
Each API transmission can return a transmit status frame indicating the success or reason for failure	Because acknowledgments are sent out of the serial interface, this provides more information about the health of the RF network and can be used to debug issues after the network has been deployed.
Received data frames indicate the sender's address	All received RF data API frames indicate the source address
Advanced addressing support	API transmit and receive frames can expose addressing fields including source and destination endpoints, cluster ID, and profile ID
Advanced networking diagnostics	API frames can provide indication of I/O samples from remote devices, and node identification messages. Some network diagnostic tools such as Trace Route, NACK, and Link Testing can only be performed in API mode.
Remote Configuration	Set/read configuration commands can be sent to remote devices to configure them as needed using the API
Simultaneous Commands	Query or set a configuration parameter while a pending command like **ND** is in progress. This cannot be done in Command mode. It is available in firmware versions 9009 or newer.

We recommend API mode when a device:

- Sends RF data to multiple destinations
- Sends remote configuration commands to manage devices in the network
- Receives RF data packets from multiple devices, and the application needs to know which device sent which packet

API mode is required when:

- Receiving I/O samples from remote devices
- Using SPI for the serial port

If the conditions listed above do not apply (for example, a sensor node, router, or a simple application), then Transparent operation might be suitable. It is acceptable to use a mixture of devices running API mode and Transparent mode in a network.

Modes of operation

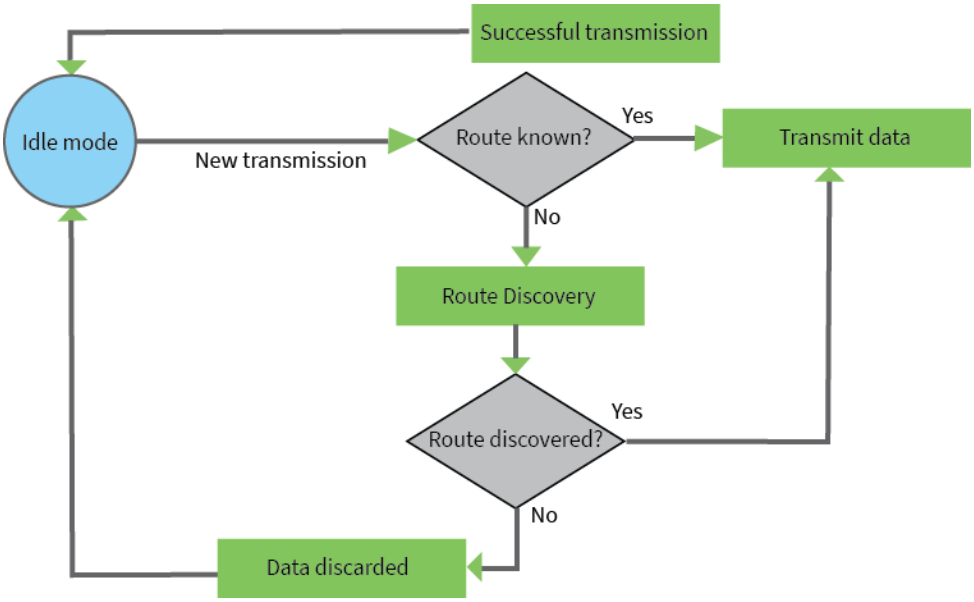
Idle mode

When not receiving or transmitting data, the device is in Idle mode. During Idle mode, the device listens for valid data on both the RF and serial ports.

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).
- Receive mode (valid RF data received through the antenna).
- Command mode (Command mode sequence issued, not available with Smart Energy software or when using the SPI port).

Transmit mode



When DigiMesh data is transmitted from one node to another, the destination node transmits a network-level acknowledgment back across the established route to the source node. This acknowledgment packet indicates to the source node that the destination node received the data packet. If the source node does not receive a network acknowledgment, it retransmits the data.

For more information, see Data transmission and routing.

Receive mode

This is the default mode for the XBee-PRO 900HP 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-PRO 900HP 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-PRO 900HP 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-PRO 900HP 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 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:

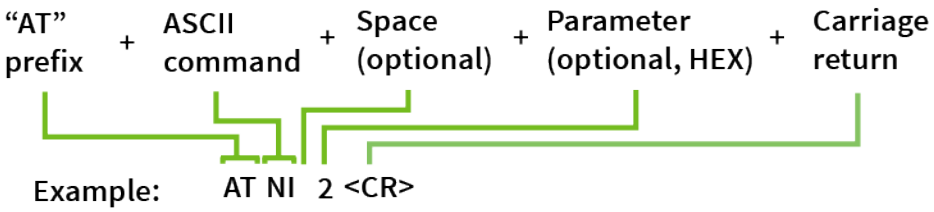
- A serial break for six seconds enters Command mode. You can issue the "break" command from a serial console, it is often a button or menu item.
- Asserting DIN (serial break) upon power up or reset enters Command mode. XCTU guides you through a reset and automatically issues the break when needed.

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-PRO 900HP 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-PRO 900HP 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. The device is almost completely off during sleep, and is incapable of sending or receiving data until it wakes up. 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. While asleep, nodes cannot receive RF messages or read commands from the UART port. For more information, see Sleep modes.

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).

The difference between a potential coordinator and a non-coordinator is that a non-coordinator node has its **SO** parameter set so that it will not participate in coordinator election and cannot ever be a sleep coordinator.

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.

In DigiMesh networks, a device functions in one of three roles:

1. A sleep coordinator.
2. A potential coordinator.
3. A non-coordinator.

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 through a process called nomination.
- 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 9).

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.

When you enable indirect messaging polling see CE (Node Messaging Options), when the device wakes, it sends a poll to the parent node. For more information, see Indirect messaging and polling.

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 ON_$\overline{\text{SLEEP}}$ line (pin pin 13) is asserted (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.

When you enable indirect messaging polling see CE (Node Messaging Options), when the device wakes, it sends a poll to the parent node. For more information, see Indirect messaging and polling.

Asynchronous cyclic sleep with pin wake up mode

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

(**SM** = 5) is similar to both the (**SM** = 1) and (**SM** = 4) modes. When the host asserts the SLEEP_REQUEST pin, the device enters a cyclic sleep mode similar to (**SM** = 4). When the host de-asserts the SLEEP_REQUEST pin, the device immediately wakes up. The device will not sleep when the SLEEP_REQUEST pin is de-asserted.

When you enable indirect messaging polling see CE (Node Messaging Options), when the device wakes, it sends a poll to the parent node. For more information, see Indirect messaging and polling.

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 does not sleep itself. At any time, the node responds to new nodes that attempt to join the sleeping network with a sync message. A sleep support node only transmits normal data when the other nodes in the sleeping network are awake.

Sleep support nodes are especially useful:

- When you use them as preferred sleep coordinator nodes.
- As aids in adding new nodes to a sleeping network.

Note Because sleep support nodes do not sleep, they should be mains powered.

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_$\overline{\text{SLEEP}}$ enabled) on a cyclic sleep node, the ON_$\overline{\text{SLEEP}}$ line asserts when the device is awake and de-asserts when the device is asleep.

If **D7** = 1, the device de-asserts $\overline{\text{CTS}}$ 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.

The sleep timer

If the device receives serial or RF data in Asynchronous cyclic sleep mode and Asynchronous cyclic sleep with pin wake up modes (**SM** = 4 or **SM** = 5), it starts a sleep timer (time until sleep).

- If the device receives any data serially or by RF link, the timer resets.
- Use ST (Wake Time) to set the duration of the timer.
- When the sleep timer expires the device returns to sleep.

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 indirect messaging poller with the **CE** command and set its **DH**:**DL** registers to match the **SH**:**SL** registers of the device that will function as the Indirect Messaging Coordinator. When you enable polling, the device sends a P2MP poll request regularly to the address specified by the **DH**:**DL** registers. When the device sends a P2MP unicast to the destination specified by the **DH**:**DL** of a polling device, 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.

The 200 K data rate firmware sends polls at least every 100 ms when awake. The 10 K data rate firmware sends polls at least every 300 ms when awake.

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. 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.

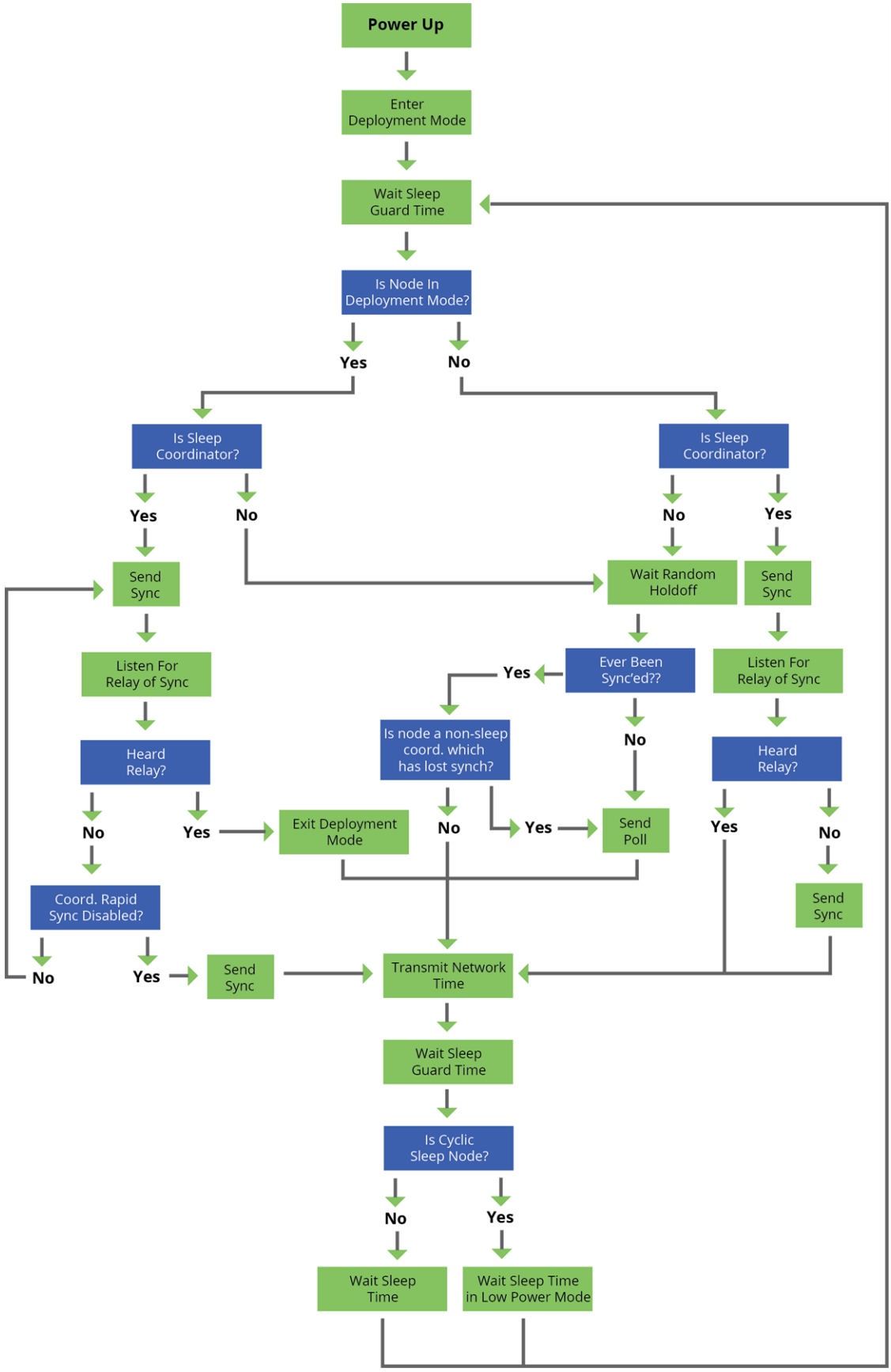
Use the **SO** (sleep options) command to disable deployment mode. This option is enabled by default.

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
Is node a non-sleep coord. which has lost synch?
No
Yes
Send Poll
No
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.

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. Also, it is not valid to have the sleep coordinator option bit set on more than one node at a time.

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, it is best to 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.
- It is not configured as a non-coordinator by setting bit 1 of **SO**.

If such a node wins out in the selection process, it becomes 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 always take priority over older sleep parameters. The age of the sleep parameters is determined by a sequence number that increments when an overriding

sync is sent.

2. Otherwise, the node with the preferred sleep coordinator bit set takes precedence.
3. Otherwise, a sleep support node—**SM 7**—takes priority over a node that is not a sleep support node—**SM 8**.
4. Otherwise, the node with highest serial number becomes the sleep coordinator.

Commissioning Pushbutton option

Use the Commissioning Pushbutton to select a device to act as the sleep coordinator.

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

Overriding syncs

Any sleep compatible node in the network that does not have the non-coordinator sleep option set can send an overriding sync and become the network sleep coordinator. An overriding sync effectively changes the synchronization of all nodes in the network to the **ST** and **SP** values of the node sending the overriding sync. It also selects the node sending the overriding sync as the network sleep coordinator. While this is a powerful operation, it may be an undesired side effect because the current sleep coordinator may have been carefully selected and it is not desired to change it. Additionally the current wake and sleep cycles may be desired rather than the parameters on the node sending the overriding sync. For this reason, it is important to know what kicks off an overriding sync.

An overriding sync occurs whenever **ST** or **SP** is changed to a value different than **OW** or **OS** respectively. For example no overriding sync will occur if **SP** is changed from **190** to **C8** if the network was already operating with **OS** at **C8**. On the other hand, if **SP** is changed from **190** to **190**—meaning no change—and **OS** is **C8**, than an overriding sync will occur because the network parameters are being changed.

Even parameters that seem unrelated to sleep can kick off an overriding sync. These are **NH**, **NN**, **RN**, and **MT**. When any of these parameters are changed, they can affect network traversal time. If such changes cause the configured value of **ST** to be smaller than the value needed for network traversal, then **ST** is increased and if that increased value is different than **OW**, then an overriding sync will occur.

For most applications, we recommend configuring the **NH**, **NN**, **RN**, and **MT** network parameters during initial deployment only. The default values of **NH** and **NN** are optimized to work for most deployments. Additionally, it would be best to set **ST** and **SP** the same on all nodes in the network while keeping **ST** sufficiently large so that it will not be affected by an inadvertent change of **NH**, **NN**, **RN**, or **MT**.

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

If you have nodes that are missing sync messages and could be going out of sync with the rest of the network, enabling an early wake gives the device a better chance to hear the sync messages that are being broadcast.

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 **NN** and **NH**.

   Based on the placement of the nodes in your network, select the appropriate values for the **NH** (Network Hops) and **NN** (Network Delay Slots) parameters.

   We optimize the default values of **NH** and **NN** 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 = **NN*****NH***(**MT**+1)*18 ms.

3. Select the duty cycle you want.
   The ratio of sleep time to wake time is the factor that has the greatest effect on the device's power consumption. Battery life can be estimated based on the following factors:
   - sleep period
   - wake time
   - sleep current
   - RX current
   - TX current
   - battery capacity

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**, **NN**, 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. In order to reduce the possibility of an unintended overriding sync, set **SP** and **ST** to the intended sleep/wake cycle on all nodes in the network. Be sure that **ST** is large enough to prevent it from being inadvertently increased by changing **NN**, **NH**, or **MT**.
7. Wait a sleep cycle for the sleeping nodes to sync themselves to the new **SP** and **ST** values.
8. Disable the preferred sleep coordinator option bit on the sleep coordinator unless you want a preferred sleep coordinator.
9. 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.

Networking methods

This section explains the basic layers and the three networking methods available on the XBee-PRO 900HP RF Modules, building from the simplest to the most complex.

The MAC and PHY layers

The PHY layer defines the physical and electrical characteristics of the network. It is responsible for managing the hardware that modulates and demodulates the RF bits.

The MAC layer is responsible for sending and receiving RF frames. As part of each packet, there is a MAC layer data header that has addressing information as well as packet options. This layer implements packet acknowledgments (ACKs), packet tracking to eliminate duplicates, and so forth.

- When a device is transmitting, it cannot receive packets.
- When a device is not sleeping, it is either receiving or transmitting.
- There are no beacons or master/slave requirements in the design of the MAC/PHY.

The XBee-PRO 900HP RF Module uses a patented method for scanning and finding a transmission. When a device transmits, it sends out a repeated preamble pattern, a MAC header, optionally a network header, followed by packet data. A receiving device is able to scan all the channels to find a transmission during the preamble, then once it has locked into that channel it attempts to receive the whole packet.

The following table shows the AT commands related to the MAC/PHY layers.

AT command	Function
CM	The Channel Mask is a user-defined list of channels that the device operates on. For additional information, see CM (Channel Mask).
HP	Change **HP** (Preamble ID) to make it so a group of devices will not interfere with another group of devices in the same vicinity. The advantage of changing this parameter is that a receiving device will not lock into a transmission of a transmitting device that does not have the same Preamble ID.
ID	Change **ID** (Network ID) to further keep devices from interfering with each other. The device matches this ID after it matches the preamble pattern and after it receives the MAC header. A unique network identifier distinguishes each network. For devices to communicate, they must be configured with the same network identifier. The **ID** parameter allows multiple networks to co-exist on the same physical channel.
PL	Sets the transmit (TX) power level. You can reduce the power level from the maximum to reduce current consumption or for testing. This comes at the expense of reduced radio range.
RR	Specifies the number of times a sending device attempts to get an ACK from a destination device when it sends a unicast packet.
MT	Specifies the number of times that a device repeatedly transmits a broadcast packet. This adds redundancy, which improves reliability.

64-bit addresses

We assign each device a unique IEEE 64-bit address at the factory. When a device is in API operating mode and it sends a packet, this is the source address that the receiving device returns.

- Use the **SH** and **SL** commands to read this address.
- The form of the address is: 0x0013A2XXXXXXXXXX.
- The first six digits are the Digi Organizationally Unique Identifier (OUI).
- The broadcast address is 0x000000000000FFFF.

Make a unicast transmission

To transmit to a specific device in Transparent operating mode:

- Set **DH**:**DL** to the **SH**:**SL** of the destination device.

To transmit to a specific device in API operating mode:

- In the 64-bit destination address of the API frame, enter the **SH**:**SL** address of the destination device.

Make a broadcast transmission

To transmit to all devices in Transparent operating mode:

- Set **DH**:**DL** to 0x000000000000FFFF.

To transmit to all devices in API operating mode:

- Set the 64-bit destination address to 0x000000000000FFFF.

The scope of the broadcast changes based on the delivery method you choose.

Delivery methods

The TO (Transmit Options) command sets the default delivery method that the device uses when in Transparent mode. In API mode, the TxOptions field of the API frame overrides the **TO** command, if non-zero.

The XBee-PRO 900HP RF Module supports three delivery methods:

- Point-to-multipoint (**TO** = 0x40).
- Repeater (directed broadcast) (**TO** = 0x80).
- DigiMesh (**TO** = 0xC0).

Point to Point / Point to Multipoint (P2MP)

This delivery method does not use a network header, only the MAC header.

In P2MP, the sending devices always send all messages directly to the destination. Other nodes do not repeat the packet. The sending device only delivers a P2MP unicast directly to the destination device, which must be in range of the sending device.

The XBee-PRO 900HP RF Module uses patented technology that allows the destination device to receive unicast transmissions directed to it, even when there is a large amount of traffic. This works best if you keep broadcast transmissions to a minimum.

A sending node repeats a P2MP broadcast transmission **MT**+1 times, but the receiving nodes do not repeat it, so like a unicast transmission, the receiving device must be in range.

All devices that receive a P2MP broadcast transmission output the data through the serial port.

P2MP throughput

The following table shows the throughput for the 10 kb/s version, using a 115.2 kb/s serial data rate.

Configuration	Data throughput
Point to point unicast, encryption disabled	8.8 kb/s
Point to point unicast, encryption enabled	8.7 kb/s

The following table shows the throughput for the 200 kb/s version, using a 115.2 kb/s serial data rate.

Configuration	Data throughput
Point to point unicast, encryption disabled	105.5 kb/s
Point to point unicast, encryption enabled	105.4 kb/s

Digi made data throughput measurements setting the serial interface rate to 115200 b/s, and measuring the time to send 100,000 bytes from source to destination. During the test, no route discoveries or failures occurred.

Repeater/directed broadcast

All of the routers in a network receive and repeat directed broadcast transmissions. Because it does not use ACKs, the originating node sends the broadcast multiple times. By default a broadcast transmission is sent four times—the extra transmissions become automatic retries without acknowledgments. This results in all nodes repeating the transmission four times. Sending frequent broadcast transmissions can quickly reduce the available network bandwidth, so use broadcast transmissions sparingly.

MAC layer

The MAC layer is the building block that is used to build repeater capability. To implement Repeater mode, we use a network layer header that comes after the MAC layer header in each packet. In this network layer there is additional packet tracking to eliminate duplicate broadcasts.

In this delivery method, the device sends both unicast and broadcast packets out as broadcasts that are always repeated. All repeated packets are sent to every device. The devices that receive the broadcast send broadcast data out their serial port.

When a device sends a unicast, it specifies a destination address in the network header. Then, only the device that has the matching destination address sends the unicast out its serial port. This is called a directed broadcast.

Any node that has a **CE** parameter set to router rebroadcasts the packet if its **BH** (broadcast hops) or broadcast radius values are not depleted. If a node has already seen a repeated broadcast, it ignores the broadcast.

The **NH** parameter sets the maximum number of hops that a broadcast transmission is repeated. The device always uses the **NH** value unless you specify a **BH** value that is smaller.

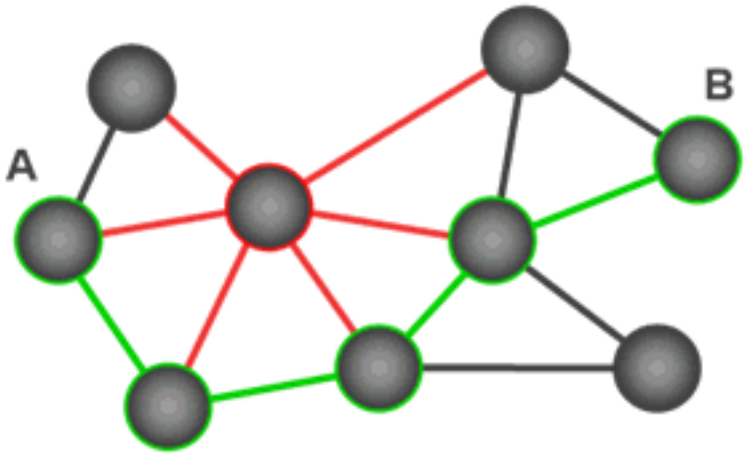
By default the **CE** parameter is set to route all broadcasts. As such, all nodes that receive a repeated packet will repeat it. If you change the **CE** parameter, you can limit which nodes repeat packets, which helps dense networks from becoming overly congested while packets are being repeated.

Transmission timeout calculations for Repeater/directed broadcast mode are the same as for DigiMesh broadcast transmissions.

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.

Related command: MR

In the same manner as the repeater delivery method, DigiMesh builds on P2MP and repeater modes. In DigiMesh, broadcasts always use the repeater delivery method, but unicasts use meshing technologies.

In the DigiMesh network layer, there are additional network layer ACKs and NACKs. Mesh networking allows messages to be routed through several different nodes to a final destination. DigiMesh firmware allows manufacturers and system integrators to bolster their networks with the self-healing attributes of mesh networking. In the event that one RF connection between nodes is lost (due to power-loss, environmental obstructions, and so on) critical data still reaches its destination due to the mesh networking capabilities embedded inside the modules. If you disable network ACKs, the network never heals.

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.

MAC retries and acknowledgments are used for transmissions between adjacent nodes in the route. NWK retries and acknowledgments are used across the entire route.

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.

Routing

A device within a mesh network determines 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 firmware uses an associative routing table to map a destination node address with its next hop. A device sends a message to the next hop address, and the message either reaches its destination or forwards to an intermediate router that routes the message on to its destination.

If a message has a broadcast address, it is broadcast to all neighbors, then all routers that receive the message rebroadcast the message **MT**+1 times. Eventually, the message reaches the entire network.

Packet tracking prevents a node from resending a broadcast message more than **MT**+1 times. This means that a node that relays a broadcast will only relay it after it receives it the first time and it will discard repeated instances of the same packet.

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 our empirical testing of throughput performance in a robust operating environment (low interference).

The results apply to the 200 kb/s version with a 115.2 kb/s serial data rate.

Configuration	Data throughput
Mesh unicast, 1 hop, encryption disabled	91.0 kb/s
Mesh unicast, 3 hop, encryption disabled	32.5 kb/s
Mesh unicast, 6 hop, encryption disabled	16.7 kb/s
Mesh unicast, 1 hop, encryption enabled	89.3 kb/s
Mesh unicast, 3 hop, encryption enabled	32.2 kb/s
Mesh unicast, 6 hop, encryption enabled	16.1 kb/s

Note We made the data throughput measurements by setting the serial interface rate to 115200 b/s, and measuring the time to send 100,000 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 Extended Transmit Status - 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 **%H** 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)

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 (Force Reset)

If you issue **FR** while the device is in Command Mode, the reset effectively exits Command mode.

Resets the device through the UART. The device responds immediately with an **OK** and performs a reset 100 ms later.

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 Binary commands are MAC/PHY commands.

AF (Available Frequencies)

You can query this read-only command to return a bitfield of the frequencies that are available in the device's region of operation. This command returns a bitfield. Each bit corresponds to a physical channel.

Note that the least significant bit in the bitmask selects the channel in the lowest frequency in the range.

Channels for these data rates are spaced 400 kHz apart.

Bit 0 – 902.400 MHz

Bit 1 – 902.800 MHz

.

.

.

Bit 31 – 914.800 MHz

.

.

.

Bit 63 – 927.600 MHz

Parameter range

0x1FFFFFF – 0x00FFFFFFFFFFFFFFFF

Default

Region	Bitfield	Channels
United States/Canada	0x00FFFFFFFFFFFFFFFF	0-63
Australia	0x00FFFFFFFE00000000	33-63
Brazil	0x00FFFFFFFE00000FFF	0-11, 33-63
Singapore	0x00FFE00000000000	63-52

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 command is a bitfield. Each bit in the bitfield corresponds to a frequency as defined in the **AF** (Available Frequencies) command. When you set a bit in **CM** and the corresponding bit in **AF** is 1, then the device can choose that channel as an active channel for communication.

A minimum of **MF** channels must be made available for the device to communicate on. You can use the **MF** command to query the minimum number of channels required for operation. If a **CM** setting would result in less than **MF** active channels being enabled, then the device returns an error. If there are

more active channels enabled than required by **MF**, then the device uses the first **MF** frequencies; higher active frequencies may be unused in favor of lower ones.

Exactly MF (Minimum Frequency Count) number of channels must be made available for the device to communicate on.

All devices in a network must use an identical set of active channels in order to communicate. Separate networks that are in physical range of each other should use different **HP** (Preamble Patterns) and/or **ID** (Network IDs) to avoid receiving data from the other network.

You may find the **ED** (Energy Detect) command useful when choosing what channels to enable or disable.

Note Channel 19 (910.000 MHz) is disabled by default. This channel has approximately 2 dBm worse receiver sensitivity than other channels. We suggest that you do not use this channel.

Parameter range

0x1FFFFFF – 0x00FFFFFFFFFFFFFFF

Default

0xFFFFFFFFFFF7FFFF

MF (Minimum Frequency Count)

You can query this read-only command to determine the minimum number of channels that you must enable with the **CM** command for proper operation in the device's region of operation.

