Digi XBee3® 802.15.4

Radio Frequency (RF) Module

User Guide

Revision history—90002273

Revision	Date	Description
A	April 2018	Initial release.
B	September 2018	S2C parity release.

Trademarks and copyright

Disclaimers

Warranty

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

www.digi.com/howtobuy/terms

Customer support

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

Product name and model

Product serial number (s)

Firmware version

Operating system/browser (if applicable)

Logs (from time of reported issue)

Trace (if possible)

Description of issue

Steps to reproduce

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

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (Digi XBee3® 802.15.4 RF Module User Guide, 90002273 B) in the subject line of your email.

Contents

AT commands

Digi XBee3® 802.15.4 RF Module User Guide

XBee3 802.15.4 RF Modules are embedded solutions providing wireless end-point connectivity to devices. These devices use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing.

The XBee3 802.15.4 RF Module supports the needs of low-cost, low-power wireless sensor networks. The devices require minimal power and provide reliable delivery of data between devices. The devices operate within the ISM 2.4 GHz frequency band.

The XBee3 802.15.4 RF Module uses XBee3 hardware and the Silicon Labs EFR32 chipset. As the name suggests, the 802.15.4 module is over-the-air compatible with our Legacy 802.15.4 modules (S1 and S2C hardware).

For information about XBee3 hardware, see the XBee3 RF Module Hardware Reference Manual.

Applicable firmware and hardware

This manual supports the following firmware:

- 802.15.4 version 20xx

It supports the following hardware:

- XBee3

Change the firmware protocol

You can switch the firmware loaded onto the XBee3 hardware to run any of the following protocols:

- Zigbee
- 802.15.4
- DigiMesh

To change protocols, use the **Update firmware** feature in XCTU and select the firmware. See the *XCTU User Guide*.

Regulatory information

See the Regulatory information section of the *XBee3 RF Module Hardware Reference Manual* for the XBee3 hardware's regulatory and certification information.

Get started

This section covers the following tasks and features:

Verify kit contents

The XBee3 802.15.4 RF Module development kit contains the following components:

Part	
XBee3 Zigbee SMT module (3)	
XBee Grove development board (3)	
Micro USB cable (3)	
Antenna - 2.4 GHz, half-wave dipole, 2.1 dBi, U.FL female, articulating (3)	
XBee stickers	

Assemble the hardware

This guide walks you through the steps required to assemble and disassemble the hardware components of your kit.

- Plug in the XBee3 802.15.4 RF Module
- Unplug an XBee3 802.15.4 RF Module

The kit includes several XBee Grove Development Boards. For more information about this hardware, see the XBee Grove Development Board documentation.

Plug in the XBee3 802.15.4 RF Module

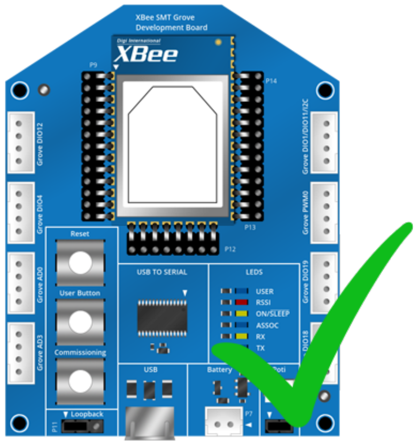
This kit includes two XBee Grove Development Boards. For more information about this hardware, visit the XBee Grove Development Board documentation.

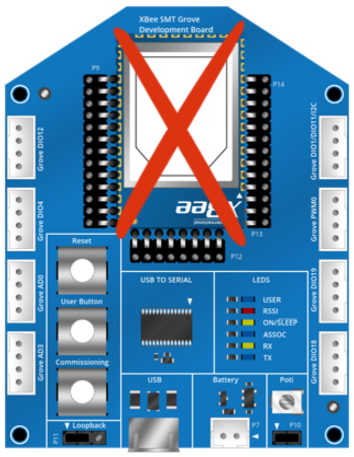
Follow these steps to connect the XBee devices to the boards included in the kit:

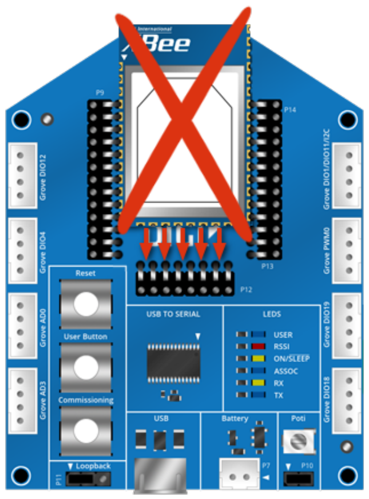
1. Plug one XBee3 802.15.4 RF Module into each XBee Grove Development Board. When you connect the development board to a PC for the first time, the PC automatically installs drivers, which may take a few minutes to complete.

   CAUTION! Never insert or remove the XBee while the power is on (either from the micro USB or a battery)!

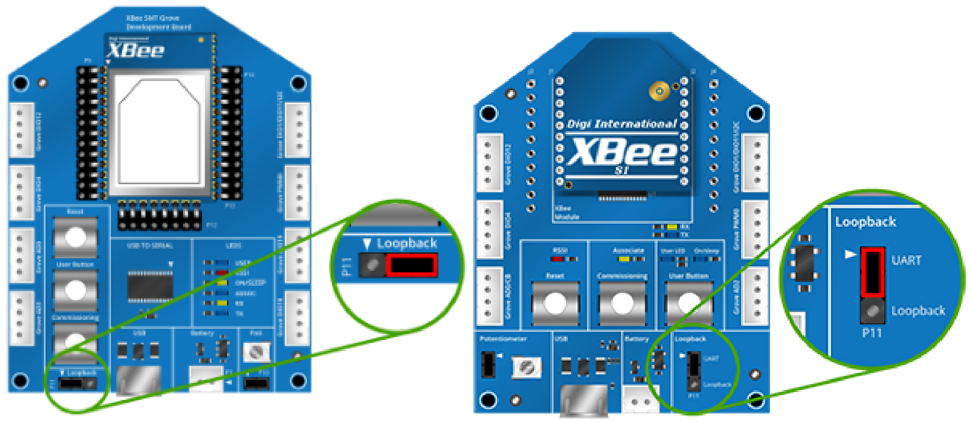
   For XBee SMT devices, align all XBee pins with the spring header and carefully push the device until it clicks firmly into the board.







2. Once theXBee3 802.15.4 RF Module is plugged into the board, connect the board to your computer using the micro USB cables provided.
3. Ensure the loopback jumper is in the UART position.



Unplug an XBee3 802.15.4 RF Module

To disconnect a device from the XBee Grove Development Board:

1. Disconnect the micro USB cable from the board so it is not powered.
2. Remove the device from the board socket, taking care not to bend any of the pins. The surface mount device uses spring pins rather than a socket and has a rectangular board cutout designed to help in removing the XBee3 802.15.4 RF Module.

CAUTION! Make sure the board is **not** powered when you remove the XBee3 802.15.4 RF Module.

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*.

Configure remote devices

You can communicate with remote devices over the air through a corresponding local device.

Note Using API mode on the local device allows you to send remote API commands.

These instructions show you how to configure the LT command parameter on a remote device.

1. Add two XBee devices to XCTU.
2. Load XBee3 802.15.4 firmware onto each device if it is not already loaded. See How to update the firmware of your modules in the *XCTU User Guide* for more information.

3. Configure the first device in API mode and name it **XBEE_A** by configuring the following parameters:

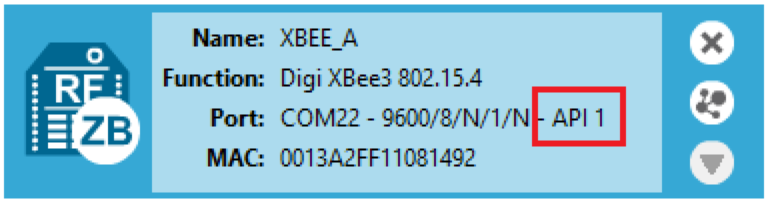
- **ID**: 2018
- **NI**: XBEE_A
- **AP**: API enabled [1]

4. Configure the second device in either API or Transparent mode, and name it **XBEE_B** by configuring the following parameters:

- **ID**: 2018
- **NI**: XBEE_B
- **AP**: 0 or 1

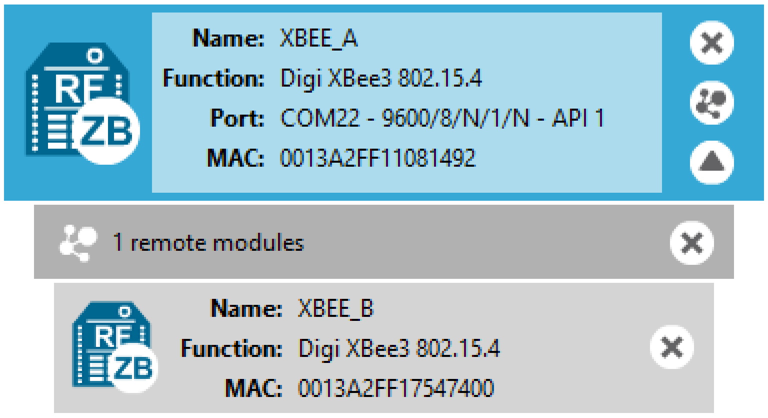
4. Disconnect XBEE_B from your computer and remove it from XCTU.
5. Connect XBEE_B to a power supply (or laptop or portable battery).

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



Name: XBEE_A
Function: Digi XBee3 802.15.4
Port: COM22 - 9600/8/N/1/N - API 1
MAC: 0013A2FF11081492

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



Name: XBEE_A
Function: Digi XBee3 802.15.4
Port: COM22 - 9600/8/N/1/N - API 1
MAC: 0013A2FF11081492

1 remote modules

Name: XBEE_B
Function: Digi XBee3 802.15.4
MAC: 0013A2FF17547400

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

   LT: FF (hexadecimal representation for 2550 ms)

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

   The remote XBee device now has a different LED blink time.

10. To return to the default LED blink times, change the **LT** parameter back to **0** for **XBEE_B**.

Configure the devices for a range test

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

   ID: 2018
   NI: LOCAL_DEVICE
   AP: API Mode Enabled [1]
4. Click the **Write radio settings** button.
5. Select the other module and click the **Default firmware settings** button.
6. Configure the following parameters:

   ID: 2018

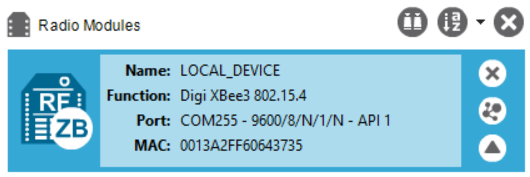
   NI: REMOTE_DEVICE

   AP: Transparent mode [0] (The remote node must be in transparent mode to loop back packets)
7. Click the **Write radio settings** button.

   After you write the radio settings for each device, their names appear in the **Radio Modules** area. The Port indicates that the LOCAL_DEVICE is in API mode.
8. Disconnect REMOTE_DEVICE from the computer, remove it from XCTU, and connect it to a power supply, laptop, or portable battery.
9. Leave LOCAL_DEVICE connected to the computer.

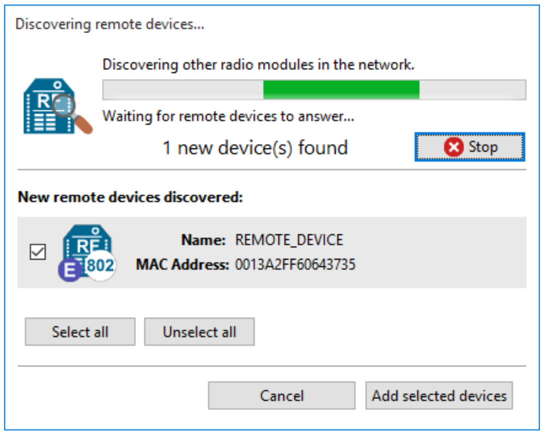
Perform a range test

1. Go to the XCTU display for radio 1.



Name: LOCAL_DEVICE
Function: Digi XBee3 802.15.4
Port: COM255 - 9600/8/N/1/N - API 1
MAC: 0013A2FF60643735

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


Discovering remote devices...

Discovering other radio modules in the network.

Waiting for remote devices to answer...

1 new device(s) found

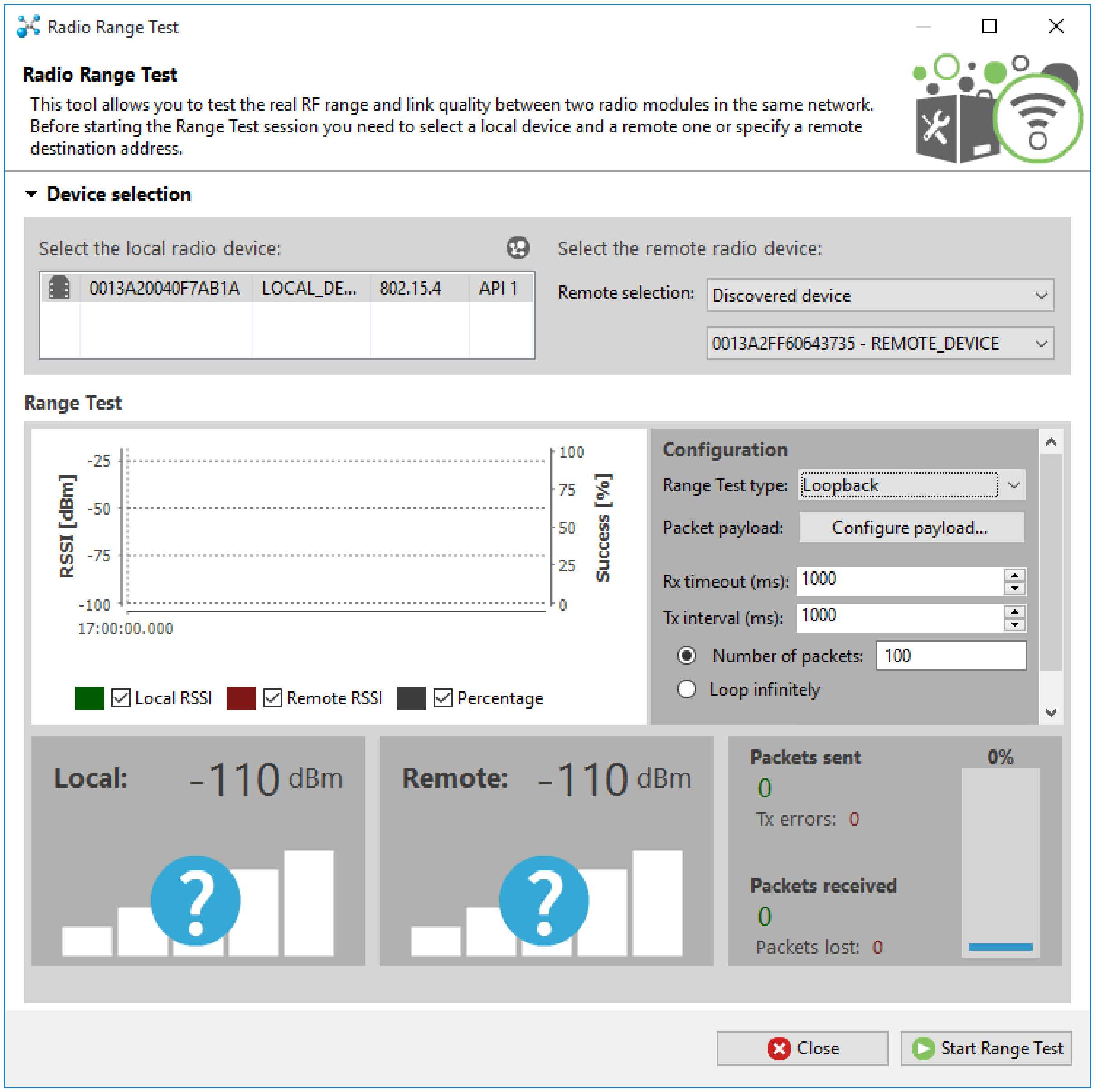
New remote devices discovered:

Name: REMOTE_DEVICE

MAC Address: 0013A2FF60643735

3. Click **Add selected devices**.

4. Click and select **Range test**. The **Radio Range Test** dialog appears.



Radio Range Test

Radio Range Test

This tool allows you to test the real RF range and link quality between two radio modules in the same network. Before starting the Range Test session you need to select a local device and a remote one or specify a remote destination address.

Device selection

Select the local radio device:

0013A20040F7AB1A	LOCAL_DE...	802.15.4	API 1

Select the remote radio device:

Remote selection: Discovered device

0013A2FF60643735 - REMOTE_DEVICE

Range Test

Configuration

Range Test type: Loopback

Packet payload: Configure payload...

Rx timeout (ms): 1000

Tx interval (ms): 1000

[✓ Number of packets: 100] Loop infinitely

Local: -110 dBm

Remote: -110 dBm

Packets sent 0 0%

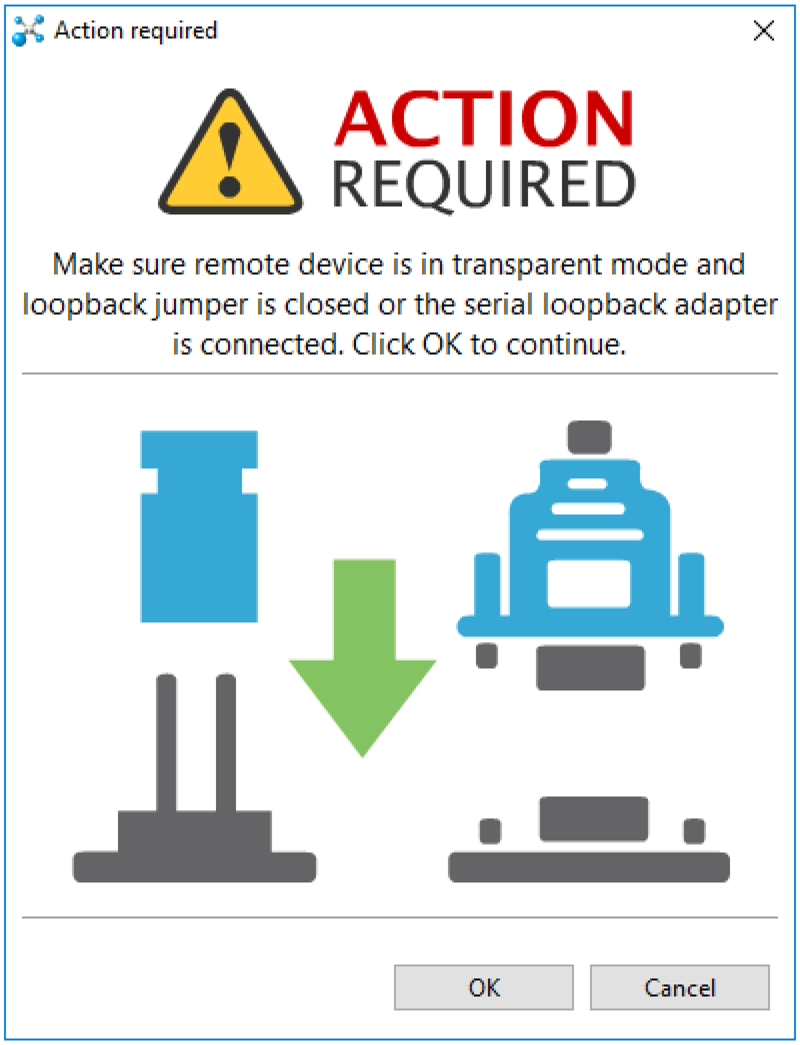
Tx errors: 0

Packets received 0

Packets lost: 0

5. Change the **Range Test type** to **Loopback**.
6. In the **Select the local radio device** area, select radio 1. XCTU automatically selects the **Discovered device** option, and the **Start Range Test** button is active.

7. Click Start Range Test to begin the range test. XCTU prompts you to enable the loopback jumper.



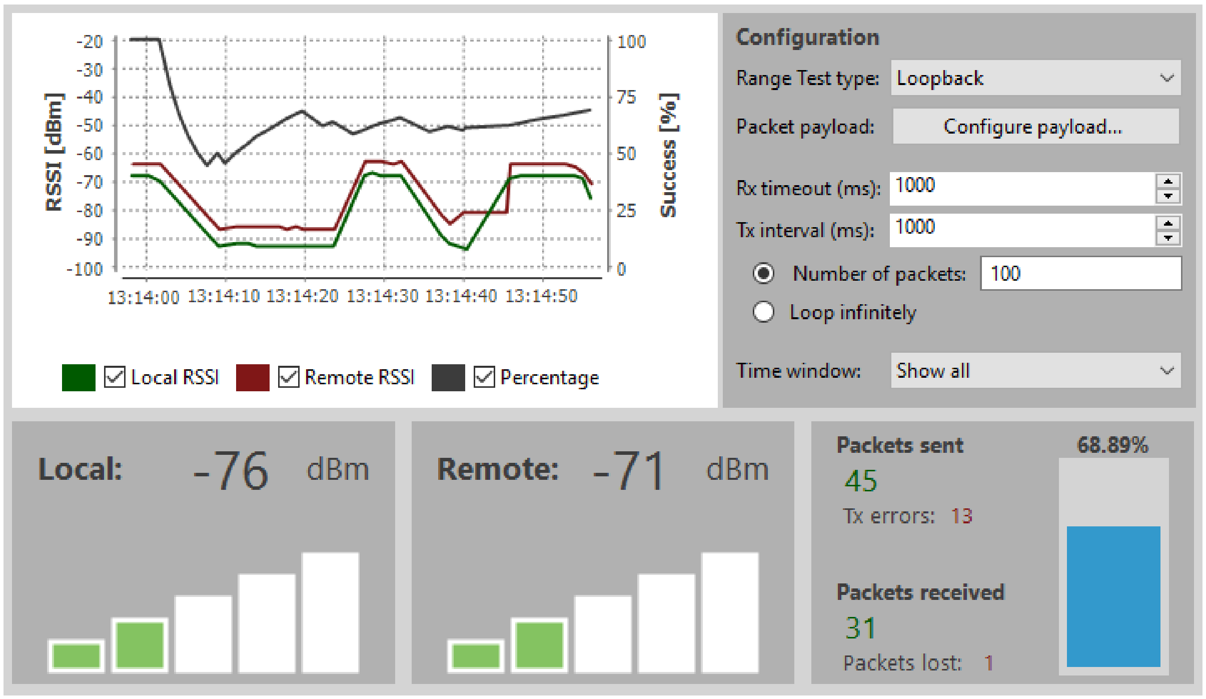
Action required

ACTION REQUIRED

Make sure remote device is in transparent mode and loopback jumper is closed or the serial loopback adapter is connected. Click OK to continue.

Plug in the XBee3 802.15.4 RF Module has pictures that show the jumper in the UART position—move the jumper to the left on the surface-mount device or down on the through-hole device puts it in loopback mode

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



Configuration

Range Test type: Loopback

Packet payload: Configure payload...

Rx timeout (ms): 1000

Tx interval (ms): 1000

● Number of packets: 100

○ Loop infinitely

Time window: Show all

Local: -76 dBm

Remote: -71 dBm

Packets sent 45

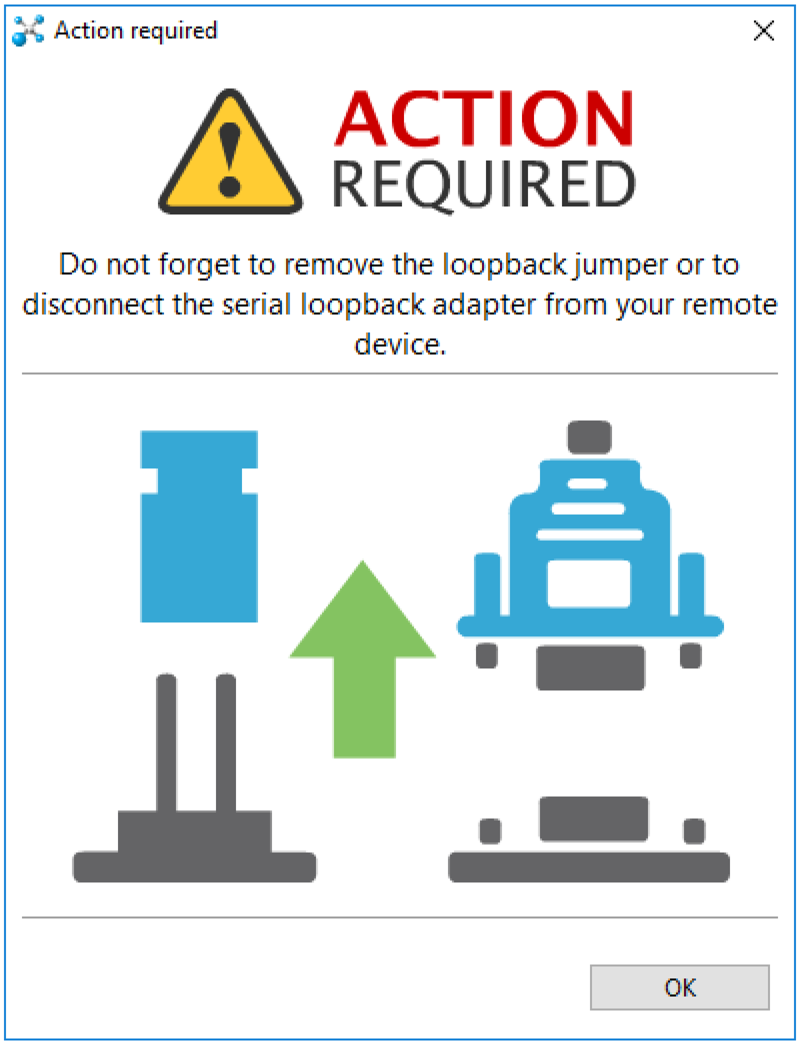
Tx errors: 13

Packets received 31

Packets lost: 1

68.89%

8. Move Radio 1 around to see the resulting signal strength at different distances. You can also test different data rates by reconfiguring the **BR** (data rate) parameter on both radios. When the test is complete, click **Stop Range Test**. XCTU displays another loopback jumper warning screen reminding you to put the loopback jumper back in its original position.


