XBee®/XBee-PRO S2C Zigbee®

RF Module

User Guide

Revision history—90002002

Revision	Date	Description
AC	May 2019	Removed Brazilian certification information for certain devices.
AD	June 2019	Added FCC publication 996369 related information.
AE	October 2019	Enhancement to the network watchdog, fixes for a number of customer reported issues.
AF	November 2019	Added Brazilian ANATEL certification information.
AG	January 2020	Added IFETEL certifications.

Trademarks and copyright

Digi, Digi International, and the Digi logo are trademarks or registered trademarks in the United States and other countries worldwide. All other trademarks mentioned in this document are the property of their respective owners.

Disclaimers

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

Warranty

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

www.digi.com/howtobuy/terms

Customer support

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

Product name and model

Product serial number (s)

Firmware version

Operating system/browser (if applicable)

Logs (from time of reported issue)

Trace (if possible)

Description of issue

Steps to reproduce

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

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (XBee/XBee-PRO® S2C Zigbee® RF Module, 90002002 AG) in the subject line of your email.

Contents

Operation

Zigbee networks

Transmission, addressing, and routing

Zigbee security

Network commissioning and diagnostics

Manage End Devices

Analog and digital I/O lines

API Operation

Frame descriptions

AT commands

Module support

Regulatory information

Migrating from XBee through-hole to XBee surface-mount devices

Manufacturing information

Load Zigbee firmware on 802.15.4 devices

XBee/XBee-PRO® S2C Zigbee® RF Module

This manual describes the operation of the XBee/XBee-PRO Zigbee RF Module, which consists of Zigbee firmware loaded onto XBee S2C and PRO S2C hardware.

The XBee/XBee-PRO Zigbee RF Modules provide wireless connectivity to end-point devices in Zigbee mesh networks. Using the Zigbee PRO Feature Set, these modules are inter-operable with other Zigbee devices, including devices from other vendors. With the XBee, users can have their Zigbee network up-and-running in a matter of minutes without configuration or additional development.

The XBee/XBee-PRO Zigbee RF Modules are compatible with other devices that use XBee Zigbee technology. These include ConnectPortX gateways, XBee and XBee-PRO Adapters, Wall Routers, XBee Sensors, and other products with the Zigbee name.

Note on product naming

Although we refer to the device as an S2C, information for the S2D is also part of the guide. The information for the S2D is part of the S2C guide because the hardware is very similar.

Applicable firmware and hardware

This manual supports the following firmware:

- 401x, 402x, 403x, 404x, 405x (S2C), 406x
- 705x

It supports the following hardware:

- XBee S2C

Firmware release notes

You can view the current release notes in the Firmware Explorer section of XCTU. For instructions on downloading and using XCTU, go to: digi.com/products/xbee-rf-solutions/xctu-software/xctu.

Technical specifications

Performance specifications

The following table describes the performance specifications for the devices.

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

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C
Indoor/urban range	Up to 60 m (200 ft)	Up to 90 m (300 ft)
Outdoor RF line-of-sight range	Up to 1200 m (4000 ft)	Up to 3200 m (2 mi)
Transmit power output (maximum)	6.3 mW (+8 dBm), boost mode 3.1 mW (+5 dBm), normal mode channel 26 max power is +3 dBm	63 mW (+18 dBm)
RF data rate	250,000 b/s	
Receiver sensitivity	-102 dBm, boost mode -100 dBm, normal mode	-101 dBm

Power requirements

The following table describes the power requirements for the XBee/XBee-PRO Zigbee RF Module.

Specification	XBee	XBee-PRO
Supply voltage	2.1 - 3.6 V	2.7 - 3.6 V
Transmit current (typical, VCC = 3.3 V)	45 mA (8 dBm, Boost mode) 33 mA (5 dBm, Normal mode)	120 mA (18 dBm)
Idle / receive current (typical, VCC = 3.3 V)	31 mA (Boost mode) 28 mA (Normal mode)	31 mA
Power-down current	<1 uA @ 25C	<1 uA @ 25C

General specifications

The following table describes the general specifications for the devices.

