# XBee/XBee-PRO DigiMesh 2.4

## Radio Frequency (RF) Module

---

## User Guide

## Revision history—90000991

| Revision | Date | Description |
|---|---|---|
| R | November 2015 | Revised the document as part of moving to online help. |
| S | January 2016 | Updated several AT commands. |
| T | February 2016 | Editorial revision to AT commands. |
| U | June 2016 | Removed the **1S** command. Fixed an error in the 0x90 frame table. Clarified the routing table size. |

## 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

## Send comments

**Documentation feedback**: To provide feedback on this document, send your comments to techcomm@digi.com.

## Customer support

**Digi Technical Support**: Digi offers multiple technical support plans and service packages to help our customers get the most out of their Digi product. For information on Technical Support plans and pricing, contact us at +1 952.912.3444 or visit us at www.digi.com/support.

Support portal login: www.digi.com/support/eservice

# Contents

## Configure the XBee/XBee-PRO DigiMesh 2.4



## Serial communication



## Work with networked devices



## Network configurations

## Sleep modes



## AT commands

## Operate in API mode

## Certifications

# XBee/XBee-PRO DigiMesh 2.4 RF Module User Guide

The XBee/XBee-PRO DigiMesh 2.4 supports the unique needs of low-cost, low-power, wireless sensor networks. The devices require minimal power and provide reliable data delivery between remote devices. The devices operate within the ISM 2.4 MHz frequency band.

These devices support routing table sizes of 32 nodes. Networks larger than this send a route discovery before each transmission. For larger networks this can be bandwidth expensive, so we offer RF optimization services to help you properly configure a network.

# Worldwide acceptance

We manufacture and certify the XBee/XBee-PRO DigiMesh 2.4s to certain industry standards. These standards enable you to understand what the devices can do and where you can use them.

The Federal Communications Commission (FCC) approves the devices for use in the United States. For details, see Agency certifications - United States. If a system contains XBee/XBee-PRO DigiMesh 2.4s, the system inherits Digi's certifications.

The devices are certified to operate in the industrial, scientific, and medical (ISM) 2.4 GHz frequency band.

We manufacture the devices under International Organization for Standardization (ISO) 9001:2000 registered standards.

We optimize the devices for use in the United States and Canada. For a complete list of agency approvals, see Certifications.

# Antenna options

Digi devices come in a variety of antenna options. The options that allow you to connect an external antenna are reverse polarity standard subminiature assembly (RPSMA) and U.FL. Typically, you make connections with either a dipole antenna with a U.FL connection, or a U.FL to RPSMA antenna adapter cable.

RPSMA is the more traditional antenna connector, however, if the device is going to be inside of an enclosure, you would need to locate the device near the edge of the enclosure to allow the connector to pass through an available bulkhead. The RPSMA connector uses the same body as a regular SMA connector, but changes the gender of the center conductor. The female RPSMA actually has a male center conductor. We equip the XBee devices with an RPSMA female plug, while the antenna is an RPSMA male jack.

The U.FL connection allows for connectivity to an external antenna. U.FL is a small antenna connection for use with a pigtail connector. A pigtail is a short (typically 4 - 6 in) cable that either terminates into an external antenna port such as an RPSMA, N or TNC connection or an antenna. You would attach the RPSMA connector to a bulkhead. These options allow you to mount the device away from the edge of the enclosure in your product and centrally locate the radio. U.FL is fragile and is not designed for multiple insertions without a specialized tool to separate the pigtail without damaging the connector; for more information, see http://www.digikey.com/product-detail/en/U.FL-LP(V)-N-2/HR5017-ND/513034.

The other available antenna options are printed circuit board (PCB) and wire antennas. We form the PCB antenna directly on the device with conductive traces. A PCB antenna performs about the same as a wire antenna.

An integrated wire antenna consists of a small wire (about 80 mm) sticking up perpendicular to the PCB. It uses a 1/4-wave wire that we solder directly to the PCB of the OEM device.

All Digi devices with antenna connectors have less than 0.1 dB loss; we do not consider one to be "better" than the other in terms of reliability or insertion loss. RF device specifications such as -110 dBm receiver sensitivity, +3 0 dBm TX power, and so forth, already include any insertion loss due to the soldered RF connector.

# Part numbers

The part numbers for these devices are available at the following link:

www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4#partnumbers

# Technical specifications

The following tables provide the device's technical specifications.

# Performance specifications

This table describes the performance specifications for the devices.

| Specification | XBee | XBee-PRO |
|---|---|---|
| Indoor / urban range | Up to 100 ft (30 m) | Up to 300 ft (90 m) standard or up to 200 ft (60 m) international variant |
| Outdoor RF line of sight range | Up to 300 ft (90 m) | Up to 1 mile (1.5 km), with a 2.0 dB dipole antenna. Up to 6 miles (10 km) with a high gain antenna. |
| Transmit power output | 1 mW (0 dBm) | 63 mW (18 dBm) standard or 10 mW (10 dBm) for the international variant |
| RF data rate | 250 kb/s | 250 kb/s |
| Serial interface data rate (software selectable) | 1200 bps - 250 kb/s (devices also support non-standard baud rates) | 1200 bps - 250 kb/s (devices also support non-standard baud rates) |
| Receiver sensitivity | -92 dBm (1% packet error rate) | -100 dBm (1% packet error rate) |

# Power requirements

The following table describes the power requirements for the devices.

| Specification | XBee | XBee-PRO |
|---|---|---|
| Supply voltage | 2.8 - 3.4 VDC | 2.8 - 3.4 VDC |
| Transmit current | 45 mA (@ 3.3 V) | 250 mA (@ 3.3 V) (150 mA for the international variant) RPSMA device only: 340 mA (@ 3.3 V) (180 mA for the international variant) |
| Idle / receive current | 50 mA (@ 3.3 V) | 55 mA (@ 3.3 V) |
| Power down current (pin sleep) | <10 µA | <10 µA |
| Power down current (cyclic sleep) | <50 µA | <50 µA |

# General specifications

The following table describes the general specifications for the devices.

| Specification | XBee | XBee-PRO |
|---|---|---|
| Operating frequency band | ISM 2.4 GHz | ISM 2.4 GHz |
| Dimensions | 2.438 cm x 2.761 cm (0.960 in x 1.087 in) | 2.438 cm x 3.294 cm (0.960 in x 1.297 in) |
| Operating temperature | -40 to 85 ℃ (industrial) | -40 to 85 ℃ (industrial) |
| Relative humidity | 0 to 95% non-condensing | 0 to 95% non-condensing |
| Antenna options | 1/4 wave wire antenna, embedded PCB antenna, RPSMA RF connector, U.FL RF connector | 1/4 wave wire antenna, RPSMA RF connector, U.FL RF connector |

## Network and security specifications

The following describes the network and security specifications for the devices.

| Specification | XBee | XBee-PRO |
|---|---|---|
| Supported network topologies | Mesh, point-to-point, point-to-multipoint, peer-to-peer | Mesh, point-to-point, point-to-multipoint, peer-to-peer |
| Number of channels (software selectable) | 16 direct sequence channels | 12 direct sequence channels |
| Addressing options | PAN ID, channel and 64-bit addresses | PAN ID, channel and 64-bit addresses |
| Encryption | 128 bit Advanced Encryption Standard (AES) | 128 bit AES |

## Regulatory approvals

The following table provides the device's regulatory approvals. See Certifications for important information.

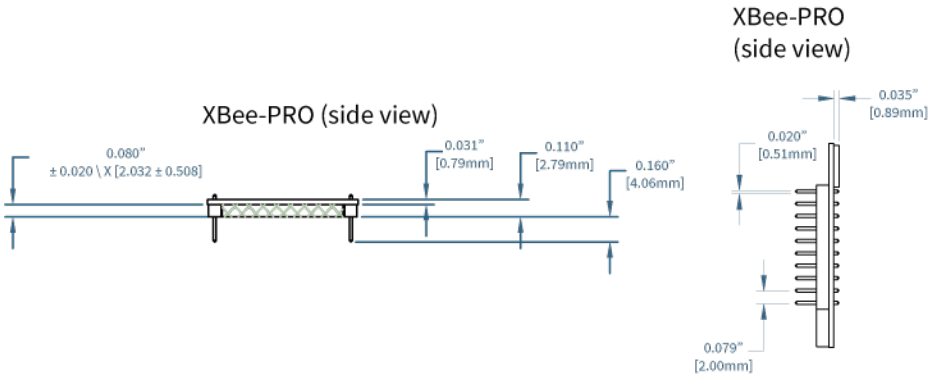
| Specification | XBee | XBee-PRO |
|---|---|---|
| United States (FCC Part 15.247) | OUR-XBEE | OUR-XBEEPRO |
| Industry Canada (IC) | 4214A-XBEE | 4214A-XBEEPRO |
| Europe (CE) | ETSI | ETSI (maximum 10 dBm transmit power output) |
| RoHS | Lead-free and RoHS compliant | Lead-free and RoHS compliant |
| Japan | R201WW07215214 | R201WW08215111 (maximum 10 dBm transmit power output) |

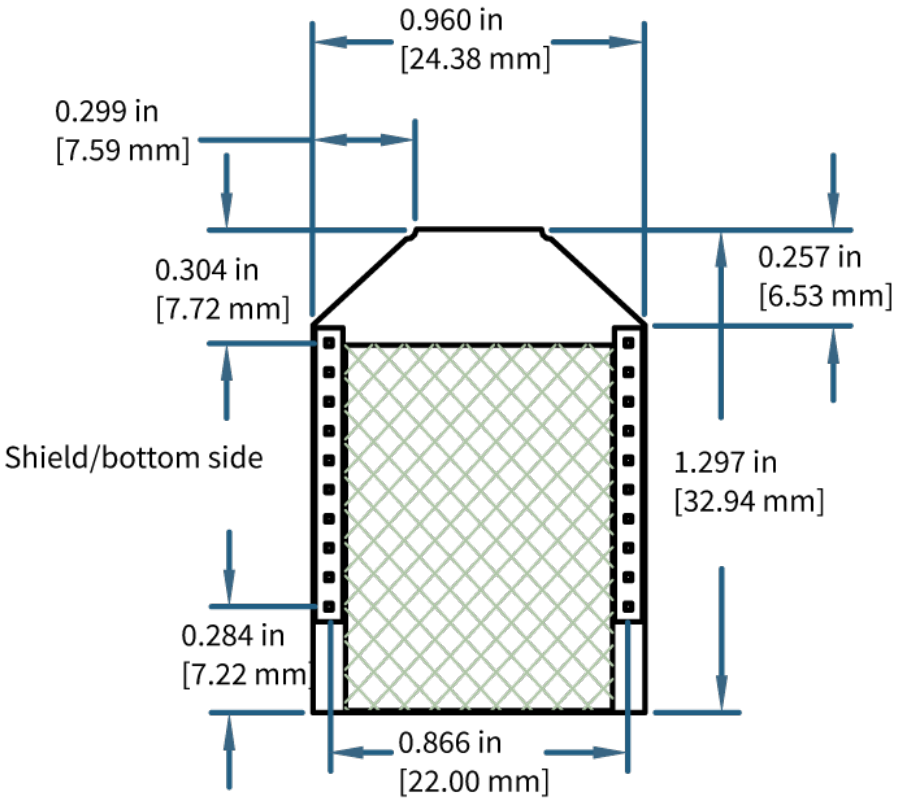
| Specification | XBee | XBee-PRO |
|---|---|---|
| Australia | C -Tick | C -Tick |
| Brazil | ANATEL 0369-15-1209 | ANATEL 0378-15-1209 |
| See Certifications for region-specific certification requirements. | | |

# Hardware

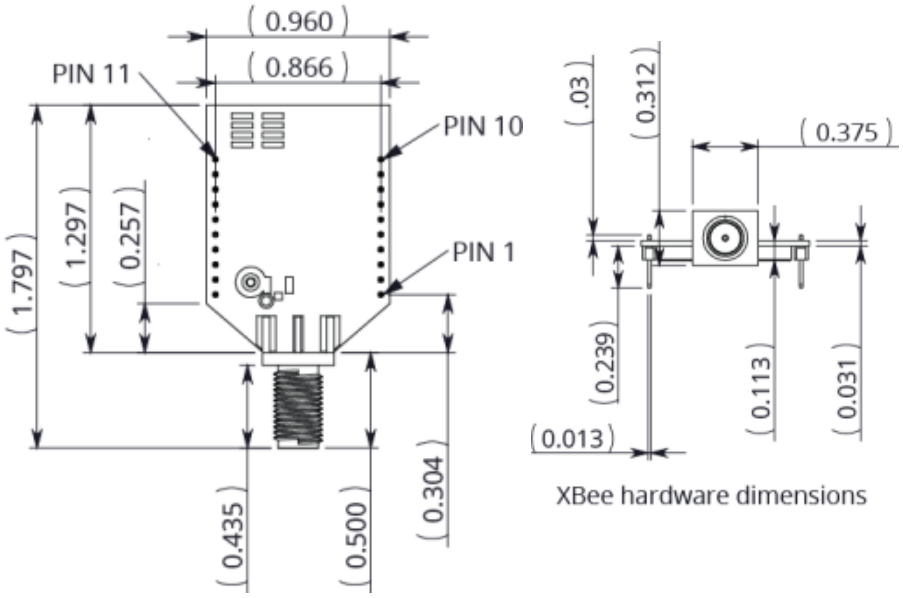
# Mechanical drawings

The following figures show the mechanical drawings for the XBee/XBee-PRO DigiMesh 2.4. The drawings do not show antenna options.

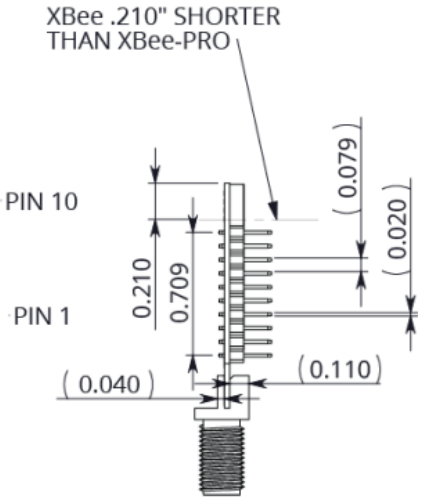



The following drawings show the RPSMA device.




XBee hardware dimensions

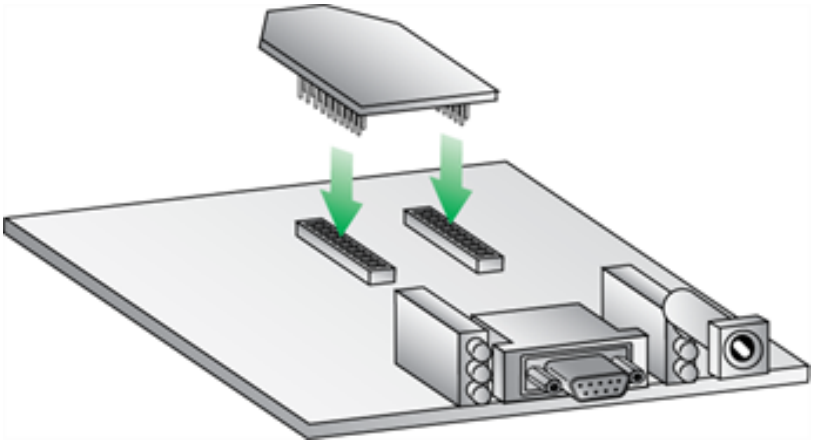



XBee hardware
(side view)

# Mounting considerations

We design the through-hole module to mount into a receptacle so that you do not have to solder the module when you mount it to a board. The development kits may contain RS-232 and USB interface boards that use two 20-pin receptacles to receive modules.

The following illustration shows the module mounting into the receptacle on the RS-232 interface board.

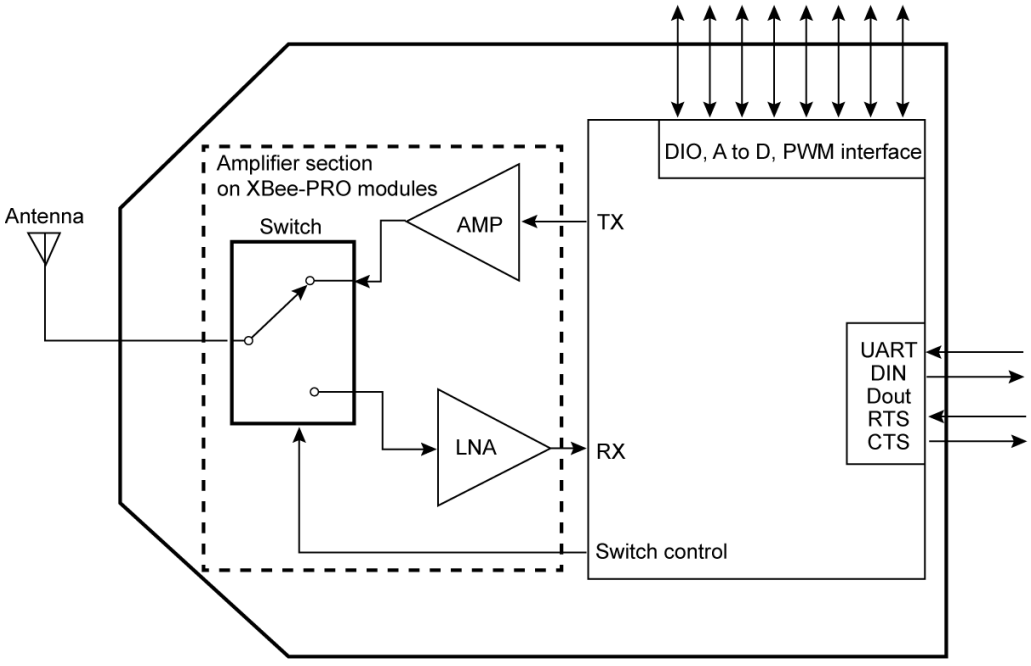


- Through-hole single-row receptacles: Samtec part number: MMS-110-01-L-SV (or equivalent)
- Surface-mount double-row receptacles: Century Interconnect part number: CPRMSL20-D-0-1 (or equivalent)
- Surface-mount single-row receptacles: Samtec part number: SMM-110-02-SM-S

**Note** We recommend that you print an outline of the module on the board to indicate the correct orientation for mounting the module.