Parameter range

1 - 50

Default

United States/Canada: 25

Australia: 25

Brazil: 25

Singapore: 11

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.

Note When using devices certified for use in Singapore, **HP** settings of 1, 2, or 3 have reduced performance compared to the other settings. Avoid these settings in this region.

Parameter range

0 - 7

Default

0

ID (Network ID)

Set or read the user network identifier.

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

Devices can only communicate with other devices that have the same network identifier and channel configured.

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

PL (TX Power Level)

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

PL= **4** is calibrated and the remaining power levels are approximate.

Parameter range

Setting	Power level
0	+7 dBm (5 mW)
1	+15 dBm (32 mW)
2	+18 dBm (63 mW)
3	+21 dBm (125 mW)
4	+24 dBm (250 mW)

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

0xA

Diagnostic commands

The following commands are diagnostic commands.

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.

If the XBee-PRO 900HP RF Module has been reset and has not yet received a packet, **DB** reports **0**.

This value is volatile (the value does not persist in the device's memory after a power-up sequence).

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. Received MAC ACK packets do not increment this counter. 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.

This value is volatile (the value does not persist in the device's memory after a power-up sequence).

Parameter range

0 - 0xFFFF

Default

0

EA (MAC ACK Failure Count)

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 unsigned value, append a hexadecimal parameter to the command. This value is volatile (the value does not persist in the device's memory after a power-up sequence).

Parameter range

0 - 0xFFFF

Default

0

TR (Transmission Failure Count)

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

0xCF

0x267

%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

0x1BE

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 Indirect Messaging Poller requests them. Setting a device as an Indirect Messaging Poller 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 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 maximum transmission hops for broadcast data transmissions.

If you set **BH** greater than **NH**, the device uses the value of **NH**. Both variants of firmware support this command.

Parameter range

0 - 0x20

Default

0

NH (Network Hops)

Sets or displays the maximum number of hops across the network. This parameter limits the number of hops. You can use this parameter to calculate the maximum network traversal time.

You must set this parameter to the same value on all nodes in the network.

Both variants are supported.

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 - 0x05

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.

Changing this value dramatically changes how long a route request takes.

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.

Note This command is supported in the 200k variant only.

Parameter range

0 - 7

Default

1

RN (Delay Slots)

Sets or displays the time delay that the transmitting device inserts before attempting to resend a packet. If the transmitting device fails to receive an acknowledgment after sending a packet, it inserts a random number of delay slots (ranging from 0 to [**RN** minus 1]) before attempting to resend the packet. Each delay slot is 38 ms.

If two devices attempt to transmit at the same time, the random time delay after packet failure only allows one device to transmit the packet successfully, while the other device waits until the channel is available for RF transmission.

RN is only applicable if:

- You enable retries using the **RR** command, or
- You insert forced delays into a transmission using the **TT** command

Parameter range

0 - 0xFF [slots]

Default

0 (no delay slots inserted)

Addressing commands

SH (Serial Number High)

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

The 64-bit source address is always enabled. This value is read-only and it never changes.

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-PRO in the factory.

The device's serial number is set at the factory and is read-only.

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.

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.

Parameter range

0 - 0xFF

Bit field:

Bit	Meaning	Description
6,7	Delivery method	b'00 = <invalid option> b'01 = Point-multipoint b'10 = Directed Broadcast (0x80) b'11 = DigiMesh (not available in the 10k product)
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

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 on the 10k build.
- Bits 4 and 5 must be set to 0.
- Bits 1, 2, and 3 cannot be set on the 10k build.

Default

0x40 (10k product)

0xC0 (200k product)

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.

- XCTU prevents you from exceeding the string limit of 20 characters for this command. If you are using another software application to send the string, you can enter longer strings, but the software on the device returns an error.

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 0 to 20 bytes in length. A carriage return or a comma automatically ends the command.

Default

0x20 (an ASCII space character)

NT (Node Discover Time)

Sets the amount of time a base node waits for responses from other nodes when using the ND (Network Discover), DN (Discover Node), and FN (Find Neighbors) commands. When a discovery is performed, the broadcast transmission includes the **NT** value to provide all remote devices with a response timeout. Remote devices wait a random time, less than **NT**, before sending their response to avoid collisions.

The **N?** command should be used to determine how long the actual discovery timeout will be based on current device configuration.

Parameter range

0x20 - 0x2EE0 (x 100 ms)

Default

0x82 (13 seconds)

NO (Node Discovery Options)

Use **NO** to suppress or include a self-response to **ND** (Node Discover) commands. When **NO** bit 1 = 1, a device performing a Node Discover includes a response entry for itself.

Use **NO** to suppress or include certain data in the **ND** (Node Discovery) response.

Parameter range

0 - 0x07 (bit field)

Bit field

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

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 (Transparent data cluster ID)

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.

Note The **AG** command is only available on products that support DigiMesh.

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 device returns an **ERROR** message if it is given without a destination node (that is without a parameter) or if the given destination node does not respond within **N?** milliseconds. If an **ERROR** is received, the device does not exit Command mode.

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 devices found in the network. For each discovered device, the following information is returned:

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>. **ND** also accepts NI (Node Identifier) as a parameter (optional). In this case, only a device that matches the supplied identifier responds.

If you send **ND** through a local API frame, the device returns each response as a separate AT_CMD_ Response packet. The data consists of the bytes listed above without the carriage return delimiters. The **NI** string ends in a **0x00** null character.

Parameter range

20-byte printable ASCII string

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 (Node Discovery Options) settings.)

RSSI OF LAST HOP<CR> (1 byte. Optionally included based on NO (Node Discovery Options) 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 AT Command (0x08) or remote AT command (0x17) API frame, each response returns as a separate AT Command Response (0x88) or Remote Command Response (0x97) frame, 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.

FN accepts a **NI** (Node Identifier) as an argument.

Parameter range

0 to 20 ASCII characters

Default

N/A

Security commands

The following AT commands are security commands.

EE (Security Enable)

Enables or disables 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 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

The following AT commands are serial interfacing commands.

BD (Baud Rate)

Values from 0 - 8 select preset standard rates.

Values at 0x39 and above select the actual baud rate if the host supports it.

Parameter range

Standard baud rates: 0x0 - 0x8

Non-standard baud rates: 0x100 to 0x6ACFC0

Value	Description
0x1	2,400 b/s
0x2	4,800 b/s
0x3	9,600 b/s
0x4	19,200 b/s
0x5	38,400 b/s
0x6	57,600 b/s
0x7	115,200 b/s
0x8	230,400 b/s

Default

0x03 (9600 b/s)

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 in the data packet.

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

0x11 - 0x16F bytes

Default

0x13F

AP (API Mode)

Set or read the API mode setting. The device can format the RF packets it receives into API frames and send them out the serial port.

Parameter range

0 - 2

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.

Parameter	Description
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. This parameter applies to both the UART and SPI interfaces.

Use **AO** to enable different API output frames.

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 Commissioning pushbutton and associate LED.

See Commissioning pushbutton.

Parameter range

0 - 4

Default

N/A

D0 (DIO0/AD0)

Sets or displays the DIO0/AD0 configuration (pin 20).

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 19).

Parameter range

0 - 6

Parameter	Description
0	Disabled
1	Commissioning button
1	SPI_$\overline{\text{ATTN}}$
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high
6	UART Data Present Indicator
6	PTI_EN

Default

0

D2 (DIO2/AD2)

Sets or displays the DIO2/AD2 configuration (pin 18).

Parameter range

0 - 5

0 - 1

Parameter	Description
0	Disabled
1	SPI_CLK
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 17).

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	SPI slave select
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D4 (DIO4)

Sets or displays the DIO4 configuration (pin 11).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	SPI_MOSI
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/ASSOCIATED_INDICATOR configuration (pin 15).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	Associate LED indicator - blinks when associated
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 16).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	$\overline{\text{RTS}}$ flow control

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

Default

0

D7 (DIO7/CTS)

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration (pin 12).

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/SLEEP_REQUEST)

Sets or displays the DIO8/SLEEP_REQUEST configuration (pin 9).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	Sleep request

Parameter	Description
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 13).

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 ().
Sets or displays the DIO10/PWM0 configuration (pin 6).

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	RSSI PWM0 output
2	PWM0 output

Parameter	Description
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 7).

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 4).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	SPI_MISO
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P3 (DIO13/DOUT)

Sets or displays the DIO13/DOUT configuration (pin 2).

Parameter range

0, 1

Parameter	Description
0	Disabled
1	UART DOUT output

Default

1

P4 (DIO14/DIN)

Sets or displays the DIO14/DIN configuration (pin 3).

Parameter range

0, 1

Parameter	Description
0	Disabled
1	UART DIN/output

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 - 0x7FFF

Default

0x0

PR (Pull-up/Down Resistor Enable)

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

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.

Bit	I/O line
0	DIO4 / AD4 / SPI_MOSI
1	DIO3 / AD3 / SPI_$\overline{\text{SSEL}}$
2	DIO2 / AD2 / SPI_SCLK
3	DIO1 / AD1 / SPI_$\overline{\text{ATTN}}$
4	DIO0 / AD0
5	DIO6 / $\overline{\text{RTS}}$
6	SLEEP_REQUEST
7	DIN / $\overline{\text{CONFIG}}$
8	DIO5 / AD5 / ASSOCIATE
9	On/$\overline{\text{SLEEP}}$
10	DIO12 / SPI_MISO
11	DIO10 / PWM0 / RSSI
12	DIO11/ PWM1
13	DIO7/$\overline{\text{CTS}}$
14	PWM0 / DOUT

Parameter range

0 - 0x7FFF (bit field)

Default

0x7FFF

M0 (PWM0 Duty Cycle)

The duty cycle of the PWM0 line (pin 6).
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 7).
Use the **P1** command to configure the line as a PWM output.

Parameter range

0 - 0x3FF

Default

0

LT (Associate 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.

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

For all other **LT** values, the firmware measures **LT** in 10 ms increments.

Parameter range

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, 1

Parameter	Description
0	1.25 V reference
1	2.5 V reference

Default

1

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

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

0 - 0xFF

Default

1

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 and transmits all enabled digital I/O and analog inputs every **IR** milliseconds. I/O Samples transmit to the address specified by **DT**.

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.

For more information, see the following commands:

- D0 (DIO0/AD0) through D9 (DIO9/ON_SLEEP).
- P0 (DIO10/RSSI/PWM0 Configuration) through P4 (DIO14/DIN).

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

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

IS (Force Sample)

Forces a read of all enabled digital and analog input lines. The data is returned through the UART or SPI.

When operating in Transparent mode (**AP** = 0), the data is returned in the following format:

All bytes are converted to ASCII:

number of samples<CR>

channel mask<CR>

DIO data<CR> (If DIO lines are enabled)

ADC channel Data<CR> (This will repeat for every enabled ADC channel)

<CR> (end of data noted by extra <CR>)

When operating in API mode (**AP** = 1), the command immediately returns an **OK** response. The data follows in the normal API format for DIO data.

Parameter range

N/A

Default

N/A

TP (Board Temperature)

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

Parameter range

0x00 - 0xFF [read-only]

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

Default

N/A

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.
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 Pin Wake. When you assert the SLEEP_RQ pin, the device enters a cyclic sleep mode similar to Asynchronous Cyclic Sleep. When you de-assert the SLEEP_RQ pin, the device immediately wakes up. The device does not sleep when you de-assert the SLEEP_RQ pin.
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 can set or clear any of the available sleep option bits.

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

Parameter range

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
4	Enable node type equality
5	Disable lone coordinator sync repeat

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

Bit	Option
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: If non-zero, **SP** determines the time to hold an indirect message for an Indirect Messaging Poller before discarding it. A Coordinator discards indirect messages after a period of (2.5 * **SP**).

Parameter range

0x1 - 1440000 (x 10 ms)

Default

0x12C (3 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) (one hour)

Default

0xBB8 (3 seconds)

WH (Wake Host Delay)

Sets or displays the wake host timer value. You can use **WH** to give a sleeping host processor sufficient time to power up after the device asserts the ON_SLEEP line.

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

0x12C

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

0xBB8

MS (Missed Sync Messages)

Reads the number of sleep or wake cycles since the device received a sync message.

Supported in the mesh firmware variant only.

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 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.

Note We recommend using the a value within the rage of 0x20 - 0x7F as those are ASCII characters.

Parameter range

0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

Default

0x2B (the ASCII plus character: **+**)

CN (Exit Command Mode)

Immediately exits Command Mode and applies pending changes.

Parameter range

N/A

Default

N/A

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)

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.

For example, if the device is version S8B, this returns 0x801.

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.

Parameter range

0 - 0xFFFFFFFF

Default

0xB0000

NP (Maximum Packet Payload Bytes)

Reads the maximum number of RF payload bytes that you can send in a transmission.

Using APS encryption (API transmit option bit enabled), reduces the maximum payload size by 9 bytes.

Using source routing (**AR** < 0xFF), further reduces the maximum payload size.

Note **NP** returns a hexadecimal value. For example, if **NP** returns 0x54, this is equivalent to 84 bytes.

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. Use the code that the device returns to determine if a node has the configuration you want.

After a firmware update this command may return a different value.

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.

API frame format

The firmware supports two API operating modes: without escaped characters and with escaped characters. Use the AP command to enable either mode. To configure a device to one of these modes, set the following AP parameter values:

- **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 operation (AP parameter = 1)

The following table shows the data frame structure when you enable **AP** = 1:

Frame fields	Byte	Description
Start delimiter	1	0x7E
Length	2 - 3	Most Significant Byte, Least Significant Byte
Frame data	4 - n	API-specific structure
Checksum	n + 1	1 byte

The firmware silently discards any data it receives prior to the start delimiter. If the device does not receive the frame correctly or if the checksum fails, the device replies with a device status frame indicating the nature of the failure.

API operation-with escaped characters (AP parameter = 2)

This mode is only available on the UART, not on the SPI serial port. The following table shows the data frame structure when you enable **AP** = 2:

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	

Escape characters

When you are sending or receiving a UART 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 XOR'd with 0x20.

Data bytes that need to be escaped:

Byte	Description
0x7E	Frame Delimiter
0x7D	Escape
0x11	XON
0x13	XOFF

Example: Raw serial data before escaping interfering bytes:

0x7E 0x00 0x02 0x23 0x11 0xCB

0x11 needs to be escaped which results in the following frame:

0x7E 0x00 0x02 0x23 0x7D 0x31 0xCB

Note In the previous example, the length of the raw data (excluding the checksum) is 0x0002 and the checksum of the non-escaped data (excluding frame delimiter and length) is calculated as:
0xFF - (0x23 + 0x11) = (0xFF - 0x34) = 0xCB.

Length

The length field specifies the total number of bytes included in the frame's data field. Its two-byte value excludes the start delimiter, the length, and the checksum.

Frame data

This field contains the information that a device receives or transmits. The structure of frame data depends on the purpose of the API frame:

Start delimiter	Length		Frame data								Checksum
			API identifier	Identifier-specific Data							
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	cmdID	cmdData							Single byte

The cmdID frame (API-identifier) indicates which API messages contains the cmdData frame (Identifier-specific data). The device sends multi-byte values big endian format.

The XBee-PRO 900HP RF Module supports the following API frames:

API frame names and IDs sent to the module

API frame names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
TX Request	0x10
Explicit TX Request	0x11
Remote Command Request	0x17

API frame names and IDs received from the device

API frame names	API ID
AT Command Response	0x88
Modem Status	0x8A
Transmit Status	0x8B
Route information packet	0x8D
Aggregate Addressing Update frame	0x8E
RX Indicator (AO=0)	0x90
Explicit Rx Indicator (AO=1)	0x91
Data Sample Rx Indicator frame	0x92
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97

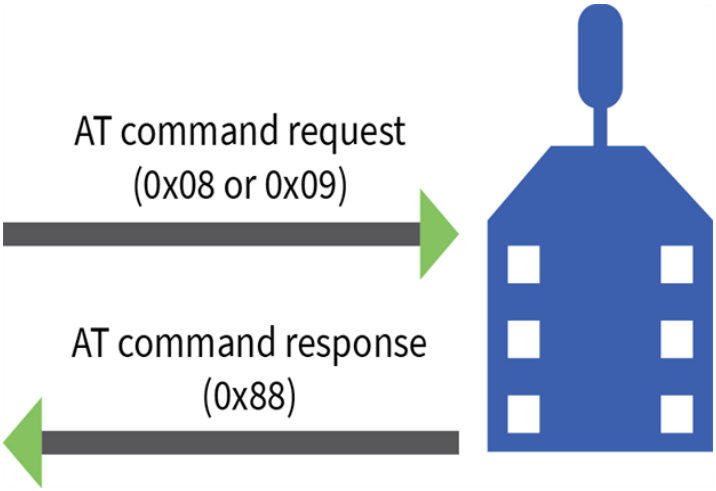
Note Requests are less than 0x80, and responses are always 0x80 or higher.

API serial exchanges

Note Using a Frame ID of 0 disables responses, which can reduce network congestion for non-critical transmissions.

AT commands

The following image shows the API frame exchange that takes place at the serial interface when sending an AT command request to read or set a device parameter. You can disable the response by setting the frame ID to 0 in the request.



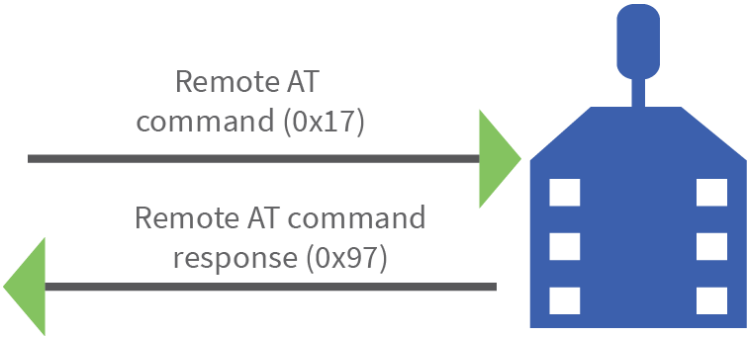
Transmit and Receive RF data

The following image shows the API frames exchange that take place at the UART 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 (0x90 or 0x91) is determined by the **AO** command.

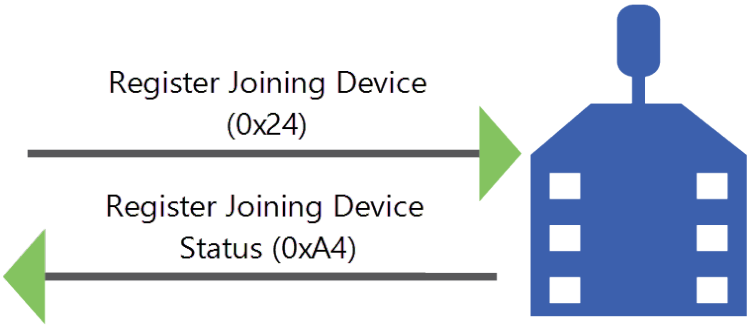
Remote AT commands

The following image shows the API frame exchanges that take place at the serial interface when sending a remote AT command. The device does not send out a remote command response frame through the serial interface if the remote device does not receive the remote command.



Device Registration

The following image shows the API frame exchanges that take place at the serial interface when registering a joining device to a trust center.



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 0x47 (the two far right digits). Subtract 0x47 from 0xFF and you get 0xB8 (0xFF - 0x47 = 0xB8). 0xB8 is the checksum for this data packet.

If an API data packet is composed with an incorrect checksum, the XBee-PRO 900HP 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

Frame descriptions

The following sections describe the API frames.

64-bit Transmit Request - 0x00

Response frame: Transmit Status - 0x89

Description

This frame type is used to send serial payload data as an RF packet to a remote device with a corresponding 64-bit IEEE address.

Note This frame format is deprecated and should only be used by customers who require compatibility with legacy Digi RF products. For new designs, we encourage you to use Transmit Request - 0x10 to initiate API transmissions.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	64-bit Transmit Request - **0x00**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**Destination address**	Set to the 64-bit IEEE address of the destination device. If set to **0x000000000000FFFF**, the broadcast address is used.
13	8-bit	**Options**	A bit field of options that affect the outgoing transmission: ■ **Bit 0**: Disable MAC ACK [**0x01**] ■ Bit 1: Reserved (set to 0) ■ **Bit 2**: Send packet with Broadcast PAN ID [**0x04**] • 802.15.4 firmwares only **Note** Option values may be combined. Set all unused bits to 0.
14-n	variable	**RF data**	The serial data to be sent to the destination. Use **NP** to query the maximum payload size that can be supported based on current settings.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to a device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**".

The corresponding Transmit Status - 0x89 response with a matching Frame ID will indicate whether the transmission succeeded.

```
7E 00 11 00 52 00 13 A2 00 12 34 56 78 00 54 78 44 61 74 61 9E
```

Frame type	Frame ID	64-bit dest address	Tx options	RF data
0x00	0x52	0x0013A200 12345678	0x00	0x547844617461
Input	*Matches response*			*"TxData"*

64-bit broadcast

Sending a broadcast transmission of the serial data "**Broadcast**" and suppressing the corresponding response by setting Frame ID to **0**.

```
7E 00 14 00 00 00 00 00 00 00 00 FF FF 00 42 72 6F 61 64 63 61 73 74 6E
```

Frame type	Frame ID	64-bit dest address	Tx options	RF data
0x00	0x00	0x00000000 0000FFFF	0x00	0x42726F616463617374
Input	*Suppress response*	*Broadcast address*		*"Broadcast"*

Local AT Command Request - 0x08

Response frame: Local AT Command Response - 0x88

Description

This frame type is used to query or set command parameters on the local device. Any parameter that is set with this frame type will apply the change immediately. If you wish to queue multiple parameter changes and apply them later, use the Queue Local AT Command Request - 0x09 instead.

When querying parameter values, this frame behaves identically to Queue Local AT Command Request - 0x09: You can query parameter values by sending this frame with a command but no parameter value field—the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Local AT Command Response - 0x88 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x88 response is the same one set by the command in the 0x08 request frame.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Local AT Command Request - **0x08**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set the local command parameter

Set the **NI** string of the radio to "**End Device**".

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will indicate whether the parameter change succeeded.

```
7E 00 0E 08 A1 4E 49 45 6E 64 20 44 65 76 69 63 65 38
```

Frame type	Frame ID	AT command	Parameter value
0x08	0xA1	0x4E49	0x456E6420446576696365
Request	*Matches response*	*"NI"*	*"End Device"*

Query local command parameter

Query the temperature of the module—**TP** command.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will return the temperature value.

```
7E 00 04 08 17 54 50 3C
```

Frame type	Frame ID	AT command	Parameter value
0x08	0x17	0x5450	(omitted)
Request	*Matches response*	*"TP"*	*Query the parameter*

Queue Local AT Command Request - 0x09

Response frame: Local AT Command Response - 0x88

Description

This frame type is used to query or set queued command parameters on the local device. In contrast to Local AT Command Request - 0x08, this frame queues new parameter values and does not apply them until you either:

- Issue a Local AT Command using the 0x08 frame
- Issue an **AC** command—queued or otherwise

When querying parameter values, this frame behaves identically to Local AT Command Request - 0x08: You can query parameter values by sending this frame with a command but no parameter value field—the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Local AT Command Response - 0x88 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x88 response is the same one set by the command in the 0x09 request frame.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Queue Local AT Command Request - **0x09**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register at a later time. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Queue setting local command parameter

Set the UART baud rate to 115200, but do not apply changes immediately.

The device will continue to operate at the current baud rate until the change is applied with a subsequent **AC** command.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will indicate whether the parameter change succeeded.

```
7E 00 05 09 53 42 44 07 16
```

Frame type	Frame ID	AT command	Parameter value
0x09	0x53	0x4244	0x07
Request	*Matches response*	*"BD"*	*7 = 115200 baud*

Query local command parameter

Query the temperature of the module (**TP** command).

The corresponding Local AT Command Response - 0x88 frame with a matching Frame ID will return the temperature value.

```
7E 00 04 09 17 54 50 3B
```

Frame type	Frame ID	AT command	Parameter value
0x09	0x17	0x5450	(omitted)
Request	*Matches response*	*"TP"*	*Query the parameter*

Transmit Request - 0x10

Response frame: Extended Transmit Status - 0x8B

Description

This frame type is used to send payload data as an RF packet to a specific destination. This frame type is typically used for transmitting serial data to one or more remote devices.

The endpoints used for these data transmissions are defined by the **SE** and **EP** commands and the cluster ID defined by the **CI** command—excluding 802.15.4. To define the application-layer addressing fields on a per-packet basis, use the Explicit Addressing Command Request - 0x11 instead.

Query the **NP** command to read the maximum number of payload bytes that can be sent.

64-bit addressing

- 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

Format

The following table provides the contents of the frame. For details on the frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Request - **0x10**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response frame. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**.
13	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
15	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to**0**—recommended—the value of **NH**specifies the broadcast radius.

Offset	Size	Frame Field	Description
16	8-bit	**Transmit options**	See the Transmit options bit field table below for available options. If set to **0**, the value of **TO** specifies the transmit options.
17-n	variable	**Payload data**	Data to be sent to the destination device. Up to **NP** bytes per packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

The available transmit options vary depending on the protocol being used. Bitfield options can be combined. Set all unused bits to **0**.

DigiMesh

0	Disable ACK [**0x01**]	Disable acknowledgments on all unicasts.
1	Disable route discoveries [**0x02**]	Disable Route Discovery on all DigiMesh unicasts.
2	Unicast NACK [**0x04**]	Enable unicast NACK messages on DigiMesh transmissions When set, a failed transmission will generate a Route Information - 0x8D frame for diagnosis.
3	Unicast trace route [**0x08**]	Enable a unicast Trace Route on DigiMesh transmissions When set, the transmission will generate a Route Information - 0x8D frame.
4	Secure Session Encryption [**0x10**]	Encrypt payload for transmission across a Secure Session. Reduces maximum payload size by 4 bytes.
5	Reserved	<set this bit to 0>
6,7	Delivery method	b'00 = <invalid option> b'01 = Point-multipoint [**0x40**] b'10 = Directed Broadcast [**0x80**] b'11 = DigiMesh [**0xC0**]

Examples

Each example is written without escapes (**AP**=**1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to a device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**". Transmit options are set to **0**, which means the transmission will send using the options set by the **TO** command.

The corresponding Transmit Status - 0x89 response with a matching Frame ID will indicate whether the transmission succeeded.

```
7E 00 14 10 52 00 13 A2 00 12 34 56 78 FF FE 00 00 54 78 44 61 74 61 91
```

Frame type	Frame ID	64-bit dest	Reserved	Bcast radius	Options	RF data
0x10	0x52	0x0013A200 12345678	0xFFFE	0x00	0x00	0x547844617461
Request	*Matches response*	*Destination*	*Unused*	*N/A*	*Will use **TO***	*"TxData"*

64-bit broadcast

Sending a broadcast transmission of the serial data "**Broadcast**" to neighboring devices and suppressing the corresponding response by setting Frame ID to **0**.

```
7E 00 17 10 00 00 00 00 00 00 00 00 FF FF FF FE 01 00 42 72 6F 61 64 63 61 73 74 60
```

Frame type	Frame ID	64-bit dest	Reserved	Bcast radius	Tx Options	RF data
0x10	0x00	0x00000000 0000FFFF	0xFFFE	0x01	0x00	0x42726F616463617374
Request	*Suppress response*	*Broadcast address*	*Unused*	*Single hop broadcast*	*Will use **TO***	*"Broadcast"*

Explicit Addressing Command Request - 0x11

Response frame: Extended Transmit Status - 0x8B

Description

This frame type is used to send payload data as an RF packet to a specific destination using application-layer addressing fields. The behavior of this frame is similar to Transmit Request - 0x10, but with additional fields available for user-defined endpoints, cluster ID, and profile ID.