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C
Operating frequency band	ISM 2.4 - 2.5 GHz	

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C
Form factor	through-hole, surface-mount	
Dimensions	through-hole: 2.438 x 2.761 cm (0.960 x 1.087 in) surface-mount: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)	through-hole: 2.438 x 3.294 cm (0.960 x 1.297 in) surface-mount: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)
Operating temperature	-40 to 85 ℃ (industrial)	
Antenna options	through-hole: PCB antenna, U.FL connector, RPSMA connector, or integrated wire surface-mount: RF pad, PCB antenna, or U.FL connector	

Networking and security specifications

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

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C
Supported network topologies	Point-to-point, point-to-multipoint, peer-to-peer, and DigiMesh	
Number of channels	16 Direct sequence channels	15 Direct sequence channels
Interface immunity	Direct Sequence Spread Spectrum (DSSS)	
Channels	11 to 26	
Addressing options	PAN ID and addresses, cluster IDs and endpoints (optional)	

Communication interface specifications

The following table provides the device's communication interface specifications.

Interface options	
UART	250 Kb/s maximum
SPI	5 Mb/s maximum (burst)

Regulatory conformity summary

This table describes the agency approvals for the devices.

Note Legacy XBee-PRO SMT (model: PRO S2C; hardware version 21xx) has different FCC and IC IDs. For more information, see Regulatory information.

Approval	XBee (surface-mount)	XBee-PRO (surface-mount)	XBee (through-hole)	XBee-PRO (through-hole)	XBee S2D (surface-mount)
United States (FCC Part 15.247)	FCC ID: MCQ-XBS2C	FCC ID: MCQ-XBPS2C (revision K and earlier) FCC ID: MCQ-PS2CSM (revision L and later)	FCC ID: MCQ-S2CTH	FCC ID: MCQ-PS2CTH	FCC ID: MCQ-S2DSM
Innovation, Science and Economic Development Canada (ISED)	IC: 1846A-XBS2C	IC: 1846A-XBPS2C (revision K and earlier) IC: 1846A-PS2CSM (revision L and later)	IC: 1846A-S2CTH	IC: 1846A-PS2CTH	IC: 1846A-S2DSM
FCC/IC Test Transmit Power Output range	-26 to +8 dBm	-0.7 to +19.4 dBm	-26 to +8 dBm	+1 to +19 dBm	-10 to +8 dBm
Europe (CE)	Yes		Yes		Yes
Australia	RCM	RCM	RCM	RCM	
Japan	R201WW10215369		R210-105563		
Brazil	ANATEL: 0616-15-1209	ANATEL: 07440-19-01209 (XBP24CZ7PIS-004, XBP24CZ7RIS-004, XBP24CZ7UIS-004)	ANATEL: 4556-15- 1209	ANATEL: 4077-15-1209	
Mexico	RCPDIXB19-1821	RCPDIPS19-1817-A1	RCPDIS219-1821-A1	RCPDIPS19-1817	

Approval	XBee (surface-mount)	XBee-PRO (surface-mount)	XBee (through-hole)	XBee-PRO (through-hole)	XBee S2D (surface-mount)
South Korea	MSIP-CRM-DIG-XBee-S2C		MSIP-CRM-DIG-XBee-S2C-TH		
RoHS	Compliant				

Serial communication specifications

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

UART pin assignments

Specifications	Device pin number	
UART pins	XBee (surface-mount)	XBee (through-hole)
DOUT	3	2
DIN / CONFIG	4	3
CTS / DIO7	25	12
RTS / DIO6	29	16

For more information on UART operation, see Operation.

SPI pin assignments

The SC2 (Serial Communication Port 2) of the Ember 357 is connected to the SPI port.

Specifications	Device pin number	
SPI pins	XBee (surface-mount)	XBee (through-hole)
SPI_SCLK	14	18
SPI_SSEL	15	17
SPI_MOSI	16	11
SPI_MISO	17	4

For more information on SPI operation, see SPI operation.