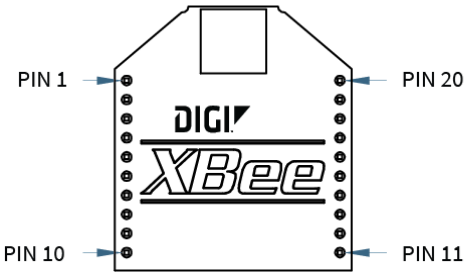
## Hardware diagram

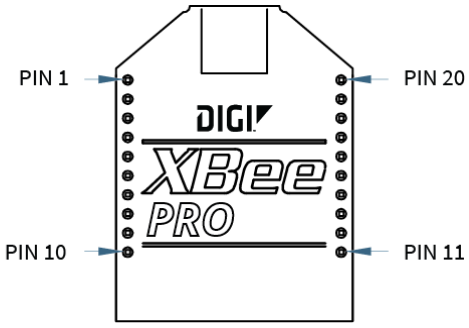
The following diagram shows a simplified view of XBee/XBee-PRO DigiMesh 2.4 hardware.



## Pin signals

The following table shows the pin signals and their descriptions.




| Pin # | Pin name | Direction | Description |
|---|---|---|---|
| 1 | Vcc | - | Power supply |
| 2 | DOUT | Output | UART data out |
| 3 | DIN/CONFIG | Input | UART data in |
| 4 | DIO12 | Either | Digital I/O 12 |
| 5 | RESET | Input/open drain output | Device reset. The reset pulse must be at least 100 µs. Drive this line as an open drain/collector. The device drives this line low when a reset occurs. Never drive this line high. |
| 6 | PWM0/RSSI/DIO10 | Either | PWM output 0 / RX signal strength indicator / Digital I/O |
| 7 | PWM/DIO11 | Either | PWM output 1 / Digital I/O 11 |
| 8 | Reserved | - | Do not connect |
| 9 | $\overline{\text{DTR}}$ / SLEEP_RQ / DIO8 | Either | Pin sleep control line or Digital I/O 8 |
| 10 | GND | - | Ground |
| 11 | AD4/ DIO4 | Either | Analog input 4 or Digital I/O 4 |
| 12 | $\overline{\text{CTS}}$/ DIO7 | Either | Clear-to-send flow control or Digital I/O 7 |
| 13 | ON/SLEEP | Output | Device Status Indicator or Digital I/O 9 |
| 14 | VREF | - | You must connect this line if you want to use analog I/O sampling. Must be between 2.6 V and Vcc. |
| 15 | Associate / DIO5/ AD5 | Either | Associated indicator, Digital I/O 5 |
| 16 | $\overline{\text{RTS}}$/ DIO6 | Either | Request-to-send flow control, Digital I/O 6 |
| 17 | AD3 / DIO3 | Either | Analog input 3 or Digital I/O 3 |
| 18 | AD2 / DIO2 | Either | Analog input 2 or Digital I/O 2 |

| Pin # | Pin name | Direction | Description |
|---|---|---|---|
| 19 | AD1 / DIO1 | Either | Analog input 1 or Digital I/O 1 |
| 20 | AD0 / DIO0 / Commissioning Pushbutton | Either | Analog input 0, Digital I/O 0, or Commissioning Pushbutton |

### *Notes*

The table specifies signal direction with respect to the device.

The device includes a 50 kΩ pull-up resistor attached to RESET.

You can configure several of the input pull-ups using the PR command.

Leave any unused pins disconnected.

## 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** command 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 The Commissioning Pushbutton.

For analog sampling, attach the VREF pin (pin 14) to a voltage reference.

# Design notes

The following guidelines help to ensure a robust design.

## Power supply design

A poor power supply can lead to poor device performance, especially if you do not keep the supply voltage within tolerance or if it is excessively noisy. To help reduce noise, place a 1.0 µF and 8.2 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 100 mV peak to peak.

## Board layout

We design XBee devices 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 devices, 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

Do not place XBee devices with the chip or integrated PCB antenna inside a metal enclosure.

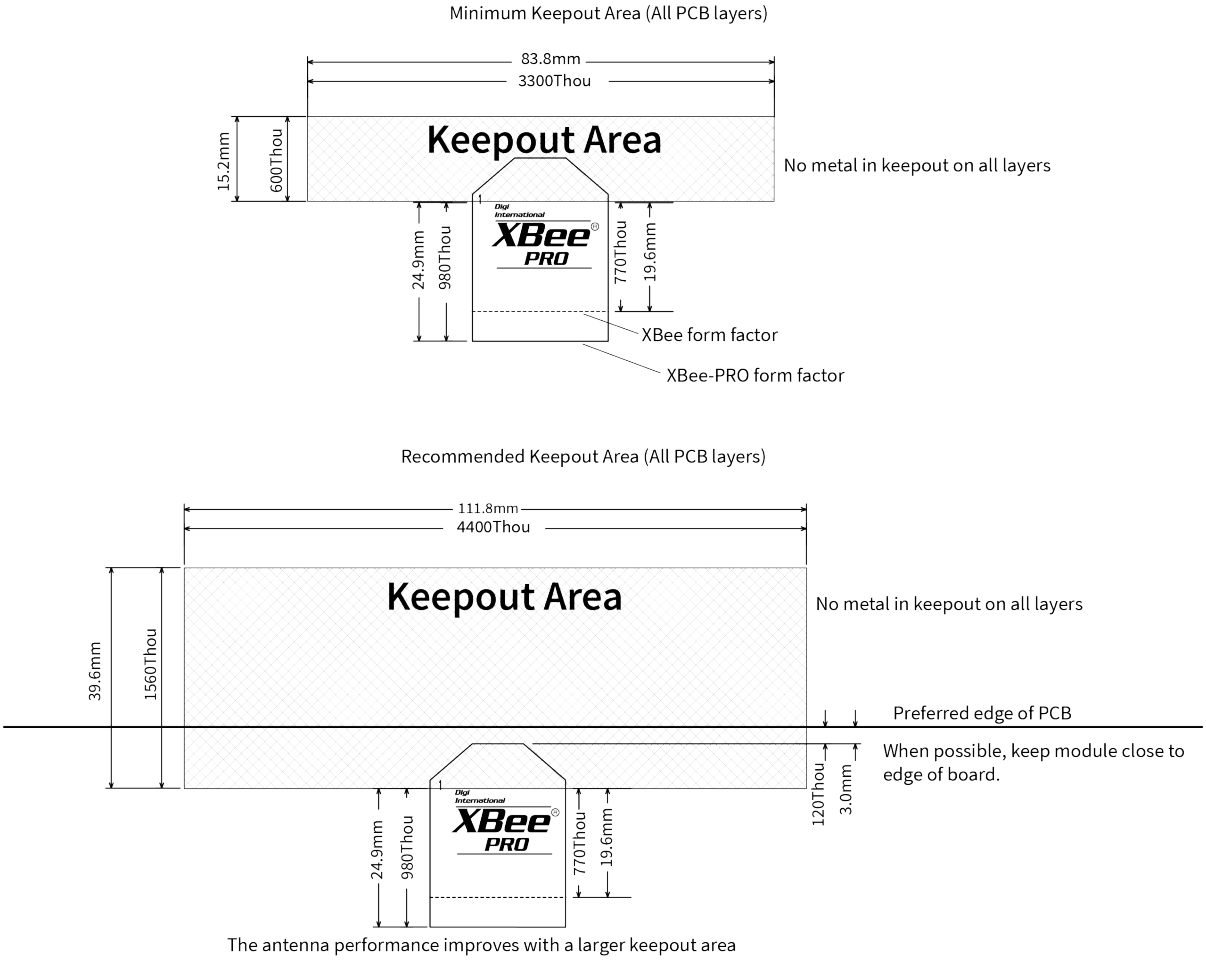
Do not place any ground planes or metal objects above or below the antenna.

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.

## Keepout area

We recommend that you allow a “keepout” area, as shown in the following drawing.

### *Through-hole keepout*




**Notes**

1. We recommend non-metal enclosures. For metal enclosures, use an external antenna.
2. Keep metal chassis or mounting structures in the keepout area at least 2.54 cm (1 in) from the antenna.
3. Maximize the distance between the antenna and metal objects that might be mounted in the keepout area.
4. These keepout area guidelines do not apply for wire whip antennas or external RF connectors. Wire whip antennas radiate best over the center of a ground plane.

# DC characteristics

The following table displays the DC characteristics (VCC = 2.8 - 3.4 VDC).

| Symbols | Parameter | Condition | Min | Typical | Max | Units |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low voltage | All digital inputs | - | - | 0.2[1] VCC | V |
| $V_{IH}$ | Input high voltage | All digital inputs | 0.8[2] VCC | - | - | V |
| $V_{OL}$ | Output low voltage | $I_{OL}$ = 2 mA, VCC >= 3.0 V | - | - | 0.18[3] VCC | V |
| $V_{OH}$ | Output high voltage | $I_{OH}$ = 2 mA, VCC >= 3.0 V | 0.82[4] VCC | - | - | V |
| $II_{IN}$ | Input leakage current | $V_{IN}$ = VCC or GND, all inputs, per pin | - | - | 0.5 | µA |

# ADC operating characteristics

The following table displays the ADC timing and performance characteristics.

| Symbols | Parameter | Condition | Min | Typical | Max | Units |
|---|---|---|---|---|---|---|
| $V_{REFH}$ | VREF-analog-to-digital converter reference range | | 2.08 | - | $V_{DDAD}$ | V |
| $I_{REF}$ | VREF-reference supply current | Enabled | - | 200 | - | µA |
| | | Disabled or sleep mode | - | < 0.01 | 0.02 | µA |
| $V_{INDC}$[5] | Analog input voltage | | $V_{SSAD}$ - 0.3 | - | $V_{SSAD}$ + 0.3 | V |

[1]Maximum electrical operating range, not valid conversion range.

[2]Maximum electrical operating range, not valid conversion range.

[3]Maximum electrical operating range, not valid conversion range.

[4]Maximum electrical operating range, not valid conversion range.

[5]Analog input must be between $V_{REFL}$ and $V_{REFH}$ for valid conversion. Values greater than $V_{REFH}$ will convert to $3FF.

# ADC timing and performance characteristics

The following table displays the ADC timing and performance characteristics.[1]

| Symbols | Parameter | Condition | Min | Typical | Max | Units |
|---|---|---|---|---|---|---|
| $R_{AS}$ | Source impedance at input[2] | | - | - | 10 | kΩ |
| RES | Ideal resolution (1 LSB)[3] | 2.08V > $V_{DDAD}$ > 3.6V | 2.031 | | 3.516 | mV |
| DNL | Differential non-linearity[4] | | - | ±0.5 | ±1.0 | LSB |
| INL | Integral non-linearity[5] | | - | ±0.5 | ±1.0 | LSB |
| $E_{ZS}$ | Zero-scale error[6] | | - | ±0.4 | ±1.0 | LSB |
| $F_{FS}$ | Full-scale error[7] | | - | ±0.4 | ±1.0 | LSB |
| $E_{IL}$ | Input leakage error[8] | | - | ±0.05 | ±5.0 | LSB |
| $E_{TU}$ | Total unadjusted error[9] | | - | ±1.1 | ±2.5 | LSB |

[1]All Accuracy numbers are based on processor and system being in WAIT state (very little activity and no I/O switching) and that adequate low-pass filtering is present on analog input pins (filter with 0.01 µF to 0.1 µF capacitor between analog input and $V_{REFL}$). Failure to observe these guidelines may result in system or microcontroller noise causing accuracy errors which will vary based on board layout and the type and magnitude of the activity. Data transmission and reception during data conversion may cause some degradation of these specifications, depending on the number and timing of packets. It is advisable to test the ADCs in your installation if best accuracy is required.

[2]RAS is the real portion of the impedance of the network driving the analog input pin. Values greater than this amount may not fully charge the input circuitry of the ATD resulting in accuracy error.

[3]The resolution is the ideal step size or 1LSB = $(V_{REFH}–V_{REFL})/1024$.

[4]Differential non-linearity is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to and from the current code.

[5]Integral non-linearity is the difference between the transition voltage to the current code and the adjusted ideal transition voltage for the current code. The adjusted ideal transition voltage is (Current Code.1/2)*(1/ $((V_{REFH}+E_{FS}).(V_{REFL}+E_{ZS}))$).

[6]Zero-scale error is the difference between the transition to the first valid code and the ideal transition to that code. The Ideal transition voltage to a given code is $(Code.1/2)*(1/(V_{REFH}.V_{REFL}))$.

[7]Full-scale error is the difference between the transition to the last valid code and the ideal transition to that code. The ideal transition voltage to a given code is $(Code.1/2)*(1/(V_{REFH}.V_{REFL}))$.

[8]Input leakage error is error due to input leakage across the real portion of the impedance of the network driving the analog pin. Reducing the impedance of the network reduces this error.

[9]Total unadjusted error is the difference between the transition voltage to the current code and the ideal straight-line transfer function. This measure of error includes inherent quantization error (1/2 LSB) and circuit error (differential, integral, zero-scale, and full-scale) error. The specified value of $E_{TU}$ assumes zero $E_{IL}$ (no leakage or zero real source impedance).

# Modes

The XBee/XBee-PRO DigiMesh 2.4 is in Receive Mode when it is not transmitting data. The device shifts into the other modes of operation under the following conditions:

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

# Transparent and API operating 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.

## 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 the AT Command interface.

## API operating mode

API operating mode is an alternative to Transparent mode. 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 originated from. The device communicates UART data in packets, also known as API frames. This mode allows for structured communications with serial devices. 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.

For more information, see API frame specifications.

## Comparing Transparent and API modes

The XBee/XBee-PRO DigiMesh 2.4 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 provides a comparison of the two modes.

| Transparent operating mode | API operating mode |
|---|---|
| When to use:<br>■ The conditions for using API mode do not apply. | When to use:<br>■ The device sends wireless data to multiple destinations.<br>■ The device configures remote devices in the network.<br>■ The device receives wireless data packets from multiple XBee devices, and the application needs to identify which devices send each packet.<br>■ The device receives I/O samples from remote XBee devices. |

| Transparent operating mode | API operating mode |
|---|---|
| Advantages:<br>■ Provides a simple interface.<br>■ It is easy for an application to support; what you send is exactly what other modules get, and vice versa.<br>■ Works very well for two-way communication between XBee devices. | Advantages:<br>■ You can set or read the configuration of remote XBee devices in the network.<br>■ You can transmit data to one or multiple destinations; this is much faster than Transparent mode where the configuration must be updated to establish a new destination.<br>■ Received data includes the sender's address.<br>■ Received data includes transmission details and reasons for success or failure.<br>■ This mode has several advanced features, such as advanced networking diagnostics, and firmware upgrades. |
| Disadvantages:<br>■ You cannot set or read the configuration of remote XBee devices in the network.<br>■ You must first update the configuration to establish a new destination and transmit data.<br>■ You cannot identify the source of received data, as it does not include the sender's address.<br>■ Received data does not include transmission details or the reasons for success or failure.<br>■ This mode does not offer the advanced features of API mode, including advanced networking diagnostics, and firmware upgrades. | Disadvantages:<br>■ The interface is more complex; data is structured in packets with a specific format.<br>■ This mode is more difficult to support; transmissions are structured in packets that need to be parsed (to get data) or created (to transmit data).<br>■ Sent data and received data are not identical; received packets include some control data and XTend vB information. |

# Additional modes

In addition to the serial communication modes, several modes apply to how devices communicate with each other.

## Command mode

Command mode is a state in which the firmware interprets incoming characters as commands. Command mode allows you to modify the device's firmware using parameters you can set using AT commands. When you want to read or set any setting of the device, you have to send it an AT command. Every AT command starts with the letters "AT", followed by the two characters that identify the command that is being issued and then by some optional configuration values. For more details, see Enter Command mode.

## Idle mode

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

## 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 modes

Sleep modes allows 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 module enters sleep mode upon pin transition, and cyclic sleep, where the module sleeps for a fixed time. While asleep, nodes cannot receive RF messages or read commands from the UART port.

The sleep modes are:

- Normal mode. Normal mode is the default for a newly powered-on node. In this mode, a node does not sleep. Normal mode nodes should be mains-powered.
- Asynchronous Pin Sleep mode. This mode allows the device to sleep and wake according to the state of the Sleep_RQ pin (pin 9).
- Asynchronous Cyclic Sleep Mode. This mode allows the device to sleep for a specified time and wake for a short time to poll.
- Asynchronous Cyclic Sleep with Pin Wake Up mode. In this mode you can wake the device up prematurely using the Sleep_RQ pin.

- Synchronous Sleep Support mode. A node in this 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 using a sync message.
- Synchronous Cyclic Sleep mode. A node 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.

### Transmit mode

When the device receives serial data and is ready to packetize it, it exits Idle mode and attempts to transmit the serial data.

## Enter Command mode

To get a device to switch into this mode, you must issue the following sequence: **GT** + **CC**(+++) + **GT**. When the device sees a full second of silence in the data stream (the guard time) followed by the string +++ (without Enter or Return) and another full second of silence, it knows to stop sending data through and start accepting commands locally.

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

Once you send the Command mode sequence, the device sends **OK** out the UART pin. The device may delay sending the **OK** if it has not transmitted all of the serial data it received.

Once 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 Receive mode.

You can customize the guard times and timeout in the device's configuration settings. For information on how to do this, see CC (Command Character), CT (Command Mode Timeout) and GT (Guard Times).

### Troubleshooting

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

## 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 the device's destination address (Low) to 0x1F.

***Respond to AT commands***

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

# Exit Command mode

1. Send the **CN** (Exit Command Mode) command 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.

# Configure the XBee/XBee-PRO DigiMesh 2.4

# Software libraries

One way to communicate with the XBee device is by using a software library. The libraries available for use with the devices include:

- XBee Java library
- XBee ANSI C 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 ANSI C Library project is a collection of portable ANSI C code for communicating with the devices in API mode.

# 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.

# Serial communication

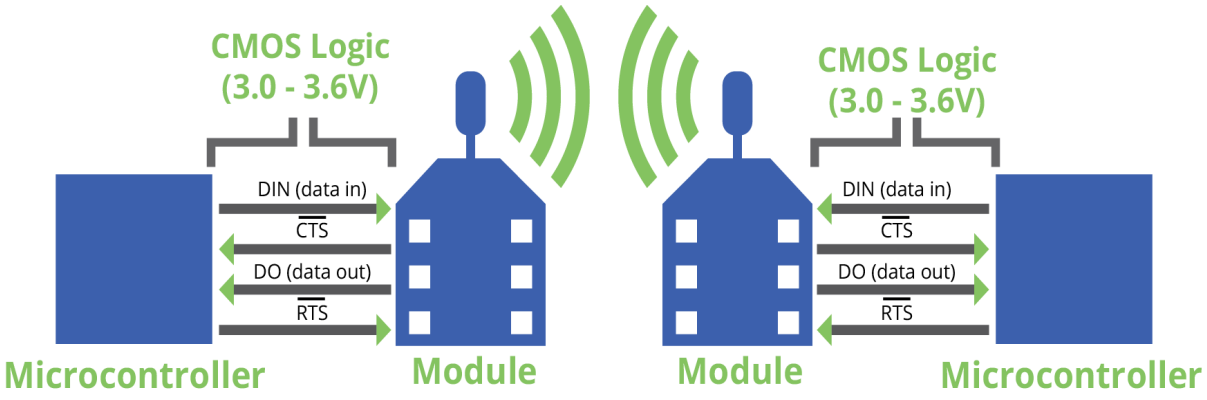
# Serial interface

The XBee/XBee-PRO DigiMesh 2.4 provides a serial interface to an RF link. The XBee/XBee-PRO DigiMesh 2.4 converts serial data to RF data and sends that data to any device in an RF network. The device can communicate through its serial port with any logic and voltage compatible universal asynchronous receiver/transmitter (UART), or through a level translator to any serial device.