This frame type is typically used for OTA updates, and serial data transmissions.

Query NP (Maximum Packet Payload Bytes) to read the maximum number of payload bytes that can be sent.

64-bit addressing

- 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

Reserved endpoints

For serial data transmissions, the **0xE8** endpoint should be used for both source and destination endpoints.

Endpoints **0xDC** - **0xEE** are reserved for special use by Digi and should not be used in an application outside of the listed purpose. The XBee 802.15.4 firmware only supports digi-specific endpoints, endpoints used outside of this range will be interpreted as the **0xE8** data endpoint.

The active Digi endpoints are:

- **0xE8** - Digi Data endpoint
- **0xE6** - Digi Device Object (DDO) endpoint
- **0xE5** - XBee3 - Secure Session Server endpoint
- **0xE4** - XBee3 - Secure Session Client endpoint
- **0xE3** - XBee3 - Secure Session SRP authentication endpoint

Reserved cluster IDs

For serial data transmissions, the **0x0011** cluster ID should be used.

The following cluster IDs can be used on the **0xE8** data endpoint:

- **0x0011**- Transparent data cluster ID
- **0x0012** - Loopback cluster ID:The destination node echoes any transmitted packet back to the source device. Cannot be used on XBee 802.15.4 firmware.

Reserved profile IDs

The Digi profile ID of **0xC105** should be used when sending serial data between XBee devices.

Format

The following table provides the contents of the frame. For details on the frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Addressing Command Request - **0x11**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**. Zigbee coordinator address is **0x0000000000000000**. When using 16-bit addressing, set this field to **0xFFFFFFFFFFFFFFFF**.
13	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
15	8-bit	**Source Endpoint**	Source endpoint for the transmission. Serial data transmissions should use **0xE8**.
16	8-bit	**Destination Endpoint**	Destination endpoint for the transmission. Serial data transmissions should use **0xE8**.
17	16-bit	**Cluster ID**	The Cluster ID that the host uses in the transmission. Serial data transmissions should use **0x11**.
19	16-bit	**Profile ID**	The Profile ID that the host uses in the transmission. Serial data transmissions between XBee devices should use **0xC105**.
21	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to **0** (recommended), the value of **NH** specifies the broadcast radius.
22	8-bit	**Transmit options**	See the Transmit options bit field table below for available options. If set to **0**, the value of **TO** specifies the transmit options.
23-n	variable	**Command data**	Data to be sent to the destination device. Up to **NP** bytes per packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

The available transmit options vary depending on the protocol being used. Bitfield options can be combined. Set all unused bits to **0**.

DigiMesh

Bit	Meaning	Description
0	Disable ACK [**0x01**]	Disable acknowledgments on all unicasts.
1	Disable route discoveries [**0x02**]	Disable Route Discovery on all DigiMesh unicasts.
2	Unicast NACK [**0x04**]	Enable unicast NACK messages on DigiMesh transmissions When set, a failed transmission will generate a Route Information - 0x8D frame for diagnosis.
3	Unicast trace route [**0x08**]	Enable a unicast Trace Route on DigiMesh transmissions When set, the transmission will generate a Route Information - 0x8D frame.
4	Secure Session Encryption [**0x10**]	Encrypt payload for transmission across a Secure Session Reduces maximum payload size by 4 bytes.
5	Reserved	<set this bit to 0>
6,7	Delivery method	b'00 = <invalid option> b'01 = Point-multipoint [**0x40**] b'10 = Directed Broadcast [**0x80**] b'11 = DigiMesh [**0xC0**]

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to an XBee device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**". Transmit options are set to **0**, which means the transmission will send using the options set by the **TO** command. This transmission is identical to a Transmit Request - 0x10 using default settings.

The corresponding Extended Transmit Status - 0x8B response with a matching Frame ID will indicate whether the transmission succeeded.

```
7E 00 1A 11 87 00 13 A2 00 12 34 56 78 FF FE E8 E8 00 11 C1 05 00 00 54 78 44 61
74 61 B4
```

Frame type	Frame ID	64-bit dest	Reserved	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0x87	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0011	0xC105	0x00	0x00	0x5478446174 61
Explicit request	*Matches response*	*Destination*	*Unused*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Use **TO***	*"TxData"*

Loopback Packet

Sending a loopback transmission to an device with the 64-bit address of **0013A20012345678** using Cluster ID **0x0012**. To better understand the raw performance, retries and acknowledgements are disabled.

The corresponding Extended Transmit Status - 0x8B response with a matching Frame ID can be used to verify that the transmission was sent.

The destination will not emit a receive frame, instead it will return the transmission back to the sender. The source device will emit the receive frame—the frame type is determined by the value of **AO**—if the packet looped back successfully.

```
7E 00 1A 11 F8 00 13 A2 00 12 34 56 78 FF FE E8 E8 00 12 C1 05 00 01 54 78 44 61
74 61 41
```

Frame type	Frame ID	64-bit dest	Reserved	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0xF8	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0012	0xC105	0x00	0x01	0x5478446174 61
Explicit request	*Matches response*	*Destination*	*Unused*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Disable retries*	*"TxData"*

Remote AT Command Request - 0x17

Response frame: Remote AT Command Response- 0x97

Description

This frame type is used to query or set AT command 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.

When querying parameter values you can query parameter values by sending this framewith a command but no parameter value field—the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Remote AT Command Response- 0x97 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x97 response is the same one set by the command in the 0x17 request frame.

Note Remote AT Command Requests should only be issued as unicast transmissions to avoid potential network disruption. Broadcasts are not acknowledged, so there is no guarantee all devices will receive the request. Responses are returned immediately by all receiving devices, which can cause congestion on a large network.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Request - **0x17**.
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device.
13	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.

Offset	Size	Frame Field	Description
15	8-bit	**Remote command options**	Bit field of options that apply to the remote AT command request: ■ **Bit 0**: Disable ACK [**0x01**] ■ **Bit 1**: Apply changes on remote [**0x02**] • If not set, changes will not applied until the device receives an **AC** command or a subsequent command change is received with this bit set ■ Bit 2: Reserved (set to 0) ■ Bit 3: Reserved (set to 0) ■ **Bit 4**: Send the remote command securely [**0x10**] **Note** Option values may be combined. Set all unused bits to 0.
16	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
18-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes—**AP** = **1**—and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set remote command parameter

Set the **NI** string of a device with the 64-bit address of **0013A20012345678** to "**Remote**" and apply the change immediately.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will indicate success.

```
7E 00 15 17 27 00 13 A2 00 12 34 56 78 FF FE 02 4E 49 52 65 6D 6F 74 65 F6
```

Frame type	Frame ID	64-bit dest	Reserved	Command options	AT command	Parameter value
0x17	0x27	0x0013A20012345678	0xFFFE	0x02	0x4E49	0x52656D6F7465
Request	*Matches response*		*Unused*	*Apply Change*	*"NI"*	*"Remote"*

Queue remote command parameter change

Change the PAN ID of a remote device so it can migrate to a new PAN, since this change would cause network disruption, the change is queued so that it can be made active later with a subsequent **AC** command or written to flash with a queued **WR** command so the change will be active after a power cycle.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will indicate success.

```
7E 00 11 17 68 00 13 A2 00 12 34 56 78 FF FE 00 49 44 04 51 D8
```

Frame type	Frame ID	64-bit dest	Reserved	Command options	AT command	Parameter value
0x17	0x68	0x0013A200 12345678	0xFFFE	0x00	0x4944	0x0451
Request	*Matches response*		*Unused*	*Queue Change*	*"ID"*	

Query remote command parameter

Query the temperature of a remote device—**TP** command.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will return the temperature value.

```
7E 00 0F 17 FA 00 13 A2 00 12 34 56 78 FF FE 00 54 50 84
```

Frame type	Frame ID	64-bit dest	Reserved	Command options	AT command	Parameter value
0x17	0xFA	0x0013A200 12345678	0xFFFE	0x00	0x5450	(omitted)
Request	*Matches response*		*Unused*	*N/A*	*"TP"*	*Query the parameter*

64-bit Receive Packet - 0x80

Request frames:

- Transmit Request - 0x10
- 64-bit Transmit Request - 0x00
- 16-bit Transmit Request - 0x01

Description

This frame type is emitted when a device configured with legacy API output—AO (API Options) = **2**—receives an RF data packet from a device configured to use 64-bit source addressing—**MY** = **0xFFFE**.

Note This frame format is deprecated and should only be used by customers who require compatibility with legacy Digi RF products. For new designs, we encourage you to use Receive Packet - 0x90 for reception of API transmissions.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame specifications.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	64-bit Receive Packet - **0x80**
4	64-bit	**64-bit source address**	The sender's 64-bit IEEE address.
12	8-bit	**RSSI**	Received Signal Strength Indicator. The Hexadecimal equivalent of (-dBm) value. For example if RX signal strength is -40 dBm, then 0x28 (40 decimal) is returned.
13	8-bit	**Options**	Bit field of options that apply to the received message: ■ Bit 0: Reserved ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: 802.15.4 only - Packet was broadcast across all PANs [**0x04**] **Note** Option values may be combined.

Offset	Size	Frame Field	Description
14-n	variable	**RF data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

A device with the 64-bit address of **0013A20087654321** sent a unicast transmission to a specific device with the payload of "**TxData**". The following frame is emitted if the destination is configured with **AO** = **2**.

```
7E 00 11 80 00 13 A2 00 12 34 56 78 5E 01 54 78 44 61 74 61 11
```

Frame type	64-bit source	RSSI	Rx options	Received data
0x80	0x0013A200 87654321	0x5E	0x01	0x547844617461
Output		*-94 dBm*	*ACK was sent*	*"TxData"*

Local AT Command Response - 0x88

Request frames:

- Local AT Command Request - 0x08
- Queue Local AT Command Request - 0x09

Description

This frame type is emitted in response to a local AT Command request. Some commands send back multiple response frames; for example, ND (Network Discover). Refer to individual AT command descriptions for details on API response behavior.

This frame is only emitted if the Frame ID in the request is non-zero.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Local AT Command Response - **0x88**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7	8-bit	**Command status**	Status code for the host's request: **0** = OK **1** = ERROR **2** = Invalid command **3** = Invalid parameter
8-n	variable	**Command data (optional)**	If the host requested a command parameter change, this field will be omitted. If the host queried a command by omitting the parameter value in the request, this field will return the value currently set on the device.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set local command parameter

Host set the NI string of the local device to "**End Device**" using a 0x08 request frame.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID is emitted as a response:

```
7E 00 05 88 01 4E 49 00 DF
```

Frame type	Frame ID	AT command	Command Status	Command data
0x88	0xA1	0x4E49	0x00	(omitted)
Response	*Matches request*	*"NI"*	*Success*	*Parameter changes return no data*

Query local command parameter

Host queries the temperature of the local device—**TP** command—using a 0x08 request frame.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID is emitted with the temperature value as a response:

```
7E 00 07 88 01 54 50 00 FF FE D5
```

Frame type	Frame ID	AT command	Command Status	Command data
0x88	0x17	0x5450	0x00	0xFFFE
Response	*Matches request*	*"TP"*	*Success*	*-2 ℃*

Transmit Status - 0x89

Request frames:

- TX Request: 64-bit address frame - 0x00
- TX Request: 16-bit address - 0x01
- User Data Relay Input - 0x2D

Description

This frame type is emitted when a transmit request completes. The status field of this frame indicates whether the request succeeded or failed and the reason.

This frame is only emitted if the Frame ID in the request is non-zero.

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 specifications.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x89**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.

Offset	Size	Frame Field	Description
5	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = No ACK received **0x02** = CCA failure **0x03** = Indirect message unrequested **0x04** = Transceiver was unable to complete the transmission **0x21** = Network ACK failure **0x22** = Not joined to network **0x2C** = Invalid frame values (check the phone number) **0x31** = Internal error **0x32** = Resource error - lack of free buffers, timers, etc. **0x34** = No Secure Session Connection **0x35** = Encryption Failure **0x74** = Message too long **0x76** = Socket closed unexpectedly **0x78** = Invalid UDP port **0x79** = Invalid TCP port **0x7A** = Invalid host address **0x7B** = Invalid data mode **0x7C** = Invalid interface. See User Data Relay Input - 0x2D. **0x7D** = Interface not accepting frames. See User Data Relay Input - 0x2D. **0x7E** = A modem update is in progress. Try again after the update is complete. **0x80** = Connection refused **0x81** = Socket connection lost **0x82** = No server **0x83** = Socket closed **0x84** = Unknown server **0x85** = Unknown error **0x86** = Invalid TLS configuration (missing file, and so forth) **0x87** = Socket not connected **0x88** = Socket not bound Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Delivery status codes

Protocol-specific status codes follow

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Successful transmission

Host sent a unicast transmission to a remote device using a TX Request: 64-bit address frame - 0x00 frame.

The corresponding 0x89 Transmit Status with a matching Frame ID is emitted as a response to the request:

```
7E 00 03 89 52 00 24
```

Frame type	Frame ID	Delivery status
0x89	0x52	0x00
Response	*Matches request*	*Success*

Modem Status - 0x8A

Description

This frame type is emitted in response to specific conditions. The status field of this frame indicates the device behavior.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Modem Status - **0x8A**

Offset	Size	Frame Field	Description
4	8-bit	**Modem status**	Complete list of modem statuses: **0x00** = Hardware reset or power up **0x01** = Watchdog timer reset **0x02** = Joined network **0x03** = Left network **0x06** = Coordinator started **0x07** = Network security key was updated **0x0B** = Network woke up **0x0C** = Network went to sleep **0x0D** = Voltage supply limit exceeded **0x0E** = Remote Manager connected **0x0F** = Remote Manager disconnected **0x11** = Modem configuration changed while join in progress **0x12** = Access fault **0x13** = Fatal error **0x3B** = Secure session successfully established **0x3C** = Secure session ended **0x3D** = Secure session authentication failed **0x3E** = Coordinator detected a PAN ID conflict but took no action **0x3F** = Coordinator changed PAN ID due to a conflict **0x32** = BLE Connect **0x33** = BLE Disconnect **0x34** = Bandmask configuration failed **0x35** = Cellular component update started **0x36** = Cellular component update failed **0x37** = Cellular component update completed **0x38** = XBee firmware update started **0x39** = XBee firmware update failed **0x3A** = XBee firmware update applying **0x40** = Router PAN ID was changed by coordinator due to a conflict **0x42** = Network Watchdog timeout expired **0x80** through **0xFF** = Stack error Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Modem status codes

Statuses for specific modem types are listed here.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Boot status

When a device powers up, it returns the following API frame:

```
7E 00 02 8A 00 75
```

Frame type	Modem Status
0x8A	0x00
Status	*Hardware Reset*

Extended Transmit Status - 0x8B

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a network transmission request completes. The status field of this frame indicates whether the request succeeded or failed and the reason. This frame type provides additional networking details about the transmission.

This frame is only emitted if the Frame ID in the request is non-zero.

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.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x8B**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
7	8-bit	**Transmit retry count**	The number of application transmission retries that occur.

Offset	Size	Frame Field	Description
8	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = MAC ACK failure **0x02** = CCA/LBT failure **0x03** = Indirect message unrequested / no spectrum available **0x21** = Network ACK failure **0x25** = Route not found **0x31** = Internal resource error **0x32** = Resource error lack of free buffers, timers, etc. **0x74** = Data payload too large **0x75** = Indirect message unrequested
9	8-bit	**Discovery status**	Complete list of delivery statuses: **0x00** = No discovery overhead **0x02** = Route discovery
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Route Information - 0x8D

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type contains the DigiMesh routing information for a remote device on the network. This route information can be used to diagnose marginal links between devices across multiple hops.

This frame type is emitted in response to a DigiMesh unicast transmission request which has Trace Routing or NACK enabled. See Trace route option and NACK messages for more information.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Route Information - **0x8D**
4	8-bit	**Source event**	Event that caused the route information to be generated: **0x11** = NACK **0x12** = Trace route
5	8-bit	**Data length**	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.
6	32-bit	**Timestamp**	System timer value on the node generating the Route Information Packet.The timestamp is in microseconds. Only use this value for relative time measurements because the time stamp count restarts approximately every hour.
10	8-bit	**ACK timeout count**	The number of MAC ACK timeouts that occur.
11	8-bit	**TX blocked count**	The number of times the transmission was blocked due to reception in progress.
12	8-bit	**Reserved**	Not used.

Offset	Size	Frame Field	Description
14	64-bit	**Destination address**	The 64-bit IEEE address of the final destination node of this network-level transmission.
21	64-bit	**Source address**	The 64-bit IEEE address of the source node of this network-level transmission.
29	64-bit	**Responder address**	The 64-bit IEEE address of the node that generates this Route Information packet after it sends (or attempts to send) the data packet to the next hop (the Receiver node).
37	64-bit	**Receiver address**	The 64-bit IEEE address of the node that the device sends (or attempts to send) the data packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Routing information

The following example represents a possible Route Information Packet. A device emits this frame when it performs a trace route enabled transmission from one device—serial number 0x0013A200 4052AAAA—to another—serial number 0x0013A200 4052DDDD—across a DigiMesh network.

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.

```
7E 00 2A 8D 12 27 6B EB CA 93 00 00 00 00 13 A2 00 40 52 DD DD 00 13 A2 00 40 52
AA AA 00 13 A2 00 40 52 BB BB 00 13 A2 00 40 52 CC CC 4E
```

Frame type	Source event	Data length	Timestamp	ACK timeout	TX Blocked	Reserved	Dest address	Source address	Responder address	Receiver address
0x8D	0x12	0x27	0x6BEBCA93	0x00	0x00	0x00	0x0013A200 4052DDDD	0x0013A200 4052AAAA	0x0013A200 4052BBBB	0x0013A200 4052CCCC
Route	*Trace Route*		*~30 minutes*	*No retries this hop*	*No error*	*N/A*	*Destination*	*Source*	*Node that sent this information*	*Next hop*

Aggregate Addressing Update - 0x8E

Description

This frame type is emitted on devices that update it addressing information in response to a network aggregator issuing an addressing update. A network aggregator is defined by a device on the network who has had the AG (Aggregator Support) command issued. A device on the network who's current **DH** and **DL** matches the address provided in the **AG** command request will update **DH** and **DL** and emit this frame.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Aggregate Addressing Update - **0x8E**
4	8-bit	**Reserved**	Reserved for future functionality. This field returns 0.
5	64-bit	**New address**	Address to which **DH** and **DL** are being set.
13	64-bit	**Old address**	Address to which **DH** and **DL** were previously set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Aggregate address update

In the following example, a device with destination address (**DH**/**DL**) of 0x0013A200 4052AAAA updates its destination address to 0x0013A200 4052BBBB.

```
7E 00 12 8E 00 00 13 A2 00 40 52 BB BB 00 13 A2 00 40 52 AA AA 19
```

Frame type	Reserved	New address	Old address
0x8E	0x00	0x0013A200 4052BBBB	0x0013A200 4052AAAA
Update	*N/A*	*What **DH**/**DL** is now set to*	*What **DH**/**DL** was set to*

Receive Packet - 0x90

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a device configured with standard API output—AO (API Options) = **0**—receives an RF data packet.

Typically this frame is emitted as a result of a device on the network sending serial data using the Transmit Request - 0x10 or Explicit Addressing Command Request - 0x11 addressed either as a broadcast or unicast transmission.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Receive Packet - **0x90**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.

Offset	Size	Frame Field	Description
14	8-bit	**Receive options**	Bit field of options that apply to the received message: ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: Reserved ■ **Bit 3**: Reserved ■ **Bit 4**: Reserved ■ **Bit 5**: Reserved ■ **Bit 6**: Reserved ■ **Bit 6, 7**: DigiMesh delivery method • b'00 = <invalid option> • b'01 = Point-multipoint [**0x40**] • b'10 = Directed Broadcast [**0x80**] • b'11 = DigiMesh [**0xC0**] **Note** Option values may be combined.
15-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

A device with the 64-bit address of **0013A20041AEB54E** sent a unicast transmission to a specific device with the payload of "**TxData**". The following frame is emitted if the destination is configured with **AO** = **0**.

```
7E 00 12 90 00 13 A2 00 41 AE B5 4E FF FE C1 54 78 44 61 74 61 C4
```

Frame type	64-bit source	Reserved	Rx options	Received data
0x90	0x0013A200 41AEB54E	0x5614	0xC1	0x547844617461
Output		*Unused*	*ACK was sent in DigiMesh mode*	*"TxData"*

Explicit Receive Indicator - 0x91

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a device configured with explicit API output—AO (API Options) bit1 set—receives a packet.

Typically this frame is emitted as a result of a device on the network sending serial data using the Transmit Request - 0x10 or Explicit Addressing Command Request - 0x11 addressed either as a broadcast or unicast transmission.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Receive Indicator - **0x91**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
14	8-bit	**Source endpoint**	Endpoint of the source that initiated transmission.
15	8-bit	**Destination endpoint**	Endpoint of the destination that the message is addressed to.
16	16-bit	**Cluster ID**	The Cluster ID that the frame is addressed to.
18	16-bit	**Profile ID**	The Profile ID that the fame is addressed to.

Offset	Size	Frame Field	Description
20	8-bit	**Receive options**	Bit field of options that apply to the received message for packets sent using Digi endpoints (0xDC-0xEE): ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: Reserved ■ **Bit 3**: Reserved ■ **Bit 4**: Reserved ■ **Bit 5**: Reserved ■ **Bit 6**: Reserved ■ **Bit 6, 7**: DigiMesh delivery method • b'00 = <invalid option> • b'01 = Point-multipoint [**0x40**] • b'10 = Directed Broadcast [**0x80**] • b'11 = DigiMesh [**0xC0**] **Note** Option values may be combined.
21-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

A device with the 64-bit address of **0013A20087654321** sent a unicast transmission to a specific device with the payload of "**TxData**". The following frame is emitted if the destination is configured with **AO** > **1**.

```
7E 00 18 91 00 13 A2 00 41 AE B5 4E FF FE E8 E8 00 11 C1 05 C1 54 78 44 61 74 61
1C
```

Frame type	64-bit source	Reserved	Source EP	Dest EP	Cluster	Profile	Rx options	Received data
0x91	0x0013A20041AEB54E	0x87BD	0xE8	0xE8	0x0011	0xC105	0xC1	0x547844617461
Explicit output		*Unused*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*ACK was sent in DigiMesh network*	*"TxData"*

I/O Sample Indicator - 0x92

Description

This frame type is emitted when a device configured with standard API output—AO (API Options) = **0**—receives an I/O sample frame from a remote device. Only devices running in API mode will send I/O samples out the serial port.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	I/O Sample Indicator - **0x92**
4	64-bit	**64-bit source address**	The sender's 64-bit IEEE address.
12	16-bit	**Reserved**	Unused, but typically **0XFFFE**.
14	8-bit	**Receive options**	Bit field of options that apply to the received message: ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] Note Option values may be combined.
15	8-bit	**Number of samples**	The number of sample sets included in the payload. This field typically reports 1 sample.

Offset	Size	Frame Field	Description
16	16-bit	**Digital sample mask**	Bit field that indicates which I/O lines on the remote are configured as digital inputs or outputs, if any: **bit 0**: DIO0 **bit 1**: DIO1 **bit 2**: DIO2 **bit 3**: DIO3 **bit 4**: DIO4 **bit 5**: DIO5 **bit 6**: DIO6 **bit 7**: DIO7 **bit 8**: DIO8 **bit 9**: DIO9 **bit 10**: DIO10 **bit 11**: DIO11 **bit 12**: DIO12 **bit 13**: DIO13 **bit 14**: DIO14 bit 15: N/A For example, a digital channel mask of **0x002F** means DIO **0**, **1**, **2**, **3**, and **5** are enabled as digital I/O.
18	8-bit	**Analog sample mask**	Bit field that indicates which I/O lines on the remote are configured as analog input, if any: **bit 0**: AD0 **bit 1**: AD1 **bit 2**: AD2 **bit 3**: AD3 **bit 7**: Supply Voltage (enabled with **V+** command)
19	16-bit	**Digital samples (if included)**	If the sample set includes any digital I/O lines (**Digital channel mask** > **0**), this field contain samples for all enabled digital I/O lines. If no digital lines are configured as inputs or outputs, this field will be omitted. DIO lines that do not have sampling enabled return 0. Bits in this field are arranged the same as they are in the Digital channel mask field.
22	16-bit variable	**Analog samples (if included)**	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 16-bit value indicating the ADC measurement of that input. Analog samples are ordered sequentially from AD0 to AD3.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

I/O sample

A device with the 64-bit address of **0013A20012345678** is configured to periodically send I/O sample data to a particular device. The device is configured with DIO3, DIO4, and DIO5 configured as digital

I/O, and AD1 and AD2 configured as an analog input.
The destination will emit the following frame:

```
7E 00 16 92 00 13 A2 00 12 34 56 78 FF FE C1 01 00 38 06 00 28 02 25 00 F8 E8
```

Frame type	64-bit source	Reserved	Rx options	Num samples	Digital channel mask	Analog channel mask	Digital samples	Analog sample 1	Analog sample 2
0x92	0x0013A200 12345678	0x87AC	0xC1	0x01	0x0038	0x06	0x0028	0x0225	0x00F8
Sample		*Unused*	*ACK was sent in mesh network*	*Single sample (typical)*	*b'00**111**000 DIO3, DIO4, and DIO5 enabled*	*b'0**11**0 AD1 and AD2 enabled*	*b'00**101**000 DIO3 and DIO5 are HIGH; DI04 is LOW*	*AD1 data*	*AD2 data*

Node Identification Indicator - 0x95

Description

This frame type is emitted when a node identification broadcast is received. The node identification indicator contains information about the identifying device, such as address, identifier string (**NI**), and other relevant data.

A node identifies itself to the network under these conditions:

- The commissioning button is pressed once.
- A **CB 1** command is issued.
- A synchronous sleep node stays awake for 30 seconds in order to receive a sync message. It also sends out an identifying message.

See ND (Network Discover) for information on the payload formatting.

See NO (Node Discovery Options) for configuration options that modify the output of this frame.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Node Identification Indicator - **0x95**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.

Offset	Size	Frame Field	Description
14	8-bit	**Options**	Bit field of options that apply to the received message: ■ Bit 0: Reserved ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: Reserved ■ Bit 4: Reserved ■ Bit 5: Reserved ■ **Bit 6, 7**: DigiMesh delivery method • b'00 = <invalid option> • b'01 = Point-multipoint [**0x40**] • b'10 = Directed Broadcast [**0x80**] • b'11 = DigiMesh [**0xC0**] **Note** Option values may be combined.
15	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
17	64-bit	**64-bit remote address**	The 64-bit address of the device that sent the Node Identification.
25	variable (2-byte minimum)	**Node identification string**	Node identification string on the remote device set by NI (Node Identifier). The identification string is terminated with a NULL byte (0x00).
27+NI	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
29+NI	8-bit	**Network device type**	What type of network device the remote identifies as: 0 = Coordinator 1 = Router 2 = End Device
30+NI	8-bit	**Source event**	The event that caused the node identification broadcast to be sent. 0 = Reserved 1 = Frame sent by node identification pushbutton event—see D0 (DIO0/AD0).
31+NI	16-bit	**Digi Profile ID**	The Digi application Profile ID—**0xC105**.
33+NI	16-bit	**Digi Manufacturer ID**	The Digi Manufacturer ID—**0x101E**.
35+NI	32-bit	**Device type identifier (optional)**	The user-defined device type on the remote device set by DD (Device Type Identifier). Only included if the receiving device has the appropriate NO (Node Discovery Options) bit set.