GPIO specifications

XBee/XBee-PRO Zigbee RF Modules have 15 General Purpose Input / Output (GPIO) ports available. The exact list depends on the device configuration as some GPIO pads are used for purposes such as serial communication.

See Enable GPIO 1 and 2 in the for more information on configuring and using GPIO ports.

GPIO electrical specification	Value
Voltage - supply	2.1 - 3.6 V

GPIO electrical specification	Value
Low Schmitt switching threshold	0.42 - 0.5 x VCC
High Schmitt switching threshold	0.62 - 0.8 x VCC
Input current for logic 0	-0.5 µA
Input current for logic 1	0.5 µA
Input pull-up resistor value	29 kΩ
Input pull-down resistor value	29 kΩ
Output voltage for logic 0	0.18 x VCC (maximum)
Output voltage for logic 1	0.82 x VCC (minimum)
Output source/sink current for pad numbers 3, 4, 5, 10, 12, 14, 15, 16, 17, 25, 26, 28, 29, 30, and 32 on the SMT modules	4 mA
Output source/sink current for pin numbers 2, 3, 4, 9, 12, 13, 15, 16, 17, and 19 on the TH modules	4 mA
Output source/sink current for pad numbers 7, 8, 24, 31, and 33 on the SMT modules	8 mA
Output source/sink current for pin numbers 6, 7, 11, 18, and 20 on the TH modules	8 mA
Total output current (for GPIO pads)	40 mA

Hardware specifications for the programmable variant

If the module has the programmable secondary processor, add the following table values to the specifications listed. For example, if the secondary processor is running at 20 MHz and the primary processor is in receive mode then the new current value will be *Itotal = Ir2 + Irx = 14 mA + 9 mA = 23 mA*, where *Ir2* is the runtime current of the secondary processor and Irx is the receive current of the primary.

Optional secondary processor specification	Add to RX, TX, and sleep currents specifications depending on mode of operation
Runtime current for 32 k running at 20 MHz	+14 mA
Runtime current for 32 k running at 1 MHz	+1 mA
Sleep current	+0.5 µA typical
For additional specifications see NXP Datasheet and Manual	MC9S08QE32

Optional secondary processor specification	Add to RX, TX, and sleep currents specifications depending on mode of operation
Minimum Reset low pulse time for EM357	+26 µS
V_{REF} Range	1.8 VDC to V_{CC}
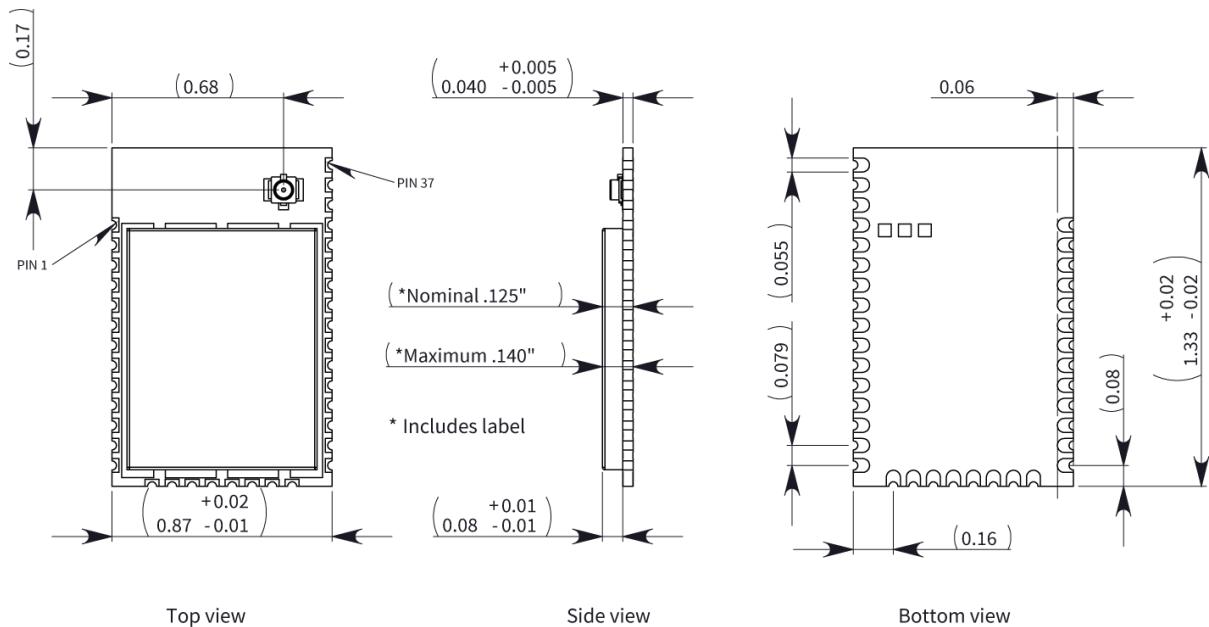
Hardware