# UART data flow

The XBee/XBee-PRO DigiMesh 2.4 device's UART performs tasks such as checking timing and parity, which is required for data communications.

Devices that have a UART interface connect directly to the pins of the XBee/XBee-PRO DigiMesh 2.4 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.

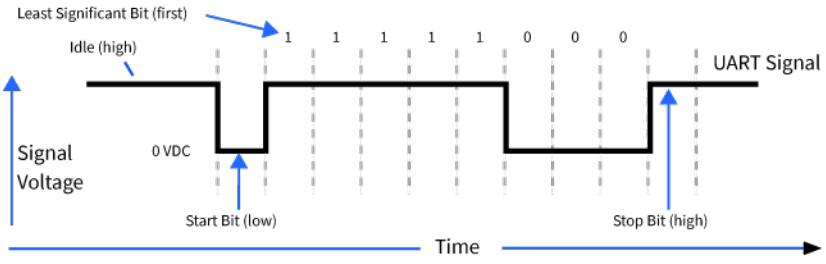



## Serial data

A device sends data to the device's UART through pin 3 (DIN) as an asynchronous serial signal. When the device is not transmitting data, the signal idles high.

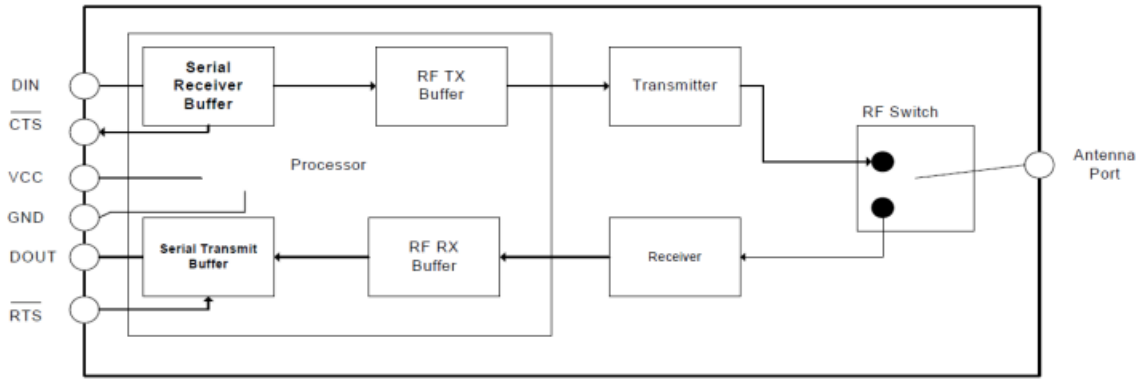
For serial communication to occur, you must configure the UART of both devices (the microcontroller and the RF module) with compatible settings for the baud rate, parity, start bits, stop bits, and data bits.

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.



# Serial buffers

The XBee/XBee-PRO DigiMesh 2.4 maintain buffers to collect serial and RF data that it receives. The serial receive buffer collects incoming serial characters and holds them until the device can process them. The serial transmit buffer collects the data it receives via the RF link until it transmits that data out the serial port. The following figure shows the process of device buffers collecting received serial data.




## Serial buffer issues

There are potential overflow and dropped packet issues, which the following section describes.

### *Serial receive buffer*

Under certain conditions, the device may not be able to process data in the serial receive buffer immediately. If a host sends large amounts of serial data to the device, the device may require CTS flow control to avoid overflowing the serial receive buffer.

Cases in which the serial receive buffer may become full and possibly overflow:

1. If the device receives a continuous stream of RF data, it does not transmit the data in the serial receive buffer until the device stops receiving RF data.
2. For mesh networking firmware, if the device transmits an RF data packet, the device may need to discover the destination address or establish a route to the destination. After transmitting the data, the device may need to retransmit the data if it does not receive an acknowledgment, or if the transmission is a broadcast. These issues could delay the processing of data in the serial receive buffer.

### *Serial transmit buffer*

If the serial transmit buffer becomes full enough that all of the data in a received RF packet will not fit in the serial transmit buffer, it drops the entire RF data packet.

Cases in which the serial transmit buffer may become full, resulting in dropped RF packets:

1. If the RF data rate is set higher than the interface data rate of the device, the device may receive data faster than it can send the data to the host. Even occasional transmissions from a large number of devices can quickly accumulate and overflow the transmit buffer.
2. If the host does not allow the device to transmit data out from the serial transmit buffer due to being held off by hardware flow control.

# Serial flow control

The RTS and CTS device pins provide RTS and/or CTS flow control. CTS flow control signals the host to stop sending serial data to the device. RTS flow control lets the host signal the device so it will not send the data in the serial transmit buffer out the UART. Use the **D6** and **D7** commands to enable RTS and CTS flow control.

## CTS flow control

CTS flow control is enabled by default; you can disable it with the **D7** command. When the serial receive buffer fills with the number of bytes specified by the **FT** parameter, the device de-asserts CTS (sets it high) to signal the host device to stop sending serial data. The device re-asserts CTS when less than FT-16 bytes are in the UART receive buffer; for more information, see FT (Flow Control Threshold).

## 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.

# Work with networked devices

# 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.

## Local configuration

You can configure devices locally using serial commands in Transparent or API mode, or remotely using remote API commands. Devices that are in API mode can send configuration commands to set or read the configuration settings of any device in the network.

## Remote configuration

When you do not have access to the device's serial port, you can use a separate device in API mode to remotely configure it. To remotely configure devices, use the following steps.

### *Send a remote command*

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

1. The 64-bit address of the remote device.
2. The correct command options value.
3. Optionally, the command and parameter data.
4. If you want a command response, set the Frame ID field to a non-zero value.

The firmware only supports unicasts of remote commands. You cannot broadcast remote commands.

XCTU has a Frames Generator tool that can assist you with building and sending a remote AT frame; see: http://www.digi.com/resources/documentation/digidocs/90001458-13/default.htm#reference/r_frames_generator_tool.htm

### *Apply changes on remote devices*

When you use remote commands to change the command parameter settings on a remote device, you must apply the parameter changes or they do not take effect. For example, if you change the **BD** parameter, the actual serial interface rate does not change on the remote device until you apply the changes. You can apply the changes using remote commands in one of three ways:

1. Set the apply changes option bit in the API frame.
2. Send an **AC** command to the remote device.
3. Send the **WR** command followed by the **FR** command to the remote device to save the changes and reset the device.

### *Remote command response*

If a local device sends a command request to a remote device, and the API frame ID is non-zero, the remote device sends a remote command response transmission back to the local device.

When the local device receives a remote command response transmission, it sends a remote command response API frame out its UART. The remote command response indicates:

1. The status of the command, which is either success or the reason for failure.
2. In the case of a command query, it includes the register value.

The device that sends a remote command does not receive a remote command response frame if:

1. It could not reach the destination device.
2. You set the frame ID to 0 in the remote command request.

# Establish and maintain network links

## Build aggregate routes

In many applications, many or all of the nodes in the network must 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, you can use the **AG** command to automatically build routes to an aggregate node in a DigiMesh network.

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

Upon deploying a DigiMesh network, you can issue the **AG** command on the desired aggregator node to cause all nodes in the network to build routes to the aggregator node. You can optionally use the **AG** 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; typically you should replacethis value with the 64-bit address of the node sending the **AG** broadcast. However, if you do not want to update the **DH**/**DL** of the device receiving the **AG** broadcast you can use the invalid address of 0xFFFE. The receiving nodes that are configured in API mode output an Aggregator Update API frame (0x8E) if they update their **DH**/**DL** address; for a description of the frame, see Aggregate Addressing Update frame - 0x8E.

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. The devices use this routing information for future DigiMesh unicast transmissions.

## DigiMesh routing examples

### *Example one:*

In a scenario where you deploy a network, and then you want to update the **DH** and **DL** registers of all the devices in the network so that they use the MAC address of the aggregator node, which has the MAC address 0x0013A200 4052C507, you could use the following technique.

1. Deploy all devices in the network with the default **DH**/**DL** of 0xFFFF.
2. Serially, send an ATAGFFFF command to the aggregator node so it sends the broadcast transmission to the rest of the nodes.

All the nodes in the network that receive the **AG** broadcast set their **DH** to 0x0013A200 and their **DL** to 0x4052C507. These nodes automatically build a route to the aggregator node.

### *Example two:*

If you want all of the nodes in the network to build routes to an aggregator node with a MAC address of 0x0013A200 4052C507 without affecting the **DH** and **DL** registers of any nodes in the network:

1. Send the ATAGFFFE command to the aggregator node. This sends an **AG** broadcast to all of the nodes in the network.
2. All of the nodes internally update only their routing table information to contain a route to the aggregator node.
3. None of the nodes update their **DH** and **DL** registers because none of the registers are set to the 0xFFFE address.

## Replace nodes

You can use the **AG** command to update the routing table and **DH**/**DL** registers in the network after you replace a device. To update only the routing table information without affecting the **DH** and **DL** registers, use the process in example two, above.

To update the **DH** and **DL** registers of the network, use example three, below.

### *Example three:*

This example shows how to cause all devices to update their **DH** and **DL** registers to the MAC address of the sending device. In this case, assume you are using a device with a serial number of 0x0013A200 4052C507 as a network aggregator, and the sending device has a MAC address of 0x0013A200 F5E4D3B2 To update the **DH** and **DL** registers to the sending device's MAC address:

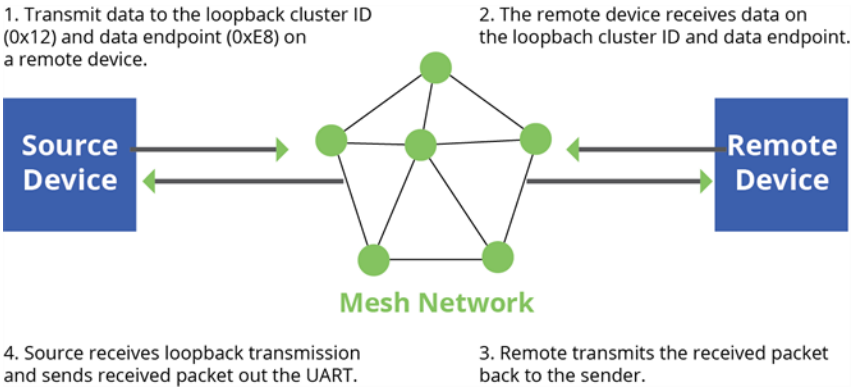
1. Replace the aggregator with 0x0013A200 F5E4D3B2.
2. Send the ATAG0013A200 4052C507 command to the new device.

# Test links in a network

For a network installation to be successful, you must determine where to place individual devices in order to establish reliable links throughout a network.

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

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



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

### *Transparent operating mode configuration (AP = 0)*

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

1. Set the **CI** command to 0x12.
2. Set the **DH** and **DL** commands to the address of the remote device.

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

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

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

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

## Test links between adjacent devices

It often helps to test the quality of a link between two adjacent modules in a network. You can use the Test Link Request Cluster ID to send a number of test packets between any two devices in a network. To clarify the example, we refer to "device A" and "device B" in this section.

To request that device B perform a link test against device A:

1. Use device A in API mode (**AP** = 1) to send an Explicit Addressing Command (0x11) frame to device B.
2. Address the frame to the Test Link Request Cluster ID (0x0014) and destination endpoint: 0xE6.
3. Include a 12-byte payload in the Explicit Addressing Command frame with the following format:

| Number of bytes | Field name | Description |
|---|---|---|
| 8 | Destination address | The address the device uses to test its link. For this example, use the device A address. |
| 2 | Payload size | The size of the test packet. Use the **NP** command to query the maximum payload size for the device. |
| 2 | Iterations | The number of packets to send. This must be a number between 1 and 4000. |

4. Device B should transmit test link packets.
5. When device B completes transmitting the test link packets, it sends the following data packet to device A's Test Link Result Cluster (0x0094) on endpoint (0xE6).
6. Device A outputs the following information as an API Explicit RX Indicator (0x91) frame:

| Number of bytes | Field name | Description |
|---|---|---|
| 8 | Destination address | The address the device used to test its link. |
| 2 | Payload size | The size of the test packet device A sent to test the link. |
| 2 | Iterations | The number of packets that device A sent. |
| 2 | Success | The number of packets that were successfully acknowledged. |
| 2 | Retries | The number of MAC retries used to transfer all the packets. |
| 1 | Result | 0x00 - the command was successful.<br>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 you want to test the link between device A (**SH**/**SL** = 0x0013A200 40521234) and device B (**SH**/**SL**=0x0013A 200 4052ABCD) by transmitting 1000 40-byte packets:

Send the following API packet to the serial interface of device A.

In the following example packet, whitespace marks fields, 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

When the test is finished, the following API frame may be received:

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

This means:

- 999 out of 1000 packets were successful.
- The device made 100 retries.
- **RR** = 10.
- maxRSSI = -80 dBm.
- minRSSI = -83 dBm.
- avgRSSI = -82 dBm.

If the Result field does not equal zero, an error has occurred. Ignore the other fields in the packet.

If the Success field equals zero, ignore the RSSI fields.

The device that sends the request for initiating the Test link and outputs the result does not need to be the sender or receiver of the test. It is possible for a third node, "device C", to request device A to perform a test link against device B and send the results back to device C to be output. It is also possible for device B to request device A to perform the previously mentioned test. In other words, the

frames can be sent by either device A, device B or device C and in all cases the test is the same: device A sends data to device B and reports the results.

## RSSI indicators

The received signal strength indicator (RSSI) measures the amount of power present in a radio signal. It is an approximate value for signal strength received on an antenna.

You can use the **DB** command to measure the RSSI on a device. **DB** returns the RSSI value measured in -dBm of the last packet the device received. This number can be misleading in multi-hop 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.

To determine the **DB** value in hardware:

1. Use the **PO** command to enable the RSSI pulse-width modulation (PWM) functionality.
2. Use the RSSI/PWM module pin (pin 6). When the device receives data, it sets the RSSI PWM duty cycle to a value based on the RSSI of the packet it receives.

This value only indicates the quality of the last hop of a multi-hop transmission. You could connect this pin to an LED to indicate if the link is stable or not.

## Discover devices

### *Discover all the devices on a network*

You can use the **ND** (Network Discovery) command to discover all devices on a network. When you send the **ND** command:

1. The device sends a broadcast **ND** command through the network.
2. All devices that receive the command send a response that includes their addressing information, node identifier string and other relevant information. For more information on the node identifier string, see NI (Node Identifier).

**ND** is useful for generating a list of all device addresses in a network.

When a device receives the network discovery command, it waits a random time before sending its own response. You can use the **NT** command to set the maximum time delay on the device that you use to send the **ND** command.

- The device that sends the **ND** includes its **NT** setting in the transmission to provide a delay window for all devices in the network.
- The default **NT** value is 0x82 (13 seconds).

## Trace route option

In many networks, it is useful to determine the route that a DigiMesh unicast takes to its destination; particularly, when you set up a network or want to diagnose problems within a network.

**Note** Because of the large number of Route Information Packet frames that a unicast with trace route enabled can generate, we suggest you only use the trace route option for occasional diagnostic purposes and not for normal operations.

The Transmit Request (0x10) frame contains a trace route option, which transmits routing information packets to the originator of the unicast using the intermediate nodes.

When a device sends a unicast with the trace route option enabled, the unicast transmits to its destination devices, which forward the unicast to its eventual destination. The destination device transmits a Route Information Packet (0x8D) frame back along the route to the unicast originator.

The Route Information Packet frame contains:

- Addressing information for the unicast.
- Addressing information for the intermediate hop.
- Other link quality information.

For a full description of the Route Information Packet frame, see Route Information Packet frame - 0x8D.

### *Trace route example*

Suppose that you successfully unicast a data packet with trace route enabled from device A to device E, through devices B, C, and D. The following sequence would occur:

- After the data packet makes a successful MAC transmission from device A to device B, device A outputs a Route Information Packet frame indicating that the transmission of the data packet from device A to device E was successful in forwarding one hop from device A to device B.
- After the data packet makes a successful MAC transmission from device B to device C, device B transmits a Route Information Packet frame to device A. When device A receives the Route Information packet, it outputs it over its serial interface.
- After the data packet makes a successful MAC transmission from device C to device D, device C transmits a Route Information Packet frame to device A (through device B). When device A receives the Route Information packet, it outputs it over its serial interface.
- After the data packet makes a successful MAC transmission from device D to device E, device D transmits a Route Information Packet frame to device A (through device C and device B). When device A receives the Route Information packet, it outputs it over its serial interface.

There is no guarantee that Route Information Packet frames will arrive in the same order as the route taken by the unicast packet. On a weak route, it is also possible for the transmission of Route Information Packet frames to fail before arriving at the unicast originator.

### *Discover devices within RF range*

- You can use the **FN** (Find Neighbors) command to discover the devices that are immediate neighbors (within RF range) of a particular device.
- **FN** is useful in determining network topology and determining possible routes.

You can send **FN** locally on a device in Command mode or you can use a local AT Command (0x08) frame.

To use **FN** remotely, send the target node a Remote AT Command frame (0x17) using **FN** as the name of the AT command.

The device you use to send **FN** transmits a zero-hop broadcast to all of its immediate neighbors. All of the devices that receive this broadcast send an RF packet to the device that transmitted the **FN** command. If you sent **FN** remotely, the target devices respond directly to the device that sent the **FN** command. The device that sends **FN** outputs a response packet in the same format as an AT Command Response (0x88) frame.