Offset	Size	Frame Field	Description
EOF-1	8-bit	**RSSI (optional)**	The RSSI of the last hop that relayed the message. Only included if the receiving device has the appropriate NO (Node Discovery Options) bit set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte—between length and checksum.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Identify remote device

A technician is replacing a DigiMesh device in the field and needs to have the its entry removed from a cloud server's database. The technician pushes the commissioning button on the old device once to send an identification broadcast. The server can use the broadcast to identify which device is being replaced and perform the necessary action.

When the node identification broadcast is sent, every device that receives the message will flash the association LED and emit the following information frame:

```
7E 00 27 95 00 13 A2 00 12 34 56 78 FF FE C2 FF FE 00 13 A2 00 12 34 56 78 4C 48
37 35 00 FF FE 01 01 C1 05 10 1E 00 14 00 08 0D
```

Frame type	64-bit source	Reserved	Options	64-bit remote	NI String	Reserved	Device type	Event	Profile ID	MFG ID
0x95	0x0013A200 12345678	0xFFFE	0xC2	0x0013A200 12345678	0x4C48373 5 00	0xFFFE	0x01	0x01	0xC105	0x101E
Identification		*Unused*	*DigiMesh broadcast*		*"LH75" + null*	*Unused*	*Router*	*Button press*	*Digi*	*Digi*

Remote AT Command Response- 0x97

Request frame: Remote AT Command Request - 0x17

Description

This frame type is emitted in response to a Remote AT Command Request - 0x17. Some commands send back multiple response frames; for example, the **ND** command. Refer to individual AT command descriptions for details on API response behavior.

This frame is only emitted if the Frame ID in the request is non-zero.

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Response - **0x97**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	64-bit	**64-bit source address**	The sender's 64-bit address.
13	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
15	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
17	8-bit	**Command status**	Status code for the host's request: **0x00** = OK **0x01** = ERROR **0x02** = Invalid command **0x03** = Invalid parameter **0x04** = Transmission failure **0x0C** = Encryption error
18-n	variable	**Parameter value (optional)**	If the host requested a command parameter change, this field will be omitted. If the host queried a command by omitting the parameter value in the request, this field will return the value currently set on the device.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set remote command parameter

Host set the **NI** string of a remote device to "**Remote**" using a Remote AT Command Request - 0x17.

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted as a response:

```
7E 00 0F 97 27 00 13 A2 00 12 34 56 78 12 7E 4E 49 00 51
```

Frame type	Frame ID	64-bit source	Reserved	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x127E	0x4E49	0x00	(omitted)
Response	*Matches request*		*Unused*	*"NI"*	*Success*	*Parameter changes return no data*

Transmission failure

Host queued the the PAN ID change of a remote device using a Remote AT Command Request - 0x17. Due to existing network congestion, the host will retry any failed attempts.

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted as a response:

```
7E 00 0F 97 27 00 13 A2 00 12 34 56 78 FF FE 49 44 04 EA
```

Frame type	Frame ID	64-bit source	Reserved	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0xFFFE	0x4944	0x04	(omitted)
Response	*Matches request*		*Unused*	*"ID"*	*Transmission failure*	*Parameter changes return no data*

Query remote command parameter

Query the temperature of a remote device—TP (Board Temperature).

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted with the temperature value as a response:

```
7E 00 11 97 27 00 13 A2 00 12 34 56 78 FF FE 54 50 00 00 2F A8
```

Frame type	Frame ID	64-bit source	Reserved	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x0013A200 12345678	0x4944	0x00	0x002F
Response	*Matches request*		*Unused*	*"TP"*	*Success*	*+47 °C*

Advanced application features

Remote configuration commands

A device in API mode has provisions to send configuration commands to remote devices using the Remote Command Request API frame. For more information, see Operate in API mode. You can use this API frame to send commands to a remote module to read or set command parameters.

Send a remote command

To send a remote command populate the Remote AT Command Request frame (0x17) with:

1. The 64-bit address and of the remote device.
2. The correct command options value.
3. The command and parameter data (optional).

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 - 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 6). 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.

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.

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.

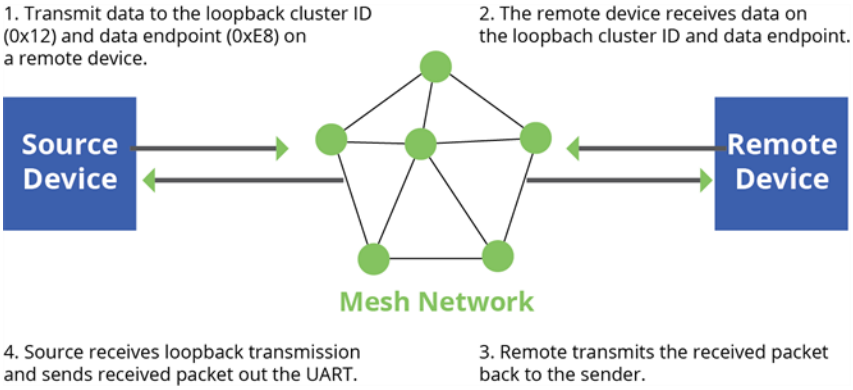
Link reliability

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.
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.

If you enable the Trace Route API option, the device sends the unicast to its destination devices, which forward the unicast to its eventual destination. The destination devices transmit a Route Information (RI) packet back along the route to the unicast originator.

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.

The destination devices contain addressing information for the unicast and intermediate hop that generates the trace route packet, and other link quality information.

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.

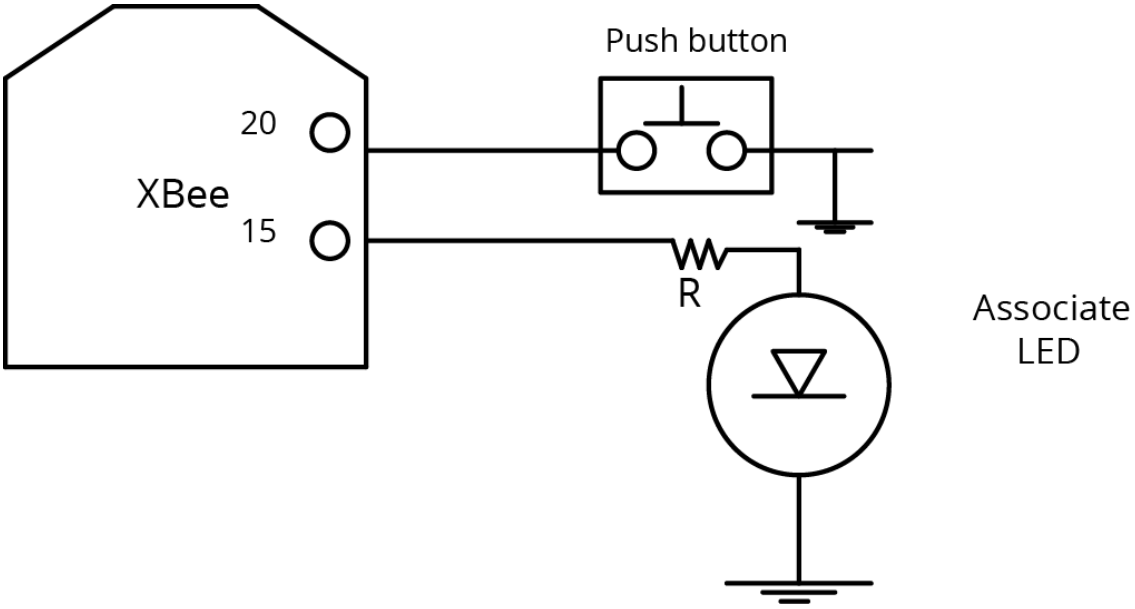
This information is useful because it allows you to identify and repair marginal links.

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



Connect a pushbutton and an LED to XBee-PRO 900HP RF Module pins 20 and 15 respectively to support the commissioning pushbutton and associate LED functionalities.

SMT RF Module

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 by setting D0 (DIO0/AD0) 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 Indicator - 0x95 out their serial interface.
1	Configured for asynchronous 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 Indicator - 0x95 out their serial interface.

Button presses	Sleep configuration and sync status	Action
1	Configured for synchronous sleep	Wakes the device 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 send a Node Identification Indicator - 0x95 out their serial interface.
2	Not configured for synchronous sleep	No effect.
2	Configured for synchronous sleep	Causes a node configured with sleeping router nomination enabled (see SO (Sleep Options)) to immediately nominate itself as the network sleep coordinator.
4	Any	Issues an RE (Restore Defaults)to restore device parameters to default values.

Use CB (Commissioning Pushbutton) to simulate button presses in software. Issue a **CB** command with a parameter set to the number of button presses you want executed. 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 a Node Identification Indicator - 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.

The Associate pin indicates the synchronization status of a sleep compatible XBee-PRO 900HP RF Module. If a device is not sleep compatible, the pin functions as a power indicator.

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

Sleep mode	LED status	Meaning
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

XBee devices support both analog input and digital I/O line modes on several configurable pins.

Queried sampling

Devices support both analog input and digital I/O line modes on several configurable pins.

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

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
CD / DIO12	4	**P2**
PWM0 / RSSI / DIO10	6	**P0**
PWM1 / DIO11	7	**P1**
$\overline{\text{DTR}}$ / SLEEP_RQ / DIO8	9	**D8**
AD4 / DIO4	11	**D4**
$\overline{\text{CTS}}$/ DIO7	12	**D7**
ON/$\overline{\text{SLEEP}}$/ DIO9	13	**D9**
ASSOC / AD5 / DIO5	15	**D5**
$\overline{\text{RTS}}$ / DIO6	16	**D6**
AD3 / DIO3	17	**D3**
AD2 / DIO2	18	**D2**
AD1 / DIO1	19	**D1**
AD0 / DIO0 / Commissioning Pushbutton	20	**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.

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.

Field	Name	Description
1	Sample sets	Number of sample sets in the packet. Always set to 1.

Field	Name	Description
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.

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

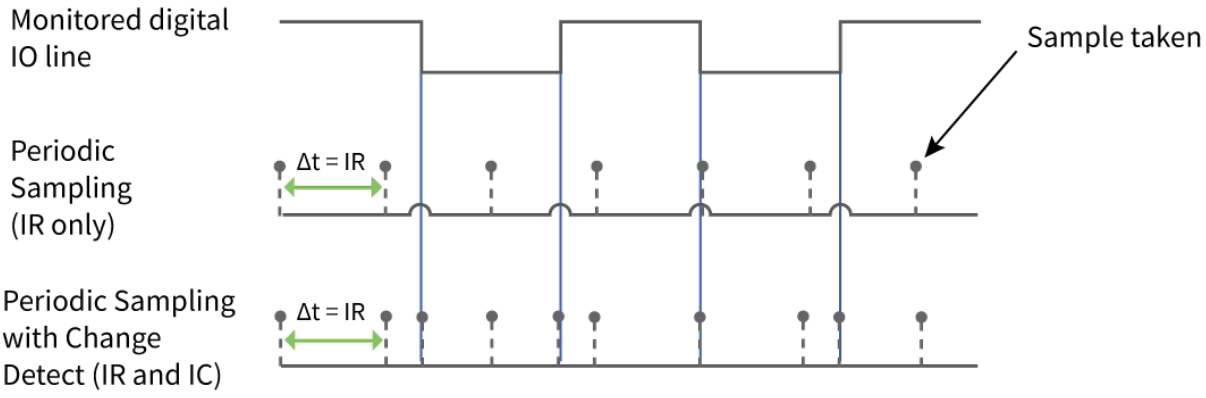
Only devices with API operating mode enabled send I/O data samples out their serial interface. Devices that are in Transparent mode (**AP** = **0**) discard the I/O data samples they receive. You must configure at least one pin as a digital or ADC input to generate sample data.

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. For more information about setting sleep modes, see Sleep modes.

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.



General Purpose Flash Memory

General Purpose Flash Memory

XBee-PRO 900HP RF Modules provide 119 512-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).

Field name	Command-specific description
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 0200 37

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.

XSC firmware

This section provides an overview of the XBee-PRO XSC firmware and the technical specifications.

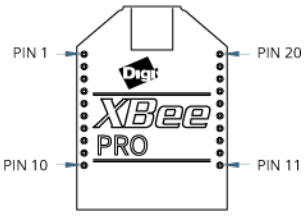
XBee-PRO XSC RF Module overview

The XBee-PRO XSC RF Modules are engineered to afford integrators with an easy-to-use RF solution that provides reliable delivery of critical data between remote devices. These modules come configured to sustain reliable long-range wireless links. The XBee-PRO XSC RF Module is a drop-in wireless solution that transfers a standard asynchronous serial data stream.

The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant, which features better performance, lower current draw, and is backward compatible with and a direct replacement for S3 radios. The S3B hardware with XSC firmware is also fully backward compatible (serial interface and over-the-air) with the 9XStream radios.

Pin signals

XBee-PRO (top view)



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

Pin	Public signal	I/O	When active	Function
1	VCC	I		Supply voltage
2	DO (Data Out)	O	N/A	Serial data exiting the module (to the UART host); See UART-interfaced data flow.
3	DI (Data In	I	N/A	Serial data entering the module (from UART host). See UART-interfaced data flow.
4	DO3 / RX LED	O	High	Pin is driven high during RF data reception; otherwise, the pin is driven low. See CD (DO3 Configuration) to enable.
5	$\overline{\text{RESET}}$[1]	I/O	Low	Re-boot module (minimum pulse is 90 µs). Open Drain configuration. Module drives reset line low momentarily on reboot and power up.
6	$\overline{\text{CONFIG}}$[2]	I	Low / high	Pin can be used as a backup method for entering Command Mode during power-up. Primary method is with **+++**. See AT commands for more information.
7		O	Driven high	Do not connect

[1]Has a pull up resistor. S3B has internal pull-up.

[2]Has a pull up resistor. S3B has internal pull-up.

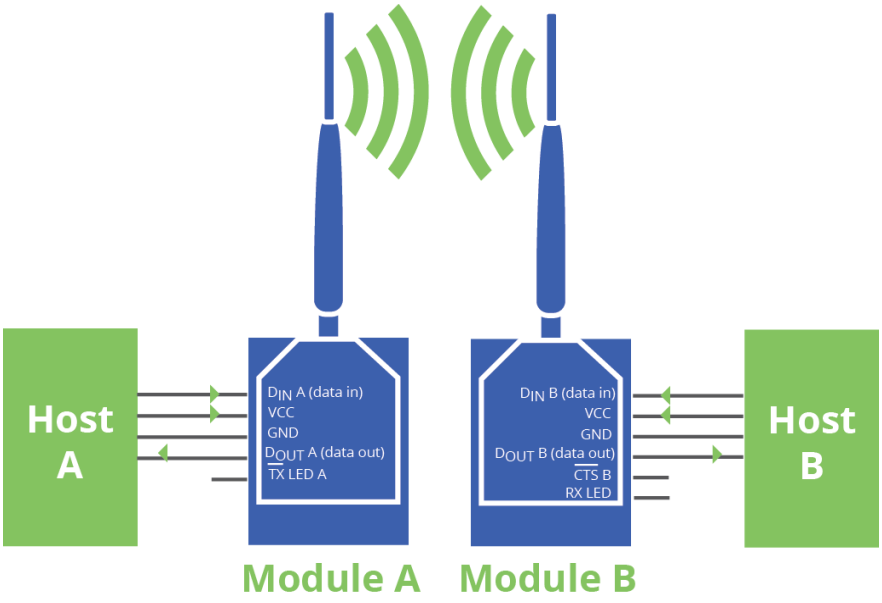
Pin	Public signal	I/O	When active	Function
8		NC		Do not connect
9	DI3 / SLEEP[1]	I	High	By default, DI3 pin is not used. To configure this pin to support sleep modes, see Sleep mode, SM (Sleep Mode) and PW (Pin Wakeup).
10	GND			Ground
11		O	Driven high	Do not connect
12	DO2 / CTS/ RS-485 Enable	O	Low	CTS (clear-to-send) flow control. When pin is driven low, UART host is permitted to send serial data to the module. Refer toUART-interfaced data flow and for more information. RS-485 Enable. To configure this pin to enable RS-485 (2-wire or 4-wire) communications, refer to UART-interfaced data flow and .
13	ON / SLEEP	O	High	High = Indicates power is on and module is not in Sleep mode. Low = Sleep mode or module is unpowered.
14	VREF	I	N/A	Not used on this module. For compatibility with other XBee devices, we recommend connecting this pin to a voltage reference if analog sampling is desired. Otherwise, connect to GND.
15	TX / PWR	O	N/A	Low = TX - Pin pulses low during transmission High = PWR - Indicates power is on and module is not in Sleep mode.
16	DI2 / RTS / CMD[2]	I	Low	RTS (request-to-send) flow control - By default, this pin is not used. To configure this pin to regulate the flow of serial data exiting the module, refer to UART-interfaced data flow and RT (DI2 Configuration). CMD. Refer to Configuration and commands and RT (DI2 Configuration) to enable binary command programming.
17		O	Driven low	Do not connect
18		O	Driven low	Do not connect
19		O	Driven low	Do not connect
20		O	Driven low	Do not connect

Electrical characteristics

When the transmitting device receives the first byte of data in the DIN buffer, it initiates the data flow sequence. As long as the TX device is not already receiving RF data, it converts the data in the DIN buffer into packets and transmits them over-the-air to the receiving device.

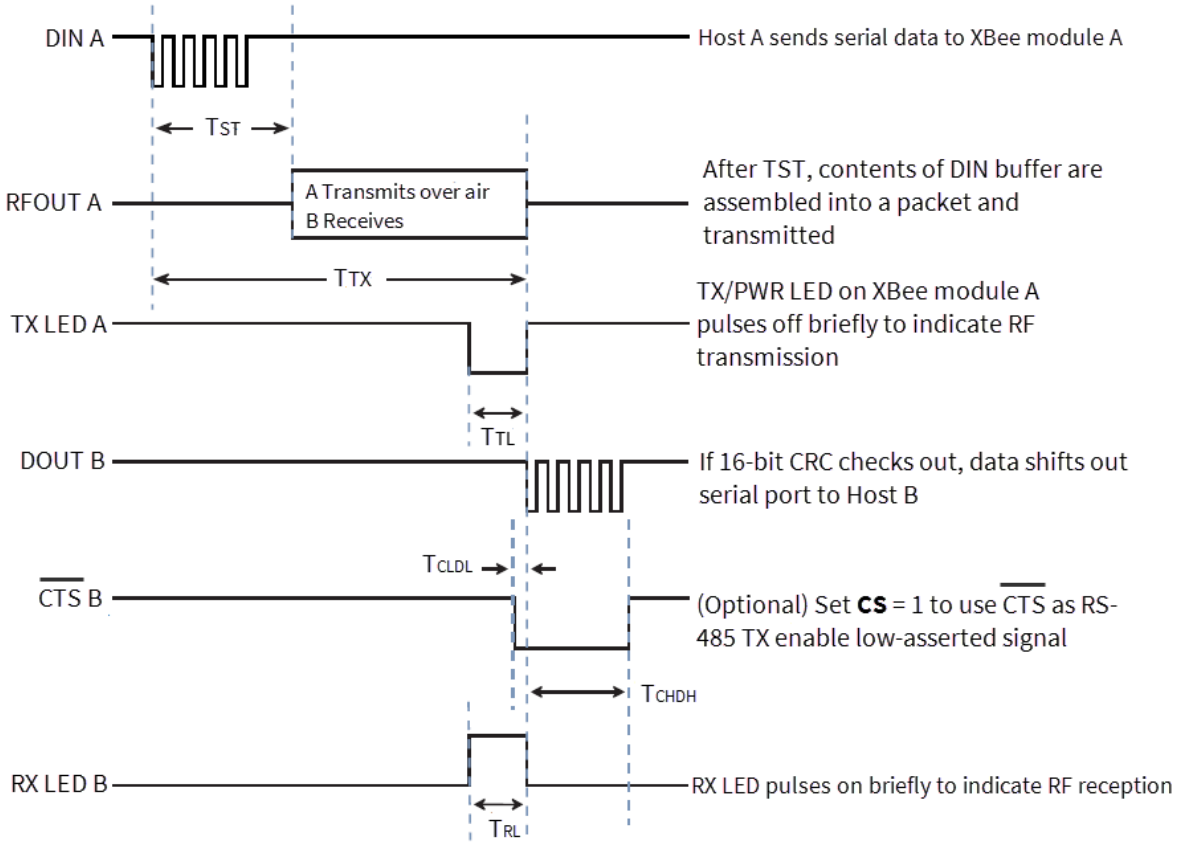
[1]Has a pull up resistor. S3B has internal pull-up.

[2]Has a pull down resistor. S3B has internal pull-up.



Timing specifications

The following figure illustrates the timing specifications. Host A and Host B correspond to a transmitting device (Host A) and receiving device (Host B).



The following table provides typical AC characteristics when **SY** = 0. The symbols correspond with the previous two figures.

Symbol	Description	9600 baud rate (32 byte packet)
T_{TX}	Latency from the time data is transmitted until received	72.0 ms
T_{TL}	Time that TX/PWR pin is driven low	16.8 ms
T_{RL}	Time that RX LED pin is driven high	25.6 ms
T_{ST}	Channel Initialization Time	35.0 ms

The following table provides typical DC characteristics when V_{CC} = 3.0 - 3.6 VDC.

Symbol	Parameter	Condition	Min	Typical	Max	Units
V_{cc}	Module supply voltage		3.0[1]		3.6	V

[1]The minimum voltage for S3B is 2.1 V, however maximum power is reduced and sensitivity may degrade.

Symbol	Parameter	Condition	Min	Typical	Max	Units
V_{IL}	Input low voltage	All input signals	-0.3		0.3 VCC	V
V_{IH}	Input high voltage	All input signals	0.7 VCC		VCC + 0.3	V
V_{OL}	Output low-level voltage	Iout = Iout_Max			0.4	V
V_{OH}	Output high-level voltage	Iout = Iout_Max	-0.4 VCC			V
IL	Input leakage current	With pull-up resistors disabled[1]		40	400	nA
IO1	Output current	Pins 2, 15 (DOUT, $\overline{TX}$/PWR)			2	mA
IO2	Output current	Pins 4, 12, 13 (DO3, $\overline{CTS}$, ON/SLEEP)			8	mA

[1] S3B can have pull-ups enabled and still maintain low leakage current.

XBee-PRO XSC specifications

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	XBee-PRO XSC (S3* hardware)	XBee-PRO XSC (S3B hardware)
Indoor/urban range	Up to 370 m (1200 ft)	Up to 610 m (2000 ft)
Outdoor line-of-sight range	Up to 9.6 km (6 mi) with dipole antenna Up to 24 km (15 mi) with high-gain antenna	Up to 14 km (9 mi) with dipole antenna Up to 45 km (28 mi) with high-gain antenna
Interface data rate	125 - 65,000 b/s (software selectable, includes non-standard baud rates)	
Throughput data rate	9,600 b/s	9.6 kb/s or 19.2 kb/s
RF data rate	10 kb/s	10 kb/s or 20 kb/s
Transmit power output	+20 dBm (100 mW)	Up to 24 dBm (250 mW) software selectable
Receiver sensitivity	-106 dBm	-109 dBm at 9600 baud -107 dBm at 19200 baud
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.		

Power requirements

This table describes the power requirements for the devices.

Specification	XBee-PRO XSC (S3* hardware)	XBee-PRO XSC (S3B hardware)
Supply voltage	3.0-3.6 VDC regulated	2.1 to 3.6 VDC
Receive current	65 mA	26 mA typical
Transmit current	265 mA	215 mA at 24 dBm
Power down current	50 uA	2.5 µA typical @3.3 v
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.		

Networking specifications

The following table provides the networking specifications for the device.

Specification	XBee-PRO XSC (S3* hardware)	XBee-PRO XSC (S3B hardware)
Frequency range	902-928 MHz (located in the 900 MHz ISM band)	
Spread spectrum	Frequency hopping	
Network topology	Point-to-point, peer-to-peer, point-to-multipoint	
Channel capacity	7 hop sequences share 25 frequencies	
Board-level serial data interface (S3B)	3V CMOS UART (5 V-tolerant)	3 V CMOS UART
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.		

General specifications

The following table describes the general specifications for the devices.

Specification	XBee-PRO XSC (S3* and S3B hardware)
Module board size	3.29 cm x 2.44 cm x 0.546 cm (1.297 x 0.962 x 0.215 in) Dimensions do not include connector/antenna or pin lengths
Weight	5 to 8 grams, depending on the antenna option
Connector	Two rows of 10 pins, 22 mm apart with 2 mm spaced male Berg-type headers
Operating temperature	-40 to 85 °C (industrial)
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.	

Antenna options

The following table describes the antenna options for the devices.

Specification	XBee-PRO XSC (S3* and S3B hardware)
Integrated wire	¼ wave monopole, 8.26 cm (3.25 in) length, 1.9 dBi gain
RF connector	Reverse-polarity SMA or U.FL
Impedance	50 Ω unbalanced
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.	

Regulatory conformity summary

This table describes the agency approvals for the XBee-PRO XSC.

Specification	XBee-PRO XSC (S3* and S3B hardware)
FCC Part 15.247	FCC ID: MCQ-XB900HP
Innovation, Science and Economic Development Canada (ISED)	IC: 1846A-XB900HP
Australia	RCM
Brazil	ANATEL 3727-12-1209
RoHS	Compliant
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.	

XBee-PRO XSC RF Module operation

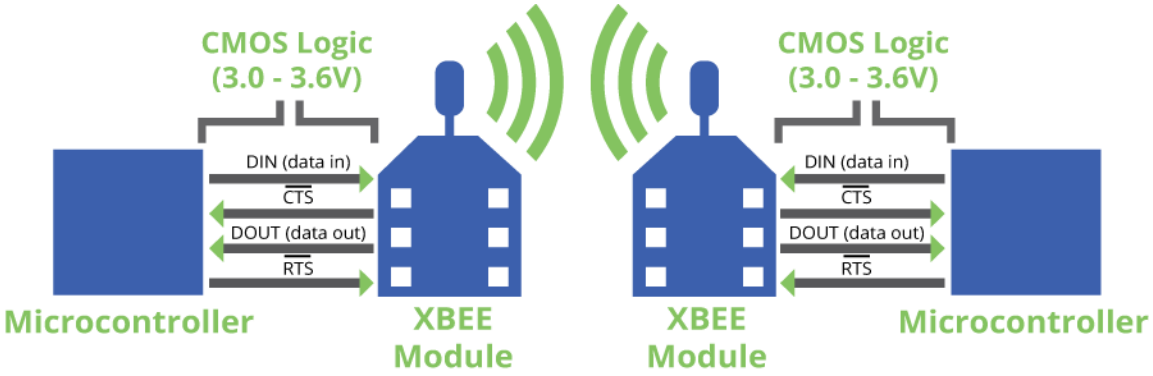
Serial communications

The XBee module interfaces to a host device through a CMOS-level asynchronous serial port. Through its serial port, the module can communicate with any UART voltage compatible device or through a level translator to any RS-232/485/422 device.