Mechanical drawings

The following mechanical drawings of the XBee/XBee-PRO Zigbee RF Modules show all dimensions in inches. The first drawing shows the XBee/XBee-PRO surface-mount model (antenna options not shown).

Top view   Side view   Bottom view

The drawings below show the XBee through-hole model

RPSMA

U.FL   ## Wire whip   ## PCB antenna
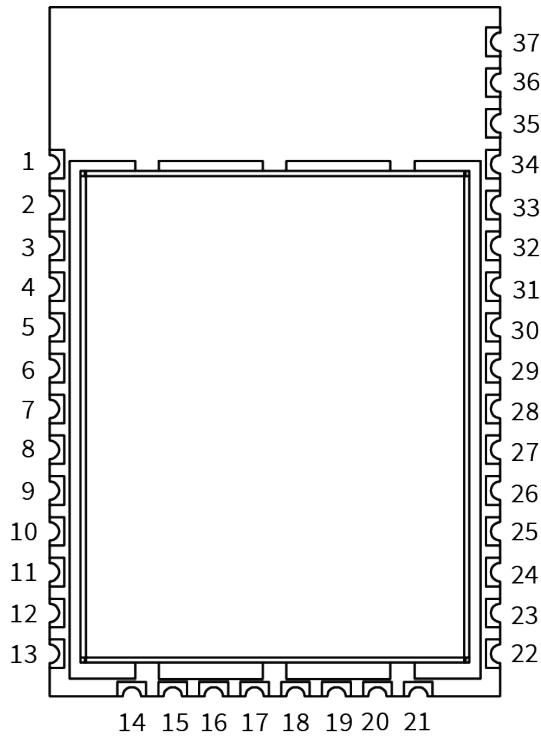
The drawings below show the XBee-PRO through-hole model.

RPSMA

U.FL   Wire whip   PCB antenna

Pin signals for the surface-mount module

The following drawing shows the surface-mount (SMT) pin locations.

The following table shows the pin signals and their descriptions for the surface-mount device.

Pin#	Name	Direction	Default state	Description
1	GND	-	-	Ground.
2	VCC	-	-	Power supply.
3	DOUT /DIO13	Both	Output	UART data out /GPIO.
4	DIN / $\overline{\text{CONFIG}}$ /DIO14	Both	Input	UART data in /GPIO.
5	DIO12	Both		GPIO.
6	$\overline{\text{RESET}}$	Input		Device reset.
7	RSSI PWM/DIO10	Both	Output	RX signal strength Indicator /GPIO.
8	PWM1/DIO11	Both	Disabled	Pulse width modulator/GPIO.
9	[reserved]	-	Disabled	Do not connect.
10	$\overline{\text{DTR}}$/SLEEP_RQ /DIO8	Both	Input	Pin sleep control Line/GPIO.
11	GND	-	-	Ground.
12	SPI_$\overline{\text{ATTN}}$/ $\overline{\text{BOOTMODE}}$/DIO19	Output	Output	Serial peripheral interface attention . Do not tie low on reset.
13	GND	-	-	Ground.
14	SPI_CLK /DIO18	Input	Input	Serial peripheral interface clock/GPIO.
15	SPI_$\overline{\text{SSEL}}$/DIO17	Input	Input	Serial peripheral interface not select/GPIO.
16	SPI_MOSI/DIO16	Input	Input	Serial peripheral interface data in/GPIO.
17	SPI_MISO/DIO15	Output	Output	Serial peripheral interface data out/GPIO.