## NACK messages

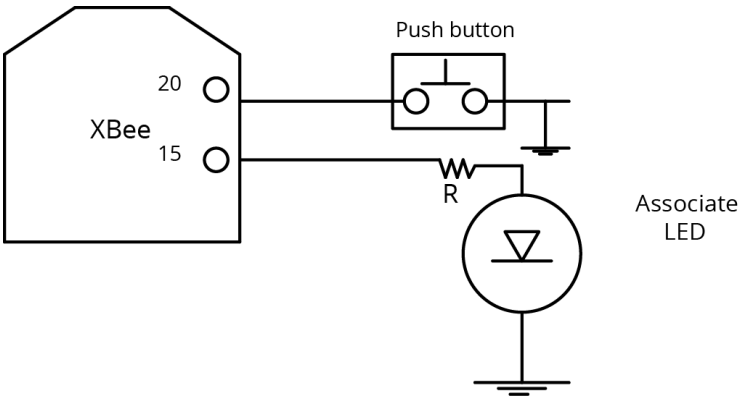
Transmit Request (0x10 and 0x11) frames contain a negative-acknowledge character (NACK) API option (Bit 2 of the Transmit Options field).

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.

## The Commissioning Pushbutton

The XBee/XBee-PRO DigiMesh 2.4 supports a set of commissioning and LED functions to help you deploy and commission devices. These functions include the Commissioning Pushbutton definitions and the associated LED functions. The following diagram shows how the hardware can support these features.




To support the Commissioning Pushbutton and its associated LED functions, connect a pushbutton and an LED to device pins 20 and 15 respectively.

### *Definitions*

To enable the Commissioning Pushbutton functionality on pin 20, set the **D0** command to 1. The functionality is enabled by default.

You must perform multiple button presses within two seconds.

The following table provides the pushbutton definitions.

| Button presses | Sleep configuration and sync status | Action |
|---|---|---|
| 1 | Not configured for sleep | Immediately sends a Node Identification broadcast transmission.<br>All devices that receive this transmission blink their Associate LED rapidly for one second.<br>All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART. |

| Button presses | Sleep configuration and sync status | Action |
|---|---|---|
| 1 | Configured for asynchronous sleep | Wakes the device for 30 seconds.<br>Immediately sends a Node Identification broadcast transmission.<br>All devices that receive this transmission blink their Associate LED rapidly for one second.<br>All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART. |
| 1 | Configured for synchronous sleep | Wakes the module for 30 seconds or until the synchronized network goes to sleep.<br>Queues a Node Identification broadcast transmission that it sends at the beginning of the next network wake cycle.<br>All devices that receive this transmission blink their Associate LED rapidly for one second.<br>All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART. |
| 2 | Not configured for synchronous sleep | No effect. |
| 2 | Configured for synchronous sleep | Causes a node configured with sleeping router nomination enabled to immediately nominate itself as the network sleep coordinator. For more information, see SO (Sleep Options). |
| 4 | Any | Sends an **RE** command to restore device parameters to default values. |

### *Use the Commissioning Pushbutton*

Use the CB command to simulate button presses in software. Send **CB** with a parameter set to the number of button presses to perform. For example, if you send **ATCB1**, the device performs the action (s) associated with a single button press.

The Node Identification Indicator (0x95) frame is similar to the Remote Command Response (0x97) frame – it contains the device's address, node identifier string (NI command), and other relevant data. All devices in API operating mode that receive the Node Identification Indicator frame send it out their UART as a Node Identification Indicator frame.

If you enable the Commissioning Pushbutton during sleep, it increases the sleeping current draw, especially in Asynchronous pin sleep (**SM** = 1) mode. When asleep, hold down the Commissioning Pushbutton for up to two seconds to wake the device from sleep, then issue the two or four button presses.

## 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/XBee-PRO DigiMesh 2.4. 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 |
| 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.

# Monitor I/O lines

| 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 |
| 6-9 | 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.

## Queried sampling

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

If you send the **IS** command from Command mode, then the device returns a carriage return delimited list containing the following fields.

| Field | Name | Description |
|---|---|---|
| 1 | Sample sets | Number of sample sets in the packet. Always set to 1. |
| 2 | Digital channel mask | Indicates which digital I/O lines have sampling enabled. Each bit corresponds to one digital I/O line on the device.<br>bit 0 = AD0/DIO0<br>bit 1 = AD1/DIO1<br>bit 2 = AD2/DIO2<br>bit 3 = AD3/DIO3<br>bit 4 = DIO4<br>bit 5 = ASSOC/DIO5<br>bit 6 = RTS/DIO6<br>bit 7 = CTS/GPIO7<br>bit 8 = DTR / SLEEP_RQ / DIO8<br>bit 9 = ON_SLEEP / DIO9<br>bit 10 = RSSI/DIO10<br>bit 11 = PWM/DIO11<br>bit 12 = CD/DIO12<br>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.<br>bit 0 = AD0/DIO0<br>bit 1 = AD1/DIO1<br>bit 2 = AD2/DIO2<br>bit 3 = AD3/DIO3<br>bit 4 = AD4/DIO4<br>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.<br>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.<br>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.

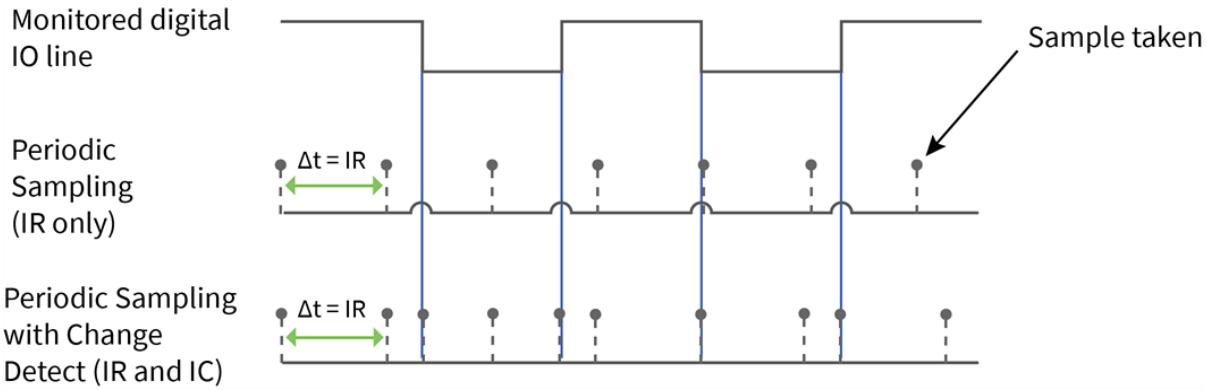
Only devices with API operating mode enabled send I/O data samples out their serial interface. Devices that are in Transparent mode (**AP** = 0) discard the I/O data samples they receive. You must configure at least one pin as a digital or ADC input to generate sample data.

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. In the figure, the gray dashed lines with a dot on top represent samples taken from the monitored DIO line. The top graph shows only **IR** samples, the bottom graph shows a combination of **IR** samples and **IC** (Change Detect). In the top graph, the humps indicate that the sample was not taken at that exact moment and needed to wait for the next **IR** sample period.



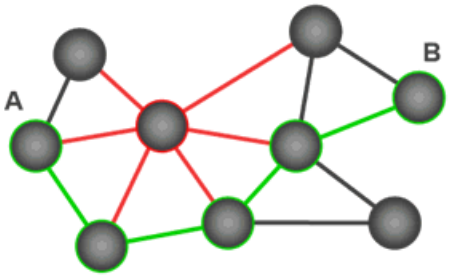
**Note** Use caution when combining Change Detect sampling with sleep modes. **IC** only causes a sample to be generated if the change takes place during a wake period. If the device is sleeping when the digital input transition occurs, then no change is detected and an I/O sample is not generated. Use **IR** in conjunction with **IC** in this instance, since **IR** generates an I/O sample upon wakeup and ensures that the change is properly observed.

# Network configurations

# 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 three 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. If a node can no longer operate, for example 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.

## Routers and end devices

You can use the **CE** command to configure devices in a DigiMesh network to act as routers or end devices. All devices in a DigiMesh network act as routers by default. Any devices that you configure as routers actively relay network unicast and broadcast traffic.

# Network identifiers

You define DigiMesh networks with a unique network identifier. Use the **ID** command to set this identifier. For devices to communicate, you must configure them with the same network identifier and the same operating channel. For devices to communicate, the **CH** and **ID** commands must be equal on all devices in the network.

The **ID** command directs the devices to talk to each other by establishing that they are all part of the same network. The **ID** parameter allows multiple DigiMesh networks to co-exist on the same physical channel.

## Operating channels

The XBee/XBee-PRO DigiMesh 2.4 operates over the 2.4 GHz band using direct sequence spread spectrum (DSSS) modulation. DSSS modulation allows the device to operate over a channel or frequency that you specify.

The 2.4 GHz frequency band defines 16 operating channels. XBee devices support all 16 channels and XBee-PRO devices support 12 of the 16 channels.

Use the **CH** command to select the operating channel on a device. **CH** tells the device the frequency to use to communicate.

For devices to communicate, the **CH** and **ID** commands must be equal on all devices in the network.

Note these requirements for communication:

- A device can only receive data from other devices within the same network (with the same **ID** value) and using the same channel (with the same **CH** value).
- A device can only transmit data to other devices within the same network (with the same **ID** value) and using the same channel (with the same **CH** value).

## 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.

If a device sends a unicast that uses both MAC and NWK retries and acknowledgments:

- Use MAC retries and acknowledgments for transmissions between adjacent devices in the route.
- Use NWK retries and acknowledgments 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.

### Broadcast addressing

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

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

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

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

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

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

## Routing

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.

## Throughput

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

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

Our empirical testing shows the following throughput performance in a robust operating environment with low interference.

| Configuration | Data throughput |
|---|---|
| 1 hop, encryption disabled | 27.0 kb/s |
| 3 hop, encryption disabled | 10.9 kb/s |
| 6 hop, encryption disabled | 5.78 kb/s |
| 1 hop, encryption enabled | 20.5 kb/s |
| 3 hop, encryption enabled | 9.81 kb/s |
| 6 hop, encryption enabled | 4.70 kb/s |

We performed data throughput measurements with the serial interface rate set to 115200 b/s, and measured the time to send 100,000 bytes from the source to the destination. During the test, there were no route discoveries or failures.

## 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:

- Transmitting a broadcast.
- Transmitting a unicast with a known route.
- Transmitting a unicast with an unknown route.
- Transmitting a unicast with a broken route.

**Note** The timeouts in this documentation are theoretical timeouts and are not precisely accurate. Your application should pad the calculated maximum timeouts by a few hundred milliseconds. When you use API operating mode, use Transmit Status (0x8B) frames 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 time depends on the **RR** parameter.

DigiMesh networks assume that the average number of MAC-level retries across a multi-hop wireless link will be three or less. The following table defines the retires and the associated time.

| RR (mac retries) | Unicast one hop time |
|---|---|
| 0 | unicastOneHopTime = 5 ms |
| 1 | unicastOneHopTime = 24 ms |
| 2 | unicastOneHopTime = 40 ms |
| 3 | unicastOneHopTime = 63 ms |

### 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**, **NN**, and **MT** parameters define the worst cast maximum broadcast delay as follows:

BroadcastTxTime=NN*NH*(MT+1)*18ms

However, if **BH** < **NH** and **!** = 0, then the formula is **NN** * **BH** * (**MT**+1) * 18ms. In that case, **BH** limits the number of hops to be less than **NH**.

### 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)

# 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).

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

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

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.

This forms a cyclic sleeping network.

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

# Normal mode

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

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

Use mains-power for devices in Normal mode.

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

When synchronized, a device in Normal mode relays sync messages that sleep-compatible nodes generate, but will 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.

## 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 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.

## Asynchronous cyclic sleep with pin wake up mode

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

This mode is a slight variation on asynchronous cyclic sleep mode (**SM** = 4) that allows you to wake a device prematurely by asserting the SLEEP_RQ pin (pin 9).

In this mode, you can wake the device after the sleep period expires, or if a high-to-low transition occurs on the SLEEP_RQ pin.

## Synchronous sleep support mode

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

A device in synchronous sleep support mode synchronizes itself with a sleeping network, but 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_SLEEP enabled) on a cyclic sleep node, the ON_SLEEP line asserts when the device is awake and de-asserts when the device is asleep.

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

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

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

# 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 the **ST** command to set the duration of the timer.
- When the sleep timer expires the device returns to sleep.

# 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.

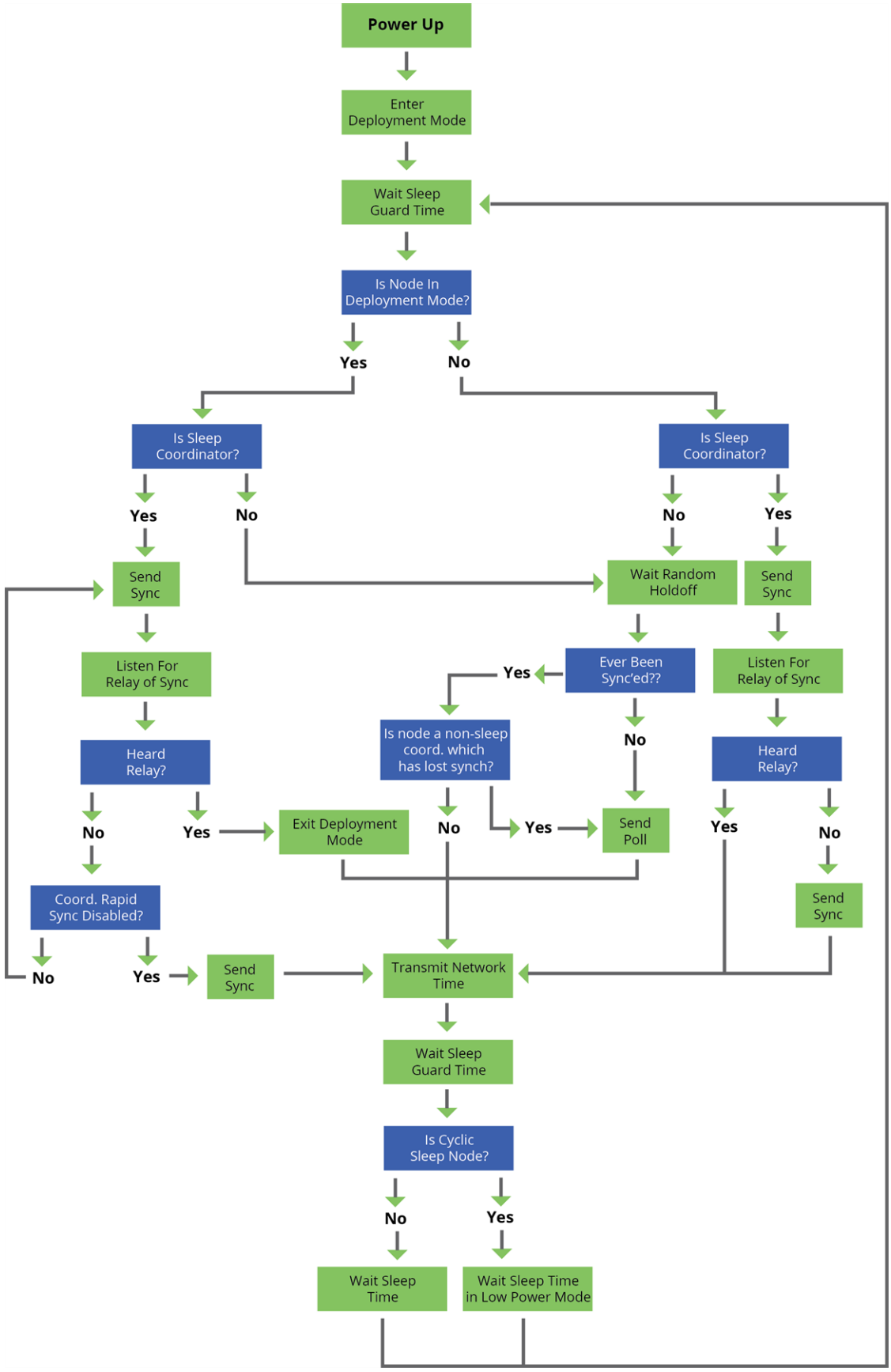
Use the **SO** (sleep options) command to disable deployment mode. This option is enabled by default.

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.

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

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

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

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

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

## Resolution criteria and selection option

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

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

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

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

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

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

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

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

## Commissioning Pushbutton option

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

### *Change sleep parameters*

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

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

For most applications, we recommend configuring the **NH**, **NN**, and **MR** network parameters during initial deployment only. The default values of **NH** and **NN** are optimized to work for most deployments.

### *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)*18ms.

3. Select the duty cycle you want.

4. Choose the sleep period and wake time.

   The wake time must be long enough to transmit the desired data as well as the sync message. The **ST** parameter automatically adjusts upwards to its minimum value when you change other AT commands that affect it (**SP**, **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. Wait a sleep cycle for the sleeping nodes to sync themselves to the new **SP** and **ST** values.
7. Disable the preferred sleep coordinator option bit on the sleep coordinator unless you want a preferred sleep coordinator.
8. Deploy the nodes to their positions.

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

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

## Add a new node to an existing network

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

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

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

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

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

## Change sleep parameters

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

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

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

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

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

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

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

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

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

## Rejoin nodes that lose sync

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

If a scenario exists where the only route connecting a subnet to the rest of the network depends on a single node, and that node fails or the wireless link fails due to changing environmental conditions (a catastrophic failure condition), then multiple subnets may arise using the same wake and sleep

intervals. When this occurs the first task is to repair, replace, and strengthen the weak link with new and/or redundant devices to fix the problem and prevent it from occurring in the future.

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

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

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

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

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

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

# Diagnostics

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

## Query sleep cycle

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

## Sleep status

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

## Missed sync messages command

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

## Sleep status API messages

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

# AT commands

# Special commands

The following commands are special commands.

## AC (Apply Changes)

Immediately applies new settings without exiting Command mode.

**Parameter range**

N/A

**Default**

N/A

## FR (Software Reset)

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

**Parameter range**

N/A

**Default**

N/A

## RE (Restore Defaults)

Restore device parameters to factory defaults.

**Parameter range**

N/A

**Default**

N/A

## WR (Write)

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

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

**Parameter range**

N/A

**Default**

N/A

# MAC/PHY commands

The following AT commands are MAC/PHY commands.

## CH (Operating Channel)

Set or read the operating channel devices used to transmit and receive data. The channel is one of three addressing configurations available to the device. The other configurations are the PAN ID (**ID** command) and destination addresses (**DL** and **DH** commands).

In order for devices to communicate with each other, they must share the same channel number. A network can use different channels to prevent devices in one network from listening to the transmissions of another. Adjacent channel rejection is 23 dB.

The command uses 802.15.4 channel numbers. Center frequency = 2405 MHz + (**CH** - 11 decimal) * 5 MHz.

**Parameter range**

0xB - 0x1A

**Default**

0xC (12 decimal)

## 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 - 0xFFFF

**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 - 0xF

**Default**

3

## CA (CCA Threshold)

Set or read the Clear Channel Assessment (CCA) threshold. Prior to transmitting a packet, the device performs a CCA to detect energy on the channel. If the device detects energy above the CCA threshold, it will not transmit the packet.