Action required

ACTION REQUIRED

Do not forget to remove the loopback jumper or to disconnect the serial loopback adapter from your remote device.

Configure the XBee3 802.15.4 RF Module

Software libraries

One way to communicate with the XBee3 802.15.4 RF Module is by using a software library. The libraries available for use with the XBee3 802.15.4 RF Module include:

- XBee Java library
- XBee Python library

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

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

Over-the-air (OTA) firmware update

The XBee3 802.15.4 RF Module supports OTA firmware updates using XCTU version 6.3.0 or higher. For instructions on performing an OTA firmware update with XCTU, see How to update the firmware of your modules in the XCTU User Guide.

OTA capability is only available when **MM** (Mac Mode) = **0** or **3**.

Custom defaults

Custom defaults allow you to preserve a subset of the device configuration parameters even after returning to default settings using RE command. This can be useful for settings that identify the device—such as NI command—or settings that could make remotely recovering the device difficult if they were reset—such as ID command.

Note You must send these commands as local AT commands, they cannot be set using Remote AT Command Request frame - 0x17.

Set custom defaults

Use %F (Set Custom Default) to set custom defaults. When the XBee3 802.15.4 RF Module receives **%F** it takes the next command it receives and applies it to both the current configuration and the custom defaults.

To set custom defaults for multiple commands, send a **%F** before each command.

Restore factory defaults

!C (Clear Custom Defaults) clears all custom defaults, so that RE command will restore the device to factory defaults. Alternatively, R1 (Restore Factory Defaults) restores all parameters to factory defaults without erasing their custom default values.

Limitations

There is a limitation on the number of custom defaults that can be set on a device. The number of defaults that can be set depends on the size of the saved parameters and the devices' firmware version. When there is no more room for custom defaults to be saved, any command sent immediately after a **%F** returns an error.

Setting a custom default that has already been set or setting a custom default to the factory default value will not reclaim the space used by the previous value. The new value takes effect but the old value still occupies space in memory, reducing the number of custom defaults that can be set. This can be remedied by using !C (Clear Custom Defaults) to clear all custom defaults when changing custom default values, and by only setting custom defaults that differ from the factory defaults.

XBee Network Assistant

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

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

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

To install the XBee Network Assistant:

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

Modes

Transparent operating mode

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

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

Serial-to-RF packetization

Data is buffered in the incoming serial buffer until one of the following causes the data to be packetized and transmitted:

1. No serial characters are received for the amount of time determined by the **RO** (Packetization Timeout) parameter. If **RO** = 0, packetization begins when a character is received.
2. The maximum number of characters that will fit in an RF packet is received. There are a number of factors that determine payload size. You can query the NP command to determine the maximum payload size based on current configuration. For more information, see Maximum payload.
3. The Command mode Sequence, **GT** + **CC** + **GT**, (including spaces) is received; this is any data in the serial receive buffer received before the sequence is transmitted. For more information, see Enter Command mode.

If the device cannot immediately transmit (for instance, if it is already receiving RF data), the serial data is stored in the serial receive buffer. The data is packetized and sent at any **RO** timeout or when **NP** bytes are received.

If the serial receive buffer becomes full, hardware flow control must be implemented in order to prevent overflow (loss of data between the host and device).

API operating mode

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

For more information, see API mode overview.

Command mode

Command mode is a state in which the firmware interprets incoming characters as commands. It allows you to modify the device's configuration using parameters you can set using AT commands. When you want to read or set any parameter of the XBee3 802.15.4 RF Module using this mode, you have to send an AT command. Every AT command starts with the letters **AT** followed by the two characters that identify the command and then by some optional configuration values.

The operating modes of the XBee3 802.15.4 RF Module are controlled by the AP command setting, but Command mode is always available as a mode the device can enter while configured for any of the operating modes.

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

Enter Command mode

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

If configured to operate in Transparent operating mode, when entering Command mode the XBee3 802.15.4 RF Module knows to stop sending data and start accepting commands locally.

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

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

You can customize the command character, the guard times and the timeout in the device's configuration settings. For more information, see CC command, CT command and GT command.

Troubleshooting

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

There are two alternative ways to enter Command mode:

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

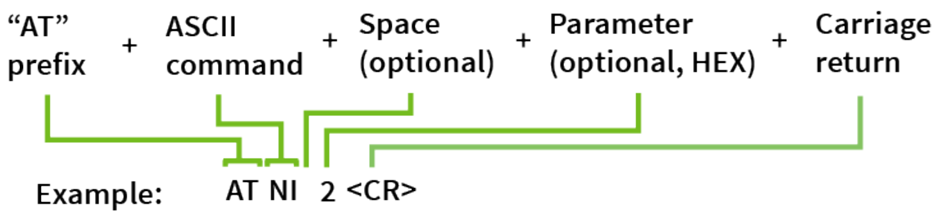
Note You must assert $\overline{\text{RTS}}$ for both of these methods, otherwise the device enters the bootloader.

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

Send AT commands

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

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



The preceding example changes NI command to **My XBee**.

Multiple AT commands

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

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

Parameter format

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

Response to AT commands

When using AT commands to set parameters the XBee3 802.15.4 RF Module responds with **OK<cr>** if successful and **ERROR<cr>** if not.

Apply command changes

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

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

Make command changes permanent

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

Send as RE command to wipe settings saved using **WR** back to their factory defaults, or custom defaults if you have set any.

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

Exit Command mode

1. Send CN 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, 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.

Idle mode

When not receiving or transmitting data, the XBee3 802.15.4 RF Module is in Idle mode. During Idle mode, the device listens for valid data on both the RF and serial ports.

If configured for Sleep support, the XBee3 802.15.4 RF Module only transitions to a low power state when in Idle mode.

Transmit mode

Transmit mode is the mode in which the device is transmitting data. This typically happens after data is received from the serial port.

Receive mode

This is the default mode for the XBee3 802.15.4 RF Module. The device is in Receive mode when it is not transmitting data. If a destination node receives a valid RF packet, the destination node transfers the data to its serial transmit buffer.

Serial communication

Serial interface

The XBee3 802.15.4 RF Module interfaces to a host device through a serial port. The device can communicate through its serial port:

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

Serial receive buffer

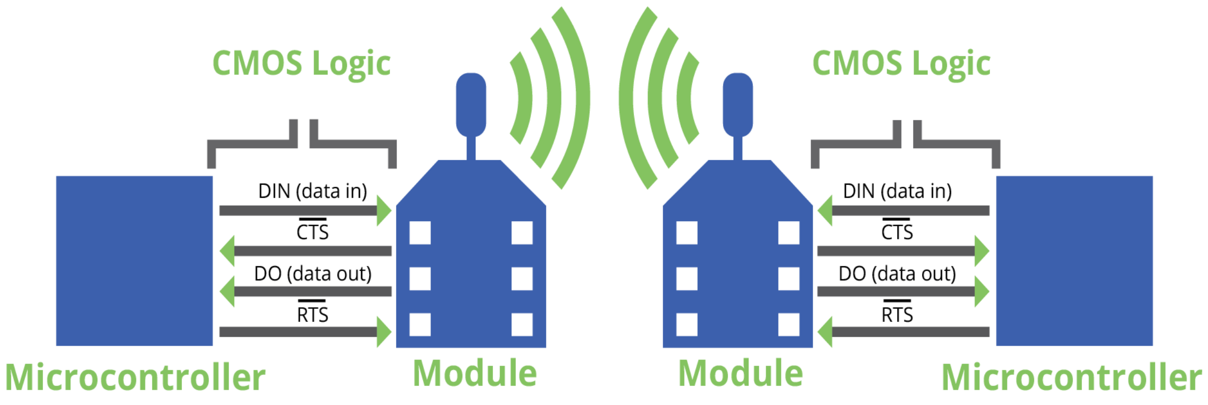
When serial data enters the device through the DIN pin or the SPI_MOSI pin, it stores the data in the serial receive buffer until the device can process it. Under certain conditions, the device may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the device such that the serial receive buffer overflows, then the device discards all incoming data until it is able to process the data in the buffer. If the UART is in use, you can avoid this by the host side by honoring clear-to-send ($\overline{CTS}$) flow control.

Serial transmit buffer

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

UART data flow

Devices that have a UART interface connect directly to the pins of the XBee3 802.15.4 RF Module as shown in the following figure. The figure shows system data flow in a UART-interfaced environment. Low-asserted signals have a horizontal line over the signal name.



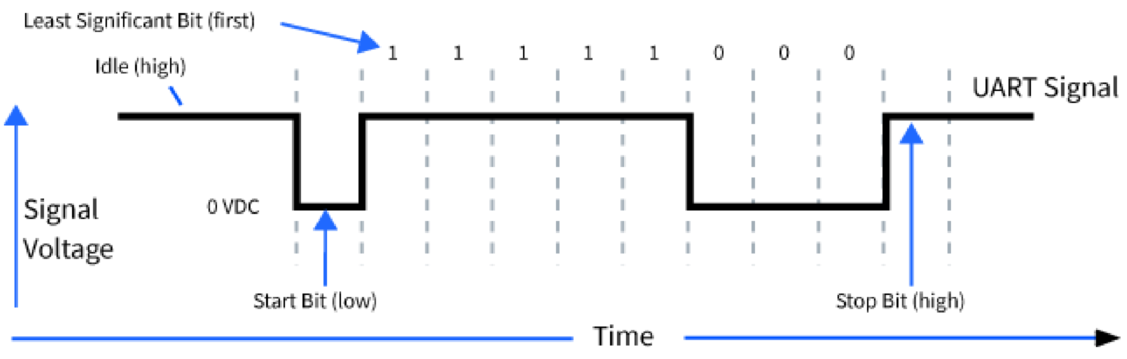
For more information about hardware specifications for the UART, see the *XBee3 Hardware Reference Manual*.

Serial data

A device sends data to the XBee3 802.15.4 RF Module's UART as an asynchronous serial signal. When the device is not transmitting data, the signals should idle high.

For serial communication to occur, you must configure the UART of both devices (the microcontroller and the XBee3 802.15.4 RF Module) with compatible settings for the baud rate, parity, start bits, stop bits, and data bits.

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

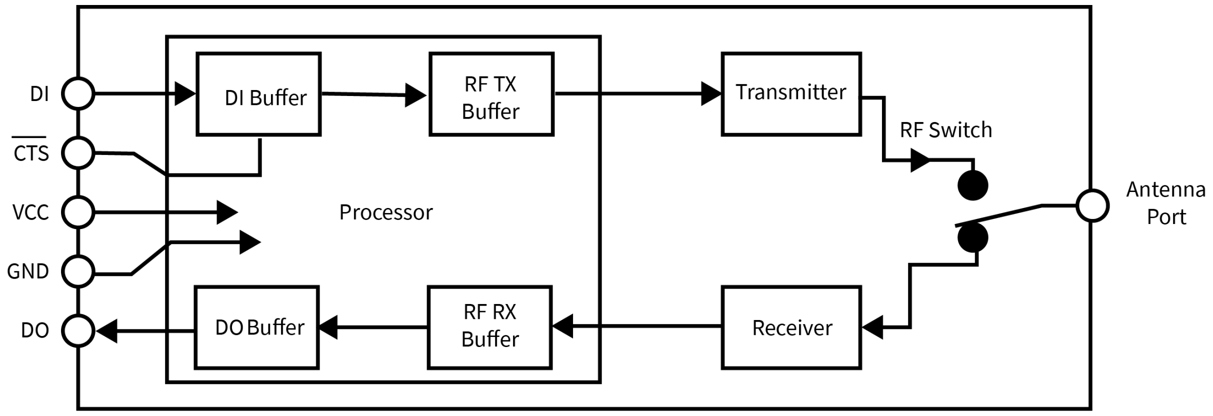


You can configure the UART baud rate, parity, and stop bits settings on the device with the **BD**, **NB**, and **SB** commands respectively. For more information, see UART interface commands.

Flow control

The XBee3 802.15.4 RF Module maintains 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.

Use D6 command and D7 command to set flow control.



Clear-to-send (CTS) flow control

If you enable CTS flow control (D7 command), when the serial receive buffer is more than **FT** bytes full, the device de-asserts CTS (sets it high) to signal to the host device to stop sending serial data. The device reasserts CTS after the serial receive buffer has less than **FT** bytes in it. See FT command to configure and read this threshold.

$\overline{\text{RTS}}$ flow control

If you send D6 command to enable $\overline{\text{RTS}}$ 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.

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

Cases in which the DO 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.

SPI operation

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

SPI communications

The XBee3 802.15.4 RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.

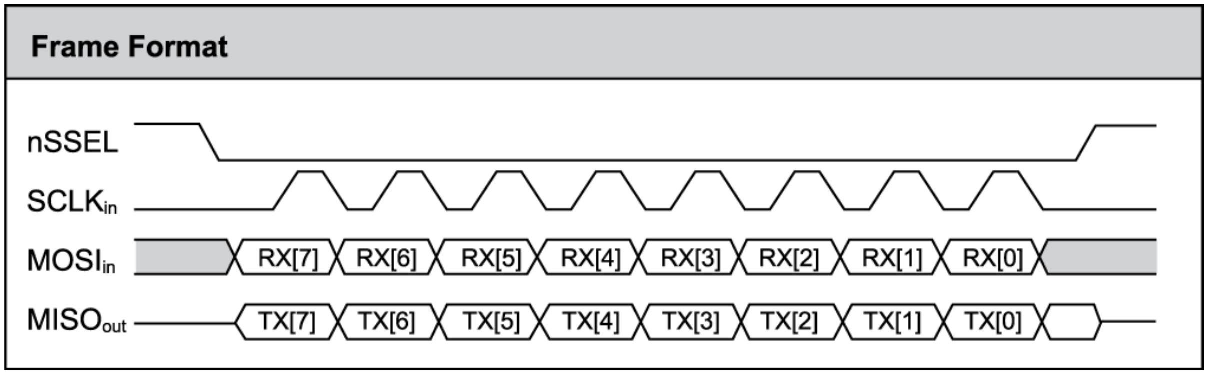
Refer to the XBee3 Hardware Reference Guide for the pinout of your device.

Signal	Direction	Function
SPI_MOSI (Master Out, Slave In)	Input	Inputs serial data from the master
SPI_MISO (Master In, Slave Out)	Output	Outputs serial data to the master
SPI_SCLK (Serial Clock)	Input	Clocks data transfers on MOSI and MISO
SPI_$\overline{\text{SSEL}}$ (Slave Select)	Input	Enables serial communication with the slave
SPI_$\overline{\text{ATTN}}$ (Attention)	Output	Alerts the master that slave has data queued to send. The XBee3 802.15.4 RF Module asserts this pin as soon as data is available to send to the SPI master and it remains asserted until the SPI master has clocked out all available data.

In this mode:

- SPI clock rates up to 5 MHz (burst) are possible.
- Data is most significant bit (MSB) first; bit 7 is the first bit of a byte sent over the interface.
- Frame Format mode 0 is used. This means CPOL= 0 (idle clock is low) and CPHA = 0 (data is sampled on the clock's leading edge).
- The SPI port only supports API Mode (**AP** = **1**).

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


Frame Format
nSSEL
SCLKin
MOSIin
MISOout
RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0]
TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]

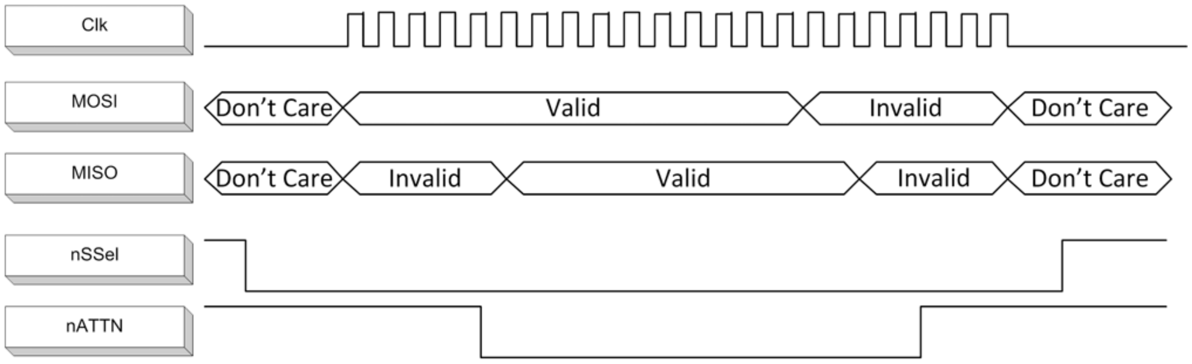
SPI mode is chip to chip communication. We do not supply a SPI communication interface on the XBee development evaluation boards included in the development kit.

Full duplex operation

When using SPI on the XBee3 802.15.4 RF Module the device uses API operation without escaped characters to packetize data. The device ignores the configuration of **AP** because SPI does not operate in any other mode. SPI is a full duplex protocol, even when data is only available in one direction. This means that whenever a device receives data, it also transmits, and that data is normally invalid. Likewise, whenever a device transmits data, invalid data is probably received. To determine whether or not received data is invalid, the firmware places the data in API packets.

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

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



Low power operation

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

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

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

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

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

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

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

Select the SPI port

To force SPI mode on through-hole devices, hold DOUT/DIO13 low while resetting the device until SPI_ATTN asserts. This causes the device to disable the UART and go straight into SPI communication mode. Once configuration is complete, the device queues a modem status frame to the SPI port, which causes the SPI_ATTN line to assert. The host can use this to determine that the SPI port is configured properly.

On surface-mount devices, forcing DOUT low at the time of reset has no effect. To use SPI mode on the SMT modules, assert the SPI_SSEL low after reset and before any UART data is input.

Forcing DOUT low on TH devices forces the device to enable SPI support by setting the following configuration values:

Through-hole	Micro and Surface-mount	SPI signal
D1 command	P9 command	$\overline{\text{ATTN}}$
D2 command	P8 command	SCLK
D3 command	P7 command	$\overline{\text{SSEL}}$
D4 command	P6 command	MOSI
P2 command	P5 command	MISO

Note The $\overline{\text{ATTN}}$ signal is optional—you can still use SPI mode if you disable the SPI_$\overline{\text{ATTN}}$ pin (**D1** on through-hole or **P9** on surface-mount devices).

As long as the host does not issue a **WR** command, these configuration values revert to previous values after a power-on reset. If the host issues a **WR** command while in SPI mode, these same parameters are written to flash, and after a reset the device continues to operate in SPI mode.

If the UART is disabled and the SPI is enabled in the written configuration, then the device comes up in SPI mode without forcing it by holding DOUT low. If both the UART and the SPI are configured (P3 command through P9 command are set to **1**) at the time of reset, then output goes to the UART until the host sends the first input to the SPI interface. As soon as the first input comes on the SPI port, then all subsequent output goes to the SPI port and the UART is disabled.

Once you select a serial port (UART or SPI), all subsequent output goes to that port, even if you apply a new configuration. Once the SPI interface is made active, the only way to switch the selected serial port back to UART is to reset the device.

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

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

Force UART operation

If you configure a device with only the SPI enabled and no SPI master is available to access the SPI slave port, you can recover the device to UART operation by holding DIN / CONFIG low at reset time.

DIN/$\overline{\text{CONFIG}}$ forces a default configuration on the UART at 9600 baud and brings up the device in Command mode on the UART port. You can then send the appropriate commands to the device to configure it for UART operation. If you write those parameters, the device comes up with the UART enabled on the next reset.

I/O support

The following topics describe analog and digital I/O line support, line passing and output control.

Legacy support

By default, the XBee3 802.15.4 RF Module is configured to operate in a legacy configuration. This provides network and application compatibility with XBee S1 802.15.4 and XBee S2C 802.15.4 devices. Use AO command to determine which outgoing API frames are emitted and what I/O lines are used for sampling.

On the source node, **AO** affects:

- Which Digital I/O lines are sampled
- What sample frame type is used for outgoing transmissions

On the destination node, **AO** affects:

- How incoming XBee3 sample frames are interpreted and what API frames are emitted

Previous 802.15.4 firmwares on the XBee S1 and XBee S2C hardware had a limited set of I/O lines available. Valid DIO lines on these devices are from D0 through D8; I/O samples are transmitted over the air using a standard I/O sample packet Legacy data format. These platforms do not have an **AO** command and always output sample data in a legacy format if possible.

For the XBee3 platform, digital I/O has been enhanced to be in parity with DigiMesh and Zigbee. You can now enable up to fourteen digital inputs for sampling: D0 through P4 as long as **AO** is not set to **2**. In order to support these additional I/O lines, an enhanced I/O sample packet is sent over the air, which is not compatible with the S1 or S2C.

By default, the XBee3 802.15.4 RF Module is configured to operate in a legacy configuration with **AO** set to **2**. This allows you to sample D0 through D8. If you configure D9 through P4 as digital I/O, they are not sampled unless you set **AO** to **0** or **1**.

For new designs, we recommend setting **AO** to **0** or **1** (Operate in API mode), which allows you to use additional I/O lines for sampling and easily allows you to switch to Zigbee or DigiMesh, as the API and I/O functionality are identical.

This table illustrates the various configuration combinations that are possible and the expected output:

Source	Source AO value	Destination	Destination AO value	Data format	API frame on receiver
XBee3	0 or 1	XBee3	0 or 1	Enhanced	I/O Data Sample Rx Indicator frame - 0x92
XBee3	0 or 1	XBee3	2	Legacy	RX (Receive) Packet: 64-bit address IO frame - 0x82 / RX Packet: 16-bit address I/O frame - 0x83
XBee3	0 or 1	S1 or S2C	N/A	N/A	N/A
XBee3	2	XBee3	0 or 1	Legacy	RX (Receive) Packet: 64-bit address IO frame - 0x82 / RX Packet: 16-bit address I/O frame - 0x83
XBee3	2	S1 or S2C	N/A	Legacy	RX (Receive) Packet: 64-bit address IO frame - 0x82 / RX Packet: 16-bit address I/O frame - 0x83

Source	Source AO value	Destination	Destination AO value	Data format	API frame on receiver
S1 or S2C	N/A	XBee3	0 or 1	Legacy	RX (Receive) Packet: 64-bit address IO frame - 0x82 / RX Packet: 16-bit address I/O frame - 0x83
S1 or S2C	N/A	XBee3	2	Legacy	RX (Receive) Packet: 64-bit address IO frame - 0x82 / RX Packet: 16-bit address I/O frame - 0x83

Refer to I/O sample data format for more information on the format of the incoming I/O sample data.

Mixed network considerations

If you use a mixed network of XBee3 and legacy S1 or S2C devices, you must set **AO** to **2** in order to transmit sample data that is compatible with these devices.

Regardless of the **AO** setting, if an XBee3 802.15.4 RF Module receives an I/O sample packet from an S1 or S2C device, it always outputs the legacy data format.

Digital I/O support

AO command determines the I/O lines available for sampling. By default, **AO** is configured to be compatible with legacy devices.

- Configure **AO** to **0** or **1** to make digital I/O available on lines DIO0 through DIO14 (D0 command - D9 command and P0 command - P4 command).
- Configure **AO** to **2** to make digital I/O available on lines DIO0 through DIO8 (**D0** - D8 command). This provides compatibility with S1 and S2C devices and is the default configuration.

See Legacy support for more information.

Digital sampling is enabled on these pins if configured as **3**, **4**, or **5** with the following meanings:

- 3 is digital input.
  - Use PR command to enable internal pull up/down resistors for each digital input. Use PD command to determine the direction of the internal pull up/down resistor. All disabled and digital input pins are pulled up by default.
- 4 is digital output low.
- 5 is digital output high.