Pin#	Name	Direction	Default state	Description
18	[reserved]*	-	Disabled	Do not connect.
19	[reserved]*	-	Disabled	Do not connect.
20	[reserved]*	-	Disabled	Do not connect.
21	[reserved]*	-	Disabled	Do not connect.
22	GND	-	-	Ground.
23	[reserved]	-	Disabled	Do not connect.
24	DIO4	Both	Disabled	GPIO.
25	$\overline{\text{CTS}}$/DIO7	Both	Output	Clear to send flow control/GPIO.
26	ON/$\overline{\text{SLEEP}}$/DIO9	Both	Output	Device status indicator/GPIO
27	VREF	Input	-	Not used for EM357. Used for programmable secondary processor. For compatibility with other XBee devices, we recommend connecting this pin to the voltage reference if analog sampling is desired. Otherwise, connect to GND.
28	ASSOCIATE/DIO5	Both	Output	Associate Indicator/GPIO.
29	$\overline{\text{RTS}}$/DIO6	Both	Input	Request to send flow control /GPIO.
30	AD3/DIO3	Both	Disabled	Analog input/GPIO.
31	AD2/DIO2	Both	Disabled	Analog input/GPIO

Pin#	Name	Direction	Default state	Description
32	AD1/DIO1	Both	Disabled	Analog input/GPIO.
33	AD0 /DIO0	Both	Input	Analog input / GPIO / Commissioning button.
34	[reserved]	-	Disabled	Do not connect.
35	GND	-	-	Ground.
36	RF	Both	-	RF I/O for RF pad variant.
37	[reserved]	-	Disabled	Do not connect.
Signal direction is specified with respect to the device. See Design notes for details on pin connections. * Refer to Writing custom firmware for instructions on using these pins if JTAG functions are needed.				

Pin signals for the through-hole module

The following table shows the pin signals and their descriptions for the through-hole module.

Pin #	Name	Direction	Default state	Description
1	VCC	-	-	Power supply
2	DOUT/DIO13	Both	Output	UART data out
3	DIN/$\overline{CONFIG}$ / DIO14	Both	Input	UART data in
4	DIO12/SPI_MISO	Both	Disabled	GPIO/SPI slave out
5	$\overline{RESET}$	Input	Input	Module reset
6	RSSI PWM/PWMO DIO10	Both	Output	RX signal strength indicator/GPIO
7	PWM1/DIO11	Both	Disabled	GPIO
8	[reserved]	-	-	Do not connect
9	DTR/$\overline{SLEEP\_RQ}$/ DIO8	Both	Input	Pin sleep control line/GPIO
10	GND	-	-	Ground
11	SPI_MOSI/DIO4	Both	Disabled	GPIO/SPI slave in

Pin #	Name	Direction	Default state	Description
12	$\overline{CTS}$/DIO7	Both	Output	Clear-to-send flow control/GPIO
13	ON_$\overline{SLEEP}$/DIO9	Both	Output	Device status indicator/GPIO
14	VREF	-	-	Not connected
15	ASSOCIATE/DIO5	Both	Output	Associate indicator/GPIO
16	$\overline{RTS}$/DIO6	Both	Input	Request to send flow control/ GPIO
17	AD3/DIO3/SPI_$\overline{SSEL}$	Both	Disabled	Analog input/GPIO/SPI slave select
18	AD2 /DIO2/SPI_CLK	Both	Disabled	Analog input/GPIO/SPI clock
19	AD1/DIO1/SPI_$\overline{ATTN}$	Both	Disabled	Analog input/GPIO/SPI attention
20	AD0/DIO0/CB	Both	Disabled	Analog input/GPIO/ Commissioning button

EM357 pin mappings

The following table shows how the EM357 pins are used on the device.