The **CA** parameter is measured in units of -dBm.

Setting the parameter to 0x00 disables CCA, otherwise the valid range is 0x24 - 0x50.

**Parameter range**

0x0 - 0x50 -dBm

**Default**

0x0 (CCA disabled)

**Example**

If you set the **CA** parameter to 60 (0x3C), the device does not transmit if it detects a signal greater than -60 dBm on the channel.

**ETSI compliance (Europe)**

Use the following settings for ETSI compliance.

| Device | Hex value | Sets to level |
|---|---|---|
| XBee | 0x3A | -58 dBm |
| XBee-PRO | 0x43 | -67 dBm |

## PL (TX Power Level)

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

For XBee, **PL** = 4, **PM** = 1 is tested at the time of manufacturing. Other power levels are approximate. On channel 26, transmitter power will not exceed -4 dBm.

**Parameter range**

0 - 4

These parameters equate to the following settings for the XBee RF module:

| Setting | Power level |
|---|---|
| 0 | -7 dBm |
| 1 | -1.7 dBm |
| 2 | -0.77 dBm |
| 3 | +0.62 dBm |
| 4 | +1.42 dBm |

These parameters equate to the following settings for the XBee-PRO RF module:

| Setting | Power level |
|---|---|
| 0 | +10 dBm |
| 1 | +12 dBm |
| 2 | +14 dBm |

| Setting | Power level |
|---|---|
| 3 | +16 dBm |
| 4 | +18 dBm |

**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**

0xA (10 retries)

## 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 - 0x3A98 (15 seconds)

**Default**

N/A

## 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 zero 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)

This command reports the received signal strength of the last RF data packet that a device receives.

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

If the device has been reset and has not yet received a packet, this variable reports 0.

The command measures RSSI in -dBm. For example if **DB** returns 0x60, then the RSSI of the last packet received was -96 dBm. This value is volatile (the value does not persist in the device's memory after a power-up sequence).

**Parameter range**

N/A

**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**

N/A

**Default**

N/A

## EA (MAC ACK Failure Count)

The number of unicast transmissions that time out awaiting a MAC ACK. This can be up to **RR** +1 timeouts per unicast when **RR** > 0.

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**

N/A

**Default**

N/A

## TR (Transmission Failure Count)

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

**Parameter range**

N/A

**Default**

N/A

### UA (Unicasts Attempted Count)

The number of unicast transmissions expecting an acknowledgment (when **RR** > 0).

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

### %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**

N/A

### %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**

N/A

## Network commands

The following commands are network commands.

### CE (Routing / Messaging Mode)

The routing and messaging mode of the device.

End devices do not propagate broadcasts and will not become intermediate nodes on a route.

**Parameter range**

0 - 2

| Parameter | Description | Routes packets |
|---|---|---|
| 0 | Standard router | Yes |
| 1 | N/A | N/A |
| 2 | End device | No |

**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**.

**Parameter range**

0 - 0x20

**Default**

0

## NH (Network Hops)

Sets or reads 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.

**Parameter range**

1 - 0x20 (1 - 32 hops)

**Default**

7

## DM (DigiMesh Options)

A bit field mask that you can use to enable or disable DigiMesh features.

Bit:

0: Disable aggregator updates. When set to 1, the device does not issue or respond to AG requests.

1: Disable Trace Route and NACK responses. When set to 1, the device does not generate or respond to Trace Route or NACK requests.

**Parameter range**

0 - 0x03 (bit field)

**Default**

0

### NN (Network Delay Slots)

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

One network delay slot is approximately 13 ms.

**Parameter range**

1 - 0xA network delay slots

**Default**

3

### MR (Mesh Unicast Retries)

Set or read the maximum number of DigiMesh network unicast 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.

**Parameter range**

0 - 7 mesh unicast retries

**Default**

1

## Addressing commands

The following AT commands are addressing commands.

### SH (Serial Number High)

Reads the upper 32 bits of the device's unique IEEE 64-bit extended address. 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)

Reads the lower 32 bits of the device's unique IEEE 64-bit extended address. 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

## 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.

The destination address is also used for I/O sampling in both Transparent and API modes.

To transmit using a 16-bit address, set **DH** to 0 and **DL** less than 0xFFFF.

0x000000000000FFFF is the broadcast address.

**Parameter range**

0 - 0xFFFFFFFF

**Default**

0

## DL (Destination Address Low)

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. The destination address is also used for I/O sampling in both Transparent and API modes.

0x000000000000FFFF is the broadcast address.

**Parameter range**

0 - 0xFFFFFFFF

**Default**

0xFFFF

## 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. The string cannot start with the space character. A carriage return or a comma automatically ends the command.

**Default**

0x20 (an ASCII space character)

## NT (Network Discovery Back-off)

Sets or reads the network discovery back-off parameter for a device. This sets the maximum value for the random delay that the device uses to send network discovery responses.

The **ND**, **DN**, and **FN** commands use **NT**.

**Parameter range**

0x20 - 0x2EE0 (x 100 ms)

**Default**

0x82 (13 seconds)

## NO (Network Discovery Options)

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

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.

**Parameter range**

0x0 - 0x7 (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** response frame out the serial interface when **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.

If you set this value to 0x12 (loopback Cluster ID), the destination node echoes any transmitted packet back to the source device.

**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.

**Parameter range**

0 - 0xFF

**Default**

0xE8

## Diagnostic - addressing commands

The following AT command is a Diagnostic - addressing command.

### N? (Network Discovery Timeout)

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

**Parameter range**

[Read-only]

**Default**

0x3D6A

## 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 XBees 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 number

**Default**

N/A

## DN (Discover Node)

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

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

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

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

When **DN** is sent as a local AT Command API frame (0x08):

1. The receiving device returns 0xFFFE followed by its 64-bit extended addresses in a Remote Command Response (0x97) frame.
2. If there is no response from a module within (**NT** * 100) milliseconds or you do not specify a parameter (by leaving it blank), the receiving device returns an ERROR message.

**Parameter range**

20-byte ASCII string

**Default**

## ND (Network Discover)

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

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

The **NT** setting determines the range of the random offset delay. The **NO** setting sets options for the Node Discovery.

For more information about options that affect the behavior of the **ND** command Refer to the description of the **NO** command for options which affect the behavior of the **ND** command.

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

**Parameter range**

N/A

**Default**

[read-only]

N/A

ASCII space character (0x20)

## FN (Find Neighbors)

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

**MY**<CR> (always 0xFFFE)

**SH**<CR>

**SL**<CR>

**NI**<CR> (Variable length)

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

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

STATUS<CR> (1 Byte: Reserved)

PROFILE_ID<CR> (2 Bytes)

MANUFACTURER_ID<CR> (2 Bytes)

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

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

<CR>

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

If you send the **FN** command through a local 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.

**Parameter range**

N/A

**Default**

N/A

# Security commands

The following AT commands are security commands.

### EE (Encryption Enable)

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

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

**Parameter range**

0 - 1

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

**Default**

0

### KY (AES Encryption Key)

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

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

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

**Parameter range**

128-bit value

**Default**

N/A

0

## Serial interfacing commands

The following AT commands are serial interfacing commands.

### BD (Baud Rate)

To request non-standard baud rates with values above 0x80, you can use the Serial Console toolbar in XCTU to configure the serial connection (if the console is connected), or click the **Connect** button (if the console is not yet connected).

When you send non-standard baud rates to a device, it stores the closest interface data rate represented by the number in the **BD** register. Read the **BD** command by sending **ATBD** without a parameter value, and the device returns the value stored in the **BD** register.

**Parameter range**

| Value | Description |
|---|---|
| 0x1 | 2,400 b/s |

| Value | Description |
|---|---|
| 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 |
| 0x9 | 460,800 b/s |
| 0xA | 921,600 b/s |

0 - 8 (standard rates) 0x39 to 0xF4240 if the host supports it

| Parameter | Description |
|---|---|
| 0x39 to 0xF4240 if the host supports it. | |

**Default**

3 (9600 b/s)

## NB (Parity)

Set or display the parity settings for UART communications.

**Parameter range**

0x00 - 0x04

| Parameter | Description |
|---|---|
| 0x00 | No parity |
| 0x01 | Even parity |
| 0x02 | Odd parity |
| 0x03 | Mark parity (forced high) |
| 0x04 | Space parity (forced low) |

**Default**

0x00

## 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.

De-assert $\overline{\text{CTS}}$ and/or send XOFF when **FT** bytes are in the UART receive buffer. Re-assert $\overline{\text{CTS}}$ when less than **FT**-16 bytes are in the UART receive buffer.

**Parameter range**

0x11 - 0xEE bytes

**Default**

0xBE

## AP (API Enable)

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

When you enable API, you must format the serial data as API frames because Transparent operating mode is disabled.

Enable API Mode. The device ignores this command when using SPI. API mode 1 is always used.

**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. |
| 2 | API Mode With Escapes. The device is in API mode and inserts escaped sequences to allow for control characters. The device passes XON (0x11), XOFF (0x13), Escape (0x7D), and start delimiter 0x7E as data. |

| Parameter | Description |
|---|---|
| 0 | API disabled (operate in Transparent mode) |
| 1 | API enabled |
| 2 | API enabled (with escaped control characters) |

**Default**

0

## AO (API Options)

The API data frame output format for RF packets received.

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

**Parameter range**

0 - 2

| 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)

**Parameter range**

0 - 4

**Default**

## D0 (DIO0/AD0)

The DIO0/AD0 pin configuration (pin 20).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 0 | Unmonitored digital input |
| 1 | Commissioning Pushbutton |
| 2 | ADC |
| 3 | Digital input |

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

**Default**

1

## D1 (DIO1/AD1)

The DIO1/AD1 pin configuration (pin 19).

**Parameter range**

0, 2 - 5

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

**Default**

0

## D2 (DIO2/AD2)

The DIO2/AD2 pin configuration (pin 18).

**Parameter range**

0, 2 - 5

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

**Default**

0

## D3 (DIO3/AD3)

The DIO3/AD3 pin configuration (pin 17).

**Parameter range**

0, 2 - 5

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

**Default**

0

## D4 (DIO4/AD4)

The DIO4/AD4 pin configuration (pin 11).

**Parameter range**

0, 2 - 5

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

**Default**

0

## D5 (DIO5/AD5/ASSOCIATED_INDICATOR)

The DIO5/AD5/ASSOCIATED_INDICATOR pin configuration (pin 15).

**Parameter range**

0, 1, 3-5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Associate LED indicator - blinks when associated |
| 2 | ADC |
| 3 | Digital input |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## D6 (DIO6/$\overline{\text{RTS}}$)

The DIO6/$\overline{\text{RTS}}$ pin configuration (pin 16).

**Parameter range**

0, 1, 3 - 5

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

**Default**

0

## D7 (DIO7/$\overline{\text{CTS}}$)

The DIO7/$\overline{\text{CTS}}$ pin 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**

1

## D8 (DIO8/SLEEP_REQUEST)

The DIO8/SLEEP_REQUEST pin configuration (pin 9).

This command enables you to configure the pin to function as a digital input. This line is also used with Pin Sleep, but pin sleep ignores the **D8** configuration. It is always used to control pin sleep, regardless of configuration of D8.

**Parameter range**

0, 1, 3 - 5

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

**Default**

1

## D9 (DIO9/ON_$\overline{\text{SLEEP}}$)

The DIO9/ON_$\overline{\text{SLEEP}}$ pin 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)

The DIO10/RSSI/PWM0 pin configuration.

The DIO10/RSSI/PWM0 pin configuration (pin 7).

**Parameter range**

0 - 5

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

**Default**

1

## P1 (DIO11/PWM1 Configuration)

The DIO11/PWM1 pin configuration (pin 7).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |

| Parameter | Description |
|---|---|
| 1 | N/A |
| 2 | PWM1 output |
| 3 | Digital input |
| 4 | Digital output, low |
| 5 | Digital output, high |

**Default**

0

## P2 (DIO12 Configuration)

The DIO12 pin configuration (pin 4).

**Parameter range**

1, 3 - 5

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

**Default**

0

## P0 (DIO10/RSSI/PWM0 Configuration)

The DIO10/RSSI/PWM0 pin configuration.

The DIO10/RSSI/PWM0 pin configuration (pin 7).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | RSSI PWM0 output |

| Parameter | Description |
|---|---|
| 2 | PWM0 output |
| 3 | Digital input |
| 4 | Digital output, low |
| 5 | Digital output, high |

**Default**

1

## P1 (DIO11/PWM1 Configuration)

The DIO11/PWM1 pin configuration (pin 7).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | N/A |
| 2 | PWM1 output |
| 3 | Digital input |
| 4 | Digital output, low |
| 5 | Digital output, high |

**Default**

0

## P2 (DIO12 Configuration)

The DIO12 pin configuration (pin 4).

**Parameter range**

1, 3 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | N/A |
| 2 | NA |
| 3 | Digital input |

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

**Default**

0

## PR (Pull-up/Down Resistor Enable)

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

| Bit | I/O line |
|---|---|
| 0 | DIO4/AD4 |
| 1 | DIO3/AD3 |
| 2 | DIO2/AD2 |
| 3 | DIO1/AD1 |
| 4 | DIO0/AD0 |
| 5 | DIO6/$\overline{\text{RTS}}$ |
| 6 | DIO8/SLEEP_REQUEST |
| 7 | DIO14/DIN/CONFIG |
| 8 | DIO5/AD6/ASSOCIATE |
| 9 | DIO9/On/$\overline{\text{SLEEP}}$ |
| 10 | DIO12 |
| 11 | DIO10/RSSI/PWM0 |
| 12 | DIO11/PWM1 |
| 13 | DIO7/CTS |
| 14 | DOUT (pin 2) |

**Parameter range**

0 - 0x7FFF (bit field)

**Default**

0x1FFF

## M0 (PWM0 Duty Cycle)

The duty cycle of the PWM0 line. Use the **P0** command to configure the line as PWM output.

**Parameter range**

0 - 0x3FF

**Default**

0

### M1 (PWM1 Duty Cycle)

The duty cycle of the PWM1 line. Use the **P1** command to configure the line as 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 when the device has joined the network.

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

For all other **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. The signal duty cycle updates with each received packet and shuts off when the timer expires.

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.

### IC (DIO Change Detect)

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, use the **IC** command to force an immediate I/O sample transmission when the DIO state changes. If sleep is enabled, the edge transition must occur during a wake period to trigger a change detect.

The data transmission contains only DIO data.

**IC** is a bitmask you can use to enable or disable edge detection on individual digital I/O lines. Only DIO0 through DIO12 can be sampled using a Change Detect.

Set unused bits to 0.

| Bit | I/O line | Module pin |
|---|---|---|
| 0 | DIO0 | 33 |
| 1 | DIO1 | 32 |
| 2 | DIO2 | 31 |
| 3 | DIO3 | 30 |
| 4 | DIO4 | 24 |
| 5 | DIO5 | 28 |
| 6 | DIO6 | 29 |
| 7 | DIO7 | 25 |
| 8 | DIO8 | 10 |
| 9 | DIO9 | 26 |
| 10 | DIO10 | 7 |
| 11 | DIO11 | 8 |
| 12 | DIO12 | 5 |

**Parameter range**

0 - 0xFFFF

0 - 0xFFFF (bit field)

**Default**

0

## IF (Sleep Sample Rate)

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

**Parameter range**

1 - 0xFF

**Default**

1

### IR (Sample Rate)

Set or read the I/O sample rate to enable periodic sampling. When set, this parameter causes the device to sample all enabled DIO and ADC at a specified interval.

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 (see D0 (DIO0/AD0)-D9 (DIO9/ON_SLEEP), P0 (DIO10/RSSI/PWM0 Configuration)-P2 (DIO12 Configuration).

**Parameter range**

0 - 0xFFFF (x 1 ms)

0, 0x32:0xFFFF (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 retuned 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

## Sleep commands

The following AT commands are sleep commands.

### SM (Sleep Mode)

Sets or reads the sleep mode of the device.

Normal mode is always awake. Pin sleep modes allow you to wake the device with the SLEEP_ REQUEST line. Asynchronous cyclic mode sleeps for **SP** time and briefly wakes, checking for activity. Sleep Support mode is always awake but can effectively communicate with **SM**8 nodes. Synchronized Cyclic Sleep nodes sleep for **SP** and wake for **ST** time.

Synchronous modes are not compatible with asynchronous modes.

| Parameter | Description |
|---|---|
| 0 | No sleep (disabled) |
| 1 | Pin hibernate |
| 2 | Pin doze |
| 4 | Cyclic Sleep Remote |
| 5 | Cyclic Sleep Remote with pin wakeup |

**Parameter range**

0 - 8

| Parameter | Description |
|---|---|
| 0 | Normal. |
| 1 | Asynchronous Pin Sleep. In this mode, the SLEEP_RQ line controls the sleep/wake state of the device. |
| 2 | N/A |
| 3 | N/A |
| 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. |
| 6 | N/A |
| 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**

0x0 - 0xFFFF

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

| Bit | Option |
|---|---|

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 Cycles Between ON_SLEEP )

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. This allows external circuitry to sleep longer than the **SP** time.

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

**Example**

Set to 1 to set ON_SLEEP high after each **SP** time (default).

If **SN** = 3, the ON_SLEEP line asserts only every third wakeup; **SN** = 9, every ninth wakeup; and so forth.

## SP (Sleep Time)

Sets or reads the device's sleep time. This command defines the amount of time the device sleeps per cycle.

**Parameter range**

1 - 0x15F900 (x 10 ms)

**Default**

0xC8

## ST (Wake Time)

Sets or reads 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.

For devices in synchronous sleep, the minimum wake time is a function of **MT**, **SP**, **NH**, **NN**, and platform dependent values. If you increase **SP**, **NH**, **NN**, or **MT**, the **ST** value raises automatically. The maximum value is one hour (0x36EE80 ms).

**Parameter range**

0x1 - 0x36EE80 (x 1 ms)

**Default**

0x7D0 (3 seconds)

### WH (Wake Host Delay)

Sets or reads 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 allows 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. |

| Bit | Description |
|---|---|
| 6 | This bit is true if the node is currently in deployment mode. |
| All other bits | Reserved. All non-documented bits should be ignored |

**Parameter range**

N/A

[Read-only]

**Default**

N/A

## 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**

N/A

**Default**

N/A

## 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, **OS** may differ from **ST**.

**Parameter range**

N/A

**Default**

N/A

## MS (Missed Sync Messages)

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

**Parameter range**

N/A

**Default**

N/A

### 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**

N/A

**Default**

N/A

## Command mode options

The following commands are Command mode option commands.