Function when AO = 0 or 1	Legacy Function when AO = 2	Micro Pin	SMT Pin	TH Pin	AT Command
DIO0	DIO0	31	33	20	D0 command
DIO1	DIO1	30	32	19	D1 command
DIO2	DIO2	29	31	18	D2 command

Function when AO = 0 or 1	Legacy Function when AO = 2	Micro Pin	SMT Pin	TH Pin	AT Command
DIO3	DIO3	28	30	17	D3 command
DIO4	DIO4	23	24	11	D4 command
DIO5	DIO5	26	28	15	D5 command
DIO6	DIO6	27	29	16	D6 command
DIO7	DIO7	24	25	12	D7 command
DIO8	DIO8	9	10	9	D8 command
DIO9	N/A	25	26	13	D9 command
DIO10	N/A	7	7	6	P0 command
DIO11	N/A	8	8	7	P1 command
DIO12	N/A	5	5	4	P2 command
DIO13	N/A	3	3	2	P3 command
DIO14	N/A	4	4	3	P4 command

I\O sampling is not available for pins P5 through P9. See the *XBee3 Hardware Reference Manual* for full pinouts and functionality.

Analog I/O support

Analog input is available on D0 through D3. Configure these pins to **2** (ADC) to enable analog sampling.

PWM output is available on P0 and P1, which can be used for Analog line passing. Use M0 command and M1 command to set a fixed PWM level.

Function	Micro Pin	SMT Pin	TH Pin	AT Command
ADC0	31	33	20	D0 command
ADC1	30	32	19	D1 command
ADC2	29	31	18	D2 command
ADC3	28	30	17	D3 command
PWM0	7	7	6	P0 command
PWM1	8	8	7	P1 command

AV (Analog Voltage Reference) specifies the analog reference voltage used for the 10-bit ADCs. Analog sample data is represented as a 2-byte value. For a 10-bit ADC, the acceptable range is from **0x0000** to **0x03FF**. To convert this value to a useful voltage level, apply the following formula:

ADC / 1023 (vREF) = Voltage

Example

An ADC value received is 0x01AE; to convert this into a voltage the hexadecimal value is first converted to decimal (0x01AE = 430). Using the default **AV** reference of 1.25 V, apply the formula as follows:

430 / 1023 (1.25 V) = 525 mV

Monitor I/O lines

You can monitor pins you configure as digital input, digital output, or analog input and generate I/O sample data. If you do not define inputs or outputs, no sample data is generated.

Typically, I/O samples are generated by configuring the device to sample I/O pins periodically (based on a timer) or when a change is detected on one or more digital pins. These samples are always sent over the air to the destination address specified with DH command and DL command.

You can also gather sample data using on-demand sampling, which allows you to interrogate the state of the device's I/O pins by issuing an AT command. You can do this on either a local or remote device via an AT command request.

The three methods to generate sample data are:

- Periodic sample (IR command)
  - Periodic sampling based on a timer
  - Samples are taken immediately upon wake (excluding pin sleep)
  - Sample data is sent to **DH**+**DL** destination address
  - Can be used with line passing
  - Requires API mode on receiver
- Change detect (IC command)
  - Samples are generated when the state of specified digital input pin(s) change
  - Sample data is sent to **DH**+**DL** destination address
  - Can be used with line passing
  - Requires API mode on receiver
- On-demand sample (IS command)
  - Immediately query the device’s I/O lines
  - Can be issued locally in Command Mode
  - Can be issued locally or remotely in API mode

These methods are not mutually exclusive and you can use them in combination with each other.

I/O sample data format

AO determines the format of the incoming and outgoing sample data.

By default, **AO** is configured to be compatible with legacy devices and outputs a legacy data format regardless of where the sample packet came from.

Legacy data format

Regardless of how I/O data is generated, the format of the sample data is always represented as a series of bytes in the following format which is compatible with the S1 802.15.4 and S2C 802.15.4 devices:

Bytes	Name	Description
1	Sample sets	Number of sample sets. This is determined by IT (Samples before TX) on the source node.
2	Digital and analog channel mask	Indicates which digital I/O and ADC lines have sampling enabled. Each bit corresponds to one digital I/O or ADC line on the device. bit 0 = DIO0 bit 1 = DIO1 bit 2 = DIO2 bit 3 = DIO3 bit 4 = DIO4 bit 5 = DIO5 bit 6 = DIO6 bit 7 = DIO7 bit 8 = DIO8 bit 9 = ADC0 bit 10 = ADC1 bit 11 = ADC2 bit 12 = ADC3 bit 13 = Reserved bit 14 = Reserved bit 15 = Reserved Example: a channel mask of 0x063C means ADC0, ADC1, DIO2, DIO3, and DIO5 are configured as digital inputs or outputs.
2	Digital data set	Each bit in the digital data set corresponds to a digital bit in the channel mask and indicates the state of the digital pin, whether high (1) or low (0). If the digital portion of the channel mask is **0**, then these two bytes are omitted as no digital I/O lines are enabled. bit 0 = DIO0 bit 1 = DIO1 bit 2 = DIO2 bit 3 = DIO3 bit 4 = DIO4 bit 5 = DIO5 bit 6 = DIO6 bit 7 = DIO7 bit 8 = DIO8 bit 9 = N/A bit 10 = N/A bit 11 = N/A bit 12 = N/A bit 13 = N/A bit 14 = N/A bit 15 = N/A
2	Analog data set (multiple)	Each enabled ADC line in the analog portion of the channel mask has a separate 2-byte value based on the number of ADC inputs on the originating device. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3. If the analog portion of the channel mask is **0**, then no analog sample bytes are included.

Enhanced data format

If you set **AO** to **0** or **1** on both the source and destination node, then the data format is represented as a series of bytes in the following format which matches the DigiMesh and Zigbee firmwares:

Bytes	Name	Description
1	Sample sets	Number of sample sets. There is always one sample set per frame.
2	Digital channel mask	Indicates which digital I/O lines have sampling enabled. Each bit corresponds to one digital I/O line on the device. bit 0 = DIO0 bit 1 = DIO1 bit 2 = DIO2 bit 3 = DIO3 bit 4 = DIO4 bit 5 = DIO5 bit 6 = DIO6 bit 7 = DIO7 bit 8 = DIO8 bit 9 = DIO9 bit 10 = DIO10 bit 11 = DIO11 bit 12 = DIO12 bit 13 = DIO13 bit 14 = DIO14 bit 15 = N/A Example: a digital channel mask of 0x002F means DIO0, 1, 2, 3 and 5 are configured as digital inputs or outputs.
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. If a bit is set, then a corresponding 2-byte analog data set is included. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3
2	Digital data set	Each bit in the digital data set corresponds to a bit in the digital channel mask and indicates the digital state of the pin, whether high (1) or low (0). If the digital channel mask is 0x0000, then these two bytes are omitted as no digital I/O lines are enabled.
2	Analog data set (multiple)	Each enabled ADC line in the analog channel mask will have a separate 2-byte value based on the number of ADC inputs on the originating device. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3. If the analog channel mask is 0x00, then no analog sample bytes is included.

API frame support

I/O samples generated using Periodic I/O sampling (**IR**) and Digital I/O change detection (**IC**) are transmitted to the destination address specified by **DH** and **DL**. In order to display the sample data, the receiver must be operating in API mode (**AP** = **1** or **2**). The sample data is represented as an I/O sample API frame.

There are three types of I/O sample frames that are supported by the XBee3 802.15.4 RF Module:

- 0x92 - Enhanced I/O sample frame
- 0x82 - Legacy 64-bit I/O sample frame
- 0x83 - Legacy 16-bit I/O sample frame

If **AO** = **0** or **1** on both the source and destination node, then a 0x92 frame is generated on the destination.

See I/O Data Sample Rx Indicator frame - 0x92 for more information on the frame's format and an example.

For all other cases, the destination node generates either a 0x82 or 0x83 frame depending on whether the source node is operating in a 16-bit or 64-bit configuration. See Addressing modes for more information.

See Legacy support for more information on what configuration options generate the various I/O frames.

On-demand sampling

You can use IS command to query the current state of all digital I/O and ADC lines on the device and return the sample data as an AT command response. If no inputs or outputs are defined, the command returns an ERROR.

On-demand sampling can be useful when performing initial deployment, as you can send **IS** locally to verify that the device and connected sensors are correctly configured. The format of the sample data matches what is periodically sent using other sampling methods. You can also send **IS** remotely using a remote AT command. When sent remotely from a gateway or server to each sensor node on the network, on-demand sampling can improve battery life and network performance as the remote node transmits sample data only when requested instead of continuously.

If you send **IS** using Command mode, then the device returns a carriage return delimited list containing the I/O sample data. If **IS** is sent either locally or remotely via an API frame, the I/O sample data is presented as the parameter value in the AT command response frame (AT Command Response frame - 0x88 or Remote Command Response frame - 0x97).

Example: Command mode

An **IS** command sent in Command mode returns the following sample data:

This example uses the enhanced I/O data format, if you use the legacy format (**AO** = **2** or data is received from an S1 or S2C device) then refer to the Legacy data format for information on how this data is structured.

Output	Description
01	One sample set
0C0C	Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 **11**00 0000 **11**00b = DIO2, 3, 10, 11)

Output	Description
03	Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 00**11**b = AD0, 1)
0408	Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 **01**00 0000 **1**000b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03D0	Analog sample data for AD0
0124	Analog sample data for AD1

Example: Local AT command in API mode

The **IS** command sent to a local device in API mode would use a AT Command Frame - 0x08 or AT Command - Queue Parameter Value frame - 0x09 frame:

```
7E 00 04 08 53 49 53 08
```

The device responds with a AT Command Response frame - 0x88 that contains the sample data:

```
7E 00 0F 88 53 49 53 00 01 0C 0C 03 04 08 03 D0 01 24 68
```

This example uses the enhanced I/O data format, if you use the legacy format (**AO** = **2** or data is received from an S1 or S2C device) then see the Legacy data format for information on how this data is structured.

Output	Field	Description
7E	Start Delimiter	Indicates the beginning of an API frame
00 0F	Length	Length of the packet
88	Frame type	AT Command response frame
53	Frame ID	This ID corresponds to the Frame ID of the 0x08 request
49 53	AT Command	Indicates the AT command that this response corresponds to 0x49 0x53 = **IS**
00	Status	Indicates success or failure of the AT command **00** = OK if no I/O lines are enabled, this will return 01 (ERROR)

Output	Field	Description
01	I/O sample data	One sample set
0C 0C		Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 **11**00 0000 **11**00b = DIO2, 3, 10, 11)
03		Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 00**11**b = AD0, 1)
04 08		Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 **01**00 0000 **10**00b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03 D0		Analog sample data for AD0
01 24		Analog sample data for AD1
68	Checksum	Can safely be discarded on received frames

Example: Remote AT command in API mode

The **IS** command sent to a remote device with an address of 0013A200 12345678 uses a Remote AT Command Request frame - 0x17:

7E 00 0F 17 87 00 13 A2 00 12 34 56 78 FF FE 00 49 53 FF

The sample data from the device is returned in a Remote Command Response frame - 0x97 frame with the sample data as the parameter value:

7E 00 19 97 87 00 13 A2 00 12 34 56 78 00 00 49 53 00 01 0C 0C 03 04 08 03 FF 03 FF 50

This example uses the enhanced I/O data format, if you use the legacy format (**AO** = **2** or data is received from an S1 or S2C device) then see Legacy data format for information on how this data is structured.

Output	Field	Description
7E	Start Delimiter	Indicates the beginning of an API frame
00 19	Length	Length of the packet
97	Frame type	Remote AT Command response frame
87	Frame ID	This ID corresponds to the Frame ID of the 0x17 request
0013A200 12345678	64-bit source	The 64-bit address of the node that responded to the request
0000	16-bit source	The 16-bit address of the node that responded to the request
49 53	AT Command	Indicates the AT command that this response corresponds to 0x49 0x53 = **IS**
00	Status	Indicates success or failure of the AT command **00** = **OK** if no I/O lines are enabled, this will return **01** (ERROR)

Output	Field	Description
01	I/O sample data	One sample set
0C 0C		Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 1100 0000 1100b = DIO2, 3, 10, 11)
03		Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 0011b = AD0, 1)
04 08		Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 0100 0000 1000b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03 D0		Analog sample data for AD0
01 24		Analog sample data for AD1
50	Checksum	Can safely be discarded on received frames

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.

Source

Use IR command to set the periodic sample rate for enabled I/O lines.

- To disable periodic sampling, set **IR** to **0**.
- For all other **IR** values, the device samples data when **IR** milliseconds elapse and transmits the sampled data to the destination address.

The DH command and DL command commands determine the destination address of the I/O samples. You must configure at least one pin as a digital I/O or ADC input on the sending node to generate sample data.

Destination

If the receiving device is operating in API operating mode the I/O data sample is emitted out of the serial port. Devices that are in Transparent operating mode discard the I/O data samples they receive unless you enable line passing.

I/O sampling upon wake

By default, a device that is configured for sleep (**SM** > **0**) that has at least one digital I/O or ADC enabled transmits an I/O sample upon wake regardless of how **IR** is configured. Sampling upon wake can be disabled by clearing bit 1 of the **SO**. For more information about setting sleep modes, see Sleep modes and SO command.

Multiple samples per packet

IT (Samples before TX) specifies how many I/O samples can be transmitted in a single OTA packet. Any single-byte value (0 - 0xFF) is accepted for input. However, the value is adjusted downward based on how many I/O samples can fit into a maximum size packet; see Maximum payload. A query of **IT** after changes are applied tells how many I/O samples will actually be gathered.

Since MM (MAC Mode) must be **0** or **3** to send I/O samples, the maximum payload in the best of conditions (short source address, short destination address, and no encryption) is 114 bytes. Seven of those bytes are used by the command header and the I/O header, leaving 107 bytes for I/O samples. The minimum I/O sample is 2 bytes. Therefore the maximum possible usable value for **IT** is 53 (or **0x35**).

Only legacy I/O frames allow for gathering multiple samples. If you set **AO** to **0** or **1**, then **IT** is not applicable and only one sample can be gathered per frame.

Example: Remote AT command in API mode

A device is configured with the following settings:

- **D0** and **D1** are set to ADC (**2**)
- **D3** is configured as a digital input (**3**)
- **AO** is set to **2**, so legacy frames are generated
- **IT** is configured to **3**, so that three samples are gathered per transmission

On the destination node, the following frame is emitted:

7E 00 1A 83 12 34 26 02 03 06 04 00 04 01 28 03 12 00 00 01 58 02 FE 00 04 01 2A 03 A0 94

Output	Field	Description
7E	Start Delimiter	Indicates the beginning of an API frame
00 1A	Length	Length of the packet
83	Frame type	Legacy 16-bit I/O Sample
12 34	16-bit Source Address	The source address of the device that sent the I/O sample
26	RSSI	The 64-bit address of the node that responded to the request
02		
03	Sample sets	The number of samples that are included in this frame
06 04	Channel mask	Mask which indicates which digital and analog lines are enabled. Even though multiple samples are being gathered, there will only ever be one channel mask per frame. (0x0604 = 0000 0110 0000 0100b = ADC0, ADC1, DIO3)

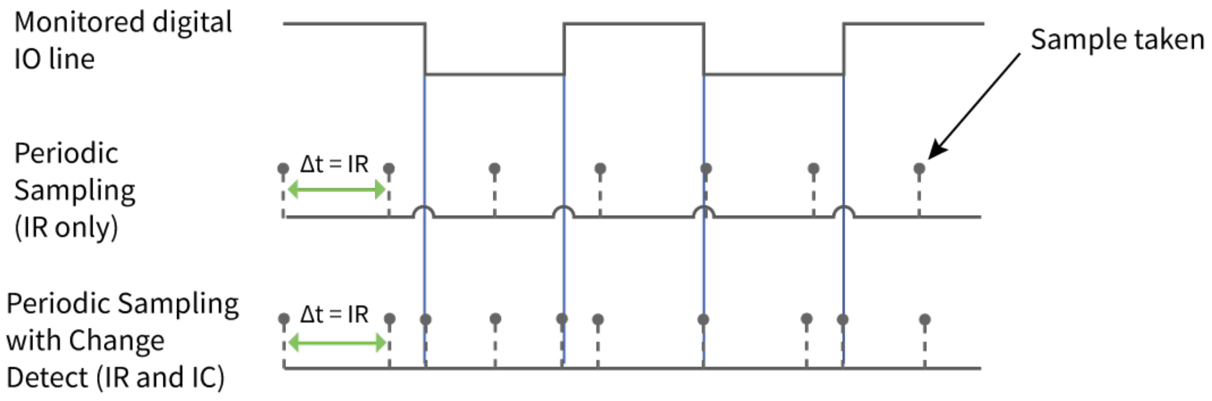
Output	Field	Description
00 04	Sample set 1	The first set of digital sample data that corresponds with the digital portion of the channel mask 0x0004 = 0000 0000 0000 0100b = DIO3 is high
01 28		Analog sample data for AD0
03 12		Analog sample data for AD1
00 00	Sample set 2	The second set of digital sample data 0x0004 = 0000 0000 0000 0000b = DIO3 is low
01 58		Second set of analog sample data for AD0
02 FE		Second set of analog sample data for AD1
00 04	Sample set 1	The third set of digital sample data 0x0004 = 0000 0000 0000 0100b = DIO3 is high
01 2A		Third set of analog sample data for AD0
03 A0		Third set of analog sample data for AD1
94	Checksum	Can safely be discarded on received frames

Digital I/O change detection

You can configure devices to transmit a data sample immediately whenever a monitored digital I/O pin changes state. IC command is a bitmask that determines 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 on the monitored digital I/O line(s) using edge detection.

Change detection is only applicable to digital I/O pins that are configured as digital input (**3**) or digital output (**4** or **5**).

The figure below shows how I/O change detection can work in combination with Periodic I/O sampling to improve sampling accuracy. 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 periodic IR samples, the bottom graph shows a combination of **IR** periodic samples and **IC** detected changes. 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.


Monitored digital IO line
Periodic Sampling (IR only)
Periodic Sampling with Change Detect (IR and IC)
Δt = IR
Δt = IR
Sample taken

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

If you enable multiple samples by setting **IT** > **1**, any change detect that occurs causes all collected periodic samples to be sent immediately, then a separate **IC** sample is sent.

I/O line passing

Line passing allows you to affect the output pins of one device by sampling the I/O pins of another. To support line passing, you must configure a device to generate I/O sample data using periodic sampling (IR command) and/or change detection (IC command).

On the device that receives I/O samples, enable line passing setting IA (I/O Input Address) with the address of the device that has the appropriate inputs enabled. This effectively binds the outputs to a particular device's input. This does not affect the ability of the device to receive I/O line data from other devices—only its ability to update enabled outputs. Set **IA** to **0xFFFF** (broadcast address) to affect the output using input data from any device on the network.

Digital line passing

Digital I/O lines are mapped in pairs; pins configured as digital input on the transmitting device affect the corresponding digital output pin on the receiving device. For example, a device that samples D5 as an input (3) only affects D5 on the receiver if D5 is configured as an output (4 or 5).

Each digital pin has an associated timeout value. When an I/O sample is received that affects a digital output pin, the pin returns to its configured state after the timeout period expires. For pins D0 through D9, the associated timeout commands are T0 command through T9 command. For pins P0 through P4, the associated timeout commands are Q0 command through **Q4**.

Digital line passing is only available on pins D0 through P3. You cannot use UART and SPI pins for line passing.

Example: Digital line passing

A sampling XBee3 802.15.4 RF Module is configured with the following settings:

AT command	Parameter value
D2 command	3 (digital input)
IR command	0x7D0 (2 seconds)
DH command	0013A200
DL command	12345678

Every two seconds, an I/O sample is generated and sent to the address specified by **DH** and **DL**. The receiver is configured with the following settings:

AT command	Parameter value
D2 command	5 (digital output low)
T2 command	0x64 (10 seconds)
IA (I/O Input Address)	0013A20087654321

When this device receives an incoming I/O sample, if the source address matches the one set by **IA**, the device sets the output of **D2** to match the input of **D2** of the receiver. This output level holds for ten seconds before the pin returns to a digital output low state.

Analog line passing

Similar to digital line passing, analog line passing pairs the Analog I/O support of one device to a PWM output of another. There are two PWM output pins that can simulate the voltage measured by the ADC inputs. Be aware that ADC inputs are on different pins than the corresponding PWM outputs: AD0 corresponds to PWM0, and AD1 corresponds to PWM1. See Analog I/O support for the pinouts.

You can set the analog line passing timeout value with PT (PWM Output Timeout), which affects both PWM output pins. You can explicitly set a PWM output level using the M0 command and M1 command commands, when an I/O sample is received that affects a PWM output pin, it returns to its configured state after the **PT** timeout period expires.

Example: Analog line passing

A sampling device is configured with the following settings:

AT command	Parameter value
D0 command	2 (ADC input)
IR command	0x7D0 (2 seconds)
DH command	0013A200
DL command	12345678

Every two seconds, an I/O sample frame is generated and sent to the address specified by **DH** and **DL**. The receiver is configured with the following settings:

AT command	Parameter value
P0	2 (PWM output)
M0	0
PT	0x12C (30 seconds)
IA	0013A20087654321

When this device receives an incoming I/O sample, if the source address matches the one set by **IA**, the device sets the PWM output of **P0** to match the ADC input of **D0** of the receiver. This output level holds for thirty seconds before the pin returns to a digital output low state.

Output sample data

If a device receives an I/O sample whose address matches that set by IA (I/O Input Address), it triggers line passing. Line passing operates whether the receiving device is operating in API or Transparent mode.

By default, if the receiver is configured for API mode, it outputs the I/O sample frame in addition to affecting output pins. You can suppress the I/O sample frame output by setting IU command to **0**. This only suppresses I/O samples that trigger line passing, a sample generated from a device whose address does not match the **IA** address is sent regardless of **IU**.

Output control

IO command controls the output levels of D0 command through D7 command that are configured as output pins (either **4** or **5**). These values override the configured output levels of the pins until they are changed again (the pins do not automatically revert to their configured values after a timeout.)

You can use **IO** to trigger a sample on change detect.

I/O behavior during sleep

When the device sleeps (**SM !** = **0**) the I/O lines are optimized for a minimal sleep current.

Digital I/O lines

Digital I/O lines set as digital output high or low maintain those values during sleep. Disabled or input pins continue to be controlled by the **PR**/**PD** settings. Peripheral pins (with the exception of CTS) are set low during sleep and SPI pins are set high. Peripheral and SPI pins resume normal operation upon wake.

Digital I/O lines that have been set using I/O line passing hold their values during sleep, however the digital timeout timer (**T0** through **T9**, and **Q0** through **Q2**) are suspended during sleep and resume upon wake.

Analog and PWM I/O Lines

Lines configured as analog inputs or PWM output are not affected during sleep. PWM lines are shut down (set low) during sleep and resume normal operation upon wake.

PWM output pins set by analog line passing are shutdown during sleep and revert to their preset values (**M0** and **M1**) on wake. This happens regardless of whether the timeout has expired or not.

Networking

Networking terms

The following table describes some common terms we use when discussing networks.

Term	Definition
Association	Establishing membership between end devices and a coordinator.
Coordinator	A full-function device (FFD) that allows end devices to associate to it and can queue and deliver indirect messages.
End device	When in the same network as a coordinator. Devices that rely on a coordinator for synchronization and can be put into states of sleep for low-power applications.
PAN	Personal Area Network. A data communication network that includes one or more end devices and optionally a coordinator.

MAC Mode configuration

Medium Access Control (MAC) Mode configures two functions:

1. Enables or disables the use of a Digi header in the 802.15.4 RF packet.
   When the Digi header is enabled (**MM** = 0 or 3), duplicate packet detection is enabled as well as certain AT commands.
   MAC Modes 1 and 2 do not include a Digi header, which disables many features of the device. All data is strictly pass-through. These modes are intended to provide some compatibility with third-party 802.15.4 devices.
2. Enables or disables MAC acknowledgment request for unicast packets.
   When MAC ACK is enabled (**MM** = 0 or 2), transmitting devices send packets with an ACK request so receiving devices send an ACK back (acknowledgment of RF packet reception) to the transmitter. If the transmitting device does not receive the ACK, it re-sends the packet up to three times or until the ACK is received.
   MAC Modes 1 and 3 disable MAC acknowledgment. Transmitting devices send packets without an ACK request so receiving devices do not send an ACK back to the transmitter.
   Broadcast messages are always sent with the MAC ACK request disabled.

The following table summarizes the functionality.

Mode	Digi header	MAC ACK
0 (default)	X	X
1		
2		X
3	X	

The default value for the **MM** configuration parameter is 0 which enables both the Digi header and MAC acknowledgment.

Clear Channel Assessment (CCA)

Prior to transmitting a packet, the device performs a CCA (Clear Channel Assessment) on the channel to determine if the channel is available for transmission. The detected energy on the channel is compared with the **CA** (Clear Channel Assessment) parameter value. If the detected energy exceeds the **CA** parameter value, the device does not transmit the packet.