Note Some lines may not go to the external device pins in the programmable secondary processor version.

EM357 pin#	EM357 pin name	XBee (SMT) pad#	XBee (TH) pin #	Other usage
12	RST	6	5	Programming
18	PA7	8	7	
19	PB3	29	16	Used for UART
20	PB4	25	12	Used for UART
21	PA0/ SC2MOSI	16	11	Used for SPI
22	PA SC2MISO	17	4	Used for SPI
24	PA2/SC2SCLK	14	18	Used for SPI
25	PA3/SC2SSEL	15	17	Used for SPI
26	PA4/PTI_EN	32	19	OTA packet tracing
27	PA5/PTI_ DATA/BOOTMODE	12	NA	OTA packet tracing, force embedded serial bootloader, and SPI attention line

EM357 pin#	EM357 pin name	XBee (SMT) pad#	XBee (TH) pin #	Other usage
29	PA6	7	6	
30	PB1/SC1TXD	3	2	Used for UART
31	PB2/SC1RXD	4	3	Used for UART
33	PC2/JTDO/SWO	26	13	JTAG (see Writing custom firmware)
34	PC3/JTDI	28	15	JTAG (see Writing custom firmware)
35	PC4/JTMS/SWDIO	5	4	JTAG (see Writing custom firmware)
36	PB0	10	9	
38	PC1/ADC3	30	17	
41	PB7/ADC2	31	18	
42	PB6/ADC1	33	20	
43	PB5/ADC0			Temperature sensor on PRO version

Design notes

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

Power supply design

A poor power supply can lead to poor radio performance, especially if you do not keep the supply voltage within tolerance or if it is excessively noisy. To help reduce noise, place a 1.0 µF and 8.2 pF capacitor as near as possible to /SMT or pin 1/TH on the PCB. If you are using a switching regulator for the power supply, switch the frequencies above 500 kHz. Limit the power supply ripple to a maximum 50 mV peak to peak.

For designs using the programmable modules, we recommend an additional 10 µF decoupling cap near (pad 2/SMT or pin 1/TH) of the module. The nearest proximity to (pad 2/SMT or pin 1/TH) of the three caps should be in the following order:

1. 8.2 pf
2. 1 µF
3. 10 µF

Board layout

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

Antenna performance

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

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

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

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

- Vehicles
- Elevators
- Ventilation ducts
- Refrigerators
- Microwave ovens
- Batteries
- Tall electrolytic capacitors

Use the following additional guidelines for optimal antenna performance:

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

Recommended pin connections

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

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

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

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

For analog sampling, attach the VREF pin (pin 14) to a voltage reference.
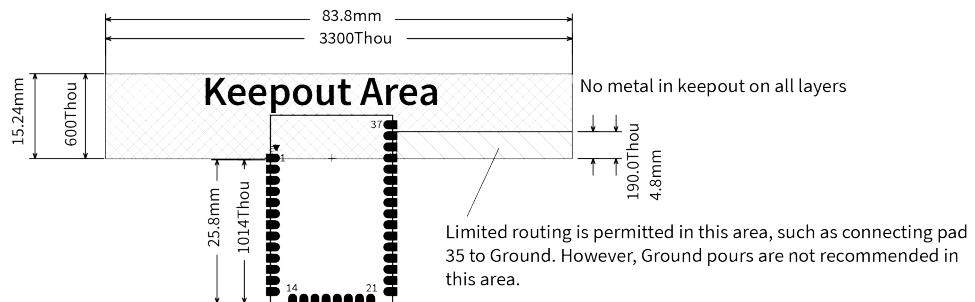
Design notes for PCB antenna devices