### CC (Command Character)

The character value the device uses to enter Command mode.

The default value (0x2B) is the ASCII code for the plus (+) character. You must enter it three times within the guard time to enter Command mode. To enter Command mode, there is also a 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**

0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

**Default**

0x2B (+)

### CT (Command Mode Timeout)

Sets or reads the Command mode timeout parameter. If a device does not receive any valid commands within this time period, it returns to Idle mode from Command mode.

**Parameter range**

2 - 0x1770 (x 100 ms)

**Default**

0x64 (10 seconds)

### CN (Exit Command mode)

Makes the device exit Command mode.

**Parameter range**

N/A

**Default**

N/A

## GT (Guard Times)

Set the required period of silence before and after the command sequence characters of the Command mode sequence (**GT** + **CC** + **GT**). The period of silence prevents inadvertently entering Command mode.

**Parameter range**

2 - 0xCE4 (x 1 ms)

**Default**

0x3E8 (one second)

The following AT commands are firmware commands.

## VL (Version Long)

Shows detailed version information including the application build date and time.

**Parameter range**

N/A

**Default**

N/A

## VR (Firmware Version)

Reads the firmware version on a device.

**Parameter range**

0 - 0xFFFFFFFF [Read-only]

**Default**

Set in the factory

## HV (Hardware Version)

Display the device's hardware version number.

**Parameter range**

0 - 0xFFFF [Read-only]

**Default**

Set in the factory

## DD (Device Type Identifier)

The Digi device type identifier value. Use this value to differentiate between multiple devices.

**Parameter range**

0 - 0xFFFFFFFF [Read-only]

**Default**

0x50000

## NP (Maximum Packet Payload Bytes)

Reads the maximum number of payload bytes that you can send in a unicast RF transmission based on the device's current configuration.

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**

N/A

## 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**

0 - 0xFFFFFFFF

**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.

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 specifications

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 command setting | Description |
|---|---|
| **AP** = 0 | Transparent operating mode, UART serial line replacement with API modes disabled. This is the default option. |
| **AP** = 1 | API operation. |
| **AP** = 2 | API operation with escaped characters (only possible on UART). |

The API data frame structure differs depending on what mode you choose.

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 discards the frame.

### *API operation (AP parameter = 1)*

We recommend this API mode for most applications. The following table shows the data frame structure when you enable this mode:

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

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

Set API to 2 to allow escaped control characters in the API frame. Due to its increased complexity, we only recommend this API mode in specific circumstances. API 2 may help improve reliability if the serial interface to the device is unstable or malformed frames are frequently being generated.

When operating in API 2, if an unescaped 0x7E byte is observed, it is treated as the start of a new API frame and all data received prior to this delimiter is silently discarded. For more information on using this API mode, refer to the following knowledge base article:

http://knowledge.digi.com/articles/Knowledge_Base_Article/Escaped-Characters-and-API-Mode-2

The following table shows the structure of an API frame with escaped characters:

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

**Escape characters**

When sending or receiving a UART data frame, you must escape (flag) specific data values 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. If not escaped, 0x11 and 0x13 are sent as is.

Data bytes that need to be escaped:

- 0x7E – Frame delimiter
- 0x7D – Escape
- 0x11 – XON
- 0x13 – XOFF

**Example** - Raw UART data frame (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.

### *Start delimiter*

This field indicates the beginning of a frame. It is always 0x7E. This allows the device to easily detect a new incoming frame.

### *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 will transmit. The structure of frame data depends on the purpose of the API frame:

| Start delimiter | Length | | Frame data | | | | | | | | Checksum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Frame type | Data | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | n | n+1 |
| 0x7E | MSB | LSB | API frame type | Data | | | | | | | Single byte |

- **Frame type** is the API frame type identifier. It determines the type of API frame and indicates how the Data field organizes the information.
- **Data** contains the data itself. This information and its order depend on the what type of frame that the Frame type field defines.

The XBee modules support the following API frames:

| API Frame Names | API ID |
|---|---|
| AT Command | 0x08 |
| AT Command - Queue Parameter Value | 0x09 |
| ZigBee Transmit Request | 0x10 |
| Explicit Addressing ZigBee Command Frame | 0x11 |
| Remote Command Request | 0x17 |
| Create Source Route | 0x21 |
| AT Command Response | 0x88 |
| Modem Status | 0x8A |
| ZigBee Transmit Status | 0x8B |
| ZigBee Receive Packet (AO=0) | 0x90 |
| ZigBee Explicit Rx Indicator (AO=1) | 0x91 |
| ZigBee I/O Data Sample Rx Indicator | 0x92 |
| XBee Sensor Read Indicator (AO=0) | 0x94 |
| Node Identification Indicator (AO=0) | 0x95 |
| Remote Command Response | 0x97 |
| Extended Modem Status | 0x98 |
| Over-the-Air Firmware Update Status | 0xA0 |
| Route Record Indicator | 0xA1 |
| Many-to-One Route Request Indicator | 0xA3 |

### *Checksum*

Checksum is the last byte of the frame and helps test data integrity. It is calculated by taking the hash sum of all the API frame bytes that came before it, except the first three bytes (start delimiter and length).

The device does not process frames sent through the serial interface with incorrect checksums, and ignores their data.

## 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.

### Escaped characters in API frames

If operating in API mode with escaped characters (**AP** parameter = 2), when you send or receive an API frame, you must escape (flag) specific data values so they do not interfere with data frame sequencing. In API operating mode with escaped characters, you must escape the following data bytes:

- 0x7E: start delimiter
- 0x7D: escape character
- 0x11: XON
- 0x13: XOFF

API operating mode with escaped characters guarantees that all the 0x7E bytes a device receives are start delimiters: this character cannot be part of any of the other frame fields (length, data, or checksum) since you must escape it.

To escape a character:

1. Insert 0x7D, the escape character.
2. Append it with the byte you want to escape, XORed with 0x20.

In API operating mode with escaped characters, the length field does not include any escape characters in the frame and the firmware calculates the checksum with non-escaped data.

## API frames

The device sends multi-byte values in big-endian format. The XBee/XBee-PRO DigiMesh 2.4 supports API frames in the following table. Request frames are less than 0x80 and responses are always 0x80 or higher.

| API frame name | API ID |
|---|---|
| AT Command | 0x08 |
| AT Command-Queue Parameter Value | 0x09 |
| Transmit Request | 0x10 |
| Explicit Addressing Command Frame | 0x11 |
| Remote Command Request | 0x17 |
| AT Command Response | 0x88 |

| API frame name | API ID |
|---|---|
| Modem Status | 0x8A |
| Transmit Status | 0x8B |
| Route Information Packet | 0x8D |
| Aggregate Addressing Update | 0x8E |
| Receive Packet (AO=0) | 0x90 |
| Explicit Rx Indicator (AO=1) | 0x91 |
| I/O Data Sample Rx Indicator | 0x92 |
| Node Identification Indicator (AO=0) | 0x95 |
| Remote Command Response | 0x97 |

## API frame exchanges

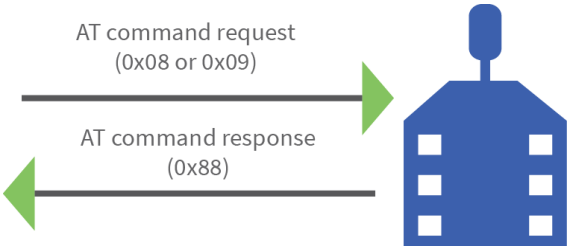
Every outgoing API frame has a corresponding response (or ACK) frame that indicates the success or failure of the outgoing API frame. This section details some of the common API exchanges that occur.

You can use the Frame ID field to correlate between the outgoing frames and associated responses.

**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 UART when you send a 0x08 AT Command Request or 0x09 AT Command-Queue Request to read or set a device parameter. To disable the 0x88 AT Command Response, set the frame ID to 0 in the request.




### *Transmit and Receive RF data*

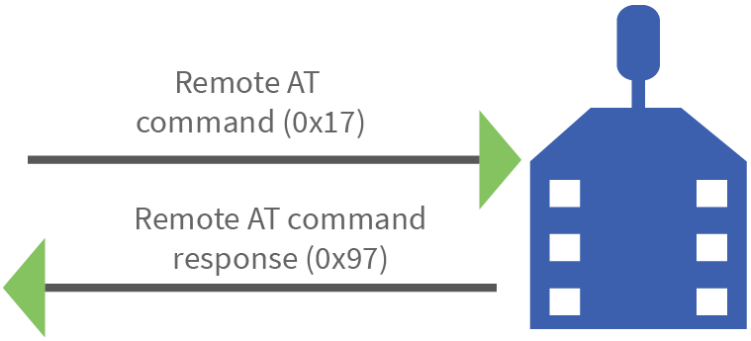
The following image shows the API exchanges that take place on the serial interface when a device sends a 0x10, or 0x11 Transmit Request to another device.


The device sends the 0x8B Transmit Status frame at the end of a data transmission unless you set the frame ID to 0 in the transmit request. If the packet cannot be delivered to the destination, the 0x8B Transmit Status frame indicates the cause of failure.

Use the **AP** command to choose the type of data frame you want to receive, either a (0x90) Receive Packet or a (0x91) Explicit Rx Indicator frame.

### *Remote AT commands*

The following image shows the API frame exchanges that take place on the serial interface when you send a 0x17 Remote AT Command frame The 0x97 Remote AT Command Response is always generated and you can use it to identify if the remote device successfully received and applied the command.



## Code to support future API frames

If your software application supports the API, you should make provisions that allow for new API frames in future firmware releases. For example, you can include the following section of code on a host microprocessor that handles serial API frames that are sent out the device's DOUT pin:

```
void XBee_HandleRxAPIFrame(_apiFrameUnion *papiFrame){
       switch(papiFrame->api_id){
               case RX_RF_DATA_FRAME:
                       //process received RF data frame
                       break;

               case RX_IO_SAMPLE_FRAME:
                       //process IO sample frame
                       break;

               case NODE_IDENTIFICATION_FRAME:
                       //process node identification frame
                       break;


               default:
                       //Discard any other API frame types that are not being used
                       break;
       }
}
```

# AT Command frame - 0x08

## Description

Use this frame to query or set device parameters on the local device. This API command applies changes after running the command. You can query parameter values by sending the 0x08 AT Command frame with no parameter value field (the two-byte AT command is immediately followed by the frame checksum).

A 0x8B response frame is populated with the parameter value that is currently set on the device.

## Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x08 |
| Frame ID | 4 | |
| AT command | 5-6 | Command name: two ASCII characters that identify the AT command. |
| Parameter value | 7-n | If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the register. |

## Example

The following example illustrates an AT Command frame when you modify the device's **NH** parameter value.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | |
| Frame type | 3 | 0x08 |
| Frame ID | 4 | 0x52 |
| AT command | 5 | 0x4E (N) |
| | 6 | 0x48 (H) |
| Parameter value (**NH**2 = two network hops) | 7 | 0x02 |
| Checksum | 8 | 0x0D |

# AT Command - Queue Parameter Value frame - 0x09

## Description

This frame allows you to query or set device parameters. In contrast to the AT Command (0x08) frame, this frame queues new parameter values and does not apply them until you issue either:

- The **AT** Command (0x08) frame (for API type)
- The **AC** command

When querying parameter values, the 0x09 frame behaves identically to the 0x08 frame. The device returns register queries immediately and not does not queue them. The response for this command is also an **AT** Command Response frame (0x88).

## Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x09 |
| Frame ID | 4 | Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response. |
| AT command | 5-6 | Command name: two ASCII characters that identify the AT command. |
| Parameter value | 7-n | If present, indicates the requested parameter value to set the given register. If no characters are present, queries the register. |

**Note** In this example, the parameter could have been sent as a zero-padded 2-byte or 4-byte value.

## Example

The following example sends a command to change the baud rate (**BD**) to 115200 baud, but does not apply the changes immediately. The device continues to operate at the previous baud rate until you apply the changes.

**Note** In this example, you could send the parameter as a zero-padded 2-byte or 4-byte value.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x05 |
| Frame type | 3 | 0x09 |

| Frame data fields | Offset | Example |
|---|---|---|
| Frame ID | 4 | 0x01 |
| AT command | 5 | 0x42 (B) |
| | 6 | 0x44 (D) |
| Parameter value (**BD**7 = 115200 baud) | 7 | 0x07 |
| Checksum | 8 | 0x68 |

## Transmit Request frame - 0x10

### Description

This frame causes the device to send payload data as an RF packet to a specific destination.

- For broadcast transmissions, set the 64-bit destination address to 0x000000000000FFFF .
- For unicast transmissions, set the 64 bit address field to the address of the desired destination node.
- Set the reserved field to 0xFFFE.
- Query the **NP** command to read the maximum number of payload bytes.

You can set the broadcast radius from 0 up to **NH**. If set to 0, the value of **NH** specifies the broadcast radius (recommended). This parameter is only used for broadcast transmissions.

You can read the maximum number of payload bytes with the **NP** command.

**Note** Using source routing reduces the RF payload by two bytes per intermediate hop in the source route.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x10 |
| Frame ID | 4 | Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response. |
| 64-bit destination address | 5-12 | MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF |
| Reserved | 13-14 | Set to 0xFFFE. |
| Broadcast radius | 15 | Sets the maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius is set to the maximum hops value. |
| Transmit options | 16 | 0x01 = Disable ACK<br>0x02 = Disable network address discovery<br>Set all other bits to **0**. |
| RF data | 17-n | Up to **NP** bytes per packet. Sent to the destination device. |

### Example

The example shows how to send a transmission to a device if you disable escaping (**AP** = 1), with destination address 0x0013A200 400A0127, and payload “TxData0A”.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x16 |
| Frame type | 3 | 0x10 |
| Frame ID | 4 | 0x01 |
| 64-bit destination address | MSB 5 | 0x00 |
| | 6 | 0x13 |
| | 7 | 0xA2 |
| | 8 | 0x00 |
| | 9 | 0x40 |
| | 10 | 0x0A |
| | 11 | 0x01 |
| | LSB 12 | 0x27 |
| 16-bit destination network address | MSB 13 | 0xFF |
| | LSB 14 | 0xFE |
| Broadcast radius | 15 | 0x00 |
| Options | 16 | 0x00 |
| RF data | 17 | 0x54 |
| | 18 | 0x78 |
| | 19 | 0x44 |
| | 20 | 0x61 |
| | 21 | 0x74 |
| | 22 | 0x61 |
| | 23 | 0x30 |
| | 24 | 0x41 |
| Checksum | 25 | 0x13 |

If you enable escaping (**AP** = 2), the frame should look like:

0x7E 0x00 0x16 0x10 0x01 0x00 0x7D 0x33 0xA2 0x00 0x40 0x0A 0x01 0x27 0xFF 0xFE 0x00 0x00 0x54 0x78 0x44 0x61 0x74 0x61 0x30 0x41 0x7D 0x33

The device calculates the checksum (on all non-escaped bytes) as [0xFF - (sum of all bytes from API frame type through data payload)].

# Explicit Addressing Command frame - 0x11

## *Description*

This frame is similar to Transmit Request (0x10), but it also requires you to specify the application-layer addressing fields: endpoints, cluster ID, and profile ID.

This frame causes the device to send payload data as an RF packet to a specific destination, using specific source and destination endpoints, cluster ID, and profile ID.

- For broadcast transmissions, set the 64-bit destination address to 0x000000000000FFFF .
- For unicast transmissions, set the 64 bit address field to the address of the desired destination node.
- Set the reserved field to 0xFFFE.

Query the **NP** command to read the maximum number of payload bytes. For more information, see The following AT commands are firmware commands..

You can read the maximum number of payload bytes with the **NP** command.

## *Format*

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x11 |
| Frame ID | 4 | Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response. |
| 64-bit destination address | 5-12 | MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF |
| Reserved | 13-14 | Set to 0xFFFE. |
| Source endpoint | 15 | Source endpoint for the transmission. |
| Destination endpoint | 16 | Destination endpoint for the transmission. |
| Cluster ID | 17-18 | The Cluster ID that the host uses in the transmission. |
| Profile ID | 19-20 | The Profile ID that the host uses in the transmission. |

| Frame data fields | Offset | Description |
|---|---|---|
| Broadcast radius | 21 | Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius set to the network maximum hops value.<br>If the broadcast radius exceeds the value of **NH** then the devices use the value of **NH** as the radius. Only broadcast transmissions use this parameter. |
| Transmission options | 22 | 0x01 = Disable ACK<br>0x02 = Disable network address discovery<br>Set all other bits to 0. |
| Data payload | 23-n | Up to **NP** bytes per packet. Sent to the destination device. |

Set all other bits to 0.

### *Example*

The following example sends a data transmission to a device with:

- 64-bit address: 0x0013A200 01238400
- Source endpoint: 0xE8
- Destination endpoint: 0xE8
- Cluster ID: 0x11
- Profile ID: 0xC105
- Payload: TxData

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x1A |
| Frame type | 3 | 0x11 |
| Frame ID | 4 | 0x01 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit destination address | MSB 5 | 0x00 |
| | 6 | 0x13 |
| | 7 | 0xA2 |
| | 8 | 0x00 |
| | 9 | 0x01 |
| | 10 | 0x23 |
| | 11 | 0x84 |
| | LSB12 | 0x00 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| Source endpoint | 15 | 0xE8 |
| Destination endpoint | 16 | 0xE8 |
| Cluster ID | 17 | 0x00 |
| | 18 | 0x11 |
| Profile ID | 19 | 0xC1 |
| | 20 | 0x05 |
| Broadcast radius | 21 | 0x00 |
| Transmit options | 22 | 0x00 |
| Data payload | 23 | 0x54 |
| | 24 | 0x78 |
| | 25 | 0x44 |
| | 26 | 0x61 |
| | 27 | 0x74 |
| | 28 | 0x61 |
| Checksum | 29 | 0xA6 |
| Checksum | 29 | 0xDD |

## Remote AT Command Request frame - 0x17

### Description

Used to query or set device parameters on a remote device. For parameter changes on the remote device to take effect, you must apply changes, either by setting the Apply Changes options bit, or by sending an **AC** command to the remote.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x17 |
| Frame ID | 4 | Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response. |
| 64-bit destination address | 5-12 | MSB first, LSB last. Set to the 64-bit address of the destination device. |
| Reserved | 13-14 | Set to 0xFFFE. |
| Remote command options | 15 | 0x02 = Apply changes on remote. If you do not set this, you must send the **AC** command for changes to take effect.<br>Set all other bits to 0. |
| AT command | 16-17 | Command name: two ASCII characters that identify the command. |
| Command parameter | 18-n | If present, indicates the parameter value you request for a given register. If no characters are present, it queries the register. Numeric parameter values are given in binary format. |

### Example

The following example sends a remote command to:

- Change the broadcast hops register on a remote device to 1 (broadcasts go to 1-hop neighbors only).
- Apply changes so the new configuration value takes effect immediately.