Also, the device inserts a delay before a transmission takes place. You can set this delay using the **RN** (Backoff Exponent) parameter. If you set **RN** to 0, there is no delay before the first CCA is performed. The **RN** parameter value is the equivalent of the “minBE” parameter in the 802.15.4 specification. The transmit sequence follows the 802.15.4 specification.

On a CCA failure, the device attempts to re-send the packet up to three additional times, meaning a total of four attempts.

CCA operations

CCA is a method of collision avoidance that is implemented by detecting the energy level on the transmission channel before starting the transmission. The CCA threshold (defined by the **CA** parameter) defines the energy level that it takes to block a transmission attempt. For example, if CCA is set to the default value of 0x41 (which is interpreted as -65 dBm) then energy detected above the -65 dBm level (for example -60 dBm) temporarily blocks a transmission attempt. But if the energy level is less than that (for example -70 dBm), the transmission is not blocked. The intent of this feature is to prevent simultaneous transmissions on the same channel.

You can disable CCA by setting **CA** to 0. Disabling CCA can improve latency in noisy environments, but it can also interfere with other devices that are operating on the same channel.

In the event that the energy level exceeds the threshold, the transmission is blocked for a random number of backoff periods. The number of backoff periods is defined by the following formula: random (2^n - 1), where *n* is defined by the **RN** parameter and increments after each CCA failure. When **RN** is set to its default value of 0, then 2^n -1 is 0, preventing any delay before the first energy detection on a new frame. However, n increments after each CCA failure, giving a greater range for the number of backoff periods between each energy detection cycle.

In the event that five energy detection cycles occur and each one detects too much energy, the application tries again 1 to 48 ms later. After the application retries are exhausted, then the transmission fails with a CCA error.

Whenever the MAC code reports a CCA failure, meaning that it performed five energy detection cycles with exponential random back-offs, and each one failed, the **EC** parameter is incremented. The **EC** parameter can be read at any time to find out how noisy the operating channel is. It continues to increment until it reaches its maximum value of 0xFFFF. To get new statistics, you can set **EC** back to 0.

Retries configuration

If you are operating in a MAC Mode that enables MAC ACK (**MM**=0 or **MM**=2), each RF packet will be sent with up to three 802.15.4 MAC-Layer retries, meaning four transmission attempts are performed. This is enabled by default and provides a minimal amount of reliability to unicast transmissions.

If you are operating in a MAC Mode that enables the Digi header (**MM**=0 or **MM**=3), then you can optionally include Application-Layer retries using the RR command command. Each Application-Layer retry attempt to send the packet using three MAC-Layer retries. This can greatly increase the reliability of unicast transmissions with a risk of reduced throughput.

Transmit status based on MAC mode and XBee retries configurations

When working in API mode, a transmit request frame sent by the user is always answered with a transmit status frame sent by the device, if the frame ID is non-zero. A Frame ID of 0 specifies that the packet should be sent without an acknowledgment.

The following tables report the expected transmit status for unicast transmissions and the maximum number of MAC and application retries the device attempts.

The tables also report the transmit status reported when the device detects energy above the CCA threshold (when a CCA failure happens).

The following table applies in either of these cases:

- Digi header is disabled.
- Digi header is enabled and XBee Retries (**RR** parameter) is equal to 0 (default configuration).

Mac ACK Config	Destination reachable			Destination unreachable			CCA failure happened		
	TX status	Retries		TX status	Retries		TX status	Retries	
		MAC	App		MAC	App		MAC	App
Enabled	00 (Success)	up to 3	0	01 (No acknowledgment received)	3	0	02 (CCA failure)	3	0
Disabled	00 (Success)	0	0	00 (Success)	0	0	02 (CCA failure)	3	0

The following table applies when:

- Digi header is enabled and XBee Retries (**RR** parameter) > 0.

Mac ACK Config	Destination reachable			Destination unreachable			CCA failure happened		
	TX status	Retries		TX status	Retries		TX status	Retries	
		MAC	App		MAC	App		MAC	App
Enabled	00 (Success)	up to 3 per app retry	up to **RR** value	21 (Network ACK Failure)	3	**RR** value	02 (CCA failure)	3	**RR** value
Disabled	00 (Success)	0	up to **RR** value	21 (Network ACK Failure)	0	**RR** value	02 (CCA failure)	3	**RR** value

Addressing

Every RF data packet sent over-the-air contains a Source Address and Destination Address field in its header. The XBee3 802.15.4 RF Module conforms to the 802.15.4 specification and supports both short 16-bit addresses and long 64-bit addresses. A unique 64-bit IEEE source address is assigned at the factory and can be read with the **SL** (Serial Number Low) and **SH** (Serial Number High) commands. A device uses its unique 64-bit address as its Source Address if its **MY** (16-bit Source Address) value is 0xFFFF or 0xFFFE. Since the default value for **MY** is **0**, devices use short source addressing by default.

Send packets to a specific device in Transparent API mode

To send a packet to a specific device using 64-bit addressing:

- Set the Destination Address (**DL** + **DH**) of the sender to match the Source Address (**SL** + **SH**) of the intended destination device.

To send a packet to a specific device using 16-bit addressing:

1. Set the **DL** parameter to equal the **MY** parameter of the intended destination device.
2. Set the **DH** parameter to 0.

Addressing modes

802.15.4 frames have a source address, a destination address, and a destination PAN ID in the over-the-air (OTA) frame. The source and destination addresses may be either long or short and the destination address may be either a unicast or a broadcast. The destination PAN ID is short and it may also be the broadcast PAN ID (**ID** is set to 0xFFFF).

In Transparent mode, the destination address is set by the **DH** and **DL** parameters, but, in API mode, it is set by the type of TX request used: TX Request: 64-bit address frame - 0x00 or TX Request: 16-bit address - 0x01 frames. In either Transparent mode or API mode, the destination PAN ID is set with the **ID** parameter, and the source address is set with the **MY** parameter if **MY** is less than **0xFFFE**, otherwise the source address is set with the device's serial number (**SH** and **SL**).

Broadcasts and unicasts

Broadcasts are identified by the 16-bit short address of **0xFFFF**. Any other destination address is considered a unicast and is a candidate for acknowledgments, if enabled.

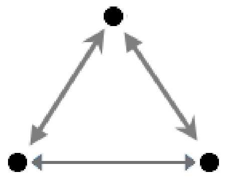
Broadcast PAN ID

The Broadcast PAN ID is also **0xFFFF**. Its effect is to traverse all PANs in the vicinity of a local device.

Short and long addresses

A short address is 16 bits and a long address is 64 bits. The short address is set with the **MY** parameter. If the short address is **0xFFFE**, then the address of the device is long and it is the serial number of the device as read by the **SH** and **SL** parameters.

Peer-to-peer networks



By default, XBee3 802.15.4 RF Modules are configured to operate within a peer-to-peer network topology and therefore are not dependent upon master/slave relationships. Our peer-to-peer architecture features fast synchronization times and fast cold start times. This default configuration accommodates a wide range of RF data applications.

To form a peer-to-peer network, set each device to the same channel and PAN ID and configure either a unique short address (**MY**) for each device or set **MY** to **0xFFFF** to use the unique long addresses.

Master/slave networks

In a Master Slave network, there is a coordinator and one or more end devices. When end devices associate to the coordinator, they become members of that Personal Area Network (PAN). As such, they share the same channel and PAN ID. PAN IDs must be unique to prevent miscommunication between PANs. Depending on the **A1** and **A2** parameters, association may assist in automatically assigning the PAN ID and the channel. These parameters are specified below based on the network role (end device or coordinator).

End device association

End device association occurs if **CE** is **0** and **A1** has bit 2 set. See the following table and A1 (End Device Association).

Bit	Hex value	Meaning
0	0x01	Allow PAN ID reassignment
1	0x02	Allow channel reassignment
2	0x04	Auto association
3	0x08	Poll coordinator on pin wake

By default, **A1** is 0, which disables association and causes a device to operate in peer-to-peer mode. When bit 2 is set, the module becomes an end device and associates to a coordinator. This is done by sending out an active scan to detect beacons from nearby networks. The active scan iterates through each channel defined by **SC** and transmits a Beacon Request command to the broadcast address and the broadcast PAN ID. It then listens on that channel for beacons from any coordinator operating on that channel. Once that time expires, the active scan selects the next channel, repeating until all the channels defined by **SC** have been scanned.

If **A1** is **0x04** (bit 0 clear, bit 1 clear, and bit 2 set), then the active scan will reject all beacons that do not match both the configured PAN ID and the configured channel. This is the best way to join a particular coordinator.

If **A1** is **0x05** (bit 0 set, bit 1 clear, and bit 2 set), then the active scan will accept a beacon from any PAN ID, providing the channel matches. This is useful if the channel is known, but not the PAN ID.

If **A1** is **0x06** (bit 0 clear, bit 1 set, and bit 2 set), then the active scan will accept a beacon from any channel, providing the PAN ID matches. This is useful if the PAN ID is known, but not the channel.

If **A1** is **0x07** (bit 0 set, bit 1 set, and bit 2 set), then the active scan will accept a beacon from any PAN ID and from any channel. This is useful when the network does not matter, but the one with the best signal is desired.

Whenever multiple beacons are received that meet the criteria of the active scan, then the beacon with the best link quality is selected. This applies whether **A1** is **0x04**, **0x05**, **0x06**, or **0x07**.

Before the End Device joins a network, the Associate LED will be on solid. After it joins a network, the Associate LED will blink twice per second. You can also query the association status with AI command or by observing modem status frames when the end device is operating in API mode.

If association parameters are changed after the end device is associated, the end device will leave the network and re-join in accordance with the new configuration parameters.

After an end device successfully joins a network, the **DH** and **DL** parameters on the device are updated to point towards the address of the coordinator it associated with. This allows communication to the coordinator to occur automatically in Transparent mode, and ensures that indirect messaging poll requests are sent to the correct address—see Direct and indirect transmission.

Additionally, after associating, an end device has MY command set to **0xFFFE**, indicating that the newly associated end device should use its 64-bit address. After associating, if you want a 16-bit address for the end device, set **MY** again.

Note **MY** is reset to **0xFFFE** if the end device needs to leave and re-associate with the coordinator.

If a coordinator changes channel or PAN ID, the end device is not informed of the change and indicates that it is still associated. You can set DA (Force Disassociation) on the end device to force it to leave the network and attempt to join again, validating that the end device can still communicate with the coordinator.

Coordinator association

A device becomes a coordinator and allows association if **CE** is 1 and **A2** has bit 2 set. See the following table and A2 (Coordinator Association).

Bit	Hex value	Meaning
0	0x01	Allow PAN ID reassignment
1	0x02	Allow channel reassignment
2	0x04	Allow association

By default, **A2** is 0, which prevents devices from associating to the coordinator. So, if **CE** is **1** and **A2** bit 2 is **0**, the device still creates a network, but end devices are unable to associate to it.

Note In this configuration, depending on the value of SP command the device might send messages indirectly—see Direct and indirect transmission.

If **A2** bit 2 is set, then joining is allowed after the coordinator forms a network.

If **A2** bit 0 is set, the coordinator performs an active scan. The active scan process sends a beacon request to the broadcast address (0xFFFF) and the broadcast PAN ID (0xFFFF) and listens for beacons responses. This process is repeated for each channel specified in **SC**.

If none of the beacons received during the active scan process match the ID parameter of the coordinator, then its ID parameter will be the PAN ID of the new network it forms. However, if a beacon response matches the PAN ID of the coordinator, the coordinator forms a PAN with a unique PAN ID.

If **A2** bit 0 is clear, then the coordinator forms a network on the PAN ID identified by the **ID** parameter, without regard to another network that might have the same PAN ID.

If **A2** bit 1 is set, the coordinator performs an energy scan, similar to the active scan. It will listen on each channel specified in the **SC** parameter. After the scan is complete, the channel with the least energy is selected to form the new network.

If **A2** bit 1 is clear, then no energy scan is performed and the **CH** parameter is used to select the channel of the new network.

If bits 0 and 1 of **A2** are both set, then an active scan is performed followed by an energy scan. However, the channels on which the active scan finds a coordinator are eliminated as possible channels for the energy scan, unless such an action would eliminate all channels. If beacons are found on all channels in the channel mask, then then the energy scan behaves the same as it would if beacons are not found on any of those channels. Therefore, the active scan will be performed on all channels in the channel mask. Then, an energy scan will be performed on the channels in the channel mask that did not find a coordinator.

Depending on the result of the active scan, the set of channels for the energy scan varies. If a PAN ID is found on all the channels in the channel mask, then the energy scan operates on all the channels in the channel mask. If at least one of the channels in the channel mask did not find a PAN ID, then the channels with PAN IDs are eliminated from consideration for the energy scan. After the energy scan completes, the channel with the least energy is selected for forming the new network.

Whenever **CE**, **ID**, **A2**, or **MY** changes, the coordinator will re-form the network. Any end devices associated to the coordinator prior to changing one of these parameters will lose association. For this reason, it is important not to change these parameters on a coordinator unless needed, or configure end devices to be flexible about what network they associate with the **A1** command.

Before the Coordinator forms a network, the Associate LED will be on solid. After it forms a network, the Associate LED will blink once per second.

Association indicators

There are two types of association indicators: Asynchronous device status messages, and on demand queries. Asynchronous device status messages occur whenever a change occurs and API mode is enabled. On demand queries occur when the **AI** command is issued, which can occur in Command mode, in API mode, or as a remote command.

Modem status messages

Not all device status messages are related with association, but for completeness all device status types reported by XBee3 802.15.4 RF Module are listed in the following table.

Type	Meaning
0x00	Hardware reset.
0x01	Watchdog reset.
0x02	End device successfully associated with a coordinator.
0x03	End device disassociated from coordinator or coordinator failed to form a new network.

Type	Meaning
0x06	Coordinator formed a new network.
0x0D	Input voltage is too high, which prevents transmissions.

Association indicator status codes

The XBee3 802.15.4 RF Module can potentially give any of the status codes in response to AI command in the following table.

Code	Meaning
0x00	Coordinator successfully started, End device successfully associated, or operating in peer to peer mode where no association is needed.
0x03	Active Scan found a PAN coordinator, but it is not currently accepting associations.
0x04	Active Scan found a PAN coordinator in a beacon-enabled network, which is not a supported feature.
0x05	Active Scan found a PAN, but the PAN ID does not match the configured PAN ID on the requesting end device and bit 0 of **A1** is not set to allow reassignment of PAN ID.
0x06	Active Scan found a PAN on a channel does not match the configured channel on the requesting end device and bit 1 of **A1** is not set to allow reassignment of the channel.
0x0C	Association request failed to get a response.
0x13	End device is disassociated or is in the process of disassociating.
0xFF	Initialization time; no association status has been determined yet.

Direct and indirect transmission

There are two methods to transmit data:

- Direct transmission: data is transmitted immediately to the Destination Address
- Indirect transmission: a packet is retained for a period of time and is only transmitted after the destination device (source address = destination address) requests the data.

Indirect transmissions can only occur on a device configured to be an indirect messaging coordinator. Indirect transmissions are useful to ensure packet delivery to a sleeping device. Indirect messaging allows messages to reliably be sent asynchronously to sleeping end devices, or operate like an incoming mailbox for a P2P network. A TX request can be made when the end device is sleeping and unable to receive RF data, and instead of being immediately send to an inoperative device, the packet is queued by the indirect messaging coordinator until the end device wakes or polls it for data.

Note that indirect messaging works best with association and end devices cyclically sleeping, but can be used in a P2P configuration by setting CE command to **1** on the device that you want to hold the indirect messages and configuring the other device to poll correctly. In the context of indirect messaging, an end device refers not just to a device with A1 (End Device Association) set to associate but the target of an indirect message. Similarly, an indirect messaging coordinator does not have to allow association (A2 (Coordinator Association)) to send messages indirectly.

Configure an indirect messaging coordinator

A device becomes an indirect messaging coordinator once CE command = **1** and SP command is not **0**. We recommend ensuring that **SP** and **ST** are set to the same values on the indirect messaging coordinator and end device, even if the indirect messaging coordinator is not configured to sleep. This is to allow the indirect messaging coordinator to send messages directly if it knows the end device is awake and sleeping cyclically.

If you are going to use a Master/Slave network with indirect messaging, ensure that the indirect messaging coordinator is also the network coordinator by allowing association (set bit 2 of A2 (Coordinator Association) to **1**).

Send indirect messages

To send an indirect message, ensure that the previous requirements are met and transmit normally. The indirect messaging coordinator queues the message until the end device requests data or the message is in the indirect queue for 2.5 times the value of **SP**.

Ensure that the message is sent to the addressed specified by MY command on the end device. If **MY** on the end device is **0xFFFF** or **0xFFFE**, then you must use the 64-bit address, otherwise use the value of **MY**. Even though an end device configured with a short address always receives direct transmissions destined to its 64-bit address, it will not receive an indirect message directed at its 64-bit address if it is configured to use a 16-bit address.

If the indirect messaging coordinator is operating in API mode, then after transmitting an indirect message the usual TX status frame (Transmit Status frame - 0x8B or TX Status frame - 0x89) is not immediately generated by the device. If the end device polls for the data within the timeout (2.5 * **SP**), then a TX status frame with status **0x00** (message sent) is sent. If the message is discarded due to the timeout expiring, the status frame is **0x03** (message purged).

After receiving a poll request and transmitting data to an end device, the indirect messaging coordinator sends all messages directly until **ST** time has elapsed. This is because after receiving RF data, the end device stays awake for **ST** time if configured in Cyclic Sleep mode (SM = 4). After **ST** time has elapsed, messages are sent indirectly again.

The Coordinator currently is able to retain up to five indirect messages.

Receive indirect messages

End devices must poll the indirect messaging coordinator in order to receive indirect messages.

There are three ways to generate a poll request:

- End devices using cyclic sleep automatically send a poll to the coordinator when they wake up unless **SO** bit 0 is set.
- End devices using pin sleep may be configured to send a poll on a pin wakeup by setting bit 3 of **A1**.
- Use FP (Force Poll) to manually send a poll to the coordinator. In Transparent mode, the poll request is not sent until the command is exited.

The poll is sent to the address located in **DH** and **DL**, so ensure that they are set to match the coordinator's source addressing mode. If the end device (**A1** bit 2 set) has associated with a coordinator (**A2** bit 2 set and **CE** = **1**), then **DH** and **DL** are automatically set to the correct values. If you use indirect messaging in a P2P network, **DH** and **DL** have to be set manually on the end device to point towards the indirect messaging coordinator.

It is more difficult to use indirect messaging with pin sleep than with cyclic sleep because the end device must wake up periodically to poll for the data from the coordinator. Otherwise, the coordinator discards the data after **SP***2.5 time. It is also important to keep the pin woke device awake for **ST** time after receiving indirect messages, otherwise the coordinator could attempt to transmit directly while the end device is asleep, and the transmission will fail. For this reason we recommend only using indirect messaging with cyclic sleep.

Encryption

The XBee3 802.15.4 RF Module supports AES 128-bit encryption. 128-bit encryption refers to the length of the encryption key entered with the **KY** command (128 bits = 16 bytes). The 802.15.4 protocol specifies eight security modes, enumerated as shown in the following table.

Level	Name	Encrypted?	Length of message integrity check	Packet length overhead
0	N/A	No	0 (no check)	0
1	MIC-32	No	4	9
2	MIC-64	No	8	13
3	MIC-128	No	16	21
4	ENC	Yes	0 (no check)	5
5	ENC-MIC-32	Yes	4	9
6	ENC-MIC-64	Yes	8	13
7	ENC-MIC-128	Yes	16	21

The XBee3 802.15.4 RF Module only supports security levels 0 and 4. It does not support message integrity checks. **EE 0** selects security level 0 and **EE 1** selects security level 4. When using encryption, all devices in the network must use the same 16-byte encryption key for valid data to get through. Mismatched keys will corrupt the data output on the receiving device. Mismatched **EE** parameters will prevent the receiving device from outputting received data.

Working from a maximum packet size of 116 bytes, encryption affects the maximum payload as shown in the following table.

Factor	Effect on maximum payload	Comment
Compatibility mode	Force to 95	If **C8** bit 0 is set, all packets are limited to 95 bytes, regardless of other factors listed below. This is how the Legacy 802.15.4 module (S1 hardware) functions.
Packet overhead	Reduce by 5	This penalty for enabling encryption is unavoidable due to the 802.15.4 protocol.
Source address	Reduce by 6	This penalty is unavoidable because the 802.15.4 requires encrypted packets to be sent with a long source address, even if a short address would otherwise be used.

Factor	Effect on maximum payload	Comment
Destination address	Reduce by 6	This penalty only applies if sending to a long address rather than a short address.
App header	Reduce by 4	The app header for encryption is 4 bytes long. This penalty only applies if **MM** = 0 or 3.

Because of the two mandatory reductions when using encryption, no packet can exceed 116 - (5+6) =105 bytes. The other options may further reduce the maximum payload to 101 bytes, 99 bytes, or 95 bytes.

When operating in API mode and not using encryption, if the source address is long, the receiving device outputs an RX Indicator (0x80) frame for received data. But, if the source address is short, the receiving device outputs a Receive Packet (0x81) frame for received data. These same rules apply for encryption if **MM** is **0** or **3**. This is possible because the four-byte encryption App header includes the short address of the sender and the long received address is not used for API output. If encryption is enabled with **MM** of **1** or **2**, then no App header exists, the source address is always long, and the receiving device in legacy API mode (**AP** = **2**) always outputs a RX Packet: 64-bit Address frame - 0x80.

Maximum payload

The absolute maximum payload size for an 802.15.4 packet is 116 bytes. Depending on module configuration, the actual maximum payload size will be reduced.

If you attempt to send an API packet with a larger payload than specified, the device responds with a Transmit Status frame (0x89 and 0x8B) with the Status field set to 74 (Data payload too large). When operating in transparent mode, if you attempt to send data larger than the maximum payload size, the data will be packetized and sent as multiple over-the-air transmissions. For more information, see Serial-to-RF packetization.

Maximum payload rules

1. If you enable transmit compatibility (**C8**) with the Legacy 802.15.4 module (S1 hardware):
   - There is a fixed maximum payload of 100 bytes
   - The rest of the rules do not apply. They apply only when you disable transmit compatibility with the Legacy 802.15.4 module.
2. The maximum achievable payload is 116 bytes. This is achieved when:
   - Not using encryption.
   - Not using the application header (**MM** is set to 1 or 2).
   - Using the short source address.
   - Using the short destination address.
3. If you are using the application header, the maximum achievable payload is reduced by:
   - 2 bytes if not using encryption (**EE** = **0**)
   - 4 bytes if using encryption (**EE** = **1**)
4. If you are using the long source address (**MY** = **0xFFFE**), the maximum achievable payload is reduced by 6 bytes (size of long address (8) - size of short address (2) = 6).

5. If you are using encryption, the source addresses are promoted to long source addresses, so the maximum achievable payload is reduced by 6 bytes.
6. If you are using the long destination address, the maximum achievable payload is reduced by 6 bytes (the difference between the 8 bytes required for a long address and the 2 bytes required for a short address).
7. If you are using encryption, the maximum achievable payload is reduced by 5 bytes.

Note You can query NP command to determine the maximum achievable payload size based on current parameters. **NP** always assumes a long destination address will be used.

Maximum payload summary tables

The following table indicates the maximum payload when using transmit compatibility with Legacy 802.15.4 modules (S1 hardware).

Encryption	
Enabled	**Disabled**
95 B	100 B

The following table indicates the maximum payload when using the application header and not using encryption. Increment the maximum payload in 2 bytes if you are not using the application header.

	Destination address	
Source address	Short	Long
Short	114 B	108 B
Long	108 B	102 B

The following table indicates the maximum payload when using the application header and using encryption. Increment the maximum payload in 4 bytes if you are not using the application header.

	Destination address	
Source address	Short	Long
Short	101 B	95 B
Long	101 B	95 B

Working with Legacy devices

The Legacy 802.15.4 module (S1 hardware) transmits packets one by one. It does not transmit a packet until it receives all expected acknowledgments of the previous packet or the timeout expires.

The XBee/XBee-PRO S2C 802.15.4 and XBee3 802.15.4 RF Modules enhance transmission by implementing a transmission queue that allows the device to transmit to several devices at the same time. Broadcast transmissions are performed in parallel with the unicast transmissions.

This enhancement in the XBee/XBee-PRO S2C 802.15.4 and XBee3 802.15.4 RF Modules can produce problematic behavior under certain conditions if the receiver is a Legacy 802.15.4 module (S1 hardware).

The conditions are:

- The sender is an XBee3 802.15.4 RF Module, and the receiver is a Legacy 802.15.4 module.
- The sender has the Digi header enabled (**MM** = 0 or 3) and **RR** (XBee Retries) > 0.
- The sender sends broadcast and unicast messages at the same time to the Legacy 802.15.4 module without waiting for the transmission status of the previous packet.