UART-interfaced data flow

Devices that have a UART interface connect directly through the pins of the XBee module as shown in the following figure.

The following diagram shows the system data flow in a UART-interfaced environment (Low-asserted signals distinguished with horizontal line over signal name).

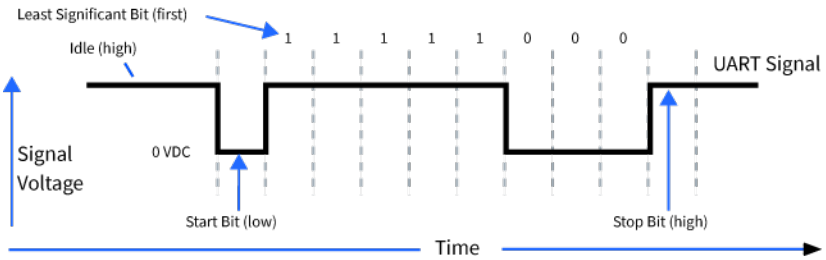


Serial data

Data enters the XBee device through the DI pin as an asynchronous serial signal. When no data is being transmitted, the signal should idle high.

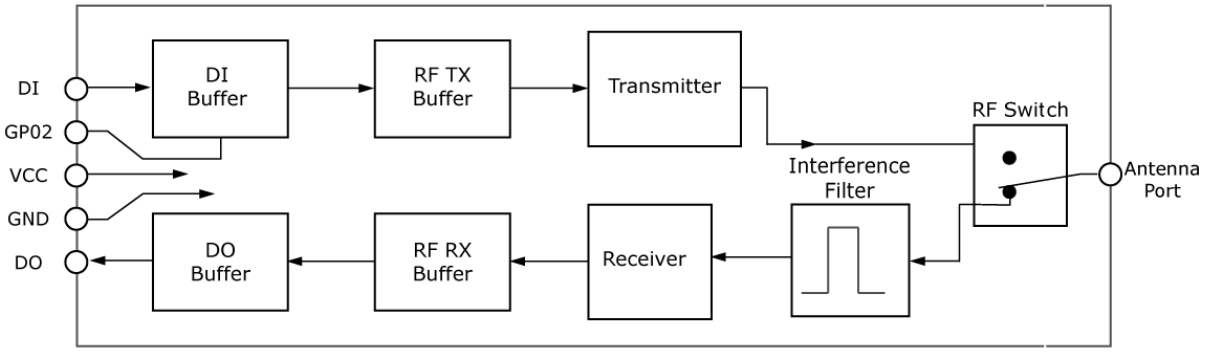
The UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communication consists of two UARTs, one being the XBee device's and the other being the microcontroller's, configured with compatible parameters (baud rate, parity, start bits, stop bits, data bits) to have successful communication. Each data packet consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high).

The following figure illustrates the serial bit pattern of data passing through the module. The UART data packet 0x1F (decimal number “31”) transmits through the XBee-PRO 900HP RF Module example data format is 8-N-1 (bits - parity - # of stop bits).



Flow control

The following internal data flow diagram shows the five most commonly-used pin signals.



Data In (DIN) buffer and flow control

When serial data enters the device through the DIN pin (pin), it stores the data in the DIN buffer until it can process the data.

When the firmware satisfies the **RB** and **RO** parameter thresholds, the device attempts to initialize an RF transmission. If the device is already receiving RF data, it stores the serial data in the device's DIN buffer.

1. The device does not receive any serial characters for the amount of time set with in the **RO** command; see RO (Packetization Timeout).
2. The device receives the maximum number of characters that fits in an RF packet.
3. The device receives the Command Mode sequence.

If the DIN buffer becomes full, you must implement hardware or software flow control in order to prevent overflow (loss of data between the host and the device). To eliminate the need for flow control:

1. Send messages that are smaller than the DIN buffer size. The size of the DIN buffer varies according to the packet size (**PK** parameter) and the parity setting (**NB** parameter) you use.
2. Interface at a lower baud rate (**BD** parameter) than the RF data rate of the firmware (**BR** parameter) of the firmware.

In the following situations, the DIN buffer may become full and overflow:

1. If you set the serial interface data rate higher than the RF data rate of the device, the device receives data from the host faster than it can transmit the data over-the-air.
2. If the device receives a continuous stream of RF data or if the device monitors data on a network, it places any serial data that arrives on the DIN pin (pin) in the DIN buffer. It transmits the data in the DIN buffer over-the-air when the device no longer detects RF data in the network.

Hardware flow control ($\overline{CTS}$)

The firmware asserts $\overline{CTS}$ before the DIN buffer is full so it has time to send the signal and the host has time to stop sending data.

When the DIN buffer is full, the firmware de-asserts $\overline{CTS}$ (high) to signal the host to stop sending data; refer to FT (Flow Control Threshold) and CS (DO2 Configuration).

The firmware re-asserts $\overline{CTS}$ after the DIN buffer has 34 bytes of memory available.

Data Out (DO) buffer and flow control

When a device receives RF data, the data enters the DOUT buffer and the device sends it out the serial port to a host device. Once the DOUT buffer reaches capacity, it loses any additional incoming RF data.

In the following situations, the DOUT buffer may become full and overflow:

1. If the RF data rate is set higher than the interface data rate of the device, the devices receives data from the transmitting device faster than it can send the data to the host.
2. If the host does not allow the device to transmit data out from the DOUT buffer because of being held off by hardware or software flow control.

Hardware flow control ($\overline{RTS}$)

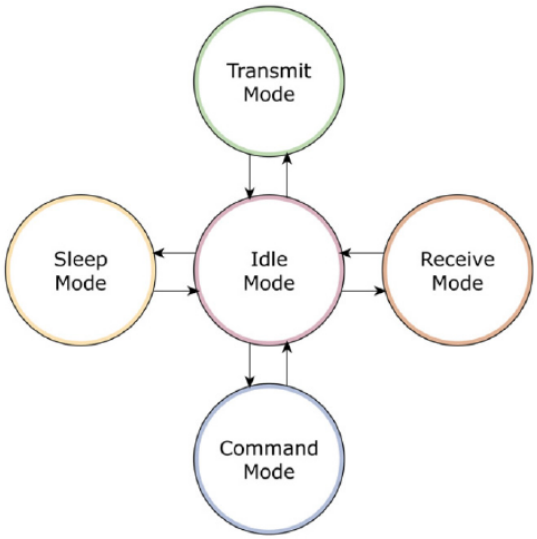
If you enable $\overline{RTS}$ for flow control (**RT** = **2**), data will not be sent out the DO Buffer as long as $\overline{RTS}$ (pin 16) is de-asserted.

Software flow control (XOFF)

You can enable XON/XOFF software flow control using FL (Software Flow Control). This option only works with ASCII data.

Operating modes

This section describes the different operating modes for the device.



Idle mode

When not receiving or transmitting data, the device is in Idle mode. During Idle mode, the device listens for valid data on both the RF and serial ports.

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).
- Receive mode (valid RF data received through the antenna).

- Command mode (Command mode sequence issued, not available with Smart Energy software or when using the SPI port).

Transmit mode

When the device received the first byte of serial data from the UART in the DI buffer, the modem attempts to shift to Transmit Mode and initiate an RF connection with other modems. After completing the transmission, the modem returns to Idle Mode.

RF transmission begins after meeting either of the following criteria:

1. **RB** bytes have been received in the DI buffer and are pending for RF transmission. For more information, see RB (Packetization Threshold).

   The **RB** parameter may be set to any value between 1 and the RF packet size (PK), inclusive. When RB = 0, the packetization threshold is ignored.
2. At least one character has been received in the DI buffer (pending for RF transmission) and **RO** time has been observed on the UART. Refer to RO (Packetization Timeout).

   - The time out can be disabled by setting RO to zero. In this case, transmission will begin after RB bytes have been received in the DI buffer.

Note RF reception must complete before the modem is able to enter into Transmit Mode.

After meeting either **RB** or **RO** conditions, the modem initializes a communications channel. Channel initialization is the process of sending an RF initializer that synchronizes receiving modems with the transmitting modem. During channel initialization, incoming serial data accumulates in the DI buffer.

The device then:

- Groups serial data in the DI buffer (for more information, see PK (Maximum RF Packet Size)
- Converts the data to RF data
- Transmits the data over-the-air until the DI buffer is empty

RF data, which includes the payload data, follows the RF initializer. The payload includes up to the maximum packet size (**PK** Command) bytes. As the transmitting modem nears the end of the transmission, it inspects the DI buffer to see if more data exists to be transmitted. This could be the case if more than **PK** bytes were originally pending in the DI buffer or if more bytes arrived from the UART after the transmission began. If more data is pending, the transmitting modem assembles a subsequent packet for transmission.

Receive mode

If a destination node receives a valid RF packet, the destination node transfers the data to its serial transmit buffer. For the serial interface to report receive data on the RF network, that data must meet the following criteria:

- ID match
- Channel match
- Address match

Sleep mode

Sleep Modes enable the device to operate at minimal power consumption when not in use. The following Sleep mode options are available:

- Pin sleep
- Cyclic sleep

For the device to transition to Sleep Mode, the module must have a non-zero **SM** (Sleep Mode) Parameter and one of the following must occur:

- The device is idle (no data transmission or reception) for a user-defined period of time. Refer to the ST (Wake Time)
- The device asserts SLEEP (only for Pin Sleep option).

In Sleep Mode, the device will not transmit or receive data until the module first transitions to Idle Mode. All Sleep Modes are enabled and disabled using the **SM** command. The device triggers transitions into and out of sleep modes are triggered by various events as shown in the table below.

Sleep mode setting	Transition into Sleep mode	Transition out of Sleep mode	Related commands	Typical power consumption (S3)*	Typical power consumption (S3B)
Pin Sleep (**SM** = 1)	microcontroller can shut down and wake devices by asserting (high) SLEEP (pin 9). The module completes a transmission or reception before activating Pin Sleep.	De-assert (low) SLEEP (pin 9).	**SM**	50 µA	2.5 µA
Cyclic Sleep (**SM** = 3-8)	Automatic transition to Sleep Mode occurs in cycles as defined by the **SM** (Sleep Mode) Command. The cyclic sleep time interval must be shorter than the "Wake-up Initializer Timer" (set by **LH** Command).	After the cyclic sleep time interval elapses. Device can be forced into Idle Mode if **PW** (Pin Wake-up) command is enabled.	**SM**, **ST**, **HT**, **LH**, **PW**	76 µA when sleeping	2.5 µA when sleeping
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.					

Pin Sleep (SM = 1)

To achieve this state, assert the SLEEP pin (high). The device remains in Pin Sleep until you de-assert the SLEEP pin.

After enabling Pin Sleep, the SLEEP pin controls whether the device is active or sleeping. When the host de-asserts SLEEP, the device is fully operational. When the host asserts SLEEP, the device transitions to Sleep mode and remains in its lowest power-consuming state until the host de-asserts the pin. This pin is only active if the device is setup to operate in this mode; otherwise the firmware ignores the pin.

Once in Pin Sleep, the device de-asserts (high) CTS (pin) , indicating that other devices should not send data to the device. The device also de-asserts (low) the TX_PWR line (pin) when the device is in Pin Sleep mode.

You cannot assert the SLEEP (pin9) until the transmission of the second byte has started.

Note The device completes a transmission or reception before activating Pin Sleep.

Cyclic Sleep Mode (SM = 4 - 8)

Cyclic Sleep modes allow device wakes according to the times designated by the cyclic sleep settings. If the device detects a wake-up initializer during the time it is awake, the device synchronizes with the transmitting device and receives data after the wake-up initializer runs its duration. Otherwise, the device returns to Sleep mode and continues to cycle in and out of activity until a wake-up initializer is detected.

While the device is in Cyclic Sleep mode, it de-asserts (high) CTS (pin) to indicate not to send data to the device. When the device awakens to listen for data, it asserts CTS and transmits any data received on the DI pin. The device also de-asserts (low) the TX_PWR (pin) when it is in Cyclic Sleep mode.

The device remains in Sleep mode for a user-defined period of time ranging from 1 second to 16 seconds (**SM** parameters 4 through 8). After this interval of time, the device returns to Idle mode and listens for a valid data packet. The listen time depends on the **BR** parameter setting. The default **BR** setting of 1 requires at least a 35 ms wake time, while the **BR** setting of 0 requires a wake time of up to 225 ms. If the device does not detect valid data on any frequency, it returns to Sleep mode. If it detects valid data, it transitions into Receive mode and receives the incoming RF packets. The device then returns to Sleep mode after a period of inactivity determined by the **ST** parameter.

You can also configure the device to wake from cyclic sleep when the SLEEP pin is de-asserted. To configure a device to operate in this manner, you must send the **PW** (Pin Wake-up) command. When you de-assert the SLEEP pin, it forces the device into Idle mode and it can begin transmitting or receiving data. It remains active until it no longer detects data for the time that **ST** specifies, at which point it resumes its low-power cyclic state.

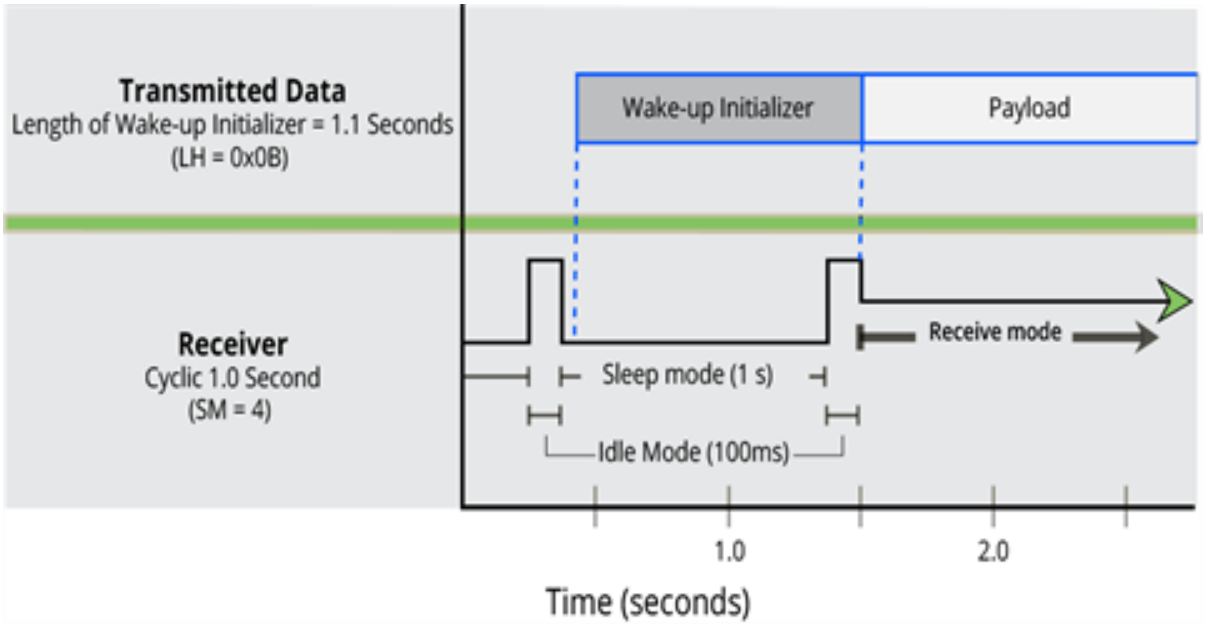
Cyclic scanning

Each RF transmission consists of an RF initializer and payload. The RF initializer contains initialization information and all receiving devices must wake during the wake-up initializer portion of data transmission in order to synchronize with the transmitting device and receive the data.

The cyclic interval time defined by the **SM** (Sleep Mode) command must be shorter than the interval time defined by **LH** (Wake-up Initializer Timer) command.

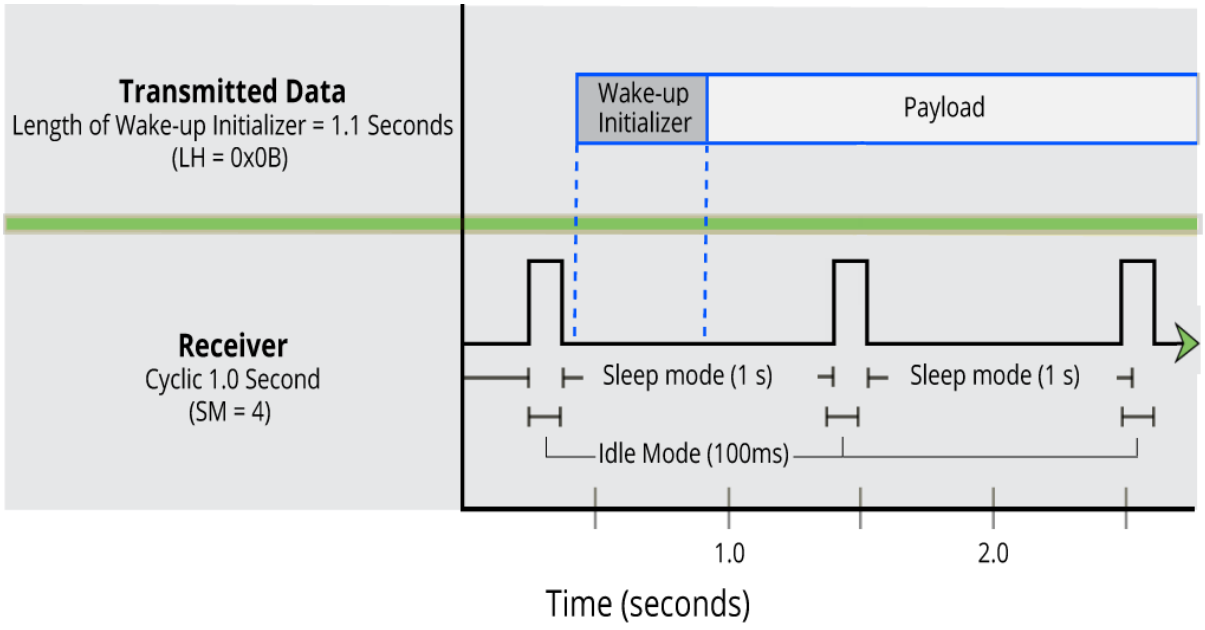
Correct configuration (LH > SM)

In the following figure, the length of the wake-up initializer exceeds the time interval of Cyclic Sleep. The receiver is guaranteed to detect the wake-up initializer and receive the accompanying payload data.



Incorrect configuration (LH < SM)

Length of wake-up initializer is shorter than the time interval of Cyclic Sleep. This configuration is vulnerable to the receiver waking and missing the wake-up initializer (and therefore also the accompanying payload data).



Command mode

To modify or read device parameters, the device must be in Command Mode, the state in which it interprets received characters on the UART as commands. Two command types are available for programming the device:

- AT commands
- Binary commands

For modified parameter values to persist in the device registry, you must save changes to non-volatile memory using **WR** (Write) Command. Otherwise, parameters are restored to previously saved values after you power the device off and then on again.

AT commands

To enter AT Command mode:

- Send the 3-character command sequence **+++** and observe guard times before and after the command characters. You can use the Terminal tab (or other serial communications software) of the XCTU Software to enter the sequence.

  Or

- Assert (low) the $\overline{\text{CONFIG}}$ pin and either turn the power going to the device off and back on. If using a Digi XBIB-R Interface Board, you can also hold the Data-In line low (also known as a break) while rebooting the device by pressing the reset button on the device assembly [device assembly = device mounted to an interface board.

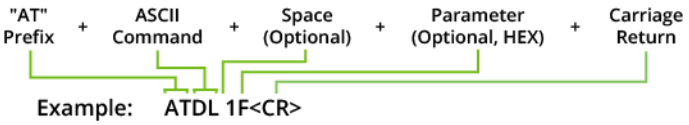
Default AT Command mode sequence (for transition to Command Mode):

- No characters sent for one second. See the BT (Guard Time Before).
- Input three plus characters ("+++") within one second. See CC (Command Sequence Character).
- No characters sent for one second. See AT (Guard Time After).

Send AT commands:

Send AT commands and parameters using the following syntax.

Syntax for sending AT commands



To read a parameter value stored in the XBee-PRO XSC RF Module register, leave the parameter field blank.

The preceding example changes the XBee-PRO XSC RF Module's Destination Address to **0x1F**. To store the new value to non-volatile (long term) memory, send the Write (**WR**) command before powering off the XBee-PRO XSC RF Module.

System response

When the device sends a command to the XBee-PRO XSC RF Module, the device parses and executes the command. If the execution of the command is successful, the device returns an **OK** message. If the execution of the command results in an error, the returns an **ERROR** message.

Exit AT Command mode:

If no valid AT Commands are received within the time specified by **CT** (Command Mode Time-out) Command, the device automatically does one of the following:

- Returns to Idle Mode
- Send **CN** (Exit Command Mode) Command

For an example of programming the XBee-PRO XSC RF Module using AT Commands and descriptions of each configurable parameter, refer to Configuration and commands.

Binary commands

Sending and receiving parameter values using binary commands is the fastest way to change operating parameters of the module. Binary commands are used most often to sample signal strength (RS parameter) or error counts; or to change module addresses and channels for polling systems when a quick response is necessary. Because sending and receiving parameter values takes place through the same data path as 'live' data (received RF payload), follow the CTS pin to distinguish between the two types of data (commands vs 'live' data).

Common questions regarding the use of binary commands:

- What are the implications of asserting CMD while live data is being sent or received?
- After sending serial data, is there a minimum time delay before CMD can be asserted?
- Is a time delay required after CMD is de-asserted before payload data can be sent?
- How can I determine the difference between live data and data received in response to a command?

You must assert CMD (pin 16) in order to send binary commands to the device. You can assert the CMD pin to recognize binary commands anytime during the transmission or reception of data. The device only check the status of the CMD signal at the end of the stop bit as the byte is shifted into the serial port. The application does not allow control over when data is received, except by waiting for dead time between bursts of communication.

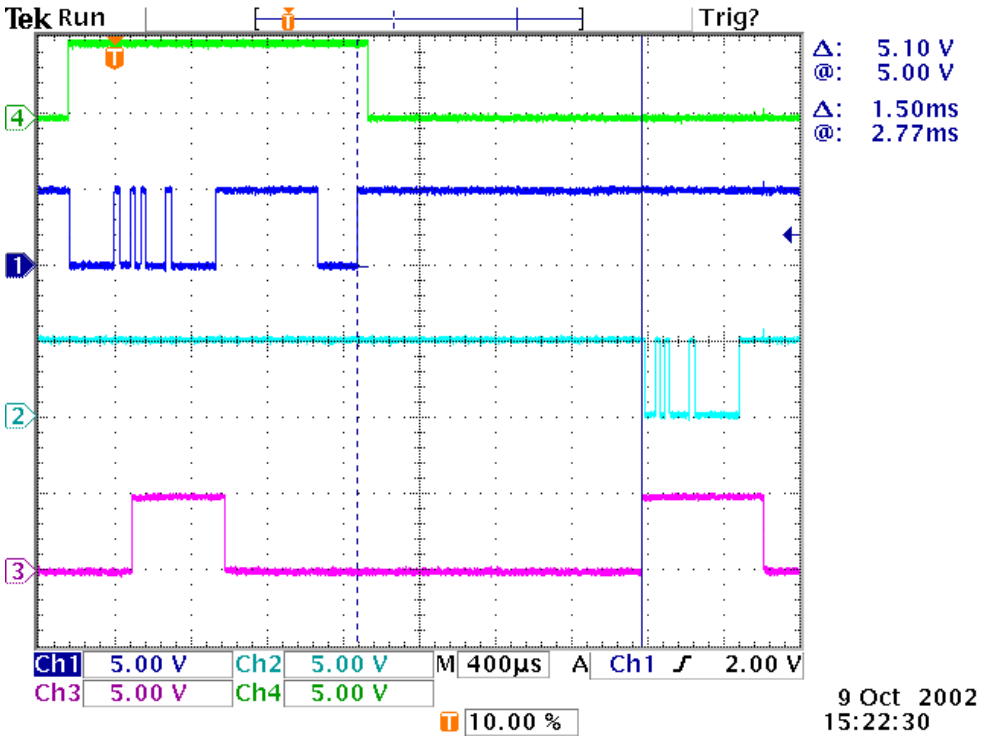
If the device sends a command in the middle of a stream of payload data to be transmitted, the command executes in the order it is received. If the radio is continuously receiving data, the radio waits for a break in the received data before executing the command. The CTS signal frames the response coming from the binary command request as shows in the following figure.

The user must observe a minimum time delay of 100 µs (after the stop bit of the command byte has been sent) before de-asserting the CMD (pin 16). The command executes after all parameters associated with the command have been sent. If all parameters are not received within 0.5 seconds, the module aborts the command and returns to Idle Mode.

Note Binary commands that return only one parameter byte must also be written with two parameter bytes, 0-padded, LSB first. Refer to for a binary programming example.

You can query commands for their current value by sending the command logically ORed (bit-wise) with the value 0x80 (hexadecimal) with CMD asserted. When the device sends the binary value (with no parameters), the current value of the command parameter is sent back through the DO pin.

Binary command write then read



Signal #4 is CMD (pin 16)

Signal #1 is the DIN (pin 3) signal to the radio

Signal #2 is the DOUT (pin 2) signal from the radio

Signal #3 is $\overline{\text{CTS}}$ (pin 12)

This graph shows a value written to a register and then read out to verify it. While not in the middle of other received data, the $\overline{\text{CTS}}$ signal outlines the data response out of the device

Note For the XBee module to recognize a binary command, you must set the **RT (DI2 Configuration)** parameter to one. If you have not enabled binary programming **RT** = **0** or **2**, the device will not recognize that the CMD pin is asserted and therefore will not recognize the data as binary commands.

Configuration and commands

Programming examples

For steps on sending AT commands to a device, refer to:

- Send AT commands
- Exit Command mode

For more information, refer to the XCTU online help at:
docs.digi.com/display/XCTU/XCTU+Overview

Connect the device to a PC

The programming examples that follow require the installation of XCTU and a serial connection to a PC. Digi stocks connector boards to facilitate interfacing with a PC.

1. Download XCTU from the Digi website:
   digi.com/products/xbee-rf-solutions/xctu-software/xctu#resources
2. After the .exe file downloads to the PC, double-click the file to launch the XCTU Setup Wizard. Follow the steps in the wizard to completely install XCTU.
3. Mount the device to an interface board, then connect the assembly to a PC.
4. Launch XCTU and click the **Add devices** tab on the upper left corner of the screen.
5. Verify that the baud rate and parity settings of the Serial/USB port match those of the device.

Note Failure to enter Command mode is commonly due to baud rate mismatch. Ensure that the **Baud Rate:** setting on the Add radio device window matches the interface data rate of the device. By default, the BD parameter = 9600 b/s.

Send binary commands

Example

Use XCTU's Serial Console tool to change the device's **DT** (Destination Address) parameter and save the new address to non-volatile memory.

This example requires XCTU and a serial connection to a PC.

To send binary commands:

1. Set the **RT** command to 1 to enable binary command programming; do this in Command mode or configure it through XCTU.
2. Drive pin 16 high to assert CMD by de-asserting the $\overline{\text{RTS}}$ line in XCTU. The device enters Binary Command mode.
3. Send hexadecimal bytes (parameter bytes must be 2 bytes long). The next four lines are examples, not required values:
   00 (Send binary command **DT**)
   0D (Least significant byte of parameter bytes)
   1A (Most significant byte of parameter bytes)
   08 (Send binary command **WR**)
4. Drive pin 16 low to de-assert CMD. After you send the commands, $\overline{\text{CTS}}$ (pin) de-asserts (driven low) temporarily. The device exits Binary Command mode.