In this example, the 64-bit address of the remote device is 0x0013A200 40401122.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x10 |
| Frame type | 3 | 0x17 |
| Frame ID | 4 | 0x01 |
| 64-bit destination address | MSB 5 | 0x00 |
| | 6 | 0x13 |
| | 7 | 0xA2 |
| | 8 | 0x00 |
| | 9 | 0x40 |
| | 10 | 0x40 |
| | 11 | 0x11 |
| | LSB 12 | 0x22 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| Remote command options | 15 | 0x02 (apply changes) |
| AT command | 16 | 0x42 (B) |
| | 17 | 0x48 (H) |
| Command parameter | 18 | 0x01 |
| Checksum | 19 | 0xF5 |

## AT Command Response frame - 0x88

### *Description*

A device sends this frame in response to an AT Command (0x08 or 0x09) frame. Some commands send back multiple frames; for example, the **ND** command.

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x88 |
| Frame ID | 4 | Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response. |
| AT command | 5-6 | Command name: two ASCII characters that identify the command. |
| Command status | 7 | 0 = OK<br>1 = ERROR<br>2 = Invalid command<br>3 = Invalid parameter |
| Command data | 8-n | The register data in binary format. If the host sets the register, the device does not return this field. |

### *Example*

If you change the **BD** parameter on a local device with a frame ID of 0x01, and the parameter is valid, the user receives the following response.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x05 |
| Frame type | 3 | 0x88 |
| Frame ID | 4 | 0x01 |
| AT command | 5 | 0x42 (B) |
| | 6 | 0x44 (D) |
| Command status | 7 | 0x00 |
| Command data | | |
| Checksum | 8 | 0xF0 |

## Modem Status frame - 0x8A

### *Description*

Devices send the status messages in this frame in response to specific conditions.

### *Format*

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8A |
| Status | 4 | 0x00 = Hardware reset<br>0x01 = Watchdog timer reset<br>0x0B = Network woke up<br>0x0C = Network went to sleep |

### *Example*

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| LSB 2 | LSB 2 | 0x02 |
| Frame type | 3 | 0x8A |
| Status | 4 | 0x00 |
| Checksum | 5 | 0x75 |

## Transmit Status frame - 0x8B

### Description

When a Transmit Request (0x10, 0x11) completes, the device sends a Transmit Status message out of the serial interface. This message indicates if the Transmit Request was successful or if it failed.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8B |
| Frame ID | 4 | Identifies the serial interface data frame being reported. If Frame ID = 0 in the associated request frame, no response frame is delivered. |
| 16-bit destination address | 5 | The 16-bit Network Address where the packet was delivered (if successful). If not successful, this address is 0xFFFD (destination address unknown). |
| | 6 | |
| Transmit retry count | 7 | The number of application transmission retries that occur. |
| Delivery status | 8 | 0x00 = Success<br>0x01 = MAC ACK failure<br>0x02 = Collision avoidance failure<br>0x21 = Network ACK failure<br>0x25 = Route not found<br>0x31 = Internal resource error<br>0x32 = Internal error |
| Discovery status | 9 | 0x00 = No discovery overhead<br>0x02 = Route discovery |

### Example

In the following example, the destination device reports a successful unicast data transmission successful and a route discovery occurred. The outgoing Transmit Request that this response frame uses Frame ID of 0x47.

| Frame Fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x07 |
| Frame type | 3 | 0x8B |

| Frame Fields | Offset | Example |
|---|---|---|
| Frame ID | 4 | 0x47 |
| Reserved | 5 | 0xFF |
| | 6 | 0xFE |
| Transmit retry count | 7 | 0x00 |
| Delivery status | 8 | 0x00 |
| Discovery status | 9 | 0x02 |
| Checksum | 10 | 0x2E |

## Route Information Packet frame - 0x8D

### Description

If you enable NACK or the Trace Route option on a DigiMesh unicast transmission, a device can output this frame for the transmission.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8D |
| Source event | 4 | 0x11 = NACK<br>0x12 = Trace route |
| Length | 5 | The number of bytes that follow, excluding the checksum. If the length increases, new items have been added to the end of the list for future revisions. |
| Timestamp | 6-9 | 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. |
| ACK timeout count | 10 | The number of MAC ACK timeouts that occur. |
| TX blocked count | 11 | The number of times the transmission was blocked due to reception in progress. |
| Reserved | 12 | Reserved, set to 0s. |
| Destination address | 13-20 | The address of the final destination node of this network-level transmission. |
| Source address | 21-28 | Address of the source node of this network-level transmission. |
| Responder address | 29-36 | Address of the node that generates this Route Information packet after it sends (or attempts to send) the packet to the next hop (the Receiver node). |
| Receiver address | 37-44 | Address of the node that the device sends (or attempts to send) the data packet. |

### Example

The following example represents a possible Route Information Packet. A device receives the packet when it performs a trace route on a transmission from one device (serial number 0x0013A200 4052AAAA) to another (serial number 0x0013A200 4052DDDD).

This particular frame indicates that the network successfully forwards the transmission from one device (serial number 0x0013A200 4052BBBB) to another device (serial number 0x0013A200 4052CCCC).

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x2A |
| Frame type | 3 | 0x8D |
| Source event | 4 | 0x12 |
| Length | 5 | 0x27 |
| Timestamp | MSB 6 | 0x9C |
| | 7 | 0x93 |
| | 8 | 0x81 |
| | LSB 9 | 0x7F |
| ACK timeout count | 10 | 0x00 |
| TX blocked count | 11 | 0x00 |
| Reserved | 12 | 0x00 |
| Destination address | MSB 13 | 0x00 |
| | 14 | 0x13 |
| | 15 | 0xA2 |
| | 16 | 0x00 |
| | 17 | 0x40 |
| | 18 | 0x52 |
| | 19 | 0xAA |
| | LSB 20 | 0xAA |
| Source address | MSB 21 | 0x00 |
| | 22 | 0x13 |
| | 23 | 0xA2 |
| | 24 | 0x00 |
| | 25 | 0x40 |
| | 26 | 0x52 |
| | 27 | 0xDD |
| | LSB 28 | 0xDD |

| Frame data fields | Offset | Example |
|---|---|---|
| Responder address | MSB 29 | 0x00 |
| | 30 | 0x13 |
| | 31 | 0xA2 |
| | 32 | 0x00 |
| | 33 | 0x40 |
| | 34 | 0x52 |
| | 35 | 0xBB |
| | LSB 36 | 0xBB |
| Receiver address | MSB 37 | 0x00 |
| | 38 | 0x13 |
| | 39 | 0xA2 |
| | 40 | 0x00 |
| | 41 | 0x40 |
| | 42 | 0x52 |
| | 43 | 0xCC |
| | LSB 44 | 0xCC |
| Checksum | 45 | 0xD2 |

## Aggregate Addressing Update frame - 0x8E

### Description

The device sends out an Aggregate Addressing Update frame on the serial interface of an API-enabled node when an address update frame (generated by the **AG** command being issued on a node in the network) causes the node to update its **DH** and **DL** registers.

For more information, refer to Establish and maintain network links.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8E |
| Format ID | 4 | Byte reserved to indicate the format of additional packet information which may be added in future firmware revisions. In the current firmware revision, this field returns 0x00. |
| New address | 5-12 | Address to which **DH** and **DL** are being set. |
| Old address | 13-20 | Address to which **DH** and **DL** were previously set. |

### Example

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x12 |
| Frame type | 3 | 0x8E |
| Format ID | 4 | 0x00 |

| Frame data fields | Offset | Example |
|---|---|---|
| New address | MSB 5 | 0x00 |
| | 6 | 0x13 |
| | 7 | 0xA2 |
| | 8 | 0x00 |
| | 9 | 0x40 |
| | 10 | 0x52 |
| | 11 | 0xBB |
| | LSB 12 | 0xBB |
| Old address | 13 | 0x00 |
| | 14 | 0x13 |
| | 15 | 0xA2 |
| | 16 | 0x00 |
| | 17 | 0x40 |
| | 18 | 0x52 |
| | 19 | 0xAA |
| | 20 | 0xAA |
| Checksum | 21 | 0x19 |

## Receive Packet frame - 0x90

### Description

When a device configured with a standard API Rx Indicator (**AO** = 0) receives an RF data packet, it sends it out the serial interface using this message type.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x90 |
| 64-bit source address | 4-11 | The sender's 64-bit address. MSB first, LSB last. |
| Reserved | 12-13 | Reserved. |
| Receive options | 14 | Bit field:<br>0x01 = Packet acknowledged<br>0x02 = Packet was a broadcast packet |
| Received data | 15-n | The RF data the device receives. |

### Example

In the following example, a device with a 64-bit address of 0x0013A200 40522BAA sends a unicast data transmission to a remote device with payload RxData. If **AO**=0 on the receiving device, it sends the following frame out its serial interface.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x12 |
| Frame type | 3 | 0x90 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source address | MSB 4 | 0x00 |
| | 5 | 0x13 |
| | 6 | 0xA2 |
| | 7 | 0x00 |
| | 8 | 0x40 |
| | 9 | 0x52 |
| | 10 | 0x2B |
| | LSB 11 | 0xAA |
| Reserved | 12 | 0xFF |
| | 13 | 0xFE |
| Receive options | 14 | 0x01 |
| Received data | 15 | 0x52 |
| | 16 | 0x78 |
| | 17 | 0x44 |
| | 18 | 0x61 |
| | 19 | 0x74 |
| | 20 | 0x61 |
| Checksum | 21 | 0x11 |

# Explicit Rx Indicator frame - 0x91

## Description

When a device configured with explicit API Rx Indicator (**AO** = 1) receives an RF packet, it sends it out the serial interface using this message type.

## Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x91 |
| 64-bit source address | 4-11 | MSB first, LSB last. The sender's 64-bit address. |
| Reserved | 12-13 | Reserved. |
| Source endpoint | 14 | Endpoint of the source that initiates transmission. |
| Destination endpoint | 15 | Endpoint of the destination where the message is addressed. |
| Cluster ID | 16-17 | The Cluster ID where the frame is addressed. |
| Profile ID | 18-19 | The Profile ID where the fame is addressed. |
| Receive options | 20 | Bit field:<br>0x01 = Packet acknowledged<br>0x02 = Packet was a broadcast packet<br>Ignore all other bits. |
| Received data | 21-n | Received RF data. |

## Example

In the following example, a device with a 64-bit address of 0x0013A200 40522BAA sends a broadcast data transmission to a remote device with payload RxData.

If a device sends the transmission:

- With source and destination endpoints of 0xE0
- Cluster ID = 0x2211
- Profile ID = 0xC105

If **AO** = 1 on the receiving device, it sends the following frame out its serial interface.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x18 |

| Frame data fields | Offset | Example |
|---|---|---|
| Frame type | 3 | 0x91 |
| 64-bit source address | MSB 4 | 0x00 |
| | 5 | 0x13 |
| | 6 | 0xA2 |
| | 7 | 0x00 |
| | 8 | 0x40 |
| | 9 | 0x52 |
| | 10 | 0x2B |
| | LSB 11 | 0xAA |
| Reserved | 12 | 0xFF |
| | 13 | 0xFE |
| Source endpoint | 14 | 0xE0 |
| Destination endpoint | 15 | 0xE0 |
| Cluster ID | 16 | 0x22 |
| | 17 | 0x11 |
| Profile ID | 18 | 0xC1 |
| | 19 | 0x05 |
| Receive options | 20 | 0x02 |
| Received data | 21 | 0x52 |
| | 22 | 0x78 |
| | 23 | 0x44 |
| | 24 | 0x61 |
| | 25 | 0x74 |
| | 26 | 0x61 |
| Checksum | 27 | 0x68 |

## Data Sample Rx Indicator frame - 0x92

### *Description*

When you enable periodic I/O sampling or digital I/O change detection on a remote device, the UART of the device that receives the sample data sends this frame out.

### *Format*

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | |
| 64-bit source address | 4-11 | The sender's 64-bit address. |
| Reserved | 12-13 | Reserved. |
| Receive options | 14 | Bit field:<br>0x01 = Packet acknowledged<br>0x02 = Packet is a broadcast packet<br>Ignore all other bits |
| Number of samples | 15 | The number of sample sets included in the payload. Always set to 1. |
| Digital channel mask | 16-17 | Bitmask field that indicates which digital I/O lines on the 17 0x1C remote have sampling enabled, if any. |
| Analog channel mask | 18 | Bitmask field that indicates which analog I/O lines on the remote have sampling enabled, if any. |
| Digital samples (if included) | 19-20 | If the sample set includes any digital I/O lines (Digital channel mask > 0), these two bytes contain samples for all enabled digital I/O lines. DIO lines that do not have sampling enabled return 0. Bits in these two bytes map the same as they do in the Digital channel mask field. |
| Analog sample | 21-22 | If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from AD0/DIO0 to AD3/DIO3. |

### *Example*

In the following example, the device receives an I/O sample from a device with a 64-bit serial number of 0x0013A20040522BAA.

The configuration of the transmitting device takes a digital sample of a number of digital I/O lines and an analog sample of AD1. It reads the digital lines to be 0x0014 and the analog sample value is 0x0225.

The complete example frame is:

7E00 1492 0013 A200 4052 2BAA FFFE 0101 001C 0200 1402 25F9

| Frame fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x14 |
| Frame-specific data | | |
| 64-bit source address | MSB 4 | 0x00 |
| | 5 | 0x13 |
| | 6 | 0xA2 |
| | 7 | 0x00 |
| | 8 | 0x40 |
| | 9 | 0x52 |
| | 10 | 0x2B |
| | LSB 11 | 0xAA |
| Reserved | MSB 12 | 0xfffe |
| | LSB 13 | 0x84 |
| Receive options | 14 | 0x01 |
| Number of samples | 15 | 0x01 |
| Digital channel mask | 16 | 0x00 |
| | 17 | 0x1C |
| Analog channel mask | 18 | 0x02 |
| Digital samples (if included) | 19 | 0x00 |
| | 20 | 0x14 |
| Analog sample | 21 | 0x02 |
| | 22 | 0x25 |
| Checksum | 23 | 0xF5 |

## Node Identification Indicator frame - 0x95

### *Description*

A device receives this frame when:

- it transmits a node identification message to identify itself
- AO=0

The data portion of this frame is similar to a network discovery response. For more information, see ND (Network Discover).

### *Format*

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x95 |
| 64-bit source address | 4-11 | MSB first, LSB last. The sender's 64-bit address. |
| Reserved | 12-13 | Reserved. |
| Receive options | 14 | Bit field:<br>0x01 = Packet acknowledged<br>0x02 = Packet was a broadcast packet<br>0x40 = Point-multipoint packet<br>0x80 = Directed broadcast packet<br>0xC0 = DigiMesh packet<br>Ignore all other bits |
| Reserved | 15-16 | Reserved. |
| 64-bit remote address | 17-24 | Indicates the 64-bit address of the remote device that transmitted the Node Identification Indicator frame. |
| NI string | 25-26 | Node identifier string on the remote device. The NI string is terminated with a NULL byte (0x00). |
| Reserved | 27-28 | Reserved. |
| Device type | 29 | 0=Coordinator<br>1=Normal Mode<br>2=End Device<br>For more options, see NO (Network Discovery Options). |
| Source event | 30 | 1=Frame sent by node identification pushbutton event - See D0 (DIO0/AD0). |
| Digi Profile ID | 31-32 | Set to the Digi application profile ID. |

| Frame data fields | Offset | Description |
|---|---|---|
| Digi Manufacturer ID | 33-34 | Set to the Digi Manufacturer ID. |
| Digi DD value (optional) | 35-38 | Reports the **DD** value of the responding device. Use the **NO** command to enable this field. |
| RSSI (optional) | 39 | Received signal strength indicator. Use the **NO** command to enable this field, . |

### *Example*

If you press the commissioning pushbutton on a remote device with 64-bit address 0x0013A200407402AC and a default **NI** string sends a Node Identification, all devices on the network receive the following node identification indicator:

A remote device with 64-bit address 0x0013A200407402AC and a default **NI** string sends a Node Identification, all devices on the network receive the following node identification indicator:

```
0x7e 0025 9500 13a2 0040 7402 acff fec2 fffe 0013 a200 4074 02ac 2000 fffe 0101
c105 101e
```

If you press the commissioning button on a remote router device with 64-bit address 0x0013A200 40522BAA, 16-bit address 0x7D84, and default **NI** string, devices on the network receive the node identification indicator.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x25 |
| Frame type | 3 | 0x95 |
| 64-bit source address | MSB 4 | 0x00 |
| | 5 | 0x13 |
| | 6 | 0xA2 |
| | 7 | 0x00 |
| | 8 | 0x40 |
| | 9 | 0x74 |
| | 10 | 0x02 |
| | LSB 11 | 0xAC |
| Reserved | 12 | 0xFF |
| | 13 | 0xFE |

| Frame data fields | Offset | Example |
|---|---|---|
| Receive options | 14 | 0xC2 |
| Reserved | 15 | 0xFF |
| | 16 | 0xFE |
| 64-bit remote address | MSB 17 | 0x00 |
| | 18 | 0x13 |
| | 19 | 0xA2 |
| | 20 | 0x00 |
| | 21 | 0x40 |
| | 22 | 0x74 |
| | 23 | 0x02 |
| | LSB 24 | 0xAC |
| NI string | 25 | 0x20 |
| | 26 | 0x00 |
| Reserved | 27 | 0xFF |
| | 28 | 0xFE |
| Device type | 29 | 0x01 |
| Source event | 30 | 0x01 |
| Digi Profile ID | 31 | 0xC1 |
| | 32 | 0x05 |
| Digi Manufacturer ID | 33 | 0x10 |
| | 34 | 0x1E |
| Digi DD value (optional) | 35 | 0x00 |
| | 36 | 0x0C |
| | 37 | 0x00 |
| | 38 | 0x00 |
| RSSI (optional) | 39 | 0x2E |
| Checksum | 40 | 0x33 |

## Remote Command Response frame - 0x97

### Description

If a device receives this frame in response to a Remote Command Request (0x17) frame, the device sends an AT Command Response (0x97) frame out the serial interface.