The effect is:

- The receiver may display duplicate packets.

The solution is:

- Set bit 0 of the **C8** (802.15.4 compatibility) parameter to 1 to enable TX compatibility mode in the XBee3 802.15.4 RF Module. This eliminates the transmission queue to avoid sending to multiple addresses simultaneously. It also limits the packet size to the levels of the Legacy 802.15.4 module.

For information on the specific differences between an XBee3 and Legacy 802.15.4 devices, refer to the Digi XBee3 802.15.4 Migration Guide.

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.

Remote configuration commands

When running in API mode, the firmware has provisions to send configuration commands to remote devices using Remote AT Command Request frame - 0x17. You can use this frame to send commands to a remote device to read or set command parameters.

CAUTION! It is important to set the short address to 0xFFFE when sending to a long address. Any other value causes the long address to be ignored. This is particularly problematic in the case where nodes are set up with default addresses of 0 and the 16-bit address is erroneously left at 0. In that case, even with a correct long address the remote command goes out to all devices with the default short address of 0, potentially resulting in harmful consequences, depending on the command.

Send a remote command

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

1. The 64-bit address of the remote device.
2. The correct command options value.
3. The command and parameter data (optional). If (and *only* if) all nodes in the PAN have unique short addresses, then remote configuration commands can be sent to 16-bit short addresses by setting the short address in the API frame for Remote AT commands. In that case, the 64-bit address is unused and does not matter.

Apply changes on remote devices

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

1. Set the Apply Changes option bit in the Remote AT Command Request frame (0x17).
2. Issue an **AC** (Apply Changes) command to the remote device.
3. Issue a **WR** + **FR** command to the remote device to save changes and reset the device.

Remote command responses

If the remote device receives a Remote AT Command Request (0x17 frame type), the remote sends an AT Command Response (0x88 frame type) back to the device that sent the remote command. The AT command response indicates the status of the command (success, or reason for failure), and in the case of a command query, it includes the parameter value.

The device that sends a remote command will not receive a remote command response frame if the frame ID in the remote command request is set to 0 , indicating that the request is sent without acknowledgment.

Node discovery

Node discovery has three variations as shown in the following table:

Commands	Syntax	Description
ND command	**ND**	Seeks to discover all nodes in the network (on the current PAN ID).
ND command	**ND**<NI String>	Seeks to discover if a particular node named <NI String> is found in the network.
DN command	**DN**<NI String>	Sets **DH**/**DL** to point to the address (64-bit or 16-bit depending on the **MY** value of the matching node) of the node whose <NI String> matches.

The node discovery command (without an **NI** string designated) sends out a broadcast to every node in the PAN ID. Each node in the PAN sends a response back to the requesting node after a jittered time delay to ensure reliable delivery.

About node discovery

The node discovery command (without an NI string designated) sends out a broadcast to every node in the PAN ID. Each node in the PAN sends a response back to the requesting node.

When the node discovery command is issued in AT command mode, all other AT commands are inhibited until the node discovery command times out, as specified by the **NT** parameter. After the timeout, an extra CRLF is output to the terminal window, indicating that new AT commands can be entered. This is the behavior whether or not there were any nodes that responded to the broadcast.

When the node discovery command is issued in API mode, the behavior is the same except that the response is output in API mode. If no nodes respond, there will be no responses at all to the node discover command. The requesting node is not able to process a new AT command until **NT** times out.

Node discovery in compatibility mode

Node discovery (without an **NI** string parameter) in compatibility mode operates the same in compatibility mode as it does outside of compatibility mode with one minor exception:

If **C8** bit 1 is set and if requesting node is operating in API mode and if no responses are received by the time **NT** times out, then an API AT command response of OK (API frame type 0x88) is sent out the serial port rather than giving no response at all, which would happen if **C8** bit 1 is not set.

Directed node discovery

The directed node discovery command (**ND** with an **NI** string parameter) sends out a broadcast to find a node in the network with a matching **NI** string. If such a node exists, it sends a response with its information back to the requesting node.

In Transparent mode, the requesting node will output an extra CRLF following the response from the designated node and the command will terminate, being ready to accept a new AT command. In the event that the requested node does not exist or is too slow to respond, the requesting node outputs an **ERROR** response after **NT** expires.

In API mode, the response from the requesting node will be output in API mode and the command will terminate immediately. If no response comes from the requested node, the requesting node outputs an error response in API mode after **NT** expires.

Directed node discovery in compatibility mode

The behavior of the Legacy 802.15.4 module (S1 hardware) varies with the default behavior described above for the directed node discovery command. The Legacy module does not complete the command until **NT** expires, even if the requested node responds immediately. After **NT** expires, it gives a successful response, even if the requested node did not respond. To enable this behavior to be equivalent to the Legacy 802.15.4 module, set bit 1 of the **C8** parameter.

Destination Node

DN command with anNI command string parameter sends out a broadcast containing the **NI** string being requested. The responding node with a matching **NI** string sends its information back to the requesting node. The local node then sets **DH**/**DL** to match the address of the responding node. As soon as this response occurs, the command terminates successfully. If operating in Command mode, an **OK** string is output and Command mode exits. In API mode another AT command may be entered.

If an **NI** string parameter is not provided, the **DN** command terminates immediately with an error. If a node with the given **NI** string doesn't respond, the **DN** command terminates with an error after **NT** times out.

Unlike **ND** (with or without an **NI** string), **DN** does not cause the information from the responding node to be output; rather it simply sets **DH**/**DL** to the address of the responding node. If the responding node has a short address, then **DH**/**DL** is set to that short address (with **DH** at 0 and **DL** set to the value of **MY**). If the responding node has a long address (**MY** is **0xFFFE**), then **DH**/**DL** are set to the **SH**/**SL** of the responding node.

Sleep support

Sleep is implemented to support installations where a mains power source is not available and a battery is required. In order to increase battery life, the device sleeps, which means it stops operating. It can be woken by a timer expiration or a pin.

Sleep modes

Sleep modes enable the device to enter states of low-power consumption when not in use. To enter Sleep mode, the following conditions must be met:

- A valid sleep mode is selected via **SM** (**SM** = **1**, **4**, or **5**)
- SLEEP_RQ/DTR (TH pin 9/SMT pin 10) is asserted (when **SM** = **1** or **5**)
- The device is idle (no data transmission or reception) for the amount of time defined by ST command (when **SM** = **4** or **5**)

The following table shows the sleep mode configurations.

Sleep mode	Description
SM 0	No sleep
SM 1	Pin sleep
SM 4	Cyclic sleep
SM 5	Cyclic sleep with pin wake-up

Pin Sleep mode (SM = 1)

Pin Sleep mode minimizes quiescent power (power consumed when in a state of rest or inactivity). In order to use Pin Sleep mode, configure D8 command (TH pin 9/SMT pin 10) for SLEEP_RQ input (**D8** = **1**). This mode is voltage level-activated; when SLEEP_RQ is asserted, the device finishes any transmit or receive activities, enters Idle mode, and then enters a state of sleep. The device does not respond to either serial or RF activity while in pin sleep.

To wake a sleeping device operating in Pin Sleep mode, de-assert SLEEP_RQ. The device wakes when SLEEP_RQ is de-asserted and is ready to transmit or receive when the CTS line is low. When waking the device, the pin must be de-asserted at least two 'byte times' after CTS goes low. This assures that there is time for the data to enter the DI buffer.

Devices with SPI functionality can use the SPI_SSEL pin instead of **D8** for pin sleep control. If **D8** = **0** and **P7** = **1**, SPI_SSEL takes the place of SLEEP_RQ and functions as described above. In order to use SPI_SSEL for sleep control while communicating on the UART, the other SPI pins must be disabled (**P5**, **P6**, and **P8** set to **0**). See Low power operation for information on using SPI_SSEL for sleep control while communicating over SPI.

Cyclic Sleep mode (SM = 4)

The Cyclic Sleep modes allow devices to periodically check for RF data. When the **SM** parameter is set to **4**, the XBee3 802.15.4 RF Module is configured to sleep, then wakes once per cycle to check for data from a coordinator. The Cyclic Sleep Remote sends a poll request to the coordinator at a specific interval set by the **SP** (Cyclic Sleep Period) parameter. The coordinator transmits any queued data addressed to that specific remote upon receiving the poll request.

If the coordinator does not respond with queued data and no UART activity is detected, the device will immediately sleep. If it detects any activity (RF or UART), then the device wakes for **ST** time. You can also set **SO** bit 8 to force the device to always wake for the full **ST** time.

ON_SLEEP goes high and CTS goes low each time the remote wakes, allowing for communication initiated by the remote host if desired.

Cyclic Sleep with Pin Wake-up mode (SM = 5)

Use this mode to wake a sleeping remote device through either the RF interface or by de-asserting SLEEP_RQ for event-driven communications. The cyclic sleep mode works as described previously with the addition of a pin-controlled wake-up at the remote device.

The SLEEP_RQ pin is level-triggered. The device wakes when a low is detected then sets $\overline{\text{CTS}}$ low as soon as it is ready to transmit or receive. The device stays awake as long as SLEEP_RQ is low; once SLEEP_RQ goes high the device returns to cyclic sleep operation.

Any activity resets the ST command timer, so the device goes back to sleep only after there is no activity for the duration of the timer.

Sleep parameters

The following AT commands are associated with the sleep modes. See the linked commands for the parameter's description, range and default values.

- SM command
- SP command
- ST command
- DP (Disassociated Cyclic Sleep Period)
- SO command

Sleep current

The following table shows the sleep current during the XBee3 802.15.4 RF Module sleep modes.

Sleep mode	SM command setting	Sleep current
Pin sleep	1	
Cyclic sleep	4	<1 µA @ 25 °C
Cyclic sleep with pin wake-up	5	<1 µA @ 25 °C

You can make devices use low sleep current by driving PWM outputs high during sleep and by using internal pull-ups/pull-downs on disabled/unused pins.

The sleep pins are set up for sleeping as specified in the following section, Sleep pins. Additionally, pins that are outputs (other than PWM outputs) continue to output the same levels during sleep. Normally, this means that pins configured for output high or low will output high or low accordingly. However, if I/O line passing overrides the output, the output level is maintained during the sleep time.

Sleep pins

The following table describes the three external device pins associated with sleep. Refer to the XBee3 Hardware Reference Guide. See the *XBee3 RF Module Hardware Reference Manual* for the pinout of your device.

Pin name	Description
DTR/SLEEP_RQ	For **SM** = 1, high puts the device to sleep and low wakes it up. For **SM** = **5**, a high to low transition wakes the device until the pin transitions back to a high state.
SPI_$\overline{\text{SSEL}}$	Alternative SLEEP_RQ line for devices operating in SPI. See Low power operation for further information.
$\overline{\text{CTS}}$	If **D7** = 1, high indicates that the device is asleep and low indicates that it is awake and ready to receive serial data.
ON_$\overline{\text{SLEEP}}$	Low indicates that the device is asleep and high indicates that it is awake.

Sleep conditions

Since instructions stop executing while the device is sleeping, it is important to avoid sleeping when the device has work to do. For example, the device will not sleep if any of the following are true:

1. The device is operating in Command mode, or in the process of getting into Command mode with the **+++** sequence.
2. The device is processing AT commands from API mode
3. The device is processing remote AT commands
4. Something is queued to the serial port and that data is not blocked by $\overline{\text{RTS}}$ flow control

If each of the above conditions are false, then sleep may still be blocked in these cases:

1. Enough time has not expired since the device has awakened.
    a. If the device is operating in pin sleep, the amount of time needed for one character to be received on the UART is enough time.
    b. If the device is operating in cyclic sleep, enough time is defined by a timer. The duration of that timer is:
        i. defined by **ST** if in **SM** 5 mode and it is awakened by a pin
        ii. 30 ms to allow enough time for a poll and a poll response
        iii. 750 ms to allow enough time for association, in case that needs to happen
    c. In addition, the wake time is extended by an additional **ST** time when new OTA data or serial data is received.
2. Sleep Request pin is not asserted when operating in pin sleep mode
3. Data is waiting to be sent OTA.

AT commands

Network and security commands

The following commands affect the 802.15.4 network.

CH command

Set or read the operating channel devices used to transmit and receive data.

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 and to reduce interference.

The command uses IEEE 802.15.4 channel numbers.

Parameter range

0xB - 0x1A

Default

0xC (channel 12)

ID command

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.

Setting **ID** to **0xFFFF** indicates a global transmission for all PANs. It does not indicate a global receive.

Parameter range

0 - 0xFFFF

Default

0x3332

C8 command

Sets or displays the operational compatibility with the Legacy 802.15.4 device (S1 hardware). This parameter should only be set when operating in a mixed network that contains XBee Series 1 devices.

Parameter range

0 - 3

Bit field:

Bit	Meaning	Setting	Description
0[1]	TX compatibility	0	Transmissions are optimized as follows: 1. Maximum transmission size is affected by multiple factors (**MM**, **MY**, **DH**, **DL**, and **EE**). See Maximum payload rules. In the best case, with no app header, short source and destination addresses, and no encryption, the maximum transmission size is 116 bytes. 2. Multiple messages can be present simultaneously on the active queue, providing they are all destined for different addresses. This improves performance.
		1	Transmissions operate like the Legacy 802.15.4 module, which means the following: 1. Maximum transmission size is 95 bytes for encrypted packets and 100 bytes for un-encrypted packets. These maximum transmission sizes are not adjusted upward for short addresses or for lack of an APP header. 2. Only one transmission message can be active at a time, even if other messages in the queue would go to a different destination address.
1	Node Discovery compatibility	0	Node discovery operates like other XBee devices and not like the Legacy 802.15.4 module. This means the following: 1. A directed **ND** request terminates after the single response arrives. This allows the device to process other commands without waiting for the **NT** to time out. 2. The device outputs an error response to the directed **ND** request if no response occurs within the time out.
		1	The module operates like the Legacy 802.15.4 module, which has the following effect: 1. When the expected response arrives, the command remains active until **NT** times out. (**NT** defaults to 2.5 seconds.) This prevents the device from processing any other AT command, even if the desired response occurs immediately. 2. When the timeout occurs, the command silently terminates and indicates success, whether or not a response occurred within the **NT** timeout.

Default

0

[1]This bit does not typically need to be set. However, when the XBee3 802.15.4 RF Module is streaming broadcasts in transparent mode to a Legacy 802.15.4 module (S1 hardware), and **RR** > 0, set this bit to avoid a watchdog reset on the Legacy 802.15.4 module.

NI command

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.

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

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

Parameter range

A string of case-sensitive ASCII printable characters from 1 to 20 bytes in length. A carriage return or a comma automatically ends the command.

Default

0x20 (an ASCII space character)

ND command

This command reports the following information after a jittered time delay. Node discover response when issued in Command mode:

16-bit Short Address (**MY** command)<CR>

Upper portion of the Long 64-bit Address (**SH** command)<CR>

Lower portion of the Long 64-bit Address (**SL** command)<CR>

Signal Strength in -dBm (**DB** command)<CR>

Node Identifier String (**NI** command)<CR>

<CR> (This is part of the response and not the end of command indicator.)

A second carriage return indicates the network discovery timeout (**NT**) has expired.

When operating in API mode and a Network Discovery is issued as a 0x08 or 0x09 frame, the response contains binary data except for the NI string in the following format:

2 bytes for Short Source Address

4 bytes for Upper Long Address

4 bytes for Lower Long Address

1 byte for the signal strength in -dBm (two's complement representation)

NULL-terminated string for NI (Node Identifier) value (maximum 20 bytes without NULL terminator)

Each device that responds to the request will generate a separate AT Command Response frame - 0x88.

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 maximum timeout (13 seconds by default), this value is sent along with the discovery broadcast and determines the random delay the remote nodes use to prevent the responses from colliding.

For more information about the options that affect the behavior of the **ND** command, see NO 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

20-byte printable ASCII string

Default

N/A

DN command

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 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:

1. The receiving device returns the 16-bit network and 64-bit extended addresses in an API Command Response 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. In the case of an ERROR, the device does not exit Command mode. Set the radius of the **DN** command using the **BH** command.

When **DN** is sent as a local AT Command Frame - 0x08:

1. The receiving device returns a success response in a AT Command Response frame - 0x88.
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

N/A

NT command

Sets or displays the amount of time a base node waits for responses from other nodes when using the **ND** (Node Discover) command. The **NT** value is transmitted along with the **ND** command; remote nodes set up a random hold-off time based on this timeout. Once the **ND** command has ended, the base discards any responses it receives.

Parameter range

0x1 - 0xFC (x 100 ms)

Default

0x19 (2.5 seconds)

NO command

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

Parameter range

0 - 1

Default

0x0

MM (MAC Mode)

Use the **MM** command to specify the operating MAC Mode; for more information see MAC Mode configuration.

The MAC Mode serves two purposes:

- Enable/disable the use of a Digi header, which enables advanced features.
- Enable/disable MAC-Layer acknowledgments.

The default configuration enables a Digi-specific header to every RF packet. This header includes information that allows for some advanced features:

- Network discovery support [ND command and DN command]
- Application-layer retries [RR command]
- Duplicate packet detection [RR command]
- Remote AT command support [Remote AT Command Request frame - 0x17]

The presence of the Digi header prevents interoperability with third-party devices. When the Digi header is disabled, encrypted data that is not valid is sent out of the UART and not filtered out. The Digi header can be disabled by setting **MM** to **1** or **2**.

When **MM** is set **1** or **3**, MAC-layer retries are disabled.

Parameter range

0 - 3

Parameter	Configuration	ACKs
0	Digi mode	With ACKs
1	802.15.4	No ACKs
2	802.15.4	With ACKs
3	Digi mode	No ACKs

Default

0

NP command

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

NP is based on multiple factors including the length of the source address, the length of the destination address, the length of the APP header, and whether or not encryption is enabled.

For the purposes of this command, it always assumes a long destination address. This means that if you select a short destination address, you will be able to send up to **NP** + 6 bytes in a single packet.

Note **NP** returns a hexadecimal value. For example, if **NP** returns 0x66, this is equivalent to 102 bytes.

Parameter range

[read-only]

Default

N/A

Coordinator/End Device configuration commands

The following commands configure the device for a master/slave 802.15.4 network.

CE command

The routing mode of the XBee3 802.15.4 RF Module.

The XBee3 802.15.4 RF Module does not allow association until bit 2 of A2 (Coordinator Association) is set.

Parameter range

0 - 1

Parameter	Description
0	End Device
1	Coordinator

Default

0

Note If **CE** = **1** and **SP** is not **0**, then all messages are sent indirectly. See Direct and indirect transmission for more information.

A1 (End Device Association)

Sets or displays the End Device association options.

Parameter range

0 - 0x0F (bit field)

Bit field:

Bit	Meaning	Setting	Description
0	Allow PanId reassignment	0	Only associates with Coordinator operating on PAN ID that matches device ID.
		1	May associate with Coordinator operating on any PAN ID.
1	Allow Channel reassignment	0	Only associates with Coordinator operating on matching **CH** channel setting.
		1	May associate with Coordinator operating on any channel.
2	Auto Associate	0	Device will not attempt association.
		1	Device attempts association until success.
3	Poll coordinator on pin wake	0	Pin Wake does not poll the Coordinator for indirect (pending) data.
		1	Pin Wake sends Poll Request to Coordinator to extract any pending data.
4 - 7	Reserved		

Default

0

A2 (Coordinator Association)

Sets or displays the Coordinator association options. These options are only applicable when configured as a coordinator by setting CE command to **1**.

Parameter range

0 - 7 (bit field)

Bit field:

Bit	Meaning	Setting	Description
0	Allow Pan ID reassignment	0	Coordinator will not perform Active Scan to locate available PAN ID. It operates on ID (PAN ID).
		1	Coordinator performs an Active Scan to determine an available **ID** (PAN ID). If a PAN ID conflict is found, the **ID** parameter will change.
1	Allow Channel reassignment	0	Coordinator will not perform Energy Scan to determine free channel. It operates on the channel determined by the **CH** parameter.
		1	Coordinator performs an Energy Scan to find the quietest channel out of the channels to be scanned determined by the **SC** parameter. The Coordinator then operates on that channel.

Bit	Meaning	Setting	Description
2	Allow Association	0	Coordinator will not allow any devices to associate to it.
		1	Coordinator allows devices to associate to it.
3 - 7	Reserved		

Default

0

SC command

Sets or displays the list of channels to scan for all Active and Energy Scans as a bit field. This affects scans initiated in AS command and ED command commands in Command mode and during End Device Association and Coordinator startup.

Parameter range

0 - 0xFFFF (bit field)

Note A parameter of **0** automatically scans the current channel configured by **CH**.

Bit field mask:

Bit	IEEE 802.15.4 channel
0	Channel 11 (0x0B)
1	Channel 12 (0x0C)
2	Channel 13 (0x0D)
3	Channel 14 (0x0E)
4	Channel 15 (0x0F)
5	Channel 16 (0x10)
6	Channel 17 (0x11)
7	Channel 18 (0x12)
8	Channel 19 (0x13)
9	Channel 20 (0x14)
10	Channel 21 (0x15)
11	Channel 22 (0x16)
12	Channel 23 (0x17)
13	Channel 24 (0x18)
14	Channel 25 (0x19)
15	Channel 26 (0x1A)

Default

0xFFFF

DA (Force Disassociation)

Causes the End Device to immediately disassociate from a Coordinator (if associated) and re-attempt to associate.

Parameter range

-

Default

-

AI command

Reads the Association status code to monitor association progress.

The following table provides the status codes and their meanings.

Status code	Meaning
0x00	Coordinator successfully started, End device successfully associated, or operating in peer to peer mode where no association is needed.
0x03	Active Scan found a PAN coordinator, but it isn't currently accepting associations.
0x05	Active Scan found a PAN, but the PAN ID doesn't match the configured PAN ID on the requesting end device and bit 0 of **A1** is not set to allow reassignment of PAN ID.
0x06	Active Scan found a PAN on a channel that does not match the configured channel on the requesting end device and bit 1 of **A1** is not set to allow reassignment of the channel.
0x0C	Association request failed to get a response.
0x13	End device is disassociated or is in the process of disassociating.
0xFF	Initialization time; no association status has been determined yet.

Parameter range

0 - 0xFF [read-only]

Default

N/A

802.15.4 Addressing commands

The following commands affect the source and destination addressing for the device.

SH command

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

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

Parameter range

0x0013A200 - 0x0013A2FF [read-only]

Default

Set in the factory

SL command

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

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

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

MY command

Sets or displays the device's 16-bit source address. Set **MY** = **0xFFFE** to disable reception of packets with 16-bit addresses. To maintain compatibility with older products, **0xFFFF** is also acceptable to disable the reception of packets with 16-bit addresses. When configured in this way, the 64-bit long source address (**SH**+**SL**) is used for outgoing messages.

Regardless of **MY**, messages addressed to the 64-bit long address of the device are always delivered.

Parameter range

0 - 0xFFFF

Default

0

DH command

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.

This destination address is also used for outgoing I/O samples 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 (**DH** = **0**, **DL** = **0xFFFF**).

Parameter range

0 - 0xFFFFFFFF

Default

0

DL command

Set or display the lower 32 bits of the 64-bit destination address. When you combine **DH** with **DL**, it defines the destination address that the device uses for transmissions in Transparent mode. This destination address is also used for outgoing I/O samples in both Transparent and API modes.

0x000000000000FFFF is the broadcast address (**DH** = **0**, **DL** = **0xFFFF**).

Parameter range

0 - 0xFFFFFFFF

Default

0

RR command

Set or reads the number of application-layer retries the device executes. Application-layer retries are only enabled if a Digi header is present via the **MM** command.

Every transmitted unicast transmission utilizes up to three MAC-Layer retries (if enabled via the **MM** command). If **RR** > 0, a failed unicast transmission will be attempted **RR** times (each application-layer retry will exhaust the three MAC-layer retries).

When transmitting a broadcast message, if **RR** = 0, only one packet is broadcast. If **RR** is > 0, then **RR** + 2 packets are sent on each broadcast. No acknowledgments are returned on a broadcast.

The **RR** value does not need to be set on all devices for retries to work. If retries are enabled, the transmitting device sets a bit in the Digi RF Packet header that requests the receiving device to send an ACK. If the transmitting device does not receive an ACK within 200 ms, it re-sends the packet within a random period up to 48 ms. Each device retry can potentially result in the MAC sending the packet four times (one try plus three retries).

Parameter range

0 - 6

Default

0

TO command

Set/read transmit options for Transparent mode.

Bit	Meaning
0	Disable MAC ACKs.
2	Send to broadcast PAN ID.

Parameter range

0 - 5

Default

0

Security commands

The following commands enable and control the encryption used for RF transmissions.

EE command

Enables or disables 128-bit Advanced Encryption Standard (AES) encryption of RD data transmissions.