The default flow control is NONE, so if you are using XCTU, $\overline{\text{CTS}}$ is not an issue. However, you can still observe the behavior of the $\overline{\text{CTS}}$ line by monitoring the $\overline{\text{CTS}}$ indicator in the terminal or console.

Special commands

The following commands are special commands.

FR (Force Reset)

If you issue **FR** while the device is in Command Mode, the reset effectively exits Command mode.

Resets the device through the UART. The device responds immediately with an **OK** and performs a reset 100 ms later.

Parameter range

N/A

Default

N/A

PL (TX Power Level)

Sets or displays the power level at which the device transmits conducted power.

The device calibrates Power lever 4 and other levels are approximate.

Binary command

0x3C (60 decimal)

Parameter range

0 - 4

Parameter	Configuration
0	+7.0 dBm, (5 mW)
1	+15.0 dBm, (32 mW)
2	+18.0 dBm, (63 mW)
3	+21.0 dBm, (125 mW)
4	+24.0 dBm, (250 mW)

Default

4

Bytes returned

1

Command mode options

The following commands are Command mode option commands.

AT (Guard Time After)

Sets or displays the time-of-silence that follows the **CC** (Command Sequence Character) of the Command mode sequence (**BT** + **CC** + **AT**). By default, one second must elapse before and after the command sequence character.

The times-of-silence surrounding the Command Sequence Character prevent the device from inadvertently entering Command mode.

Binary command

0x05 (5 decimal)

Parameter range

0x02 – 0xFFFF [x 100 milliseconds]

Default

0xA (1 second)

Bytes returned

2

BT (Guard Time Before)

Sets the **DI** pin silence time that must precede the Command Sequence Character (**CC** command) of the Command mode sequence.

Binary command

0x04 (4 decimal)

Parameter range

2 - 0xFFFF [x 100ms]

Default

0x0A (1 second)

Bytes returned

2

CC (Command Sequence Character)

Sets or displays the character the device uses between guard times of the AT Command mode sequence. The AT 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.

Binary command

0x13 (19 decimal)

Parameter range

0x0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

Default

0x2B (ASCII “+”)

Bytes returned

1

CD (DO3 Configuration)

Selects or reads the behavior of the DO3/RX LED line.

Parameter range

0 - 3

Parameter	Configuration
0	RX LED
1	Default high
2	Default low
3	Reserved
4	Assert only when you have sent the packet addressed to the device.

Default

0

Bytes returned

1

CN (Exit Command Mode)

Makes the device exit Command mode.

Binary command

0x09 (9 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

CT (Command Mode Timeout)

Set or read 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.

Use the **CN** (Exit Command mode) command to exit Command mode manually.

Binary command

0x06 (6 decimal)

Parameter range

0x02 - 0xFFFFF [x 100 milliseconds]

Default

0xC8 (20 seconds)

Bytes returned

2

E0 (Echo Off)

Turns off the character echo in Command mode.

By default, echo is off.

Binary command

0x0A (10 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

E1 (Echo On)

Enables character echo in Command mode. Each character that you type echoes back to the terminal when **E1** is active. **E0** (Echo Off) is the default.

Binary command

0x0B (11 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

PC (Power-up to Transparent operating mode)

Sets the device to power-up directly into Command mode from reset or power-on. If you enable the **PC** Command with the **SM** Parameter set to 1, you can use the DI3 (pin 9) to enter Command mode.

If the **PC** command is enabled with the **SM** parameter set to 1, DI3 (pin 9) can be used to enter the module into Command mode. When the device de-asserts (low) the DI3 pin, it wakes-up in Command mode.

This behavior allows device DTR emulation.

Binary command

0x1E (30 decimal)

Parameter range

0 - 1

Parameter	Configuration
0	Power-up to Idle mode
1	Power-up to Command mode

Default

0

Bytes returned

1

Networking and security commands

The following AT commands are networking and security commands.

AM (Auto-set MY)

Sets the **MY** (Source Address) parameter from the factory-set serial number of the device. The address consists of bits 29, 28 and 13-0 of the serial number, in that order.

Sending **AM** displays the address.

Note This command is only supported on S3B modules.

Binary command

0x3A (58 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

MD (RF Mode)

Sets or displays the settings that enable the Peer-to-peer or Repeater modes on the device.

Repeater Mode enables longer range via an intermediary device. When **MD**=3, the module acts as a "store and forward" repeater. The device repeats any packets not addressed to this node. A Repeater End Node handles repeated messages, but will not repeat the message over-the-air.

For more information, see Repeater mode.

Binary command

0x32 (50 decimal)

Parameter range

0, 3, 4

Parameter	Configuration
0	Peer-to-Peer (transparent operation)
3	Repeater & End Node
4	End Node

Default

0

Bytes returned

1

MY (Source Address)

Sets or displays the Source Address of a device.

For more information, see Addressing.

Binary command

0x2A (42 decimal)

Parameter range

Default

0xFFFF (Disabled - **DT** (Destination Address) parameter serves as both source and destination address).

Bytes returned

2

Network commands

The following commands are network commands.

DT (Destination Address)

Sets or displays the networking address of a device. The devices use three filtration layers:

- Vendor ID Number (**ID**)
- Channel (**HP**)
- Destination Address (**DT**)

The **DT** command assigns an address to a device that enables it to communicate with only devices that have the same address. All devices that share the same Destination Address can communicate freely with each other. Devices in the same network with a different Destination Address than the transmitter listen to all transmissions to stay synchronized, but will not send any of the data out their serial ports.

Binary command

0x00 (0 decimal)

Parameter range

0 - 0xFFFF

Default

0

Bytes returned

2

HP (Preamble ID)

Set or read the device's hopping channel number. A channel is one of three layers of filtration available to the device.

In order for devices to communicate with each other, the devices must have the same channel number since each channel uses a different hopping sequence. Devices can use different channels to prevent devices in one network from listening to transmissions of another.

When a device receives a packet it checks **HP** before the network ID, as it is encoded in the preamble and the network ID is encoded in the MAC header.

Binary command

0x11 (17 decimal)

Parameter range

0 - 6

Default

0

Bytes returned

1

HT (Time before Wake-up Initializer)

Sets or displays the time of inactivity (no serial or RF data is sent or received) before a transmitting (TX) RF device sends a wake-up initializer. The main purpose of this command is to prevent devices from sending the Long Header with every data packet. For more information on long headers, see LH (Wakeup Initializer Timer).

For RX devices operating in Cyclic Sleep mode (**SM** = 4-8), set **HT** to be shorter than the **ST** command.

The TX device sends a wake-up initializer, which instructs all receiving (RX) devices to remain awake to receive RF data.

From the perspective of the RX device: after **HT** time elapses and the inactivity timeout (**ST** command) is met, the RX device goes into cyclic sleep. In cyclic sleep, the RX device wakes once per sleep interval (**SM** command) to check for a wake-up initializer. When it detects a wake-up initializer, the device stays awake to receive data. The wake-up initializer must be longer than the cyclic sleep interval to ensure that sleeping devices detect incoming data.

When **HT** time elapses, the TX device knows it needs to send a wake-up initializer for all RX devices to remain awake and receive the next transmission.

Binary command

0x03 (3 decimal)

Parameter range

0 - 0xFFFF [x 100 ms]

Default

0xFFFF (wake-up initializer will not be sent)

Bytes returned

2

ID (Network ID)

Sets or displays the Vendor Identification Number (VID) of the device. Devices must have matching VIDs in order to communicate. If the device uses OEM network IDs, 0xFFFF uses the factory value.

Binary command

0x27 (39 decimal)

Parameter range

0x10 - 0x7FFF (user-settable)

0x8000 - 0xFFFF (factory-set)

Default

N/A

MK (Address Mask)

Sets or read the device's Address Mask.

All RF data packets contain the Destination Address of the transmitting (TX) device. When a device receives a packet, the TX device's Destination Address is logically combined bitwise (in other words, joined with AND) with the Address Mask of the receiving (RX) device. The resulting value must match

the Destination Address or Address Mask of the RX device for the packet to be received and sent out the RX device's DO (Data Out) pin. If the combined value does not match the Destination Address or Address Mask of the RX device, it discards the packet.

The firmware treats all 0 values as irrelevant and ignores them. For more information, see Addressing.

Binary command

0x12 (18 decimal)

Parameter range

0 - 0xFFFF

Default

0xFFFF

Note The Destination address (**DT** parameter) of the transmitting device must match the destination address of the receiving device.)

Bytes returned

2

RN (Delay Slots)

Sets or displays the time delay that the transmitting device inserts before attempting to resend a packet. If the transmitting device fails to receive an acknowledgment after sending a packet, it inserts a random number of delay slots (ranging from 0 to (**RN** minus 1)) before attempting to resend the packet. Each delay slot is 38 ms.

If two devices attempt to transmit at the same time, the random time delay after packet failure only allows one device to transmit the packet successfully, while the other device waits until the channel is available for RF transmission.

RN is only applicable if:

- You enable retries using the **RR** command, or
- You insert forced delays into a transmission using the **TT** command

Binary command

0x19 (25 decimal)

Parameter range

0 - 0xFF [slots]

Default

0 (no delay slots inserted)

Bytes returned

1

RR (Unicast Mac Retries)

Sets or displays the maximum number of retries sent for a given RF packet. When you enable RR (RR > 0), it enables RF packet retries and ACKs.

After transmitting a packet, the transmitting device waits to receive an ACK from a receiving device. If it does not receive the ACK in the time that **RN** specifies, it transmits the original packet again. The transmitting device transmits the RF packet repeatedly until it receives an ACK or until it sends the packet **RR** times.

Note You must have retries enabled for all modules in the network for retries to work.

Binary command

0x18 (24 decimal)

Parameter range

0 - 0xFF

Default

0 (disabled)

Bytes returned

1

SY (Time Before Initialization)

Keeps a communication channel open as long as the device transmits or receives before the active connection expires. You can use this command to reduce latency in a query/response sequence and set it 100 ms longer than the delay between transmissions. This command allows multiple Modules to share a hopping channel for a given amount of time after receiving data.

By default, all packets include an RF initializer that contains channel information used to synchronize any listening receivers to the transmitter's hopping pattern. Once a new device comes within range, it is able to instantly synchronize to the transmitter and start receiving data. If no new devices are introduced into the system, the synchronization information becomes redundant once devices have become synchronized.

The **SY** command allows the devices to remove this information from the RF Initializer after the initial synchronization. For example, changing the SY Parameter to 0x14 (20 decimal) allows all devices to remain in sync for 2 seconds after the last data packet was received. The device does not send synchronization information unless transmission stops for more than 2 seconds. This command allows significant savings in packet transmission time.

The **SY** command is not supported above a value of 5 when interfacing an XBee-PRO XSC S3B with a 9XStream.

We do not recommend this command for use in an interference-prone environment. Interference can break up the session making the communications channel unavailable until the **SY** time expires. If you set **SY** to zero, the channel session opens and closes with each transmission, resulting in a more robust link with increased latency.

Binary command

0x17 (23 decimal)

Parameter range

0 – 0xFF [x 100 milliseconds]

Default

0 (Disabled - channel initialization information is sent with each RF packet.)

Bytes returned

1

TT (Streaming Limit)

Sets or displays the limit on the number of bytes that a device can send before issuing a random delay.

If a device is sending a continuous stream of RF data, it inserts a delay that stops its transmission and gives other devices time to transmit once it sends **TT** bytes of data. The random delay it inserts lasts between 1 and **RN** + 1 delay slots where each delay slots lasts 38 ms.

You can use **TT** to simulate full-duplex behavior.

Binary command

0x1A (26 decimal)

Parameter range

0xFFFF (0 = disabled)

Default

0xFFFF (65535 decimal)

Bytes returned

2

Serial interfacing commands

The following commands are 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

When the host sends a value above 0x7D, 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)	S3* BD parameter stored (HEX)	S3B BD parameter stored (HEX)
0	1200	0	0
4	19,200	4	4
6	57600	6	5
12C	300	12B	12B
E100	57600	E883	E10D
* The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.			

Binary command

0x15 (21 decimal)

Parameter ranges

0 - 6 (standard rates)

0x7D – 0xFFFF (125d – 65535d) (non-standard rates)

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

Default

Set to equal device's factory-set RF data rate.

Bytes returned

2

CS (DO2 Configuration)

Sets or displays the behavior of the DO2 pin signal. This output can provide RS-232 flow control and controls the TX enable signal for RS-485 or RS-422 operations.

By default, DO2 provides RS-232 Clear-to-Send ($\overline{\text{CTS}}$) flow control.

Binary command

0x1F (31 decimal)

Parameter range

0 - 4

Parameter	Configuration
0	RS-232 $\overline{\text{CTS}}$ flow control
1	RS-485 TX enable low
2	Static high
3	RS-485 TX enable high
4	Static low

Default

0

Bytes returned

1

FL (Software Flow Control)

Configures software flow control. Use the device as the DO2 (pin 12) to implement Hardware flow control to regulate when serial data can be transferred to the device .

The XON character used is 0x11 (17 decimal).

The XOFF character used is 0x13 (19 decimal).

Binary command

0x07 (7 decimal)

Parameter range

0 - 1

Parameter Value	Configuration
0	Disable software flow control
1	Enable software flow control

Default

0

Bytes returned

1

FT (Flow Control Threshold)

Sets or displays the flow control threshold.

De-assert CTS when the number of bytes specified by the **FT** parameter are in the DIN buffer. Re-assert CTS when less than FT - 16 bytes are in the UART receive buffer.

Binary command

0x24 (36 decimal)

Parameter range

DI buffer size minus 0x11 bytes

Default

0xBBF - DI buffer size minus 0x11 (17 decimal)

NB (Parity)

Set or read the parity settings for UART communications.

Parameter range

0 - 4 (S3* hardware)

0 - 5 (S3B hardware)

***** The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.

Parameter	Configuration
0	8-bit (no parity)
1	8-bit even
2	8-bit odd
3	8-bit mark
4	8-bit space
5	9-bit data (S3B Hardware)

Default

0

Bytes returned

1

PK (Maximum RF Packet Size)

Sets or displays the maximum size of RF packets that a device in Transparent operating mode (**AP** = **0**) transmits. You can use the maximum packet size along with the **RB** and **RO** parameters to implicitly set the channel dwell time.

Changes to the **PK** parameter may have a secondary effect on the RB (Packetization Threshold) parameter. **RB** must always be less than or equal to **PK**. If you change **PK** to a value that is less than the current value of **RB**, the **RB** value lowers to be equal to **PK**.

Note This command is only supported on S3B hardware.

Binary command

0x29 (41 decimal)

Parameter range

0x2 - 0x100 [Bytes]

Default

0x40 (64 decimal)

Bytes returned

2

RB (Packetization Threshold)

Sets or displays the character threshold value.

RF transmission begins after a device receives data in the DIN buffer and meets either of the following criteria:

- The UART receives **RB** characters
- The UART receive lines detect **RO** character times of silence after receiving at least 1 byte of data

If a device lowers **PK** below the value of **RB**, **RB** is automatically lowered to match the PK value.

If **RO** = 0, the device must receive **RB** bytes before beginning transmission.

RB and **RO** criteria only apply to the first packet of a multi-packet transmission. If data remains in the DIN buffer after the first packet, transmissions continue in a streaming manner until there is no data left in the DIN buffer.

Binary command

0x20 (32 decimal)

Parameter range

1 - 0x100 (bytes) (Maximum value equals the current value of **PK** Parameter (up to 0x100 HEX (800 decimal))

Default

1

Bytes returned

2

RO (Packetization Timeout)

Set or read the number of character times of inter-character silence required before transmission begins. For information on how **RO** works with the **RB** command, see RB (Packetization Threshold).

After a device receives a serial byte and it receives no other byte before the RO timeout, the tranmission begins.

Binary command

0x21 (33 decimal)

Parameter range

0 - 0xFFFF [x 200 µs]

Default

0

Bytes returned

2

RT (DI2 Configuration)

Sets or displays the behavior of the DI2/$\overline{\text{RTS}}$/CMD line. You must use **RT** to enable $\overline{\text{RTS}}$ flow control or binary programming.

Binary command

0x16 (22 decimal)

Parameter range

0 - 2

Parameter	Configuration
0	Disabled
1	Binary Command enable
2	$\overline{\text{RTS}}$ flow control enable

Default

0 (disabled)

Bytes returned

1

Diagnostic commands

The following commands are diagnostic commands.

ER (Receive Count Error)

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.

Sets or displays the receive error. The error-count records the number of packets partially received then aborted on a reception error. This value returns to 0 after a reset and is not non-volatile (that is, the value does not persist in the module's memory after a power-up sequence).

Once the “Receive Error Count” reaches its maximum value (up to 0xFFFF), it remains at its maximum count value until the maximum count value is explicitly changed or you reset the device.

To reset the counter to any 16-bit unsigned value, append a hexadecimal parameter to the command. This value is volatile (the value does not persist in the device's memory after a power-up sequence).

Binary command

0x0F (15 decimal)

Parameter range

0 - 0xFFFF

Default

0

Bytes returned

2

GD (Receive Good 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 command.

Parameter range

0 - 0xFFFF

Default

0

Bytes returned

2

RE (Restore Defaults)

Restore device parameters to factory defaults.

RE does not cause the device to store default values to non-volatile (persistent) memory. You must send the **WR** command prior to power-down or reset to save the default settings in the device's non-volatile memory.

Binary command

0x0E (14 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

RP (RSSI PWM Timer)

Enables a pulse-width modulated (PWM) output on the CONFIG pin. We calibrate the pin to show the difference between received signal strength and the sensitivity level of the device. PWM pulses vary from zero to 95 percent. Zero percent means the RF signal the device receives is at or below the device's sensitivity level.

The following table shows dB levels above sensitivity and PWM values. The total time period of the PWM output is 8.32 ms. PWM output consists of 40 steps, so the minimum step size is 0.208 ms.

dB above sensitivity	PWM percentage (high period / total period)
10	47.5%
20	62.5%
30	77.5%

A non-zero value defines the time that PWM output is active with the RSSI value of the last RF packet the device receives. After the set time when the device has not received RF packets, it sets the PWM output low (0 percent PWM) until the device receives another RF packet. It also sets PWM output low at power-up. A parameter value of 0xFF permanently enables PWM output and always reflects the value of the last received RF packet.

The PWM output and Config input share the CONFIG /RSSI pin. When the device is powered, the Config pin is an input. During the power-up sequence, if RP is a non-zero value, the firmware configures the Config pin as an output and sets it low until the device receives the first RF packet. With a non-zero RP parameter, the CONFIG pin is an input for RP ms after power up.

The PWM output and Config input share the CONFIG pin. When the device is powered, the Config pin is an input. During the power-up sequence, the device reads Config pin to determine whether to go to AT command mode. The Config pin is then configured as an output and set to low until the device receives the first RF packet. With a non-zero RP parameter, the CONFIG pin is an input for RP ms after power up.

Binary command

0x22 (34 decimal)

Parameter range

0 - 0xFF [x 100 ms]

Default

0 (disabled)

Bytes returned

1

RZ (DI Buffer Size)

Reads the size of the DI buffer of the UART receiving device.

Multiply the **DI** buffer size by 1.5 to determine the **DO** buffer size.

Binary command

0x2C (44 decimal)

Parameter range

Read-only

Default

N/A

Bytes returned

1

Note This command is only supported on S3B modules.

RS (RSSI)

Returns the signal level of the last packet received.

You can use this reading to determine range characteristics of devices under various conditions of noise and distance.

Once you issues the command, the device returns a value between 0x6 and 0x36, where 0x36 represents a very strong signal level and 0x4 indicates a low signal level.

Binary command

0x1C (28 decimal)

Parameter range

0x06 – 0x36 [read-only]

Default

N/A

Bytes returned

1

SH (Serial Number High)

Reads the device's serial number high word.

Binary command

0x25 (37 decimal)

Parameter range

0x0 - 0xFFFF [read-only]

Default

N/A

Bytes returned

2

SL (Serial Number Low)

Reads the serial number low word of the device.

Binary command

0x26 (38 decimal)

Parameter range

0 - 0xFFFF [read-only]

Default

N/A

Bytes returned

2

TR (Transmission Failure Count)

Records the number of retransmit failures.

The device increments this number each time a packet is not acknowledged within the number of retransmits specified by the **RR** (Retries) Command. It therefore counts the number of packets that have not been successfully received and have been dropped.

The TR Parameter is not non-volatile and resets to zero each time the device is reset.

Binary command

0x1B (27 decimal)

Parameter range

0 - 0xFFFF

Default

0

Bytes returned

2

VR (Firmware Version - Short)

Reads the firmware version on a device.

Firmware versions contain four significant digits: A.B.C.D. If B = 2, the device is programmed for operation in Australia only.

Binary command

0x14 (20 decimal)

Parameter range

[read-only]: 0 - 0xFFFF

Default

N/A

Bytes returned

2

Sleep commands

The following commands are sleep commands.

FH (Force Wakeup Initializer)

Forces the device to send a wake-up initializer on the next transmission.

Only use **FH** with cyclic sleep modes active on remote devices.

You do not need to issue the **WR** (Write) command with **FH**.

Binary command

0x0D (13 decimal)

Parameter range

N/A

Default

N/A

Bytes returned

N/A

HT (Time before Wake-up Initializer)

Sets or displays the time of inactivity (no serial or RF data is sent or received) before a transmitting (TX) RF device sends a wake-up initializer. The main purpose of this command is to prevent devices from sending the Long Header with every data packet. For more information on long headers, see LH (Wakeup Initializer Timer).

For RX devices operating in Cyclic Sleep mode (**SM** = 4-8), set **HT** to be shorter than the **ST** command.

The TX device sends a wake-up initializer, which instructs all receiving (RX) devices to remain awake to receive RF data.

From the perspective of the RX device: after **HT** time elapses and the inactivity timeout (**ST** command) is met, the RX device goes into cyclic sleep. In cyclic sleep, the RX device wakes once per sleep interval (**SM** command) to check for a wake-up initializer. When it detects a wake-up initializer, the device stays awake to receive data. The wake-up initializer must be longer than the cyclic sleep interval to ensure that sleeping devices detect incoming data.

When **HT** time elapses, the TX device knows it needs to send a wake-up initializer for all RX devices to remain awake and receive the next transmission.

Binary command

0x03 (3 decimal)

Parameter range

0 - 0xFFFF [x 100 ms]

Default

0xFFFF (wake-up initializer will not be sent)

Bytes returned

2

LH (Wakeup Initializer Timer)

Sets or displays the duration of time during which the wake-up initializer is sent. When receiving devices are in Cyclic Sleep Mode, they power-down after a period of inactivity as specified by the **ST** parameter and will periodically wake and listen for data transmissions. In order for the receiving devices to remain awake, they must detect ~35 ms of the wake-up initializer.

You must use **LH** whenever a receiving device is operating in Cyclic Sleep mode. The wake-up initializer time must be longer than the cyclic sleep time, which is set by the **SM** (Sleep Mode) parameter. If the wake-up initializer time is less than the Cyclic Sleep interval, the connection is at risk of missing the wake-up initializer transmission.

To view a diagram of the correct configuration, see SM (Sleep Mode).

Binary command

0x0C (12 decimal)

Parameter range

0 - 0xFF [x100 milliseconds]

Default

1

Bytes returned

1

PW (Pin Wakeup)

Enables or disables the sleep pin.

Under normal operation, a device in Cyclic Sleep mode cycles from an active state to a low-power state at regular intervals until it is ready to receive data. If you set **PW** to 1, you can use the SLEEP pin (pin 26) to wake the device from Cyclic Sleep. When you de-assert (low) the SLEEP pin, the device is operational and will not go into Cyclic Sleep.

Once you assert the SLEEP pin, the device remains active for the period of time specified by the **ST** parameter and returns to Cyclic Sleep mode if no data is ready to transmit. **PW** is only valid if Cyclic Sleep is enabled.

Binary command

0x1D (29 decimal)

Parameter range

0 - 1

Parameter	Configuration
0	Disabled
1	Enabled

Default

0

Bytes returned

1

SM (Sleep Mode)

Sets or displays the device's sleep mode settings, which configure the device to run in states that require minimal power consumption.

By default, Sleep Mode is disabled and the device remains continually active. The **SM** command allows the device to run in a lower-power state and be configured in one of the eight settings. Cyclic Sleep settings wake the device after the amount of time designated by the **SM** command.

If the device detects a wake-up initializer during the time it is awake, it synchronizes with the transmitter and starts receiving data after the wake-up initializer runs its duration. Otherwise, the device returns to Sleep Mode and continues to cycle in and out of inactivity until it detects the Wake-up Initializer.

If a Cyclic Sleep setting is chosen, the ST, LH and HT parameters must also be set as described in Sleep mode.

Binary command

0x01

Parameter range

0, 1 3 - 8

Parameter	Description
0	Disabled
1	Pin Sleep
3	Cyclic 0.5 second sleep (RF module wakes every 0.5 seconds)
4	Cyclic 1 second sleep (RF module wakes every 1.0 seconds)
5	Cyclic 2 second sleep
6	Cyclic 4 second sleep
7	Cyclic 8 second sleep
8	Cyclic 16 second sleep

Default

0

Bytes returned

1

ST (Wake Time)

Sets or displays the amount of time (in tenths of seconds) that the device remains inactive before entering Sleep mode. For example, if you set **ST** to 0x64 (100 decimal), the device enters Sleep mode after 10 seconds of inactivity (no transmitting or receiving).

You can only use this command if you use **SM** to select Cyclic Sleep or Serial Port Sleep mode settings; see SM (Sleep Mode).

Binary command

0x02 (2 decimal)

Parameter range

0x10 – 0xFFFF [x 100 milliseconds]

Default

0x64 (10 seconds)

Bytes returned

2

Network configurations

Network topologies

The device supports three different network topologies:

- Point-to-point
- Point-to-multipoint
- Peer-to-peer

Point-to-point networks

This following section provides the RF communication type and RF mode for point-to-point networks.

Definition

Point-to-point means that an RF data link exists between two devices.

Sample network profile (Broadcast communications)

Use default values for all devices.

Sample network profile (Acknowledged communications)

Note Assume default values for all parameters that are not in this list. These profiles do not reflect addressing implementations.

1. Use XCTU or an alternative terminal program to send the **AM** command. See AM (Auto-set MY) for details.
2. Set the destination address to 0xFFFF, send: **DT FFFF**

Basic RF modes

Streaming, Multi-Transmit, Repeater.

Acknowledged RF mode

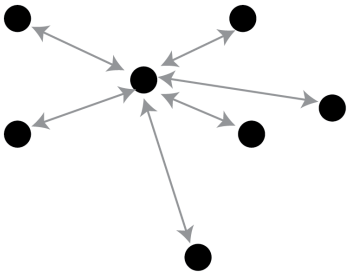
Acknowledged mode.

Point-to-multipoint networks

This following section provides the RF communication type and RF mode for point-to-multipoint networks.

Definition

Point-to-multipoint refers to a network with RF data links between one base and multiple remotes.



Sample network profile (Broadcast communications)

Note Assume default values for all parameters that are not in this list. These profiles do not reflect addressing implementations.

Base:

1. Send **MY 0** to set the source address to 0x00.
2. Send **DT FFFF** to set the destination address to 0xFFFF.

Remotes:

1. Use XCTU or another terminal program to send the **AM** command. See AM (Auto-set MY) for details.
2. Send **DT 0** to set the destination address to 0x00.