Some commands, such as the **ND** command, may send back multiple frames.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x97 |
| Frame ID | 4 | This is the same value that is passed in to the request. |
| 64-bit source (remote) address | 5-12 | The address of the remote device returning this response. |
| Reserved | 13-14 | Reserved. |
| AT commands | 15-16 | The name of the command. |
| Command status | 17 | 0 = OK<br>1 = ERROR<br>2 = Invalid Command<br>3 = Invalid Parameter |
| Command data | 18-n | The value of the requested register. |

### Example

If a device sends a remote command to a remote device with 64-bit address 0x0013A200 40522BAA to query the **SL** command, and if the frame ID = 0x55, the response would look like the following example.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x13 |
| Frame type | 3 | 0x97 |
| Frame ID | 4 | 0x55 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source (remote) address | MSB 5 | 0x00 |
| | 6 | 0x13 |
| | 7 | 0xA2 |
| | 8 | 0x00 |
| | 9 | 0x40 |
| | 10 | 0x52 |
| | 11 | 0x2B |
| | LSB 12 | 0xAA |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| 16-bit source (remote) address | MSB 13 | 0x7D |
| | LSB 14 | 0x84 |
| AT commands | 15 | 0x53 (S) |
| | 16 | 0x4C (L) |
| Command status | 17 | 0x00 |
| Command data | 18 | 0x40 |
| | 19 | 0x52 |
| | 20 | 0x2B |
| | 21 | 0xAA |
| Checksum | 22 | 0xF4 |

# Certifications

# Agency certifications - United States

## United States (FCC)

XBee/XBee-PRO 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:

Contains FCC ID: OUR-XBEE/OUR-XBEEPRO[1]

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, please take note of the following instructions:

- For XBee modules where the antenna gain is less than 13.8 dBi, no additional SAR testing is required. The 20 cm separation distance is not required for antenna gain less than 13.8 dBi.
- For XBee modules where the antenna gain is greater than 13.8 dBi and for all XBee-PRO modules, 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.

[1]The FCC ID for the XBee is “OUR-XBEE.” The FCC ID for the XBee-PRO is “OUR-XBEEPRO.”

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.

## RF exposure statement

If you are an integrating the XBee into another product, you must include the following Caution statement in product manuals to alert users of FCC RF exposure compliance:

**CAUTION!** To satisfy FCC RF exposure requirements for mobile transmitting devices, a separation distance of 20 cm or more should be maintained between the antenna of this device and persons during device operation. To ensure compliance, operations at closer than this distance is not recommended. The antenna used for this transmitter must not be co-located in conjunction with any other antenna or transmitter.

## FCC-approved antennas (2.4 GHz)

You can install XBee and XBee-PRO RF modules using antennas and cables constructed with standard connectors (Type-N, SMA, TNC, and so forth) if you perform the installation professionally and according to FCC guidelines. If a non-professional performs the installation, you must use non-standard connectors (RPSMA, RPTNC, and so forth).

The modules are FCC-approved for fixed base station and mobile applications on channels 0x0B - 0x1A (XBee) and 0x0C - 0x17 (XBee-PRO). If you mount the antenna at least 20 cm (8 in) from nearby persons, the FCC considers the application to be a mobile application. You must test antennas that are not in the table to comply with FCC Section 15.203 (Unique Antenna Connectors) and Section 15.247 (Emissions).

**XBee RF Modules** (1 mW): XBee modules are tested and approved for use with the antennas listed in the first and second tables below.

**XBee-PRO RF Modules** (63 mW): XBee-PRO modules are tested and approved for use with the antennas listed in the first and third tables below.

The antennas in the following tables are approved for use with this module. We do not carry all of these antenna variants. Contact Digi Sales for the available antennas.

**Antennas approved for use with the XBee/XBee-PRO RF Modules (cable loss is not required)**

| Part number | Type (description) | Gain | Application* | Minimum separation |
|---|---|---|---|---|
| A24-HASM-450 | Dipole (half-wave articulated RPSMA - 4.5”) | 2.1 dBi | Fixed/mobile | 20 cm |
| 29000095 | Dipole (half-wave articulated RPSMA - 4.5”) | 2.1 dBi | Fixed/mobile | 20 cm |

| Part number | Type (description) | Gain | Application* | Minimum separation |
|---|---|---|---|---|
| A24-HABSM | Dipole (articulated RPSMA) | 2.1 dBi | Fixed | 20 cm |
| A24-HABUF-P5I | Dipole (half-wave articulated bulkhead mount U.FL. with 5” pigtail) | 2.1 dBi | Fixed | 20 cm |
| A24-HASM-525 | Dipole (half-wave articulated RPSMA - 5.25") | 2.1 dBi | Fixed/mobile | 20 cm |
| A24-QI | Monopole (integrated whip) | 1.5 dBi | Fixed | 20 cm |
| A24-C1 | Surface-mount | -1.5 dBi | Fixed/mobile | 20 cm |
| 29000430 | Integrated PCB Antenna | -0.5 dBi | Fixed/mobile | 20 cm |
| * If you are using the RF module in a portable application or if the module is used in a handheld device and the antenna is less than 20 cm from the human body when the device is in operation: The integrator may be responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1091 and FCC Guidelines for Human Exposure to Radiofrequency Electromagnetic Fields, OET Bulletin and Supplement C. See the note under FCC notices for more information. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person. | | | | |

**Antennas approved for use with the XBee RF Modules (cable loss is shown if required)**

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| Yagi class antennas | | | | | |
| A24-Y4NF | Yagi (4-element) | 6.0 dBi | Fixed | 2 m | |
| A24-Y6NF | Yagi (6-element) | 8.8 dBi | Fixed | 2 m | 1.7 dB |
| A24-Y7NF | Yagi (7-element) | 9.0 dBi | Fixed | 2 m | 1.9 dB |
| A24-Y9NF | Yagi (9-element) | 10.0 dBi | Fixed | 2 m | 2.9 dB |
| A24-Y10NF | Yagi (10-element) | 11.0 dBi | Fixed | 2 m | 3.9 dB |

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| A24-Y12NF | Yagi (12-element) | 12.0 dBi | Fixed | 2 m | 4.9 dB |
| A24-Y13NF | Yagi (13-element) | 12.0 dBi | Fixed | 2 m | 4.9 dB |
| A24-Y15NF | Yagi (15-element) | 12.5 dBi | Fixed | 2 m | 5.4 dB |
| A24-Y16NF | Yagi (16-element) | 13.5 dBi | Fixed | 2 m | 6.4 dB |
| A24-Y16RM | Yagi (16-element, RPSMA connector) | 13.5 dBi | Fixed | 2 m | 6.4 dB |
| A24-Y18NF | Yagi (18-element) | 15.0 dBi | Fixed | 2 m | 7.9 dB |
| Omni-directional class antennas | | | | | |
| A24-F2NF | Omni-directional (fiberglass base station) | 2.1 dBi | Fixed/mobile | 20 cm | |
| A24-F3NF | Omni-directional (fiberglass base station) | 3.0 dBi | Fixed/mobile | 20 cm | |
| A24-F5NF | Omni-directional (fiberglass base station) | 5.0 dBi | Fixed/mobile | 20 cm | |
| A24-F8NF | Omni-directional (fiberglass base station) | 8.0 dBi | Fixed | 2 m | |
| A24-F9NF | Omni-directional (fiberglass base station) | 9.5 dBi | Fixed | 2 m | 0.2 dB |
| A24-F10NF | Omni-directional (fiberglass base station) | 10.0 dBi | Fixed | 2 m | 0.7 dB |

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| A24-F12NF | Omni-directional (fiberglass base station) | 12.0 dBi | Fixed | 2 m | 2.7 dB |
| A24-F15NF | Omni-directional (fiberglass base station) | 15.0 dBi | Fixed | 2 m | 5.7 dB |
| A24-W7NF | Omni-directional (base station) | 7.2 dBi | Fixed | 2 m | |
| A24-M7NF | Omni-directional (mag-mount base station) | 7.2 dBi | Fixed | 2 m | |
| Panel class antennas | | | | | |
| A24-P8SF | Flat panel | 8.5 dBi | Fixed | 2 m | 1.5 dB |
| A24-P8NF | Flat panel | 8.5 dBi | Fixed | 2 m | 1.5 dB |
| A24-P13NF | Flat panel | 13.0 dBi | Fixed | 2 m | 6 dB |
| A24-P14NF | Flat panel | 14.0 dBi | Fixed | 2 m | 7 dB |
| A24-P15NF | Flat panel | 15.0 dBi | Fixed | 2 m | 8 dB |
| A24-P16NF | Flat panel | 16.0 dBi | Fixed | 2 m | 9 dB |
| * If you are using the RF module in a portable application or if the module is used in a handheld device and the antenna is less than 20 cm from the human body when the device is in operation: The integrator may be responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1091 and FCC Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields, OET Bulletin and Supplement C. See the note under FCC notices for more information. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person. | | | | | |

### *High gain antenna summary*

The following antenna types have been tested and approved for use with the XBee module:

**Antenna type: Yagi** The XBee RF Module was tested and approved with 15 dBi antenna gainand with 7.9 dB cable loss. You can use any Yagi type antenna with 7.1 dBi gain or less with no cable loss.

**Antenna type: omni-directional** The XBee RF Module was tested and approved with 15 dBi antenna gain with 5.7 dB cable loss. You can use any omni-directional antenna with 9.3 dBi gain or less with no cable loss.

**Antenna type: flat panel** The XBee RF Module was tested and approved with 16 dBi antenna gain with 9.0 dB cable loss. You can use any flat panel antenna with 7.0 dBi gain or less with no cable loss.

**Antennas approved for use with the XBee-PRO DigiMesh 2.4 RF Modules (cable loss is required)**

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| Yagi class antennas | | | | | |
| A24-Y4NF | Yagi (4-element) | 6.0 dBi | Fixed | 2 m | 8.1 dB |
| A24-Y6NF | Yagi (6-element) | 8.8 dBi | Fixed | 2 m | 10.9 dB |
| A24-Y7NF | Yagi (7-element) | 9.0 dBi | Fixed | 2 m | 11.1 dB |
| A24-Y9NF | Yagi (9-element) | 10.0 dBi | Fixed | 2 m | 12.1 dB |
| A24-Y10NF | Yagi (10-element) | 11.0 dBi | Fixed | 2 m | 13.1 dB |
| A24-Y12NF | Yagi (12-element) | 12.0 dBi | Fixed | 2 m | 14.1 dB |
| A24-Y13NF | Yagi (13-element) | 12.0 dBi | Fixed | 2 m | 14.1 dB |
| A24-Y15NF | Yagi (15-element) | 12.5 dBi | Fixed | 2 m | 14.6 dB |
| A24-Y16NF | Yagi (16-element) | 13.5 dBi | Fixed | 2 m | 15.6 dB |
| A24-Y16RM | Yagi (16-element, RPSMA connector) | 13.5 dBi | Fixed | 2 m | 15.6 dB |
| A24-Y18NF | Yagi (18-element) | 15.0 dBi | Fixed | 2 m | 17.1 dB |
| Omni-directional class antennas | | | | | |
| A24-F2NF | Omni-directional (fiberglass base station) | 2.1 dBi | Fixed/Mobile | 20 cm | 4.2 dB |
| A24-F3NF | Omni-directional (fiberglass base station) | 3.0 dBi | Fixed/Mobile | 20 cm | 5.1 dB |

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| A24-F5NF | Omni-directional (fiberglass base station) | 5.0 dBi | Fixed/Mobile | 20 cm | 7.1 dB |
| A24-F8NF | Omni-directional (fiberglass base station) | 8.0 dBi | Fixed | 2 m | 10.1 dB |
| A24-F9NF | Omni-directional (fiberglass base station) | 9.5 dBi | Fixed | 2 m | 11.6 dB |
| A24-F10NF | Omni-directional (fiberglass base station) | 10.0 dBi | Fixed | 2 m | 12.1 dB |
| A24-F12NF | Omni-directional (fiberglass base station) | 12.0 dBi | Fixed | 2 m | 14.1 dB |
| A24-F15NF | Omni-directional (fiberglass base station) | 15.0 dBi | Fixed | 2 m | 17.1 dB |
| A24-W7NF | Omni-directional (base station) | 7.2 dBi | Fixed | 2 m | 9.3 dB |
| A24-M7NF | Omni-directional (mag-mount base station) | 7.2 dBi | Fixed | 2 m | 9.3 dB |
| Panel class antennas | | | | | |
| A24-P8SF | Flat panel | 8.5 dBi | Fixed | 2 m | 8.6 dB |
| A24-P8NF | Flat panel | 8.5 dBi | Fixed | 2 m | 8.6 dB |
| A24-P13NF | Flat panel | 13.0 dBi | Fixed | 2 m | 13.1 dB |
| A24-P14NF | Flat panel | 14.0 dBi | Fixed | 2 m | 14.1 dB |
| A24-P15NF | Flat panel | 15.0 dBi | Fixed | 2 m | 15.1 dB |
| A24-P16NF | Flat panel | 16.0 dBi | Fixed | 2 m | 16.1 dB |

| Part number | Type (description) | Gain | Application* | Minimum separation | Required cable loss |
|---|---|---|---|---|---|
| A24-P19NF | Flat panel | 19.0 dBi | Fixed | 2 m | 19.1 dB |

* If you are using the RF module in a portable application or if the module is used in a handheld device and the antenna is less than 20 cm from the human body when the device is in operation: The integrator may be responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1091 and FCC Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields, OET Bulletin and Supplement C. See the note under FCC notices for more information. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person.

# Agency certifications - Australia (C-Tick)

These products comply with requirements to be used in end products in Australia. All products with EMC and radio communications must have a registered C-Tick mark. Registration to use the compliance mark will only be accepted from Australian manufacturers or importers, or their agent, in Australia.

## Labeling requirements

In order to place a C-Tick mark on an end product, a company must comply with 1 or 2 below:

1. Have a company presence in Australia.
2. Have a company, distributor, or agent in Australia that will sponsor importing the end product.

Contact Digi for questions about locating a contact in Australia.

# Agency certifications - Brazil Agência Nacional de Telecomunicações (Anatel)

The XBee RF modules with 802.15.4 or DigiMesh firmware (models noted in the following conformity information) comply with Brazil ANATEL standards in Resolution No. 506. The following information is required in the user manual for the product containing the radio and on the product containing the radio (in Portuguese):

The XBee-PRO RF modules with 802.15.4 or DigiMesh firmware (models noted in conformity information below) comply with Brazil ANATEL standards in Resolution No. 506. The following information is required in the user manual for the product containing the radio and on the product containing the radio (in Portuguese):

### Modelo XBee-Pro S3B:


Este equipamento opera em caráter secundário, isto é, não tem direito a proteção contra interferência prejudicial, mesmo de estações do mesmo tipo, e não pode causar interferência a sistemas operando em caráter primário."

## Agency certifications - Industry Canada (IC)

### Labeling requirements

IC requires you to place a clearly visible label on the outside of the final product enclosure, displaying the following text:

Contains Model XBee Radio, IC: 4214A-XBEE

Contains Model XBee-PRO Radio, IC: 4214A-XBEEPRO

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.

## Agency certifications - European Telecommunications Standards Institute (ETSI)

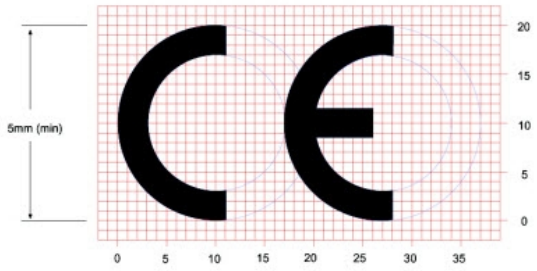
The RF Module is certified for use in several European countries; for a complete list, go to www.digi.com.

If the RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product to the European harmonized EMC and low-voltage/safety standards. A Declaration of Conformity must be issued for each of these standards and kept on file as described in Annex II of the R&TTE Directive.

Furthermore, the manufacturer must maintain a copy of the RF Module user manual documentation and ensure the final product does not exceed the specified power ratings, antenna specifications, and/or installation requirements as specified in the user manual. If any of these specifications are exceeded in the final product, a submission must be made to a notified body for compliance testing to all required standards.

## OEM labeling requirements

The “CE” marking must be affixed to a visible location on the OEM product.




The CE mark shall consist of the initials “CE” taking the following form:

- If the CE marking is reduced or enlarged, the proportions given in the above graduated drawing must be respected.
- The CE marking must have a height of at least 5 mm except where this is not possible on account of the nature of the apparatus.
- The CE marking must be affixed visibly, legibly, and indelibly.

## Restrictions

The power output of the XBee-PRO RF Modules must not exceed 10 dBm. The power level is set using the **PL** command. The International Variant of this product is internally limited to 10 dBm.

France imposes restrictions on the 2.4 GHz band. Go to www.art-telecom.Fr or contact Digi for more information.

## Declarations of conformity

Digi has issued Declarations of Conformity for the XBee RF Modules concerning emissions, EMC and safety.

**Note** Digi does not list the entire set of standards that must be met for each country. Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. For more information relating to European compliance of an OEM product incorporating the XBee RF Module, contact Digi, or refer to the following web sites:
• CEPT ERC 70-03E, technical requirements, European restrictions and general requirements: available at www.ero.dk/.
• R&TTE Directive, equipment requirements and market placement: available at www.ero.dk/.

## Approved antennas

When you integrate high-gain antennas, European regulations stipulate the EIRP power maximums. Use the following guidelines to determine which antennas to use when you design an application.

The following antenna types are tested and approved for use with the XBee Module:

### *Yagi*

RF module was tested and approved with 15 dBi antenna gain with 1 dB cable-loss (EIRP Maximum of 14 dBm). Any Yagi type antenna with 14 dBi gain or less can be used with no cable-loss.

### *Omni-directional*

RF module was tested and approved with 15 dBi antenna gain with 1 dB cable-loss (EIRP Maximum of 14 dBm). Any Omni-directional type antenna with 14 dBi gain or less can be used with no cable-loss.

### *Flat panel*

RF module was tested and approved with 19 dBi antenna gain with 4.8 dB cable-loss (EIRP Maximum of 14.2 dBm). Any Flat Panel type antenna with 14.2 dBi gain or less can be used with no cable-loss.

The embedded XBee-PRO was tested and approved for use with the following antennas:

**Note** At 10 dBm transmit power, the **PL** parameter value must equal 0, or you must use the international variant.

### *Dipole*

2.1 dBi, omni-directional, articulated RPSMA, Digi part number A24-HABSM.

### *Chip antenna*

-1.5 dBi.

### *Attached monopole whip*

1.5 dBi.

### *Integrated PCB antenna*

-0.5 dBi

The RF modem encasement was designed to accommodate the RPSMA antenna option.

# Agency certifications - Japan (Telec)

In order to use the XBee-PRO in Japan, you must order the International version. The International XBee-PRO RF Modules are limited to a transmit power output of 10 dBm .

## Labeling requirements

A clearly visible label on the outside of the final product enclosure must display the following text:

R201WW07215214 (XBee)

R201WW08215111 (XBee-PRO)