The firmware uses the 802.15.4 Default Security protocol and uses AES encryption with a 128-bit key. AES encryption dictates that all devices in the network use the same key, and that the maximum RF packet size is 95 bytes if Tx compatibility is enabled (you set bit 0 of **C8**). If **C8**, bit 0 is not set, see Maximum payload.

When encryption is enabled, the device always uses its 64-bit long address as the source address for RF packets. This does not affect how the **MY** (Source Address), **DH** (Destination Address High) and **DL** (Destination Address Low) parameters work.

If **MM** (MAC Mode) is set to 1 or 2 and **AP** (API Enable) parameter > 0:

With encryption enabled and a 16-bit short address set, receiving devices can only issue RX (Receive) 64-bit indicators. This is not an issue when **MM** = **0** or **3**.

If a device with a non-matching key detects RF data, but has an incorrect key:

When encryption is enabled, non-encrypted RF packets received are rejected and are not sent out the UART.

Parameter range

0 - 1

Parameter	Description
0	Encryption Disabled
1	Encryption Enabled

Default

0

KY command

Sets the 128-bit network security key value that the device uses for encryption and decryption.

This command is write-only and cannot be read. If you attempt to read **KY**, the device returns an **OK** status.

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

The entire payload of the packet is encrypted using the key and the CRC is computed across the ciphertext.

Parameter range

128-bit value (up to 16 bytes)

Default

0

RF interfacing commands

The following AT commands affect the RF interface of the device.

PL command

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

Note If operating on channel 26 (**CH** = 0x1A), output power will be capped and cannot exceed 8 dBm regardless of the **PL** setting.

Parameter range

0 - 4

The following table shows the TX power versus the **PL** setting.

PL setting	XBee3 TX power	XBee3-PRO TX power
4	8 dBm	19 dBm
3	5 dBm	15 dBm
2	2 dBm	8 dBm
1	-1 dBm	3 dBm
0	-5 dBm	-5 dBm

Default

4

PP command

Display the operating output power based on the current configuration (channel and **PL** setting). The values returned are in dBm, with negative values represented in two's complement; for example:

-5 dBm = 0xFB.

Parameter range

0 - 0xFF [read-only]

Default

N/A

CA (CCA Threshold)

Defines 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. The CCA threshold is set upon device initialization, any change to the CCA threshold must be written to flash with the **WR** command and the module reset (power cycle or **FR** command) before the new threshold is observed.

You can set **CA** to 0 to disable CCA; this can improve latency but may cause interference with other 2.4GHz devices when transmitting. You can disable and enable CCA at runtime, which does not require a power cycle.

Parameter range

0 (disabled), 0x28 - 0x64 (-dBm)

Default

0x41

RN command

Defines the minimum value of the back-off exponent in the CSMA-CA algorithm. The Carrier Sense Multiple Access - Collision Avoidance (CSMA-CA) algorithm was engineered for collision avoidance (random delays are inserted to prevent data loss caused by data collisions.

If **RN** = 0, there is no delay between a request to transmit and the first iteration of CSMA-CA.

Unlike CSMA-CD, which reacts to network transmissions after collisions have been detected, CSMA-CA acts to prevent data collisions before they occur. As soon as a device receives a packet that is to be transmitted, it checks if the channel is clear (no other device is transmitting). If the channel is clear, the packet is sent over-the-air. If the channel is not clear, the device waits for a randomly selected period of time, then checks again to see if the channel is clear. After a time, the process ends and the data is lost.

Parameter range

0 - 5 (exponent)

Default

0

DB command

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

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

If the XBee3 802.15.4 RF Module has been reset and has not yet received a packet, **DB** reports **0**.

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

Parameter range

0 - 0xFF [read-only]

Default

N/A

MAC diagnostics commands

The following AT commands are MAC/PHY commands.

AS command

Sends a Beacon Request to a Broadcast address (**0xFFFF**) and Broadcast PAN (**0xFFFF**) on every channel in the scan channel mask—SC command. Active Scan can only be performed locally and returns an ERROR if attempted remotely.

A PanDescriptor is created and returned for every Beacon received from the scan. Each PanDescriptor contains the following information:

CoordAddress (**SH** + **SL** parameters)<CR>

Note If **MY** on the coordinator is set less than 0xFFFF, the **MY** value is displayed.

CoordPanID (**ID** parameter)<CR>

CoordAddrMode <CR>

0x02 = 16-bit Short Address

0x03 = 64-bit Long Address

Channel (**CH** parameter) <CR>

SecurityUse<CR>

ACLEntry<CR>

SecurityFailure<CR>

SuperFrameSpec<CR> (2 bytes):

bit 15 - Association Permitted (MSB)

bit 14 - PAN Coordinator

bit 13 - Reserved

bit 12 - Battery Life Extension

bits 8-11 - Final CAP Slot

bits 4-7 - Superframe Order

bits 0-3 - Beacon Order

GtsPermit<CR>

RSSI<CR> (- RSSI is returned as -dBm)

TimeStamp<CR> (3 bytes)

<CR> (A carriage return indicates the end of the PanDescriptor)

The Active Scan returns one PanDescriptor response per discovered network. Each PanDescriptor has a trailing carriage return <CR> to indicate the end of the frame. The sequence of PanDescriptors has a final trailing carriage return (3 <CR> in sequence indicate the end of the active scan).

If using API Mode, no <CR>'s are returned and a separate response frame is generated for each PanDescriptor. For more information, see Operate in API mode.

Parameter range

N/A

Default

N/A

ED command

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

N/A

Default

N/A

EA command

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

0 - 0xFFFF

Default

0x0

EC (CCA Failures)

Sets or displays the number of frames that were blocked and not sent due to CCA failures or receptions in progress. If CCA is disabled (**CA** is **0**), then this count only increments for frames that are blocked due to receive in progress. When this count reaches its maximum value of **0xFFFF**, it stops counting.

You can reset **EC** to **0** (or any other value) at any time to make it easier to track errors. This value is volatile (the value does not persist in the device's memory after a power-up sequence).

Parameter range

0 - 0xFFFF

Default

0x0

Sleep settings commands

The following commands enable and configure the low power sleep modes of the device.

SM command

Sets or displays the sleep mode of the device.

By default, Sleep Modes are disabled (**SM** = 0) and the device remains in Idle/Receive mode. When in this state, the device is constantly ready to respond to either serial or RF activity.

When operating in Pin Sleep (**SM** = 1), D8 command must be set as a peripheral (**D8**=1) in order for the device to sleep.

Parameter range

0 - 5

Parameter	Description
0	No sleep (disabled)
1	Pin sleep
2	Reserved
3	Reserved
4	Cyclic Sleep Remote
5	Cyclic Sleep Remote with pin wakeup

Default

0

SP command

Sets and reads the duration of time that a remote device sleeps. After the cyclic sleep period is over, the device wakes and checks for data. If data is not present, the device goes back to sleep. The maximum sleep period is 4 hours (**SP** = 0x15F900).

The **SP** parameter is only valid if you configure the end device to operate in Cyclic Sleep (**SM** = 4-5). Coordinator and End Device **SP** values should always be equal.

To send direct messages on a coordinator, set **SP** = 0. If the device is a coordinator (CE command = **1**) and **SP** is not **0**, the device sends all transmissions indirectly, meaning end devices have to poll the coordinator to receive data—FP (Force Poll) or using cyclic sleep.

End Device: **SP** determines the sleep period for cyclic sleeping remotes. The maximum sleep period is 4 hours (0x15F900).

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

Parameter range

0x0 - 0x15F900 (x 10 ms) (4 hours)

Default

0x0

ST command

Sets or displays the wake time of the device.

The **ST** parameter is only valid for end devices configured with Cyclic Sleep settings (**SM** = 4 - 5) and for coordinators. Upon waking the device polls for queued indirect messages and UART data. If it does not detect activity, the device immediately sleeps. The device only stays awake for **ST** time if RF or UART activity is detected upon wakeup or bit 8 of SO command is set to **1**.

Coordinator and End Device **ST** values must be equal.

Parameter range

0x1 - 0x36EE80 (x 1 ms)

Default

0x7D0 (2 seconds)

DP (Disassociated Cyclic Sleep Period)

Sets or displays the sleep period for cyclic sleeping remotes that are configured for Association but that are not associated to a Coordinator. For example, if a device is configured to associate and is configured as a Cyclic Sleep remote, but does not find a Coordinator, it sleeps for **DP** time before reattempting association.

Parameter range

1 - 0x15F900 (x 10 ms)

Default

0x3E8 (10 seconds)

SO command

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.

Parameter range

0 - 0x103

Bit field:

Bit	Setting	Meaning	Description
0	0	Normal operations	A device configured for cyclic sleep will poll for data on waking
	1	Disable wakeup poll	A device configured for cyclic sleep will not poll for data on waking
1	0	Normal operations	A device configured in a sleep mode with ADC/DIO sampling enabled will automatically perform a sampling on wakeup
	1	Suppress sample on wakeup	A device configured in a sleep mode with ADC/DIO sampling enabled will not automatically sample on wakeup
8	0	Normal operations	A device configured for cyclic sleep will wake only momentarily after the expiration of **SP**
	1	Always wake for **ST** time	A device configured for cyclic sleep will always remain awake for **ST** time before returning to sleep
Set all other option bits to 0.			

Default

0

FP (Force Poll)

The **FP** command is deferred until changes are applied. This prevents indirect messages from arriving at the end device while it is operating in Command mode.

Parameter range

N/A

Default

N/A

UART interface commands

The following commands affect the UART serial interface.

BD command

This command configures the serial interface baud rate for communication between the UART port of the device and the host. Standard baud rates can be set using a parameter value of 0 - 8.

Non-standard interface data rates

The firmware interprets any value from 0x4B0 through 0x3D090 as an actual baud rate. When the firmware cannot configure the exact rate specified, it configures the closest approximation to that rate. For example, to set a rate of 57600 b/s send the following command line: **ATBDE100**. Then, to find out the closest approximation, send **ATBD** to the console window. It sends back a response of 0xE0D1, which is the closest approximation to 57600 b/s attainable by the hardware.

Note When using XCTU, you can only set and read non-standard interface data rates using the XCTU **Terminal** tab. You cannot access non-standard rates through the **Modem Configuration** tab.

The following table provides some example **BD** parameters sent versus the parameters stored.

BD parameter sent (HEX)	Interface data rate (b/s)	BD parameter stored (HEX)
0	1200 (standard)	0
4	19,200 (standard)	4
7	115,200 (standard)	7
E100	57,600	E139
1C200	115,200	1C273

Parameter range

Standard baud rates: 0x0 - 0x0A

Non-standard baud rates: 0x12C - 0x0EC400

Parameter	Description
0x0	1200 b/s
0x1	2400 b/s
0x2	4800 b/s
0x3	9600 b/s
0x4	19200 b/s
0x5	38400 b/s
0x6	57600 b/s
0x7	115200 b/s
0x8	230400 b/s
0x9	460,800 b/s
0xA	921,600 b/s

Default

3 (9600 baud)

NB command

Set or read the serial parity settings for UART communications.

The device does not actually calculate and check the parity. It only interfaces with devices at the configured parity and stop bit settings for serial error detection.

Parameter range

0 - 2

Parameter	Description
0	No parity
1	Even parity
2	Odd parity

Default

0

SB command

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

Parameter range

0 - 1

Parameter	Configuration
0	One stop bit
1	Two stop bits

Default

0

FT command

Set or display the flow control threshold.

The device de-asserts $\overline{\text{CTS}}$ when **FT** bytes are in the UART receive buffer. It re-asserts $\overline{\text{CTS}}$ when less than **FT** bytes are in the UART receive buffer.

Parameter range

0x0C - 0xA2 bytes

Default

0x81

RO command

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.

The **RO** command only applies to Transparent mode, it does not apply to API mode.

Parameter range

0 - 0xFF (x character times)

Default

3

AP command

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

For more information, see Serial modes.

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

Parameter range

0 - 2

Parameter	Description
0	API disabled (operate in Transparent mode)

Parameter	Description
1	API enabled
2	API enabled (with escaped control characters)

Default

0

AO command

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.
2	Legacy 802.15.4 API Indicator - 0x80/0x81. Also restricts the Digital Input sampling to **D0** through **D8** and allows for OTA compatibility with legacy S1 and S2C devices.

Default

2

AZ (Extended API Options)

Optionally output additional ZCL messages that would normally be masked by the XBee application.

Use this when debugging OTA firmware updates by enabling client-side messages to be sent out of the serial port.

Parameter range

0 - 2

Parameter	Description
0	Suppress ZCL output
1	Reserved
2	Output supported ZCL packets

Default

0

Command mode options

The following commands affect how Command mode operates.

CC command

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. For more information, see Enter Command mode.

Parameter range

0 - 0xFF

Default

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

CT command

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

Parameter range

2 - 0x1770 (x 100 ms)

Default

0x64 (10 seconds)

GT command

Set the required period of silence before and after the command sequence characters of the Command mode sequence, **GT** + **CC** + **GT** (including spaces). The period of silence prevents inadvertently entering Command mode. For more information, see Enter Command mode.

Parameter range

0x2 - 0x6D3 (x 1 ms)

Default

0x3E8 (one second)

CN command

Executable command. **CN** immediately exits Command mode and applies pending changes.

Parameter range

N/A

Default

N/A

UART pin configuration commands

The following commands are related to pin configuration for the UART interface.

D6 command

Sets or displays the DIO6/$\overline{\text{RTS}}$ configuration (Micro pin 27/SMT pin 29/TH 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 command

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration (Micro pin 24/SMT pin 25/TH 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 enable, low Tx (0 V on transmit, high when idle)
7	RS-485 enable, high Tx (high on transmit, 0 V when idle)

Default

1

P3 command

Sets or displays the DIO13/UART_DOUT configuration (Micro pin 3/SMT pin 3/TH pin 2).

Parameter range

0, 1, 3 - 5

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

Default

1

P4 command

Sets or displays the DIO14/UART_DIN configuration (Micro pin 4/SMT pin 4/TH pin 3).

Parameter range

0, 1, 3 - 5

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

Default

1

SPI interface commands

The following commands affect the SPI serial interface on SMT and MMT variants. These commands are not applicable to the through-hole variant of the XBee3; see **D1** through **D4** and **P2** for through-hole SPI support.

P5 command

Sets or displays the DIO15/SPI_MISO configuration (Micro pin 16/SMT pin 17). This only applies to surface-mount and micro devices.

Parameter range

0, 1, 4, 5

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

Default

1

P6 command

Sets or displays the DIO16/SPI_MOSI configuration (Micro pin 15/SMT pin 16). This only applies to surface-mount and micro devices.

Parameter range

0, 1, 4, 5

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

Default

1

P7 command

Sets or displays the DIO17/SPI_$\overline{\text{SSEL}}$ configuration (Micro pin 14/SMT pin 15). This only applies to surface-mount and micro devices.

Parameter range

0 - 1, 4, 5

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

Default

1

P8 command

Sets or displays the DIO18/SPI_CLK configuration (Micro pin 13/SMT pin 14). This only applies to surface-mount and micro devices.

Parameter range

0, 1, 4, 5

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

Default

1

P9 command

Sets or displays the DIO19/SPI_$\overline{\text{ATTN}}$ configuration (Micro pin 11/SMT pin 12). This only applies to surface-mount and micro devices.

Parameter range

0, 1, 4, 5

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

Default

1

I/O settings commands

The following commands configure the various I/O lines available on the XBee3 802.15.4 RF Module.

D0 command

Sets or displays the DIO0/ADC0/CB configuration (TH pin 20/SMT pin 33).

Parameter range

0 - 5

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

Default

1

CB command

Use **CB** to simulate Commissioning Pushbutton presses in software.

You can enable a physical commissioning pushbutton with D0 command.

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

Parameter range

1, 4

Parameter	Description
1	Keeps device awake for 30 seconds.
4	Restore defaults (equivalent to sending an RE command).

Default

N/A

D1 command

Sets or displays the DIO1/ADC1/TH_SPI_$\overline{\text{ATTN}}$ configuration (Micro pin 30/SMT pin 32/TH pin 19).

Parameter range

SMT/MMT: 0, 2 - 5

TH: 0 - 5

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{ATTN}}$ for the through-hole device N/A for surface-mount device
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D2 command

Sets or displays the DIO2/ADC2/TH_SPI_CLK configuration (Micro pin 29/SMT pin 31/TH pin 18).

Parameter range

SMT/MMT: 0, 2 - 5

TH: 0 - 5

Parameter	Description
0	Disabled
1	SPI_CLK for through-hole devices N/A for surface-mount devices
2	ADC

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

Default

0

D3 command

Sets or displays the DIO3/ADC3/TH_SPI_$\overline{\text{SSEL}}$ configuration (Micro pin 28/SMT pin 30/TH pin 17).

Parameter range

SMT/MMT: 0, 2 - 5

TH: 0 - 5

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{SSEL}}$ for the through-hole device N/A for surface-mount device
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D4 command

Sets or displays the DIO4/TH_SPI_MOSI configuration (Micro pin 23/SMT pin 24/TH pin 11).

Parameter range

SMT/MMT: 0, 3 - 5

TH: 0, 1, 3 - 5

Parameter	Description
0	Disabled
1	SPI_MOSI for the through-hole device N/A for the surface-mount and micro device

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

Default

0

D5 command

Sets or displays the DIO5/ASSOCIATED_INDICATOR configuration (Micro pin 26/SMT pin 28/TH pin 15).

Parameter range

0, 1, 3 - 5

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

Default

1

D8 command

Sets or displays the DIO8/$\overline{\text{DTR}}$/SLP_RQ configuration (Micro pin 9/SMT pin 10/TH pin 9).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	$\overline{\text{DTR}}$/Sleep_Request (used with pin sleep and cyclic sleep with pin wake)
2	N/A
3	Digital input

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

Default

1

D9 command

Sets or displays the DIO9/ON_$\overline{\text{SLEEP}}$ configuration (Micro pin 25/SMT pin 26/TH pin 13).

Parameter range

0, 1, 3 - 5

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

Default

1

P0 command

Sets or displays the DIO10/RSSI/PWM0 configuration (Micro pin 7/SMT pin 7/TH pin 6).

When configured as RSSI PWM output, the device outputs a PWM signal with a duty cycle equivalent to the dBm of the received packet.

Use RP command to configure the timeout.

When configured as PWM output (**2**): you can use **M0** to explicitly control the PWM0 output. When used with Analog line passing, PWM0 corresponds with ADC0.

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	RSSI PWM output

Parameter	Description
2	PWM0 output. M0 command or I/O line passing control the value.
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

P1 command

Sets or displays the DIO11/PWM1 configuration (Micro pin 8/SMT pin 8/TH pin 7).

When configured as PWM output (**2**): you can use **M1** to explicitly control the PWM1 output. When used with Analog line passing, PWM corresponds with ADC1.

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	N/A
2	PWM1 output. M1 command or I/O line passing control the value.
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P2 command

Sets or displays the DIO12/TH_SPI_MISO configuration (Micro pin 5/SMT pin 5/TH pin 4).

Parameter range

SMT/MMT: 0, 3 - 5

TH: 0, 1, 3 - 5

Parameter	Description
0	Disabled

Parameter	Description
1	SPI_MISO for the through-hole device N/A for the surface-mount and micro device
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

PR command

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

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

The **PD** (Pull Direction) parameter determines the direction of the internal pull-up/down resistor.

PR and **PD** only affect lines that are configured as digital inputs (**3**) or disabled (**0**).

By default, pull-up resistors are enabled on all disabled I/O lines.

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

Bit	I/O line
0	DIO4
1	DIO3
2	DIO2
3	DIO1
4	DIO0
5	DIO6/$\overline{\text{RTS}}$
6	DIO8/DTR/Sleep_Rq
7	UART_DIN
8	DIO5/Associate
9	DIO9/Awake_$\overline{\text{SLEEP}}$
10	DIO12
11	DIO10/PWM-RSSI
12	DIO11

Bit	I/O line
13	DIO7/$\overline{\text{CTS}}$
14	UART_DOUT
15	DIO15
16	N/A
17	N/A
18	N/A
19	N/A

Parameter range

0 - 0xFFFFF (bit field)

Default

0xFFFF

Example

Sending the command **ATPR 6F** turn bits 0, 1, 2, 3, 5 and 6 ON, and bits 4 and 7 OFF. The binary equivalent of 0x6F is 01101111. Bit 0 is the right-most digit in the binary bit field.

PD command

See PR command for the bit mappings.

Parameter range

Through-hole: 0 - 0xFFFF

SMT/MMT: 0 - 0xFFFFF

Default

0xFFFF

M0 command

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

If IA (I/O Input Address) is set correctly and P0 command is configured as PWM0 output, incoming AD0 samples automatically modify the PWM0 value. See PT (PWM Output Timeout).

To configure the duty cycle of PWM0:

1. Enable PWM0 output (**P0** = **2**).
2. Change **M0** to the desired value.
3. Apply settings (use **CN** or **AC**).

The PWM period is 64 µs and there are 0x03FF (1023 decimal) steps within this period. When **M0** = **0** (0% PWM), 0x01FF (50% PWM), 0x03FF (100% PWM), and so forth.

Parameter range

0 - 0x3FF

Default

0

M1 command

If IA (I/O Input Address) is set correctly and P1 command is configured as PWM1 output, incoming AD0 samples automatically modify the PWM1 value. See PT (PWM Output Timeout).

To configure the duty cycle of PWM1:

1. Enable PWM1 output (**P1** = 2).
2. Change **M1** to the desired value.
3. Apply settings (use **CN** or **AC**).

The PWM period is 64 µs and there are 0x03FF (1023 decimal) steps within this period. When **M1** = **0** (0% PWM), 0x01FF (50% PWM), 0x03FF (100% PWM), and so forth.

Parameter range

0 - 0x3FF

Default

0

RP command

The PWM timer expiration in 0.1 seconds. **RP** sets the duration of pulse width modulation (PWM) signal output on the RSSI pin. The pin signal duty cycle updates with each received packet and shuts off when the timer expires. This command is only applicable when **P0** is set to **1** which enables RSSI PWM output.

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

Parameter range

0 - 0xFF (x 100 ms), 0xFF

Default

0x28 (four seconds)

LT command

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

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

Parameter range

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

Default

0

I/O sampling commands

The following commands configure I/O sampling on an originating device. Any I/O sample generated by this device is sent to the address specified by **DH** and **DL**. You must configure at least one I/O line as an input or output for a sample to be generated.

IS command

Immediately forces an I/O sample to be generated for the digital and analog I/O lines that are configured for the local device. If you issue the command to the local device, the sample data is sent out the local serial interface. If sent remotely, the sample is taken on the destination and the sample data is returned as an AT Command Response frame - 0x88.

If the device receives ERROR as a response to an **IS** query, there are no valid I/O lines to sample.

Refer to On-demand sampling for more information on using this command and examples.

Standard I/O capability

If AO command is set to **2**, the XBee3 802.15.4 RF Module's **IS** I/O options are D0 command - D8 command and four analog channels: AD0/DIO0 - AD3/DIO3.

When operating in Transparent mode (AP command = **0** and AO command = **2**), the data is returned in the following format:

All bytes are converted to ASCII:

number of samples<CR>

AIO/DIO mask (Bits 0 - 8 are digital I/O; Bits 9 - 12 analog channels)<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.

Extended I/O capability

If **A0** is set to **0** or **1**, the XBee3 802.15.4 RF Module's **IS** I/O options are D0 command - D9 command and P0 command - P4 command and four analog channels AD0/DIO0 - AD3/DIO3.

When operating in Transparent mode (**AP** = **0** and **AO** = **0**, AO = 1), the data is returned in the following format:

All bytes are converted to ASCII:

number of samples<CR>

DIO mask (Bits 0 - 14 are digital I/O<CR>

AIO mask (Bits 0 - 3 are Analog channels<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

IR command

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 command-D8 command, P0 command-P2 command.

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

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

IC command

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** - **P5**). If the device detects a change on an enabled digital I/O pin, it immediately transmits a digital I/O sample to the address specified by **DH** + **DL**. 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 DIO15 can be sampled using a Change Detect.

Bit field

Bit	I/O line	Device pin
0	DIO0	Micro pin 31/SMT pin 33/TH pin 20
1	DIO1	Micro pin 30/SMT pin 32/TH pin 19
2	DIO2	Micro pin 29/SMT pin 31/TH pin 18
3	DIO3	Micro pin 28/SMT pin 30/TH pin 17
4	DIO4	Micro pin 23/SMT pin 24/TH pin 11
5	DIO5	Micro pin 26/SMT pin 28/TH pin 15
6	DIO6	Micro pin 27/SMT pin 29/TH pin 16
7	DIO7	Micro pin 24/SMT pin 25/TH pin 12
8	DIO8	Micro pin 9/SMT pin 10/TH pin 9

Bit	I/O line	Device pin
9	DIO9	Micro pin 25/SMT pin 26/TH pin 13
10	DIO10	Micro pin 7/SMT pin 7/TH pin 6
11	DIO11	Micro pin 8/SMT pin 8/TH pin 7
12	DIO12	Micro pin 5/SMT pin 5/TH pin 4

Parameter range

0 - 0x7FFF

Default

0

AV (Analog Voltage Reference)

The analog voltage reference used for A/D sampling.