Sample network profile (Acknowledged communications)

Note Assume the default value for all parameters that are not in this list. These profiles do not reflect addressing implementations.

Base:

1. Send **MY 0** to set the source address to 0x00.
2. Send **DT FFFF** to set the destination address to 0xFFFF.
3. Send **RR 3** to set the number of retries to 3.

Remotes:

1. Use XCTU or another terminal program to send the **AM** command.
2. Send **DT FFFF** to set the destination address to 0xFFFF.
3. Send **RR 3** to set the number of retries to 3.

Basic RF modes

Streaming, Multi-Transmit, Repeater, and Polling.

Acknowledged RF mode

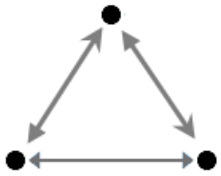
Acknowledged and Polling.

Peer to peer networks

This following section provides the RF communication type and RF mode for peer-to-peer networks.

Definition

In Peer-to-peer networks, devices remain synchronized without the use of master/slave dependencies. Each device shares the roles of master and slave. Digi's peer-to-peer architecture features fast synch times (35 ms to synchronize devices) and fast cold start times (50 ms before transmission).



Sample network profile (Broadcast communications)

Note Assume default values for all parameters that are not in this list. These profiles do not reflect addressing implementations.

Use default values for all devices.

Sample network profile (Acknowledged communications)

Note Assume default values for all parameters that are not in this list. These profiles do not reflect addressing implementations.

All devices:

1. Send **MY 0** to set the source address to 0x00.
2. Send **DT FFFF** to set the destination address to 0xFFFF.
3. Send **RR 3** to set the number of retries to 3.

Basic RF modes

Streaming.

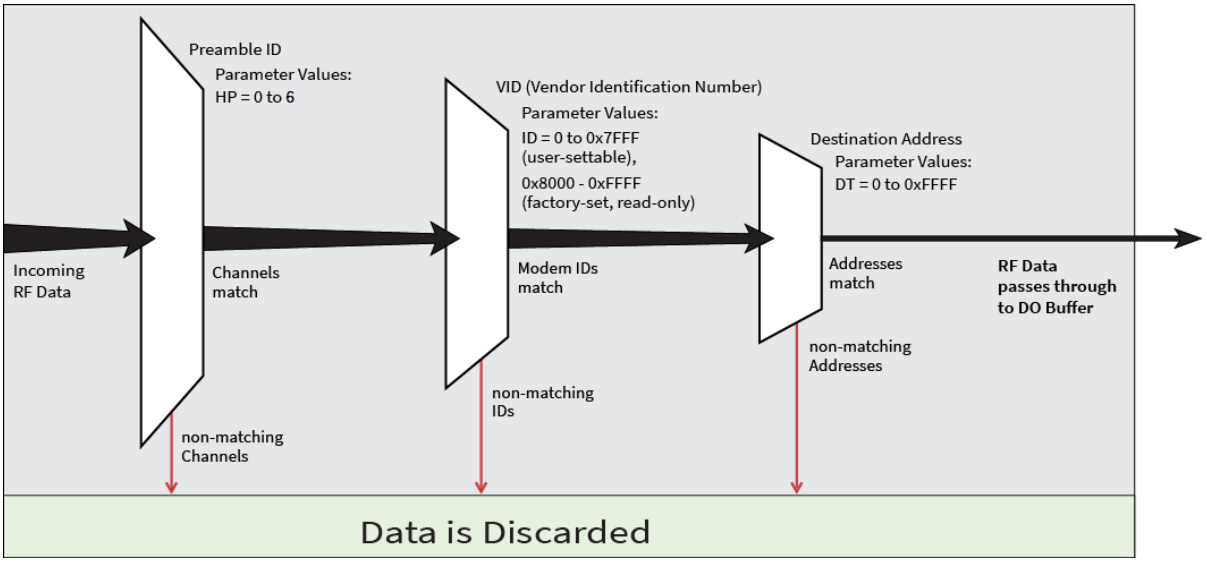
Acknowledged RF mode

Acknowledged.

Addressing

Each RF packet contains addressing information that the receiving devices use to filter incoming RF data. Receiving devices inspect the Preamble ID (**HP** parameter), Vendor Identification Number (**ID** parameter) and Destination Address (**DT** parameter) in each RF packet. A receiving device discards all data that does not pass through all three network security layers.

The following image illustrates the addressing layers in the RF packet header.



Address recognition

Transmissions can be addressed to a specific device or group of devices using the **DT** (Destination Address) and **MK** (Address Mask) parameters. The transmitting device dictates whether the packet is intended for a specific device (local address) or multiple devices (global address) by comparing the packet's **DT** parameter to its own **MK** parameter.

Basic communications

Basic communications includes two sub-types:

- Broadcast. By default, the XBee-PRO 900HP RF Modules communicate through Broadcast communications and within a peer-to-peer network topology. When any device transmits, all other devices within range receive the data and pass it directly to their host device.
- Addressed. If addressing parameters match, the device forwards the RF data it receives to the DOUT buffer; otherwise, it discards the RF data.

When using Basic communications, the integrator handles any functions, such as acknowledgments, at the application layer. The Broadcast modes provide transparent communications, meaning that the RF link replaces a wired link.

Streaming mode (default)

Characteristics: Highest data throughput

- Lowest latency and jitter
- Reduced immunity to interference
- Transmissions never acknowledged (ACK) by receiving module(s)

Required parameter values (TX Module): **RR** (Retries) = 0

Related commands: Networking (**DT**, **MK**, **MY**), Serial Interfacing (**PK**, **RB**, **RO**, **TT**)

Recommended use: Mode is most appropriate for data systems more sensitive to latency and/or jitter than to occasional packet loss.

Streaming Mode data flow

- Streaming Mode state diagram (TX Module)
- Events and processes in this mode are common to all of the other RF Modes.

Note When the device is streaming data, **RB** and **RO** parameters are only observed on the first packet. After transmission begins, the TX event continues uninterrupted until the DI buffer is empty or it reaches the streaming limit (**TT** Command). As with the first packet, the payload of each subsequent packet includes up to the maximum packet size (**PK** Command).

The transmitting device specifies the streaming limit (**TT** Command) as the maximum number of bytes the transmitting device can send in one transmission event. After it reaches the **TT** parameter threshold, the transmitting module forces a random delay of 1 to **RN** delay slots (exactly 1 delay slot if **RN** = 0).

The device sends subsequent packets without an RF initializer since receiving devices stay synchronized with the transmitting device for the duration of the transmission event (from preceding packet information). However, due to interference, some receiving devices may lose data (and synchronization to the transmitting device), particularly during long transmission events.

Once the transmitting device sends all pending data or has reaches the **TT** limit, the transmission event ends.

The transmitting device will not transmit again for exactly **RN** delay slots if the local (that is, the transmitting device's) **RN** parameter is set to a non-zero value.

The receiving devices will not transmit for a random number of delays between 0 and (**RN**-1) if the local (that is, the receiving module's) **RN** parameter is set to a non-zero value.

These delays are intended to lessen congestion following long bursts of packets from a single transmitting device, during which several receiving devices may have become ready to transmit.

Repeater mode

Characteristics

- Self-organizing - No route configuration is necessary.
- Self-healing / Fault-tolerant.
- Low power consumption and Minimized interference.
- Network throughput is determined by number of hops, not by number of repeaters. Multiple repeaters within range of source node count as one hop.
- Supports "transparent" multi-drop mode or addressed data filtering mode.
- Duplicate RF packets are automatically filtered out.
- All packets propagate to every node in the network (filtering rules apply).
- Broadcast communications - each packet comes out every node exactly once.
- Addressed communications - all devices see every packet. Only the device with a matching address forwards it to the DO buffer (UART IN).
- Devices transmit and forward data entering the network on any device through every repeater device until it reaches the ends of the network.
- Each repeater repeats a packet only once.

Constraints

- Requires that each device have a unique **MY** (Source Address) parameter.
- System must introduce just one packet at a time to the network for transmission (256 bytes max).
- Each hop (H) decreases the network throughput by a factor of 1/(H+1). Additional repeaters add network redundancy without decreasing throughput.

Required parameter values (TX module): **MD** = 3 or 4, **MY** = unique value. Send the **AM** (Auto-set **MY**) and **WR** (Write) commands to all devices in the network).

Related commands: Networking (**MD**, **DT**, **MY**, **AM**), Serial Interfacing (**RN**, **PK**, **RO**, **RB**).

Recommended use: Use in networks where intermediary nodes required to relay data to devices that are beyond the transmission range of the base device.

Theory of operation

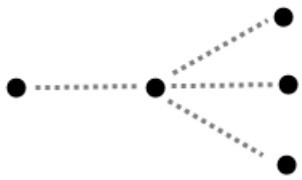
Integrators can extend the effective range and reliability of a data radio system by forwarding traffic through one or more repeaters.

Instead of using routing tables and path discovery to establish dynamic paths through a network, the repeater system uses a sophisticated algorithm to propagate each RF packet through the entire network.

The network supports RF packets of up to 256 bytes. The repeater network can operate using broadcast or addressed communications for multi-drop networks and works well in many systems with no special configuration.

When in Repeater Mode, the network repeats each message among all available nodes exactly one time. This mechanism eliminates the need for configuring specific routes. The network is self-organizing and self-healing so that the system is able to receive transmissions in the event of a device going down.

Sample Repeater network topology:



To summarize, the system sends and receives 64 bytes 5 times per second for a throughput of 320 bytes per second with no repeaters and 128 bytes per second with 2 repeaters. Generally, the network throughput decreases by a factor of 1/(R+1), with R representing the number of repeaters between the source and destination.

Configuration instruction (Basic Broadcast communications)

Assign each device a unique **MY** (source) address. The **AM** (Auto-set MY) command configures a unique source address based on device serial number.

Enable Basic Broadcast Communications (**DT** = 0xFFFF) or Addressed Broadcast Communications (**DT** specifies a destination).

Configure **PK**, **RO** and **RB** to ensure that **RF** packet aligns with protocol packet. (for example, **PK**=0x100, **RB**=0x100, **RO** depends on baud rate).

Configure one or more repeaters in the system (**MD** = 3).

Configure remote nodes as destinations (**MD** = 4). This ensures that the remote node waits for the repeater traffic to subside before transmitting a response.

The preceding configuration instructions reflect configuration for a Basic Broadcast Repeater system. To configure a Basic Addressed Repeater system, use the **DT** (Destination Address) parameter to assign unique addresses to each device in the network.

Algorithm details

- Packet ID (PID) consists of the transmitting device **MY** address and packet serial number.
- Devices ignore incoming packets with a PID already found in the PID buffer.
- Each device maintains a PID buffer 8 deep of previously received packets (managed as FIFO).

Packets may be shifted out the serial port and/or repeated depending on the **DT** parameter contained in the RF packet.

The following table shows the **DT** (Destination Address) parameter truth.

Address match	Send out serial port?	Repeat?
Global	Yes	Yes
Local	Yes	Yes
None	No	Yes

Repeat delay based on RSSI

A transmitted packet may be received by more that one repeater at the same time. To reduce the probability of repeaters transmiting at the same time that may result in a collision and possible data loss, an algorithm allows a variable back-off prior to retransmission of the packet by a repeater. Devices that receive the packet with a stronger RF signal (RSSI) have the first opportunity to retransmit the packet.

Use the **RN** (Delay Slots) parameter to configure the delay.

- Set **RN**=0 (no delays) for small networks with few repeaters or repeaters that are not within range of each other.
- Set **RN**=1 for systems with 2 to 5 repeaters that may be within range of each other.

Use the following formula to compute the actual length of the delay:

Delay (ms) = L * DS

DS = (-41-RSSI)/(10*RN)+RandomInt(0,RN)

Where:

- L is the length of the transmitted packet in milliseconds.
- DS is the number of delay slots to wait.
- RSSI is the received signal strength in dBm.
- **RN** is the value of the **RN** register.
- RandomInt(A,B) is a function that returns a random integer from A to B-0.

Response packet delay

As a packet propagates through the repeater network, if any node receives the data and generates a quick response, the response must be delayed so it does not collide with subsequent retransmissions

of the original packet. To reduce collisions, both repeater and end node radios in a repeater network delay transmission of data shifted in the serial port allowing any repeaters within range to complete their retransmissions.

The time for this delay is computed by the followinformula:

Maximum Delay (ms) = L * DS

DS = (((41(-100))/10)*__RN__)+__RN__+1

Where:

- L is the length of the transmitted packet in milliseconds.
- DS is the number of delay slots to wait.
- RSSI is the received signal strength in dBm.
- **RN** is the value of the **RN** register.

Use case - broadcast repeater network

Consider devices R1 through R10, each communicating to a PLC using the ModBus protocol and spaced evenly in a line. All ten nodes are configured as 'destinations & repeaters' within the scope of Basic Broadcast Communications (**MD**=3, **AM**, **DT**=0xFFFF, **PK**=0x100, **RO**=0x03, **RB**=0x100, **RN**=1). The Base Host (BH) shifts payload that is destined for R10 to R1. R1 initializes RF communication and transmits payload to nodes R2 through R5 which are all within range of R1. Modules R2 through R5 receive the RF packet and retransmit the packet simultaneously; they also send data out the serial ports to the PLCs.

The following table shows the commands used to configure repeater functions.

AT command	Binary command	AT command name	Range	# Bytes returned	Factory default
AM	0x3A (58d)	Auto-set MY	-	-	-
DT	0x00 (0d)	Destination Address	0-0xFFFF	2	0
MD	0x3C (60d)	RF Mode	3-4	1	0
MY	0x2A (42d)	Source Address	0-0xFFFF	2	0xFFFF
RN	0x19 (25d)	Delay Slots	0-0xFF [slots]	1	0
WR	0x08 (8d)	Write	-	-	-

Bandwidth considerations

Using broadcast repeaters in a network reduces the overall network data throughput as each repeater buffers an entire packet before retransmitting it. For example: if the destination is within range of the transmitter and the packet is 32 bytes long, the transmission takes approximately 72 ms on a 9600 baud XSC device. If that same packet has to propagate through two repeaters, it takes 72 ms to arrive at the first repeater, another 72 ms to get to the second and a final 72 ms to get to the destination, for a total of 216 ms.

Considering UART transfer times (~1ms/byte at 9600 baud), a server to send a 32 byte query and receive a 32 byte response is ~200 ms, which allows for 5 polls per second. With the two repeaters in the path, the same query/response sequence would take about 500 ms for 2 polls per second.

Acknowledged mode

Characteristics: Reliable delivery through positive acknowledgments for each packet

Throughput, latency, and jitter vary depending on the quality of the channel and the strength of the signal.

Recommended use: Acknowledge Mode configuration is appropriate when reliable delivery is required between modules. If messages are smaller than 256 bytes, use **RB** and **RO** commands to align **RF** packets with application packets.

Required parameter values (TX device): **RR** (Retries) >= 1

Related commands: Networking (**DT**, **MK**, **RR**), Serial Interfacing (**PK**, **RN**, **TT**, **RO**, **RB**)

The following table shows a sample network profile.

Device	Parameter settings (assume default values for parameter not listed)
All	ATRR A [set the number of retries to 0x0A] ATRN 5 [set the number of delay slots to 5]

Acknowledged mode connection sequence

After sending a packet while in Acknowledged Mode, the transmitting device listens for the ACK (acknowledgment). If it receives the ACK, it either sends a subsequent packet (if more transmit data is pending), or waits for exactly **RN** random delay slots before allowing another transmission (if no more data is pending for transmission).

- If the transmitting device does not receive the ACK within the allotted time, it retransmits the packet with a new RF initializer following the ACK slot. There is no delay between the first ACK slot and the first retransmission. Subsequent retransmissions incur a delay of a random number of delay slots, between 0 and **RN**.
- If **RN** is set to 0 on the transmitting device, there are never any back-off delays between retransmissions. During back-off delays, the transmitting device enters Idle Mode and may receive RF data. This can increase the back-off delay, as the radio cannot return to RF transmit (or retransmit) mode as long as it is receiving RF data.

After receiving and acknowledging a packet, the receiving device moves to the next frequency and listens for either a retransmission or new data for a specific period of time. Even if the transmitting device indicates no more pending transmit data, it may have not received the previous ACK and may retransmit the packet (potentially with no delay after the ACK slot). In this case, the receiving device always detects the immediate retransmission, which holds off the communications channel and reduces collisions.

Receiving devices acknowledge each retransmission they receive, but they only pass the first copy of a packet received out the UART. **RB** and **RO** parameters are not applied to subsequent packets. This means that once transmission has begun, it continues uninterrupted until the **DI** buffer is empty or reaches its streaming limit (**TT**).

As with the first packet, the payload of each subsequent packet includes up to the maximum packet size (**PK** parameter).

- The transmitting device checks for more pending data near the end of each packet.
- The streaming limit (**TT** parameter) specifies the maximum number of bytes the transmitting device sends in one transmission event, which may consist of many packets and retries. If it reaches the **TT** parameter, the transmitting device forces a random delay of 1 to **RN** delay slots (exactly 1 delay slot if **RN** is zero).

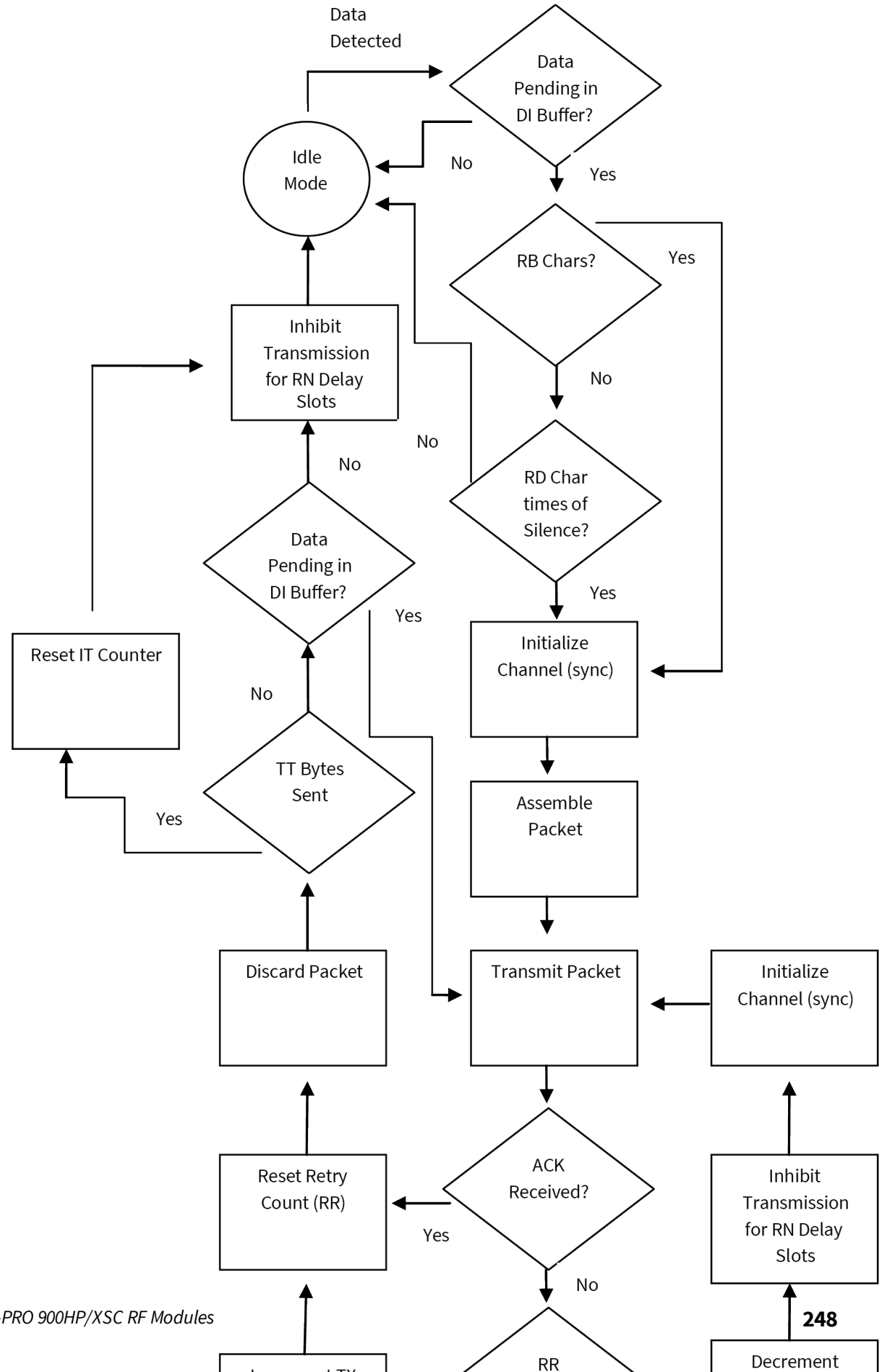
- The device counts each packet only once toward **TT**, no matter how many times the packet is retransmitted.

Subsequent packets in acknowledged mode are similar to those in streaming mode, with the addition of an acknowledgment between each packet, and the possibility of retransmissions. The device sends subsequent packets without an RF initializer, because the receiving devices are already synchronized to the transmitting device from the preceding packets and they remain synchronized for the duration of the transmission event.

Each packet retransmission includes an RF initializer. Once the transmitting device sends all pending data or reaches the **TT** limit, the acknowledged transmission event is completed.

- The transmitting device does not transmit again for exactly **RN** delay slots, if the local **RN** parameter is set to a nonzero value.
- The receiving device does not transmit for a random number of delay slots between 0 and (**RN**-1), if the local **RN** parameter is set to a nonzero value.

These delays can lessen congestion following long bursts of packets from a single transmitting device when several receiving devices may be ready to transmit.


Data Detected
Data Pending in DI Buffer?
No
Yes
Idle Mode
RB Chars?
Yes
No
Inhibit Transmission for RN Delay Slots
No
No
RD Char times of Silence?
Yes
Data Pending in DI Buffer?
Yes
Reset IT Counter
No
Initialize Channel (sync)
TT Bytes Sent
Yes
Assemble Packet
Discard Packet
Transmit Packet
Initialize Channel (sync)
Reset Retry Count (RR)
ACK Received?
Inhibit Transmission for RN Delay Slots
Yes
No
RR
Decrement

S3B hardware certifications

Agency certifications - United States

United States (FCC)

The XBee-PRO 900HP/XBee-PRO XSC RF Modules comply with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

To fulfill FCC certification requirements, the OEM must comply with the following regulations:

- The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.
- XBee/XBee-PRO RF modules may only be used with antennas that have been tested and approved for use with this module. For more information, see .

OEM labeling requirements

WARNING! As an Original Equipment Manufacturer (OEM) you must ensure that FCC labeling requirements are met. You must include a clearly visible label on the outside of the final product enclosure that displays the following content:

XBee-PRO 900HP and XBee-PRO XSC

Required FCC Label for OEM products containing the XBee-PRO 900HP/XBee-PRO XSC RF Module:

> Contains FCC ID: MCQ-XB900HP
>
> The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: *(i.)* this device may not cause harmful interference and *(ii.)* this device must accept any interference received, including interference that may cause undesired operation.

FCC notices

IMPORTANT: The XBee/XBee-PRO RF Module has been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

IMPORTANT: OEMs must test final product to comply with unintentional radiators (FCC section 15.107 and 15.109) before declaring compliance of their final product to Part 15 of the FCC rules.

IMPORTANT: The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, the device must undergo SAR testing.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can

radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect equipment and receiver to outlets on different circuits.
- Consult the dealer or an experienced radio/TV technician for help.

Limited modular approval

This is an RF module approved for Limited Modular use operating as a mobile transmitting device with respect to section 2.1091 and is limited to OEM installation for Mobile and Fixed applications only. During final installation, end-users are prohibited from access to any programming parameters. Professional installation adjustment is required for setting module power and antenna gain to meet EIRP compliance for high gain antenna(s).

Final antenna installation and operating configurations of this transmitter including antenna gain and cable loss must not exceed the EIRP of the configuration used for calculating MPE. Grantee (Digi) must coordinate with OEM integrators to ensure the end-users and installers of products operating with the module are provided with operating instructions to satisfy RF exposure requirements.

The FCC grant is valid only when the device is sold to OEM integrators. Integrators are instructed to ensure the end-user has no manual instructions to remove, adjust or install the device.

FCC-approved antennas

CAUTION! This device has been tested with Reverse Polarity SMA connectors with the antennas listed in the tables of this section. When integrated into OEM products, fixed antennas require installation preventing end-users from replacing them with non-approved antennas. Antennas not listed in the tables must be tested to comply with FCC Section15.203 (unique antenna connectors) and Section 15.247 (emissions).

CAUTION! The FCC requires that all spread spectrum devices operating within the Unlicensed radio frequency bands must limit themselves to a maximum radiated power of 4 Watts EIRP. Failure to observe this limit is a violation of our warranty terms, and shall void the user's authority to operate the equipment.

This can be stated: RF power - cable loss + antenna gain <= 36 dBm eirp.

Fixed base station and mobile applications

Digi RF Modules are pre-FCC approved for use in fixed base station and mobile applications. When the antenna is mounted at least 20cm (8") from nearby persons, the application is considered a mobile application.

Portable applications and SAR testing

If the module will be used at distances closer than 20cm to all persons, the device may be required to undergo SAR testing. Co-location with other transmitting antennas closer than 20cm should be avoided.

RF exposure statement

You must include the following Caution statement in OEM product manuals.