Parameter range

0 - 2

Parameter	Description
0	1.25 V reference
1	2.5 V reference
2	VDD reference

Default

0

IT (Samples before TX)

Sets or displays the number of samples to collect before transmitting data. The maximum number of samples is dependent on the number of enabled I/O lines and the maximum payload available.

If **IT** is set to a number too big to fit in the maximum payload, it is reduced such that it will fit. A query of **IT** after setting it reports the actual number of samples in a packet.

Parameter range

0x1 - 0xFF

Default

1

IF command

The number of sleep cycles that elapse between periodic I/O samples.

Parameter range

1 – 0xFF

Default

1

IO command

Sets digital output levels. This allows DIO lines setup as outputs to be changed through Command mode.

Parameter range

8-bit bit map; each bit represents the level of an I/O line set up as an output

Default

N/A

I/O line passing commands

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

IA (I/O Input Address)

The source address of the device to which outputs are bound.

To disable I/O line passing, set all bytes to **0xFF**.

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

Parameter range

0 - 0xFFFF FFFF FFFF FFFF

Default

0xFFFFFFFFFFFFFFFF (I/O line passing disabled)

IU command

IU disables or enables I/O API UART output when line passing is enabled if the received sample has a source address that matches IA (I/O Input Address) or if **IA** is set to **0xFFFF**.

Note To enable API output, you must set AP command to an API mode (**AP** = **1** or **2**).

Parameter range

0 - 1

Parameter	Description
0	Disabled
1	Enabled

Default

1

T0 command

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

Parameter range

0 - 0xFF

Default

0

T1 command

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

Parameter range

0 - 0xFF

Default

0

T2 command

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

Parameter range

0 - 0xFF

Default

0

T3 command

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

Parameter range

0 - 0xFF

Default

0

T4 command

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

Parameter range

0 - 0xFF

Default

0

T5 command

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

Parameter range

0 - 0xFF

Default

0

T6 command

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

Parameter range

0 - 0xFF

Default

0

T7 command

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

Parameter range

0 - 0xFF

Default

0

T8 command

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

Parameter range

0 - 0xFF

Default

0

T9 command

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

Parameter range

0 - 0xFF

Default

0

Q0 command

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

Parameter range

0 - 0xFF

Default

0

Q1 command

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

Parameter range

0 - 0xFF

Default

0

Q2 command

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

Parameter range

0 - 0xFF

Default

0

PT (PWM Output Timeout)

Specifies how long both PWM outputs (**P0**, **P1**) output a given PWM signal before it reverts to the configured value (**M0**/**M1**). If set to **0**, there is no timeout. This timeout only affects these pins when they are configured as PWM output.

Parameter range

0 - 0xFF (x 100 ms)

Default

0xFF

Diagnostic commands - firmware/hardware information

The following AT commands provide information about the device's hardware and firmware.

VR command

Reads the firmware version on a device.

Parameter range

0x2000 - 0x2FFF

Default

Set in the firmware

VL command

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

Parameter range

N/A

Default

N/A

VH command

Reads the bootloader version of the device.

Parameter range

N/A

Default

N/A

HV command

Display the hardware version number of the device.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in firmware

%C (Hardware/Software Compatibility)

Specifies what firmware is compatible with this device's hardware. **%C** is compared to the to the "compatibility_number" field of the firmware configuration xml file. Firmware with a compatibility number lower than the value returned by **%C** cannot be loaded onto the board. If an invalid firmware is loaded, the device will not boot until a valid firmware is reloaded.

Parameter range

[read-only]

Default

N/A

%V command

Reads the voltage on the Vcc pin in mV.

Parameter range

0 - 0xFFFF (in mV) [read only]

Default

N/A

TP command

The current module temperature in degrees Celsius. The temperature is represented in two's complement, as shown in the following example:

1 °C = 0x0001 and -1°C = 0xFFFF

Parameter range

0 - 0xFFFF (Celsius) [read-only]

Default

N/A

DD command

Stores the Digi device type identifier value. Use this value to differentiate between multiple types of devices.

Parameter range

0 - 0xFFFFFFFF

Default

0x130000

CK command

Reads the cyclic redundancy check (CRC) of the current AT command configuration settings to determine if the configuration has changed.

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

Parameter range

0 - 0xFFFF [read-only]

Default

N/A

FR command

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

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

Parameter range

N/A

Default

N/A

Memory access commands

This section details the executable commands that provide memory access to the device.

AC (Apply Changes)

This command applies changes to all command parameters configured in Command mode.

Any of the following also applies changes the same as issuing an **AC** command:

- Exiting Command mode with a **CN** command.
- Exiting Command mode via timeout.
- Receiving a 0x08 API command frame.
- Issuing a 0x08 Local AT Command API frame.
- Issuing a remote 0x17 AT Command API frame with option bit 1 set.

Example: Altering the UART baud rate with the **BD** command does not change the operating baud rate until after an **AC** command is received; at this point, the interface immediately changes baud rates.

Parameter range

N/A

Default

N/A

WR command

Immediately writes parameter values to non-volatile flash memory so they persist through a power cycle. Operating network parameters are persistent and do not require a **WR** command for the device to reattach to the network.

Note Once you issue a **WR** command, do not send any additional characters to the device until after you receive the **OK** response. Use the **WR** command sparingly; the device's flash supports a limited number of write cycles.

Parameter range

N/A

Default

N/A

RE command

Restore device parameters to factory defaults.

Does not exit out of Command mode.

Parameter range

N/A

Default

N/A

Custom default commands

The following commands are used to assign custom defaults to the device. Send RE command to restore custom defaults. You must send these commands as local AT commands, they cannot be set using Remote AT Command Request frame - 0x17.

%F (Set Custom Default)

When **%F** is received, the XBee3 802.15.4 RF Module takes the next command received and applies it to both the current configuration and the custom defaults, so that when defaults are restored with RE command the custom value is used.

Parameter range

N/A

Default

N/A

!C (Clear Custom Defaults)

Clears all custom defaults. This command does not change the current settings, but only changes the defaults so that RE command restores settings to the factory values.

Parameter range

N/A

Default

N/A

R1 (Restore Factory Defaults)

Restores factory defaults, ignoring any custom defaults set using %F (Set Custom Default).

Parameter range

N/A

Default

N/A

Operate in API mode

API mode overview

As an alternative to Transparent operating mode, you can use API operating mode. API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol. The API specifies how commands, command responses and device status messages are sent and received from the device using the serial interface or the SPI interface.

We may add new frame types to future versions of the firmware, so we recommend building the ability to filter out additional API frames with unknown frame types into your software interface.

Use the AP command to set the operation mode

Use AP command to specify the operation mode:

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

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

API frame format

An API frame consists of the following:

- Start delimeter
- Length
- Frame data
- Checksum

API operation (AP parameter = 1)

This is the recommended API mode for most applications. The following table shows the data frame structure when you enable this mode:

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

Any data received prior to the start delimiter is silently discarded. If the frame is not received correctly or if the checksum fails, the XBee replies with a radio status frame indicating the nature of the failure.

API operation with escaped characters (AP parameter = 2)

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

When operating in API 2, if an unescaped 0x7E byte is observed, it is treated as the start of a new API frame and all data received prior to this delimiter is silently discarded. For more information on using this API mode, see the Escaped Characters and API Mode 2 in the Digi Knowledge base.

API escaped operating mode works similarly to API mode. The only difference is that when working in API escaped mode, the software must escape any payload bytes that match API frame specific data, such as the start-of-frame byte (0x7E). The following table shows the structure of an API frame with escaped characters:

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

Start delimiter field

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

Escaped characters in API frames

If operating in API mode with escaped characters (**AP** parameter = 2), when sending or receiving a serial data frame, specific data values must be escaped (flagged) so they do not interfere with the data frame sequencing. To escape an interfering data byte, insert 0x7D and follow it with the byte to be escaped (XORed with 0x20).

The following data bytes need to be escaped:

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

To escape a character:

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

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

Example: escape an API frame

To express the following API non-escaped frame in API operating mode with escaped characters:

Start delimiter	Length	Frame type	Frame Data	Checksum
			Data	
7E	00 0F	17	01 00 13 A2 00 40 AD 14 2E FF FE 02 4E 49	6D

You must escape the 0x13 byte:

1. Insert a 0x7D.
2. XOR byte 0x13 with 0x20: 13 ⊕ 20 = 33

The following figure shows the resulting frame. Note that the length and checksum are the same as the non-escaped frame.

Start delimiter	Length	Frame type	Frame Data	Checksum
			Data	
7E	00 0F	17	01 00 7D 33 A2 00 40 AD 14 2E FF FE 02 4E 49	6D

The length field has a two-byte value that specifies the number of bytes in the frame data field. It does not include the checksum field.

Length field

The length field is a two-byte value that specifies the number of bytes contained in the frame data field. It does not include the checksum field.

Frame data

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

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

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

Multi-byte values are sent big-endian.

Calculate and verify checksums

To calculate the checksum of an API frame:

1. Add all bytes of the packet, except the start delimiter 0x7E and the length (the second and third bytes).
2. Keep only the lowest 8 bits from the result.
3. Subtract this quantity from 0xFF.

To verify the checksum of an API frame:

1. Add all bytes including the checksum; do not include the delimiter and length.
2. If the checksum is correct, the last two digits on the far right of the sum equal 0xFF.

Example

Consider the following sample data packet: **7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8**

Byte(s)	Description
7E	Start delimiter
00 0A	Length bytes
01	API identifier
01	API frame ID
50 01	Destination address low
00	Option byte
48 65 6C 6C 6F	Data packet
B8	Checksum

To calculate the check sum you add all bytes of the packet, excluding the frame delimiter **7E** and the length (the second and third bytes):

7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8

Add these hex bytes:

01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F = 247

Now take the result of 0x247 and keep only the lowest 8 bits which, in this example, is 0x47 (the two far right digits). Subtract 0x47 from 0xFF and you get 0xB8 (0xFF - 0x47 = 0xB8). 0xB8 is the checksum for this data packet.

If an API data packet is composed with an incorrect checksum, the XBee3 802.15.4 RF Module will consider the packet invalid and will ignore the data.

To verify the check sum of an API packet add all bytes including the checksum (do not include the delimiter and length) and if correct, the last two far right digits of the sum will equal FF.

01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F + B8 = 2FF

Frame descriptions

The following sections describe the API frames.

TX Request: 64-bit address frame - 0x00

Description

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

Format

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

Frame data fields	Offset	Description
Frame type	3	0x00
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK, which is a TX Status frame - 0x89 that indicates the packet was transmitted successfully. If set to **0**, the device does not send a response.
64-bit destination address	5-12	Set to the 64-bit address of the destination device. If set to 0x000000000000FFFF, the broadcast address is used.
Options	13	0x01 = Disable ACK 0x04 = Send packet with Broadcast PAN ID. Set all other bits to 0.
RF data	14-n	The RF data length can be up to 110 bytes, but may be less depending on other factors discussed in Maximum payload.

TX Request: 16-bit address - 0x01

Description

A TX Request message causes the device to transmit data as an RF Packet.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x01
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent TX Status frame - 0x89. If set to **0**, the device does not send a response.
16-bit destination address	5-6	Set to the 16-bit address of the destination device. Broadcast = 0xFFFF.
Options	7	0x01 = Disable ACK. 0x04 = Send packet with Broadcast PAN ID. Set all other bits to 0.
RF data	8-n	The RF data length can be up to 116 bytes, but may be less depending on other factors discussed in Maximum payload.

AT Command Frame - 0x08

Description

Use this frame to query or set command parameters on the local device. This API command applies changes after running the command. You can query parameter values by sending the AT Command Frame - 0x08 with no parameter value field (the two-byte AT command is immediately followed by the frame checksum). Any parameter that is set with this frame type will apply the change immediately. If you wish to queue multiple parameter changes and apply them later, use the AT Command - Queue Parameter Value frame - 0x09 instead.

When an AT command is queried, a AT Command Response frame - 0x88 response frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x88 response is the same one set by the command in the 0x08 frame.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x08
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent response (0x88). If set to 0, the device does not send a response.
AT command	5-6	Command name: two ASCII characters that identify the AT command.
Parameter value	7-n	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the register.

Example

The following example illustrates an AT Command frame where the device's **SL** parameter value is queried.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x04
Frame type	3	0x08

Frame data fields	Offset	Example
Frame ID	4	0x13
AT command	5	0x53 (S)
	6	0x4C (L)
Parameter value (optional)		
Checksum	8	0x45

The following example illustrates an AT Command frame when you modify the device's **DL** parameter value to a broadcast address of 0xFFFF. A non-zero Frame ID can be used to correlate the AT command request with the corresponding response frame.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x08
Frame type	3	0x08
Frame ID	4	0x4D
AT command	5	0x44 (D)
	6	0x4C (L)
Parameter value	7-10	0xFF 0xFF
Checksum	11	0x1C

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 sets 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 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 format.

Frame data fields	Offset	Description
Frame type	3	0x09
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent response (0x88). 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 (**BD7** = 115200 baud) (optional)	7-n	If present, indicates the requested parameter value to set the given register. If no characters are present, queries the register.

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 or 16-bit address field to the address of the desired destination node.
- If transmitting to a 64-bit destination, set the 16-bit address field to **0xFFFE**, otherwise set the 64-bit destination address field to **0xFFFFFFFFFFFFFFFF**.
- 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.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x10
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
64-bit destination address	5-12	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = **0x000000000000FFFF**. If transmitting to a 16-bit address, set this field to **0xFFFFFFFFFFFFFFFF**.
16-bit destination address	13-14	Set to the 16-bit address of the destination device, or set to **0xFFFE** if sending to the 64-bit address of the end device.
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.
Reserved	16	Set 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

Frame data fields	Offset	Example
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
Reserved	15-16	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.
- If sending to a 16-bit address, set the 64-bit address to **0xFFFFFFFFFFFFFFFF**, otherwise set the 16-bit address to **0xFFFE**.

Query the **NP** command to read the maximum number of payload bytes. For more information, see 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 format.

Frame data fields	Offset	Description
Frame type	3	0x11
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to **0**, the device does not send a response.
64-bit destination Address	5-12	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = **0x000000000000FFFF**. Set to **0xFFFFFFFFFFFFFFFF** if transmitting to a 16-bit destination.
Reserved	13-14	Set to the 16-bit address of the destination device.
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.
Reserved	21-22	Set to **0**.
Data Payload	23-n	Data that is sent to the destination device.

Transmit Options bit field

See Transmit Request frame - 0x10.

Example

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

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

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x1A
Frame type	3	0x11
Frame ID	4	0x01
64-bit destination address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x01
	10	0x23
	11	0x84
	LSB12	0x00
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
Reserved	21-22	0x00

Frame data fields	Offset	Example
Data payload	23	0x54
	24	0x78
	25	0x44
	26	0x61
	27	0x74
	28	0x61
Checksum	29	0xA6

Remote AT Command Request frame - 0x17

Description

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

Format

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

Frame data fields	Offset	Description
Frame type	3	0x17
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent response (0x97). If set to **0**, the device does not send a response.
64-bit destination address	5-12	Broadcast = 0x000000000000FFFF. This field is ignored if the 16-bit network address field equals anything other than 0xFFFF.
16-bit destination address	13-14	Set to match the 16-bit network address of the destination, MSB first, LSB last. Set to 0xFFFF if 64-bit addressing is being used.
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.

RX Packet: 64-bit Address frame - 0x80

Description

When a device configured with legacy API Rx Indicator (**AO** = **2**) receives an RF data packet from a device configured to use 64-bit addressing (**MY** = **0xFFFE**), it sends this frame out the serial interface.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x80
64-bit source address	4-11	The sender's 64-bit address.
RSSI	12	Received Signal Strength Indicator. The Hexadecimal equivalent of (-dBm) value. For example if RX signal strength is -40 dBm, then 0x28 (40 decimal) is returned.
Options	13	Bit field: 0 = [reserved]. 1 = Packet was a broadcast packet. 2 = Packet was broadcast across all PANs. 3-7 = [reserved].
Received data	14-n	The RF data that the device receives.

Receive Packet: 16-bit address frame - 0x81

Description

When a device configured with legacy API Rx Indicator (**AO** = **2**) receives an RF packet from a device configured to use 16 bit addressing (**MY** < **0xFFFE**), it sends this frame out the serial interface.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x81
Source address	4-5	MSB first LSB last
RSSI	6	RSSI = hexadecimal equivalent of -dBm value. For example, if RX signal strength = -40 dBm, it returns 0x28 (40 decimal).
Options	7	Bit 0 = [reserved]. Bit 1 = Packet was a broadcast packet. Bit 2 = Packet was broadcast across all PANs. Bits 3 - 7 = [reserved].
RF data	8-n	The RF data that the device receives.

RX (Receive) Packet: 64-bit address IO frame - 0x82

Description

When the device receives an I/O sample from a remote device configured to use 64-bit addressing, the I/O data is sent out the UART using this frame type

Format

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

Frame data fields	Offset	Total number of samples	Description
Frame type	3	N/A	0x82
64-bit source address	4-11	N/A	MSB first, LSB last.
RSSI	12	N/A	RSSI: Hexadecimal equivalent of (-dBm) value. For example, if RX signal strength = -40 dBm, the device returns 0x28 (40 decimal).
Status	13	N/A	bit 0 = reserved bit 1 = Address broadcast bit 2 = PAN broadcast bits 3-7 = [reserved]
Number of samples	14	N/A	Total number of samples.
Channel Indicator (see bit field table below)	15	MSB	Indicates which inputs have sampling enabled (if any). Each bit represents either a DIO line or ADC channel. Bit set to 1 if channel is active
	16	LSB	
Digital samples (if enabled) (see bit field table below)	17	MSB	If any of the DIO lines are enabled in the Channel indicator, these two bytes contain samples for all enabled DIO lines. DIO lines that do not have sampling enabled return 0. If no DIO line is enabled, no bytes are included in the frame.
	18	LSB	

Frame data fields	Offset	Total number of samples	Description
	19	ADC0 MSB	If the sample set includes any ADC data, each enabled analog input returns a two-byte value indicating the A/D measurement of that input. ADC channel data is represented as an unsigned 10-bit value right-justified on a 16-bit boundary. Analog samples are ordered sequentially from AD0 to AD5.
	20	ADC0 LSB	
	...	N/A	
	n -1	ADCn MSB	
	n	ADCn LSB	

The following table shows the Channel Indicator and Digital Samples bit fields.

Bit field	Description
Reserved	3 bits
A3 - A0	4 analog bits
D8 - D0	9 digital bits

RX Packet: 16-bit address I/O frame - 0x83

Description

When the device receives an I/O sample from a remote device configured to use 16-bit addressing, the I/O data is sent out the UART using this frame type.

Format

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

Frame data fields	Offset	Total number of samples	Description
Frame type	3	N/A	0x83
Source Address	4-5	N/A	MSB first, LSB last.
RSSI	6	N/A	RSSI: Hexadecimal equivalent of (-dBm) value. For example, if RX signal strength = -40 dBm, the device returns 0x28 (40 decimal).
Options	7	N/A	bit 0 = reserved bit 1 = Address broadcast bit 2 = PAN broadcast bits 3-7 = [reserved]
Number of samples	8	N/A	Total number of samples.
Channel Indicator (see bit field table below)	9	MSB	Indicates which inputs have sampling enabled (if any). Each bit represents either a DIO line or ADC channel. Bit set to 1 if channel is active.
	10	LSB	
Digital Samples (if enabled) (see bit field table below)	11	MSB	If any of the DIO lines are enabled in the Channel indicator, these two bytes contain samples for all enabled DIO lines. DIO lines that do not have sampling enabled return 0. If no DIO line is enabled, no bytes are included in the frame.
	12	LSB	

Frame data fields	Offset	Total number of samples	Description
Analog samples	13	ADC0 MSB	If the sample set includes any ADC data, each enabled analog input returns a two-byte value indicating the A/D measurement of that input. ADC channel data is represented as an unsigned 10-bit value right-justified on a 16-bit boundary. Analog samples are ordered sequentially from AD0 to AD5.
	14	ADC0 LSB	
	…		
	n - 1	ADCn MSB	
	n	ADCn LSB	

The following table shows the Channel Indicator bit field.

Bit field	Description
Reserved	3 bits
A3 - A0	4 analog bits
D8 - D0	9 digital bits

AT Command Response frame - 0x88

Description

A device sends this frame in response to an AT Command Frame - 0x08 and a AT Command - Queue Parameter Value frame - 0x09. Some commands send back multiple frames; for example, the **ND** command. This command ends by sending a frame with a status of **0** (OK) and no value. In the particular case of **ND**, a frame is received via a remote node in the network and when the process is finished, the AT command response is received. For details on the behavior of **ND**, see ND command.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x88
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent request (0x08 or 0x09). If set to **0** in the request frame, 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 1 = ERROR 2 = Invalid command 3 = Invalid parameter 4 = Tx failure
Command data		The register data in binary format. If the host sets the register, the device does not return this field.

Example

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

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05

Frame data fields	**Offset**	**Example**
Frame type	3	0x88
Frame ID	4	0x01
AT command	5	0x42 (B)
	6	0x44 (D)
Command status	7	0x00
Command data		(No command data implies the parameter was set rather than queried)
Checksum	8	0xF0

TX Status frame - 0x89

Description

When a TX request: TX Request: 64-bit address frame - 0x00 or TX Request: 16-bit address - 0x01 is complete, the device sends an 0x89 TX Status frame. Other transmit request frames (0x10, 0x11) will send an 0x8B status frame. This message indicates if the packet transmitted successfully or if there was a failure.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x89
Frame ID	4	Identifies the TX Request frame being reported. If the Frame ID = 0 in the TX Request, no TX Status frame is given.
Status	5	0x00 = Success 0x01 = No ACK received 0x02 = CCA failure 0x03 = Indirect message unrequested 0x21 = Network ACK failure 0x31 = Internal error 0x74 = The payload in the frame was larger than allowed

Notes:

- A status of 0x01 occurs when all MAC and Application-Layer retries have expired and no ACK is received.
- If the transmitter sends an outgoing transmission as a broadcast (destination address = 0x000000000000FFFF), status 0x01 and 0x21 will never be returned because broadcasts are sent unacknowledged.

Example

The following example shows a successful status received.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x03

Frame data fields	Offset	Example
Frame type	3	0x89
Frame ID	4	0x01
Status	5	0x00
Checksum	6	0x75

Modem Status frame - 0x8A

Description

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

Format

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

Frame data fields	Offset	Description
Frame type	3	0x8A
Status	4	0x00 Hardware reset 0x01 Watchdog timer reset 0x02 = End device successfully associated with a coordinator 0x03 = End device disassociated from coordinator or coordinator failed to form a new network 0x06 = End device successfully associated with a coordinator 0x0D Input voltage is too high, which prevents transmissions

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 an 0x8B Transmit Status message out of the serial interface. This message indicates if the Transmit Request was successful or if it failed.

Note Broadcast transmissions are not acknowledged and always return a status of 0x00, even if the delivery failed.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x8B
Frame ID	4	The Frame ID of the response will be the same value that was used in the originating Tx request.
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 0x01 = MAC ACK Failure 0x02 = CCA failure 0x03 = Indirect message unrequested 0x21 = Network ACK Failure 0x31 = Internal resource error 0x74 = Data payload too large
Reserved	9	

Example

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

Frame Fields	Offset	Example
Start delimiter	0	0x7E

Frame Fields	Offset	Example
Length	MSB 1	0x00
	LSB 2	0x07
Frame type	3	0x8B
Frame ID	4	0x47
Reserved	5	0xFF
	6	0xFE
Transmit retry count	7	0x00
Delivery status	8	0x00
Reserved	9	0x02
Checksum	10	0x2E

Receive Packet frame - 0x90

Description

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

Format

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

Frame data fields	Offset	Description	
Frame type	3	0x90	
64-bit source address	4-11	The sender's 64-bit address. MSB first, LSB last.	
Reserved	12-13	16-bit source address.	
Receive options	14	**Bit**	**Interpretation**
		0	Packet acknowledged
		1	Broadcast packet
		2 - 5	Reserved
		6 - 7	Delivery mode:
			b 00 Invalid
			b 01 Point-to-multipoint
			b 10 Repeater mode
			b 11 DigiMesh
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

Frame data fields	Offset	Example
Length	MSB 1	0x00
	LSB 2	0x12
Frame type	3	0x90
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 command = 1) receives an RF packet, it sends it out the serial interface using this message type.

The Cluster ID and endpoints must be used to identify the type of transaction that occurred.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x91
64-bit source address	4-11	MSB first, LSB last. The sender's 64-bit address.
Reserved	12-13	16-bit source address.
Source endpoint	14	Endpoint of the source that initiates transmission.
Destination endpoint	15	Endpoint of the destination where the message is addressed.
Cluster ID	16-17	The Cluster ID where the frame is addressed.
Profile ID	18-19	The Profile ID where the fame is addressed.
Receive options	20	Bit field: 0x00 = Packet acknowledged 0x01 = Packet was a broadcast packet 0x06, 0x07: b’01 = Point-Multipoint b’10 = Repeater mode (directed broadcast) b’11 = DigiMesh Ignore all other bits.
Received data	21-n	Received RF data.

Example

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

If a device sends the transmission:

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

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

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x18
Frame type	3	0x91
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	12	0xFF
	13	0xFE
Source endpoint	14	0xE0
Destination endpoint	15	0xE0
Cluster ID	16	0x22
	17	0x11
Profile ID	18	0xC1
	19	0x05
Receive options	20	0x02
Received data	21	0x52
	22	0x78
	23	0x44
	24	0x61
	25	0x74
	26	0x61
Checksum	27	0x68

I/O 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 format.

Frame data fields	Offset	Description
Frame type	3	0x92
64-bit source address	4-11	The sender's 64-bit address.
Reserved	12-13	Reserved.
Receive options	14	Bit field: 0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet Ignore all other bits
Number of samples	15	The number of sample sets included in the payload. Always set to 1.
Digital channel mask	16-17	Bitmask field that indicates which digital I/O lines on the remote have sampling enabled, if any.
Analog channel mask	18	Bitmask field that indicates which analog I/O lines on the remote have sampling enabled, if any.
Digital samples (if included)	19-20	If the sample set includes any digital I/O lines (Digital channel mask > 0), these two bytes contain samples for all enabled digital I/O lines. DIO lines that do not have sampling enabled return 0. Bits in these two bytes map the same as they do in the Digital channel mask field.
Analog sample	21-n	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from ADO/DIO0 to AD3/DIO3.

Example

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

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

The complete example frame is:

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x14
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	0xFF
	LSB 13	0xFE
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

Remote Command Response frame - 0x97

Description

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

Some commands, such as the **ND** command, may send back multiple frames. For details on the behavior of **ND**, see ND command.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x97
Frame ID	4	This is the same value that is passed into the request. The request is a 0x17 frame.
64-bit source (remote) address	5-12	The long address of the remote device returning this response.
16-bit source (remote) address	13 - 14	The short address of the remote device returning this response.
AT commands	15-16	The name of the command.
Command status	17	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter 4 = Remote Command Transmission Failed
Command data	18-n	The value of the requested register in hexadecimal notation (non-ASCII).

OTA firmware upgrade process for 802.15.4

OTA/OTB file

The .ota file extension is for a file which contains an OTA Firmware upgrade image. The .otb file extension is for a file which contains an OTA combined upgrade image containing both the bootloader as well as the firmware. However, the way we process both the files remain the same.

The OTA header

The OTA firmware uses a specific firmware file with a .ota extension. We recommend parsing the OTA header from the OTA file first to obtain the firmware version, manufacturer code, image type and the size of the GBL file. These parameters are required to generate the rest of the OTA firmware upgrade messages. The format of the OTA header is:

Bytes	Field name	Description
4	OTA upgrade file identifier	Has to match 0x0BEEF11E in little endian. If it is not, then the OTA file is not a valid upgrade file.
2	OTA Header version	0x0001
2	OTA Header length	Length of the OTA Header.
2	OTA Header Field control	Bit mask that indicates if additional information is included in the image. (Read the Security Credential Version in this table).
2	Manufacturer Code	0x101E
2	Image Type	0x0000
4	File Version	The version of the firmware upgrade image.
2	Stack Version	This is set to 2 by default.
32	OTA Header String	Usually contains the Firmware image name followed by 0xFFs. For example, **FFFFFFFFFFFFFFFlbg.10F3_42MD_3BX** which is XB3_DM24-3F01.gblFFFFFFFFFFFFFFF in little endian
4	Image Size	Contains the size of the .gbl file for the firmware.
0/1	Security Credential version	If bit 0 of the OTA Header Field Control is set to 1, this indicates the security credential version type that the client is required to have, before it will install the image (set to 2).
0/8	Upgrade File Destination	If bit 1 of the OTA Header Field Control is set to 1, this indicates that this OTA file contains security credential/certificate 577 data or other type of information that is specific to a particular device. Currently, we do not use this feature.
0/2	Hardware/Software Compatibility	If bit 2 of the OTA Header Field Control is set to 1.

The file version field contains additional hardware/software compatibility information. We recommend that if you intend to perform an OTA update, you use the OTA header extracted from the file so that you can avoid undesired behavior.

Hardware/software compatibility

The Hardware Software Compatibility number ensures that an incompatible firmware is not flashed on to the XBee3 802.15.4 RF Module. To obtain this value, query %C (Hardware/Software Compatibility) on the target device. You can successfully update the device to a firmware if, and only if, the value of **%C** of the image is greater than or equal to the value returned by the device when you query **%C**.

Parse the image blocks

To parse the image blocks:

1. Divide the contents of the underlying .gbl file into 48 byte blocks for encrypted networks and 56 byte blocks for unencrypted networks
2. Create Image Block Requests around the image blocks; see Image Block request.

Note The .gbl file is placed at an offset of 75 bytes and so it is important to start parsing the image from that point in the file.

The Image Block size for 802.15.4 is 64 bytes for both encrypted and non-encrypted networks.

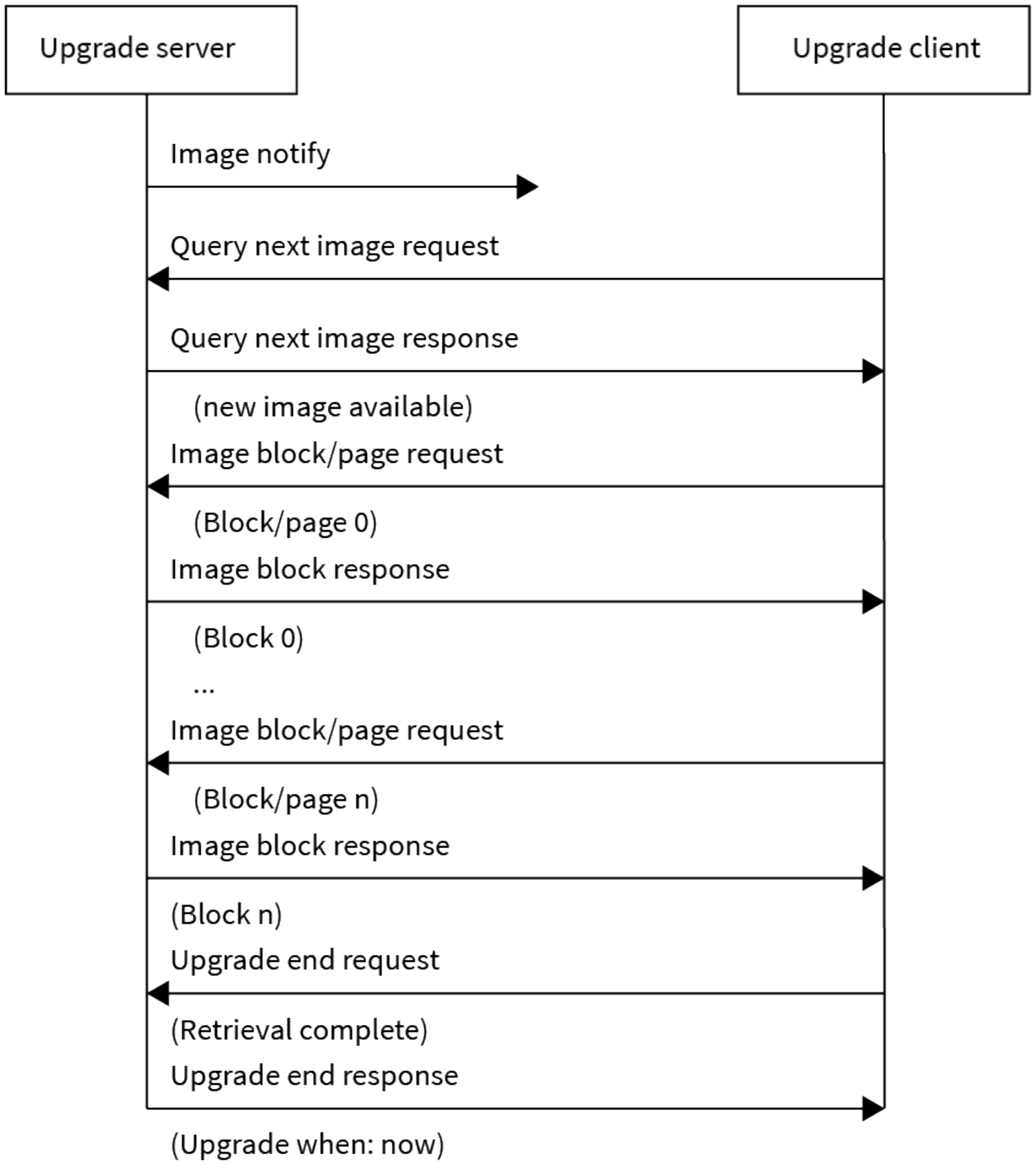
Storage

The OTA firmware image blocks are received and stored in a separate internal flash slot that is allotted exclusively for this purpose. Once all the image bytes are written to the slot, the new image must be validated by the current application before it can be used.

If the new image is deemed invalid, the running 802.15.4 firmware rejects the image and continues operating with the current, valid application.

ZCL OTA messaging

The following figure provides the messaging sequence between the Server (updater node) and the Client (target node).



ZCL message output

By default ZCL messages are not printed to the UART on the client. To see these messages, set AZ (Extended API Options) to **2**. ZCL messages received by the server are always printed to the UART.

Image Notify

The server sends the Image Notify message to the client informing the device of the presence of an update image. The Image Notify message is sent when the upgrade process is initiated from the server.

Create the Image Notify request

The Image Notify Request is an explicit transmit frame (0x11 type) passed into the server with the following information:

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x21	
Frame Type	3	0x11	
Frame ID	4	0x01	
64-bit destination address	MSB 5	0x00	
	6	0x13	
	7	0xA2	
	8	0xFE	
	9	0x00	
	10	0x00	
	11	0x00	
	LSB 12	0x03	
16-bit destination address	MSB 13	0x28	
	LSB 14	0x2F	
Source Endpoint	15	0xE8	
Destination Endpoint	16	0xE8	
Cluster ID	MSB 17	0x00	
	LSB 18	0x19	
Profile ID	MSB 19	0xC1	
	LSB 20	0x05	
Broadcast radius	21	0x00	
Transmit options	22	0x00	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	23	0x09	
		Transaction sequence number	24	0x01	
	ZCL payload	Command ID	25	0x00	Image Notify Command ID
		Payload type	26	0x03	Contains Jitter, Image Type, Firmware Version
		Query jitter	27	0x00	
		Manufacturer ID	LSB 28	0x1E	Digi's Manufacturer ID in Little Endian
			MSB 29	0x10	
		Image type	LSB 30	0x00	Image type should be 0x0000 unless the server doesn't want to update the client
			MSB 31	0x00	
		Firmware version	LSB 32	0x01	Firmware version of the new update file in Little Endian. In this example, the version is 0x1001
			33	0x10	
			34	0x00	
			MSB 35	0x00	
Checksum			36	0xE5	

Query Next Image request

The client device sends the Query Next Image request message to the server to indicate it is ready to receive a firmware image and is sent as a response to an Image Notify message. The client sends information about the existing firmware version as a part of this message. The server emits the following frame after receiving the request from the client:

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x1E	
Frame Type	3	0x91	

Frame data fields			Offset	Example	Comments
64-bit source address			MSB 4	0x00	
			5	0x13	
			6	0xA2	
			7	0xFE	
			8	0x00	
			9	0x00	
			10	0x00	
			LSB 11	0x03	
16-bit source address			MSB 12	0x28	
			LSB 13	0x2F	
Source Endpoint			14	0xE8	
Destination Endpoint			15	0xE8	
Cluster ID			MSB 16	0x00	
			LSB 17	0x19	
Profile ID			MSB 18	0xC1	
			LSB 19	0x05	
Receive options			20	0x01	
Data payload	ZCL frame header	Frame control	21	0x01	
		Transaction sequence number	22	0x00	

Frame data fields			Offset	Example	Comments
	ZCL payload	Command ID	23	0x01	Query Next Image request
		Field control	24	0x00	
		Manufacturer ID	LSB 25	0x1E	
			MSB 26	0x10	
		Image type	LSB 27	0x00	
			MSB 28	0x00	
		Firmware version	LSB 29	0x00	
			30	0x10	
			31	0x00	
			MSB 32	0x00	
Checksum			33	0x71	

Query Next Image response

The server obtains the information sent by the Client in the Query Next Image request and determines if it has a suitable image for the client. It then sends a Query Next Image response with one of the following status messages as appropriate:

- 0x00 - SUCCESS: The server is authorized to upgrade the client with the image.
- 0x98 - NO_IMAGE_AVAILABLE: The server is authorized to update the client but does not have a new OTA update image available.
- 0x7E - NOT_AUTHORIZED: The server is not authorized to update the client.

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x24	
Frame Type	3	0x11	
Frame ID	4	0x01	

Frame data fields	Offset	Example	Comments
64-bit destination address	MSB 5	0x00	
	6	0x13	
	7	0xA2	
	8	0xFE	
	9	0x00	
	10	0x00	
	11	0x00	
	LSB 12	0x03	
16-bit destination address	MSB 13	0x28	
	LSB 14	0x2F	
Source Endpoint	15	0xE8	
Destination Endpoint	16	0xE8	
Cluster ID	MSB 17	0x00	
	LSB 18	0x19	
Profile ID	MSB 19	0xC1	
	LSB 20	0x05	
Broadcast radius	21	0x00	
Transmit options	22	0x00	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	23	0x09	
		Transaction sequence number	24	0x01	
	ZCL payload	Command ID	25	0x02	Query Next Image Response
		Status	26	0x00	Success = 0x00 No Image Available = 0x98 Not Authorized = 0x7E
		Manufacturer ID	LSB 27	0x1E	
			MSB 28	0x10	
		Image type	LSB 29	0x00	
			MSB 30	0x00	
		Firmware version	LSB 31	0x01	Firmware version of the new update file in Little Endian. In this example, the version is 0x1001
			32	0x10	
			33	0x00	
			MSB 34	0x00	
		Image Size	LSB 35	0x2E	
			36	0xF3	
			37	0x02	
			MSB 38	0x00	
Checksum			39	0xE5	

Image Block request

The Client generates Image Block requests to request the server for bytes of the OTA firmware image. Each image block is 64 byte long. The client also sends the file offset as a way to keep the synchronization of every block intact.

The Image Block requests are repeated by the client until all the blocks of the image are successfully obtained. The size of the OTA upgrade image is usually obtained by the client in the Query Next Image response message and hence it knows the exact number of Image Block requests it needs to send.

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x1E	
Frame Type	3	0x91	
64-bit source address	MSB 4	0x00	
	5	0x13	
	6	0xA2	
	7	0xFE	
	8	0x00	
	9	0x00	
	10	0x00	
	LSB 11	0x03	
16-bit source address	MSB 12	0x28	
	LSB 13	0x2F	
Source Endpoint	14	0xE8	
Destination Endpoint	15	0xE8	
Cluster ID	MSB 16	0x00	
	LSB 17	0x19	
Profile ID	MSB 18	0xC1	
	LSB 19	0x05	
Receive options	20	0x01	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	21	0x01	
		Transaction sequence number	22	0x01	
	ZCL payload	Command ID	23	0x03	Image Block Request
		Field control	24	0x00	
		Manufacturer ID	LSB 25	0x1E	
			MSB 26	0x10	
		Image type	LSB 27	0x00	
			MSB 28	0x00	
		Firmware version	LSB 29	0x01	
			30	0x10	
			31	0x00	
			MSB 32	0x00	
		File Offset	LSB 33	0x00	0x0 for the first request. Offset by multiples of Image Block size. For example, 0x00000000 for the first request, 0x00000040, 0x00000080 and so on.
			34	0x00	
			35	0x00	
			LSB 36	0x00	
		Image Block Size	37	0x40	
Checksum			38	0x2D	

Image Block response

The server generates an Image Block response upon receiving an Image Block request command. It responds with a SUCCESS status on being able to retrieve the data for the client. The server uses the file offset sent by the client to determine the location of the requested data within the OTA upgrade image.

If you wish to cancel the update process, send an ABORT (0x95) status.

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x65	
Frame Type	3	0x11	
Frame ID	4	0x01	
64-bit destination address	MSB 5	0x00	
	6	0x13	
	7	0xA2	
	8	0xFE	
	9	0x00	
	10	0x00	
	11	0x00	
	LSB 12	0x03	
16-bit destination address	MSB 13	0x28	
	LSB 14	0x2F	
Source Endpoint	15	0xE8	
Destination Endpoint	16	0xE8	
Cluster ID	MSB 17	0x00	
	LSB 18	0x19	
Profile ID	MSB 19	0xC1	
	LSB 20	0x05	
Broadcast radius	21	0x00	
Transmit options	22	0x00	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	23	0x09	
Data payload		Transaction sequence number	24	0x02	
Data payload	ZCL payload	Command ID	25	0x05	Image Block Response
Data payload	ZCL payload	Status	26	0x00	Success = 0x00 Abort = 0x95
Data payload	ZCL payload	Manufacturer ID	LSB 27	0x1E	
Data payload	ZCL payload		MSB 28	0x10	
Data payload	ZCL payload	Image type	LSB 29	0x00	
Data payload	ZCL payload		MSB 30	0x00	
Data payload	ZCL payload	Firmware version	LSB 31	0x01	
Data payload	ZCL payload		32	0x10	
Data payload	ZCL payload		33	0x00	
Data payload	ZCL payload		MSB 34	0x00	
Data payload	ZCL payload	File Offset	LSB 35	0x00	
Data payload	ZCL payload		36	0x00	
Data payload	ZCL payload		37	0x00	
Data payload	ZCL payload		MSB 38	0x00	
Data payload	ZCL payload	Image Block Size	39	0x40	64 byte blocks
Data payload	ZCL payload	Image Block Data	40-104	0xEB-0x00	An image block of the size mentioned in Image Block Size
Checksum			106	0x4E	

Upgrade End request

The Upgrade End request is generated by the client after it verifies the received firmware image to ensure its integrity and validity. If the image fails any integrity checks, the client sends an Upgrade End request command to the upgrade server with INVALID_IMAGE as the status. If the image passes all integrity checks, the client sends an Upgrade End request command to the upgrade server with SUCCESS as the status.

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x1E	
Frame Type	3	0x91	
64-bit source address	MSB 4	0x00	
	5	0x13	
	6	0xA2	
	7	0xFE	
	8	0x00	
	9	0x00	
	10	0x00	
	LSB 11	0x03	
16-bit source address	MSB 12	0x28	
	LSB 13	0x2F	
Source Endpoint	14	0xE8	
Destination Endpoint	15	0xE8	
Cluster ID	MSB 16	0x00	
	LSB 17	0x19	
Profile ID	MSB 18	0xC1	
	LSB 19	0x05	
Receive options	20	0x01	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	21	0x01	
		Transaction sequence number	22	0x30	
	ZCL payload	Command ID	23	0x06	Upgrade End Request
		Status	24	0x00	Success = 0x00 Invalid Image = 0x96 Abort = 0x95 Require More Image = 0x99
		Manufacturer ID	LSB 25	0x1E	
			MSB 26	0x10	
		Image type	LSB 27	0x00	
			MSB 28	0x00	
		Firmware version	LSB 29	0x01	
			30	0x10	
			31	0x00	
			MSB 32	0x00	
Checksum			38	0x3B	

Upgrade End response

If the server receives an Upgrade End request with a SUCCESS status, it generates an Upgrade End response along with the time at which the device should upgrade to the new image.

Frame data fields	Offset	Example	Comments
Start delimiter	0	0x7E	
Length	MSB 1	0x00	
	LSB 2	0x24	
Frame Type	3	0x11	
Frame ID	4	0x01	

Frame data fields	Offset	Example	Comments
64-bit destination address	MSB 5	0x00	
	6	0x13	
	7	0xA2	
	8	0xFE	
	9	0x00	
	10	0x00	
	11	0x00	
	LSB 12	0x03	
16-bit destination address	MSB 13	0x28	
	LSB 14	0x2F	
Source Endpoint	15	0xE8	
Destination Endpoint	16	0xE8	
Cluster ID	MSB 17	0x00	
	LSB 18	0x19	
Profile ID	MSB 19	0xC1	
	LSB 20	0x05	
Broadcast radius	21	0x00	
Transmit options	22	0x00	

Frame data fields			Offset	Example	Comments
Data payload	ZCL frame header	Frame control	23	0x09	
		Transaction sequence number	24	0x01	
	ZCL payload	Command ID	25	0x07	Upgrade End response
		Manufacturer ID	LSB 26	0x1E	
			MSB 27	0x10	
		Image type	LSB 28	0x00	
			MSB 29	0x00	
		Firmware version	LSB 30	0x01	
			31	0x10	
			32	0x00	
			MSB 33	0x00	
		Current Time	LSB 34	0xF0	32 bit unsigned integer Seconds since Epoch
			35	0x1A	
			36	0x53	
			MSB 37	0x21	
		Upgrade Time	LSB38	0x00	
			39	0x1B	
			40	0x53	
			MSB 41	0x21	
Checksum			38	0xE5	

OTA error handling

ZCL OTA status code	Value	Description
SUCCESS	0x00	Successful operation
ABORT	0x95	Failed when client or server decides to abort the upgrade process
NOT_AUTHORIZED	0x7E	Server is not authorized to upgrade the client
INVALID_IMAGE	0x96	Invalid OTA upgrade image. For example, the image failed signature validation or CRC.
WAIT_FOR_DATA	0x97	Server does not have data block available yet
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for a particular client
MALFORMED_ COMMAND	0x80	The command received is badly formatted or has incorrect parameters
UNSUP_CLUSTER_ COMMAND	0x81	Such command is not supported on the device
REQUIRE_MORE_ IMAGE	0x99	The client still requires more OTA upgrade image files in order to successfully upgrade

Default response commands

The OTA framework has a command ID **0xB** reserved for error messages that are sent by the target device. Default response commands are transmitted by the target device by wrapping the ZCL payload in a Explicit Addressing Command frame - 0x11. The table below shows the ZCL Payload contents.

Note This is an example for a default response that has been received by an OTA source device. You can see that it is an Explicit Rx Indicator frame - 0x91.

Start Delimiter	8	7E
Length	16	00 17
Frame Type	8	91
Source Address	64	FF FF FF FF FF FF FF FF
Source Address	16	FF FF
Source Endpoint	8	E8
Destination Endpoint	8	E8
Cluster ID	16	00 19
Profile ID	16	C1 05

Receive Options	8	C1
RF Data (ZCL payload. Hex In Little Endian)	Frame Control	00
	Sequence Number	00
	Command ID	0B
	Erring Command	02
	Status	8A
Checksum	F2	

The example above reports an error on the **Query Next Image Response(Erring Command: 0x02)** command informing the server that there is an attempt to update to the same firmware version as the one that is running on the target radio (Status : **0x8A**).

The following table explains the different error statuses which occur at different stages in the OTA upgrade process.

Command ID	ZCL OTA command	Status	XCTU message
0x0B Default Response	0x02 Query Next Image Response	0x80	Incorrect Query Next Image Response Format
		0x85	Attempting to upgrade to invalid firmware (Bad Image Type, Wrong Mfg ID, Wrong HW/SW compatibility(%C))
		0x89	Image size is too big
		0x8A	Please ensure that the image you are attempting to upgrade has a different version than the current version
		0x01	ZCL OTA Message Out of Sequence
	0x05 Image Block Response	0x80	Incorrect Image Block Response Format
		0x01	ZCL OTA Message Out of Sequence
		0x87	Upgrade File Mismatch
	0x08 Upgrade End Response	0X87	Wrong Upgrade File

When the source device or the server receives a default response frame with a command ID of **0x0B** and the erring command is **0x02** that is, the **Query Next Image Response**, it means there is something wrong with the **Query Next Image Response** sent by the server. Similarly, if the erring command is **0x05** that is, the **Image Block Response**, it means there is something wrong with the **Image Block Response** sent by the server, and the same applies to **Upgrade End Response** where there is an error on the **Upgrade End response** message sent by the server.

Upgrade End Request error statuses

The status field in the Upgrade End request informs the server of any errors during the download or verification of the OTA firmware update image on the client. The error codes that could be reported are:

ZCL OTA Command	Status	Error Message
0x06 Upgrade End Request	0x94	Client Timed Out
	0x96	Invalid OTA Image
	0x95	Client Aborted Upgrade
	0x05	Storage Erase Failed
	0x87	Contact Tech Support (Highly unlikely to occur)