CAUTION! This equipment is approved only for mobile and base station transmitting devices. Antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

not

FCC-approved antennas (900 MHz)

The antennas in the tables below have been approved for use with this device. Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Antennas approved for use with the XBee-PRO 900HP RF Module

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
Omni-directional antennas					
A09-F0	Fiberglass Base Station	RPN	0 dBi	Fixed	0dB
A09-F1	Fiberglass Base Station	RPN	1.0 dBi	Fixed	0dB
A09-F2	Fiberglass Base Station	RPN	2.1 dBi	Fixed	0dB
A09-F3	Fiberglass Base Station	RPN	3.1 dBi	Fixed	0dB
A09-F4	Fiberglass Base Station	RPN	4.1 dBi	Fixed	0dB
A09-F5	Fiberglass Base Station	RPN	5.1 dBi	Fixed	0dB
A09-F6	Fiberglass Base Station	RPN	6.1 dBi	Fixed	0dB
A09-F7	Fiberglass Base Station	RPN	7.1 dBi	Fixed	0dB
A09-F8	Fiberglass Base Station	RPN	8.1 dBi	Fixed	0dB
A09-F9	Base Station	RPSMAF	9.2dBi	Fixed	0dB
A09-W7	Wire Base Station	RPN	7.1 dBi	Fixed	0dB
A09-F0	Fiberglass Base Station	RPSMA	0 dBi	Fixed	0dB

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-F1	Fiberglass Base Station	RPSMA	1.0 dBi	Fixed	0dB
A09-F2	Fiberglass Base Station	RPSMA	2.1 dBi	Fixed	0dB
A09-F3	Fiberglass Base Station	RPSMA	3.1 dBi	Fixed	0dB
A09-F4	Fiberglass Base Station	RPSMA	4.1 dBi	Fixed	0dB
A09-F5	Fiberglass Base Station	RPSMA	5.1 dBi	Fixed	0dB
A09-F6	Fiberglass Base Station	RPSMA	6.1 dBi	Fixed	0dB
A09-F7	Fiberglass Base Station	RPSMA	7.1 dBi	Fixed	0dB
A09-F8	Fiberglass Base Station	RPSMA	8.1 dBi	Fixed	0dB
A09-M7	Base Station	RPSMAF	7.2dBi	Fixed	0dB
A09-W7SM	Wire Base Station	RPSMA	7.1 dBi	Fixed	0dB
A09-F0TM	Fiberglass Base Station	RPTNC	0 dBi	Fixed	0dB
A09-F1TM	Fiberglass Base Station	RPTNC	1.0 dBi	Fixed	0dB
A09-F2TM	Fiberglass Base Station	RPTNC	2.1 dBi	Fixed	0dB
A09-F3TM	Fiberglass Base Station	RPTNC	3.1 dBi	Fixed	0dB
A09-F4TM	Fiberglass Base Station	RPTNC	4.1 dBi	Fixed	0dB
A09-F5TM	Fiberglass Base Station	RPTNC	5.1 dBi	Fixed	0dB
A09-F6TM	Fiberglass Base Station	RPTNC	6.1 dBi	Fixed	0dB
A09-F7TM	Fiberglass Base Station	RPTNC	7.1 dBi	Fixed	0dB
A09-F8TM	Fiberglass Base Station	RPTNC	8.1 dBi	Fixed	0dB
A09-W7TM	Wire Base Station	RPTNC	7.1 dBi	Fixed	0dB

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-HSM-7 [1]	Straight half-wave	RPSMA	3.0 dBi	Fixed / Mobile	0dB
A09-HASM-675	Articulated half-wave	RPSMA	2.1 dBi	Fixed/ Mobile	0dB
A09-HABMM-P6I	Articulated half-wave w/ 6” pigtail	MMCX	2.1 dBi	Fixed/ Mobile	0dB
A09-HABMM-6-P6I	Articulated half-wave w/ 6” pigtail	MMCX	2.1 dBi	Fixed/ Mobile	0dB
A09-HBMM-P6I	Straight half-wave w/ 6” pigtail	MMCX	2.1 dBi	Fixed/ Mobile	0dB
A09-HRSM	Right angle half-wave	RPSMA	2.1 dBi	Fixed	0dB
A09-HASM-7*	Articulated half-wave	RPSMA	2.1 dBi	Fixed	0dB
A09-HG	Glass mounted half-wave	RPSMA	2.1 dBi	Fixed	0dB
A09-HATM	Articulated half-wave	RPTNC	2.1 dBi	Fixed	0dB
A09-H	Half-wave dipole	RPSMA	2.1 dBi	Fixed	0dB
A09-HBMMP6I	1/2 wave antenna	MMCX	2.1dBi	Mobile	0dB
A09-QBMMP6I	1/4 wave antenna	MMCX	1.9 dBi	Mobile	0dB
A09-QI	1/4 wave integrated wire antenna	Integrated	1.9 dBi	Mobile	0dB
29000187	Helical	Integrated	-2.0 dBi	Fixed/ Mobile	0dB
A09-QW	Quarter-wave wire	Permanent	1.9 dBi	Fixed/ Mobile	0dB
A09-QSM-3H	Heavy duty quarter-wave straight	RPSMA	1.9 dBi	Fixed/ Mobile	0dB
A09-QBMM-P6I	Quarter-wave w/ 6” pigtail	MMCX	1.9 dBi	Fixed/ Mobile	0dB

[1]Installers should apply additional torque to screw on the antenna.

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-QHRN	Miniature Helical Right Angle solder	Permanent	-1 dBi	Fixed/ Mobile	0dB
A09-QHSN	Miniature Helical Right Angle solder	Permanent	-1 dBi	Fixed/ Mobile	0dB
A09-QHSM-2	2” Straight	RPSMA	1.9 dBi	Fixed/ Mobile	0dB
A09-QHRSM-2	2" Right angle	RPSMA	1.9 dBi	Fixed/ Mobile	0dB
A09-QHRSM-170	1.7" Right angle	RPSMA	1.9 dBi	Fixed/ Mobile	0dB
A09-QRSM-380	3.8" Right angle	RPSMA	1.9 dBi	Fixed/ Mobile	0dB
A09-QAPM-520	5.2” Articulated screw mount	Permanent	1.9 dBi	Fixed/ Mobile	0dB
A09-QSPM-3	3” Straight screw mount	Permanent	1.9 dBi	Fixed/ Mobile	0dB
A09-QAPM-3	3” Articulated screw mount	Permanent	1.9 dBi	Fixed/ Mobile	0dB
A09-QAPM-3H	3” Articulated screw mount	Permanent	1.9 dBi	Fixed/ Mobile	0dB
A09-DPSM-P12F	omni directional magnetic mount w/ 12ft pigtail	RPSMA	3.0 dBi	Fixed	0dB
A09-D3NF-P12F	omni directional magnetic mount w/ 12ft pigtail	RPN	3.0 dBi	Fixed	0dB
A09-D3SM-P12F	omni directional w/ 12ft pigtail	RPSMA	3.0 dBi	Fixed	0dB
A09-D3PNF	omni directional permanent mount	RPN	3.0 dBi	Fixed	0dB
A09-D3TM-P12F	omni directional w/ 12ft pigtail	RPTNC	3.0 dBi	Fixed	0dB
A09-D3PTM	omni directional permanent mount	RPTNC	3.0 dBi	Fixed	0dB
A09-M0SM	Mag Mount	RPSMA	0 dBi	Fixed	0dB
A09-M2SM	Mag Mount	RPSMA	2.1 dBi	Fixed	0dB
A09-M3SM	Mag Mount	RPSMA	3.1 dBi	Fixed	0dB

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-M5SM	Mag Mount	RPSMA	5.1 dBi	Fixed	0dB
A09-M7SM	Mag Mount	RPSMA	7.1 dBi	Fixed	0dB
A09-M8SM	Mag Mount	RPSMA	8.1 dBi	Fixed	0dB
A09-M0TM	Mag Mount	RPTNC	0 dBi	Fixed	0dB
A09-M2TM	Mag Mount	RPTNC	2.1 dBi	Fixed	0dB
A09-M3TM	Mag Mount	RPTNC	3.1 dBi	Fixed	0dB
A09-M5TM	Mag Mount	RPTNC	5.1 dBi	Fixed	0dB
A09-M7TM	Mag Mount	RPTNC	7.1 dBi	Fixed	0dB
A09-M8TM	Mag Mount	RPTNC	8.1 dBi	Fixed	0dB
Yagi antennas					
A09-Y6	2 Element Yagi	RPN	6.1 dBi	Fixed/ Mobile	0dB
A09-Y7	3 Element Yagi	RPN	7.1 dBi	Fixed/ Mobile	0dB
A09-Y8	4 Element Yagi	RPN	8.1 dBi	Fixed/ Mobile	0dB
A09-Y9	4 Element Yagi	RPN	9.1 dBi	Fixed/ Mobile	0dB
A09-Y10	5 Element Yagi	RPN	10.1 dBi	Fixed/ Mobile	0dB
A09-Y11	6 Element Yagi	RPN	11.1 dBi	Fixed/ Mobile	0dB
A09-Y12	7 Element Yagi	RPN	12.1 dBi	Fixed/ Mobile	0dB
A09-Y13	9 Element Yagi	RPN	13.1 dBi	Fixed/ Mobile	0.8dB

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-Y14	10 Element Yagi	RPN	14.1 dBi	Fixed/ Mobile	1.8dB
A09-Y14	12 Element Yagi	RPN	14.1 dBi	Fixed/ Mobile	1.8dB
A09-Y15	13 Element Yagi	RPN	15.1 dBi	Fixed/ Mobile	2.8dB
A09-Y15	15 Element Yagi	RPN	15.1 dBi	Fixed/ Mobile	2.8dB
A09-Y6TM	2 Element Yagi	RPTNC	6.1 dBi	Fixed/ Mobile	0dB
A09-Y7TM	3 Element Yagi	RPTNC	7.1 dBi	Fixed/ Mobile	0dB
A09-Y8TM	4 Element Yagi	RPTNC	8.1 dBi	Fixed/ Mobile	0dB
A09-Y9TM	4 Element Yagi	RPTNC	9.1 dBi	Fixed/ Mobile	0dB
A09-Y10TM	5 Element Yagi	RPTNC	10.1 dBi	Fixed/ Mobile	0dB
A09-Y11TM	6 Element Yagi	RPTNC	11.1 dBi	Fixed/ Mobile	0dB
A09-Y12TM	7 Element Yagi	RPTNC	12.1 dBi	Fixed/ Mobile	0dB
A09-Y13TM	9 Element Yagi	RPTNC	13.1 dBi	Fixed/ Mobile	0.8dB
A09-Y14TM	10 Element Yagi	RPTNC	14.1 dBi	Fixed/ Mobile	1.8dB
A09-Y14TM	12 Element Yagi	RPTNC	14.1 dBi	Fixed/ Mobile	1.8dB

Part number	Type	Connector	Gain	Application	Cable loss or power reduction for S3B device
A09-Y15TM	13 Element Yagi	RPTNC	15.1 dBi	Fixed/ Mobile	2.8dB
A09-Y15TM	15 Element Yagi	RPTNC	15.1 dBi	Fixed/ Mobile	2.8dB

FCC publication 996369 related information

In publication 996369 section D03, the FCC requires information concerning a module to be presented by OEM manufacturers. This section assists in answering or fulfilling these requirements.

2.1 General

No requirements are associated with this section.

2.2 List of applicable FCC rules

This module conforms to FCC Part 15.247.

2.3 Summarize the specific operational use conditions

Certain approved antennas require attenuation for operation. For the XBee-PRO 900HP RF Module, see FCC-approved antennas (900 MHz).

Host product user guides should include the antenna table if end customers are permitted to select antennas.

2.4 Limited module procedures

Not applicable.

2.5 Trace antenna designs

While it is possible to build a trace antenna into the host PCB, this requires at least a Class II permissive change to the FCC grant which includes significant extra testing and cost. If an embedded trace or chip antenna is desired, contact a Digi sales representative for information on how to engage with a lab to get the modified FCC grant.

2.6 RF exposure considerations

For RF exposure considerations see RF exposure statement and FCC-approved antennas (900 MHz).

Host product manufacturers need to provide end-users a copy of the “RF Exposure” section of the manual: RF exposure statement.

2.7 Antennas

A list of approved antennas is provided for the XBee-PRO 900HP RF Modules. See FCC-approved antennas (900 MHz).

2.8 Label and compliance information

Host product manufacturers need to follow the sticker guidelines outlined in OEM labeling requirements.

2.9 Information on test modes and additional testing requirements

Contact a Digi sales representative for information on how to configure test modes for the XBee-PRO 900HP RF Module.

2.10 Additional testing, Part 15 Subpart B disclaimer

All final host products must be tested to be compliant to FCC Part 15 Subpart B standards. While the XBee-PRO 900HP RF Module was tested to be complaint to FCC unintentional radiator standards, FCC

Part 15 Subpart B compliance testing is still required for the final host product. This testing is required for all end products. XBee-PRO 900HP RF Module Part 15 Subpart B compliance does not affirm the end product's compliance.

See FCC notices for more details.

ISED (Innovation, Science and Economic Development Canada)

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Labeling requirements

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product enclosure must display the following text:

Contains IC: 1846A-XB900HP

The integrator is responsible for its product to comply with IC ICES-003 and FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Transmitters for detachable antennas

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the tables in FCC-approved antennas (900 MHz) or with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device. The required antenna impedance is 50 ohms.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Detachable antenna

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire àl'établissement d'une communication satisfaisante.

Brazil ANATEL

Mexico IFETEL

Manufacturer: Digi International

Country: USA

Brand: Digi

Model: XBee PROS3B

Tariff Code (HS): 8517-62-14

IFETEL (IFT) number RCPDIXB19-1822 applies to these XBee 900 HP radios:

- XBP9B-DMWT-002
- XBP9B-DMST-002
- XBP9B-DMUT-002
- XBP9B-DPWT-001
- XBP9B-DPST-001
- XBP9B-DPUT-001
- XBP9B-DMWTB002
- XBP9B-DMSTB002
- XBP9B-DMUTB002
- XBP9B-DMWTB002A
- XBP9B-DMUTB002A

OEM labeling requirements

WARNING! The Original Equipment Manufacturer (OEM) must ensure that Mexico IFT labeling requirements are met.

The IFETEL number for the XBee 900HP product must be listed either on the end product, on the packaging, in the manual, or in the software with the following phrase:

> “Este equipo contiene el módulo XBee-PROS3B con Número IFETEL: RCPDIXB19-1822”

or

> “Este equipo contiene el módulo con IFT #: RCPDIXB19-1822”

The following paragraph must also be present in the User Manual for the end product:

> “La operación de este equipo está sujeta a las siguientes dos condiciones: (1) es posible que este equipo o dispositivo no cause interferencia perjudicial y (2) este equipo o dispositivo debe aceptar cualquier interferencia, incluyendo la que pueda causar su operación no deseada.”

IDA (Singapore) certification

Labeling

The labeling of equipment is per Info-communications Development Authority of Singapore (Singapore IDA, www.ida.gov.sg/). This license is only for the Digi XB900HP radio and not the final product, so

customers must be aware that they should find a consultant who is aware of the requirements and can guide them through the process of obtaining a license for their product with Singapore IDA. The license number is DA105737.

Frequency band

The available frequency band for Singapore is 920 MHz to 925 MHz. The Digi radio cannot interfere with other services and is not in a protected band.

Antenna gain

The maximum allowed antenna gain is 2.1 dBi, which is the gain of a dipole.

Legacy S3B hardware certifications

Agency certifications - United States

United States (FCC)

The XBee-PRO XSC RF Modules comply with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

To fulfill FCC certification requirements, the OEM must comply with the following regulations:

- The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.
- XBee/XBee-PRO RF modules may only be used with antennas that have been tested and approved for use with this module; refer to FCC-approved antennas (2.4 GHz).

OEM labeling requirements

WARNING! As an Original Equipment Manufacturer (OEM) you must ensure that FCC labeling requirements are met. You must include a clearly visible label on the outside of the final product enclosure that displays the following content:

XBee PRO S3

Note The S3 hardware variant is a legacy design that is obsolete. New and old designs should use the S3B hardware variant.

Required FCC Label for OEM products containing the XBee-PRO XSC RF Module:

Contains FCC ID: MCQ-XBEEXSC

The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: *(i.)* this device may not cause harmful interference and *(ii.)* this device must accept any interference received, including interference that may cause undesired operation.

Or

XBee PRO S3B

Contains FCC ID: MCQ-XBPS3B

The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: *(i.)* this device may not cause harmful interference and *(ii.)* this device must accept any interference received, including interference that may cause undesired operation.

FCC notices

IMPORTANT: The XBee-PRO XSC OEM RF Module has been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

IMPORTANT: OEMs must test final product to comply with unintentional radiators (FCC section 15.107 and 15.109) before declaring compliance of their final product to Part 15 of the FCC rules.

IMPORTANT: The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, please take note of the following instructions the device must undergo SAR testing.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect equipment and receiver to outlets on different circuits.
- Consult the dealer or an experienced radio/TV technician for help.

Limited modular approval

This is an RF module approved for Limited Modular use operating as a mobile transmitting device with respect to section 2.1091 and is limited to OEM installation for Mobile and Fixed applications only. During final installation, end-users are prohibited from access to any programming parameters. Professional installation adjustment is required for setting module power and antenna gain to meet EIRP compliance for high gain antenna(s).

Final antenna installation and operating configurations of this transmitter including antenna gain and cable loss must not exceed the EIRP of the configuration used for calculating MPE. Grantee (Digi) must coordinate with OEM integrators to ensure the end-users and installers of products operating with the module are provided with operating instructions to satisfy RF exposure requirements.

The FCC grant is valid only when the device is sold to OEM integrators. Integrators are instructed to ensure the end-user has no manual instructions to remove, adjust or install the device.

FCC-approved antennas

CAUTION! This device has been tested with Reverse Polarity SMA connectors with the antennas listed in the tables of this section. When integrated into OEM products, fixed antennas require installation preventing end-users from replacing them with non-approved antennas. Antennas not listed in the tables must be tested to comply with FCC Section 15.203 (unique antenna connectors) and Section 15.247 (emissions).

Fixed base station and mobile applications

Digi RF Modules are pre-FCC approved for use in fixed base station and mobile applications. When the antenna is mounted at least 20cm (8") from nearby persons, the application is considered a mobile application.

Portable applications and SAR testing

If the module will be used at distances closer than 20cm to all persons, the device may be required to undergo SAR testing. Co-location with other transmitting antennas closer than 20cm should be avoided.

RF exposure statement

You must include the following Caution statement in OEM product manuals.

CAUTION! This equipment is approved only for mobile and base station transmitting devices. Antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

ISED (Innovation, Science and Economic Development Canada)

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Labeling requirements

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product enclosure must display the following text.

Contains IC: 1846A-XB900HP

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B-Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Contains IC: 1846A-XBEEXSC or Contains IC: 1846A-XBPS3B

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B-Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Antenna options: 900 MHz antenna listings

The antennas in the tables below have been approved for use with this module. Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
Omni-directional antennas					
A09-F0	Fiberglass Base Station	RPN	0 dBi	Fixed	0dB
A09-F1	Fiberglass Base Station	RPN	1.0 dBi	Fixed	0dB
A09-F2	Fiberglass Base Station	RPN	2.1 dBi	Fixed	0dB
A09-F3	Fiberglass Base Station	RPN	3.1 dBi	Fixed	0dB
A09-F4	Fiberglass Base Station	RPN	4.1 dBi	Fixed	0dB
A09-F5	Fiberglass Base Station	RPN	5.1 dBi	Fixed	0dB
A09-F6	Fiberglass Base Station	RPN	6.1 dBi	Fixed	0dB
A09-F7	Fiberglass Base Station	RPN	7.1 dBi	Fixed	0dB
A09-F8	Fiberglass Base Station	RPN	8.1 dBi	Fixed	0dB
A09-F9	Base Station	RPSMAF	9.2 dBi	Fixed	0dB
A09-W7	Wire Base Station	RPN	7.1 dBi	Fixed	0dB
A09-F0	Fiberglass Base Station	RPSMA	0 dBi	Fixed	0dB
A09-F1	Fiberglass Base Station	RPSMA	1.0 dBi	Fixed	0dB
A09-F2	Fiberglass Base Station	RPSMA	2.1 dBi	Fixed	0dB

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
A09-F3	Fiberglass Base Station	RPSMA	3.1 dBi	Fixed	0dB
A09-F4	Fiberglass Base Station	RPSMA	4.1 dBi	Fixed	0dB
A09-F5	Fiberglass Base Station	RPSMA	5.1 dBi	Fixed	0dB
A09-F6	Fiberglass Base Station	RPSMA	6.1 dBi	Fixed	0dB
A09-F7	Fiberglass Base Station	RPSMA	7.1 dBi	Fixed	0dB
A09-F8	Fiberglass Base Station	RPSMA	8.1 dBi	Fixed	0dB
A09-M7	Base Station	RPSMA	7.2 dBi	Fixed	0dB
A09-W7SM	Wire Base Station	RPSMA	7.2 dBi	Fixed	0dB
A09-F0TM	Fiberglass Base Station	RPTNC	0 dBi	Fixed	0dB
A09-F1TM	Fiberglass Base Station	RPTNC	1.0 dBi	Fixed	0dB
A09-F2TM	Fiberglass Base Station	RPTNC	2.1 dBi	Fixed	0dB
A09-F3TM	Fiberglass Base Station	RPTNC	3.1 dBi	Fixed	0dB
A09-F4TM	Fiberglass Base Station	RPTNC	4.1 dBi	Fixed	0dB
A09-F5TM	Fiberglass Base Station	RPTNC	5.1 dBi	Fixed	0dB
A09-F6TM	Fiberglass Base Station	RPTNC	6.1 dBi	Fixed	0dB
A09-F7TM	Fiberglass Base Station	RPTNC	7.1 dBi	Fixed	0dB
A09-F8TM	Fiberglass Base Station	RPTNC	8.1 dBi	Fixed	0dB
A09-HSM	Wire Base Station	RPTNC	7.1 dBi	Fixed/Mobile	0dB
A09 HSM-7	Straight half-wave	RPSMA	3.0 dbi	Fixed/Mobile	0dB
A09-HASM-675	Articulated half-wave	RPSMA	2-1 dBi	Fixed/Mobile	0dB

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
A09-HABMM-P61	Articulated half-wave w/6" pigtail	MMCX	2-1 dBi	Fixed/Mobile	0dB
A09-HABMM-6-P61	Articulated half-waAve w/6" pigtail	MMCX	2-1 dBi	Fixed/Mobile	0dB
A09-HBMM-P61	Straight half-wave w/6" pigtail	MMCX	2-1 dBi	Fixed/Mobile	0dB
A09-HRSM	Right angle half-wave	RPSMA	2-1 dBi	Fixed	0dB
A09-HASM-7*	Articulated half-wave	RPSMA	2-1 dBi	Fixed	0dB
A09-HG	Glass mounted half-wave	RPSMA	2-1 dBi	Fixed	0dB
A09-HATM	Articulated half-wave	RPTNC	2-1 dBi	Fixed	0dB
A09-H	Half-wave dipole	RPSMA	2-1 dBi	Fixed	0dB
A09-HBMMP61	1/2 wave antenna	MMCX	2-1 dBi	Mobile	0dB
A09-QBMMP61	1/4 wave antenna	MMCX	1.9 dBi	Mobile	0dB
A09-QI	1/4 wave integrated wire antenna	Integrated	1.9 dBi	Mobile	0dB
29000187	Helical	Integrated	-2.0 dBi	Fixed/Mobile	0dB
A09-QBMM-P61	Quarter-wave w/6" pigtail	MMCX	1.9 dBi	Fixed/Mobile	0dB
A09-QHRN	Miniature helical right angle solder	Permanent	-1 dBi	Fixed/Mobile	0dB
A09-QHSN	Miniature helical right angle solder	Permanent	-1 dBi	Fixed/Mobile	0dB
A09-QHSM-2	2" Straight	RPSMA	1.9 dBi	Fixed/Mobile	0dB
A09-QHRSM-2	2" Right angle	RPSMA	1.9 dBi	Fixed/Mobile	0dB
A09-QHRSM-170	1.7" Right angle	RPSMA	1.9 dBi	Fixed/Mobile	0dB
A09-QRSM-380	3.8" Right angle	RPSMA	1.9 dBi	Fixed/Mobile	0dB
A09-QAPM-520	5.2" Articulated screw mount	Permanent	1.9 dBi	Fixed/Mobile	0dB

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
A09-QSPM-3	3" Straight screw mount	Permanent	1.9 dBi	Fixed/Mobile	0dB
A09-QAPM-3	3" Articulated screw mount	Permanent	1.9 dBi	Fixed/Mobile	0dB
A09-QAPM-3H	3" Articulated screw mount	Permanent	1.9 dBi	Fixed/Mobile	0dB
A09-DPSM-P12F	omni directional permanent mount w/12 ft pigtail	RPSMA	3.0 dBi	Fixed	0dB
A09-D3NF-P12F	omni directional magnetic mount w/ 12ft pigtail	RPN	3.0 dBi	Fixed	0dB
A09-D3SM-P12F	omni directional w/ 12ft pigtail	RPSMA	3.0 dBi	Fixed	0dB
A09-D3PNF	omni directional permanent mount	RPN	3.0 dBi	Fixed	0dB
A09-D3TM-P12F	omni directional w/ 12ft pigtail	RPTNC	3.0 dBi	Fixed	0dB
A09-D3PTM	omni directional permanent mount	RPTNC	3.0 dBi	Fixed	0dB
A09-M0SM	Mag mount	RPSMA	0 dBi	Fixed	0dB
A09-M2SM	Mag mount	RPSMA	2.1 dBi	Fixed	0dB
A09-M3SM	Mag mount	RPSMA	3.1 dBi	Fixed	0dB
A09-M5SM	Mag mount	RPSMA	5.1 dBi	Fixed	0dB
A09-M7SM	Mag mount	RPSMA	7.1 dBi	Fixed	0dB
A09-M8SM	Mag mount	RPSMA	8.1 dBi	Fixed	0dB
A09-M0TM	Mag mount	RPTNC	0 dBi	Fixed	0dB
A09-M2TM	Mag mount	RPTNC	2.1 dBi	Fixed	0dB
A09-M3TM	Mag mount	RPTNC	3.1 dBi	Fixed	0dB
A09-M5TM	Mag mount	RPTNC	5.1 dBi	Fixed	0dB

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
A09-M7TM	Mag mount	RPTNC	7.1 dBi	Fixed	0dB
A09-M8TM	Mag mount	RPTNC	8.1 dBi	Fixed	0dB
Yagi antennas					
A09-Y6	2 Element Yagi	RPN	6.1 dBi	Fixed/Mobile	0dB
A09-Y7	3 Element Yagi	RPN	7.1 dBi	Fixed/Mobile	0dB
A09-Y8	4 Element Yagi	RPN	8.1 dBi	Fixed/Mobile	0dB
A09-9	4 Element Yagi	RPN	9.1 dBi	Fixed/Mobile	0dB
A09-Y10	5 Element Yagi	RPN	10.1 dBi	Fixed/Mobile	0dB
A09-Y11	6 Element Yagi	RPN	11.1 dBi	Fixed/Mobile	0dB
A09-Y12	7 Element Yagi	RPN	12.1 dBi	Fixed/Mobile	0dB
A09-Y13	9 Element Yagi	RPN	13.1 dBi	Fixed/Mobile	0dB
A09-Y14	10 Element Yagi	RPN	14.1 dBi	Fixed/Mobile	0dB
A09-Y14	12 Element Yagi	RPN	14.1 dBi	Fixed/Mobile	0dB
A09-Y15	13 Element Yagi	RPN	15.1 dBi	Fixed/Mobile	0dB
A09-Y15	15 Element Yagi	RPN	15.1 dBi	Fixed/Mobile	0dB

Part number	Type	Connector	Gain	Application	Cable Loss or power reduction for S3B device
A09-Y6TM	2 Element Yagi	RPTNC	6.1 dBi	Fixed/Mobile	0dB
A09-Y7TM	3 Element Yagi	RPTNC	7.1 dBi	Fixed/Mobile	0dB
A09-Y8TM	4 Element Yagi	RPTNC	8.1 dBi	Fixed/Mobile	0dB
A09-Y9TM	4 Element Yagi	RPTNC	9.1 dBi	Fixed/Mobile	0dB
A09-Y10TM	5 Element Yagi	RPTNC	10.1 dBi	Fixed/Mobile	0dB
A09-Y11TM	6 Element Yagi	RPTNC	11.1 dBi	Fixed/Mobile	0dB
A09-Y12TM	7 Element Yagi	RPTNC	12.1 dBi	Fixed/Mobile	0dB
A09-Y13TM	9 Element Yagi	RPTNC	13.1 dBi	Fixed/Mobile	0dB
A09-Y14TM	10 Element Yagi	RPTNC	14.1 dBi	Fixed/Mobile	0dB
A09-Y14TM	12 Element Yagi	RPTNC	14.1 dBi	Fixed/Mobile	0dB
A09-Y15TM	13 Element Yagi	RPTNC	15.1 dBi	Fixed/Mobile	0dB
A09-Y15TM	15 Element Yagi	RPTNC	15.1 dBi	Fixed/Mobile	0dB

Transmitters with detachable antennas

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the table above with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Detachable antenna

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peutfonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvépour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillageradioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne etson gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépassepas l'intensité nécessaire àl'établissement d'une communication satisfaisante.

Brazil ANATEL