Position PCB antenna devices so there are no ground planes or metal objects above or below the antenna. For best results, do not place the device in a metal enclosure, as this may greatly reduce the range. Place the device at the edge of the PCB on which it is mounted. Make sure the ground, power and signal planes are vacant immediately below the antenna section.
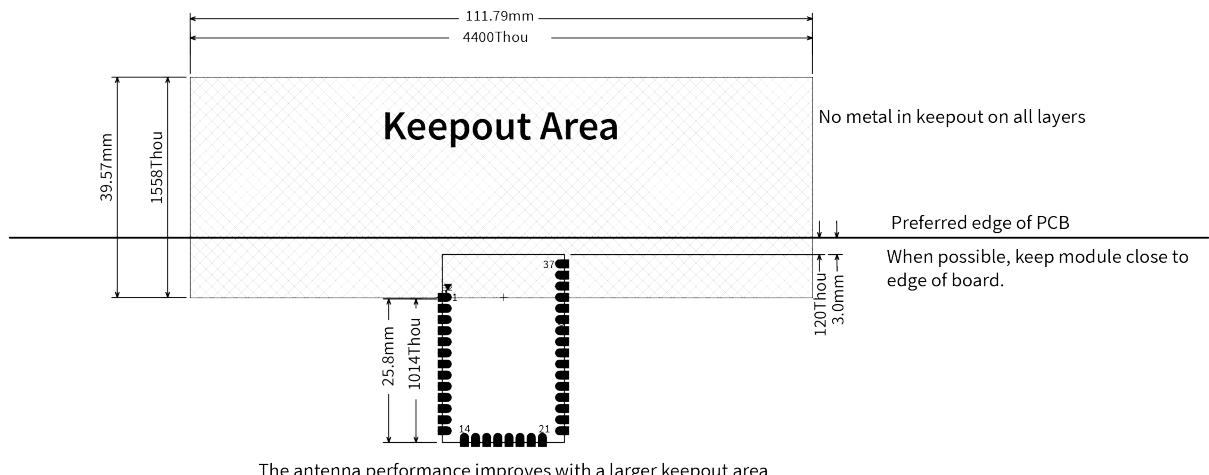
The following drawings illustrate important recommendations when you are designing with PCB antenna devices. For optimal performance, do not mount the device on the RF pad footprint described in the next section, because the footprint requires a ground plane within the PCB antenna keep out area.

Surface-mount keepout area

Minimum Keepout Area for PCB Antenna (All layers)

83.8mm
3300Thou

15.24mm
600Thou

Keepout Area

No metal in keepout on all layers

37

1

25.8mm
1014Thou

190.0Thou
4.8mm

Limited routing is permitted in this area, such as connecting pad 35 to Ground. However, Ground pours are not recommended in this area.

14   21

Recommended Keepout Area for PCB Antenna (All layers)

111.79mm
4400Thou

39.57mm
1558Thou

Keepout Area

No metal in keepout on all layers

Preferred edge of PCB

37

When possible, keep module close to edge of board.

120Thou
3.0mm

1

25.8mm
1014Thou

14   21

The antenna performance improves with a larger keepout area

Through-hole keepout area

Minimum Keepout Area (All PCB layers)

Keepout Area

83.8mm
3300Thou

15.2mm / 600Thou

No metal in keepout on all layers

24.9mm / 980Thou

770Thou / 19.6mm

Digi International XBee® PRO

XBee form factor

XBee-PRO form factor

Recommended Keepout Area (All PCB layers)

Keepout Area

111.8mm
4400Thou

39.6mm / 1560Thou

No metal in keepout on all layers

Preferred edge of PCB

When possible, keep module close to edge of board.

120Thou / 3.0mm

24.9mm / 980Thou

770Thou / 19.6mm

Digi International XBee® PRO

The antenna performance improves with a larger keepout area

Notes

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

Design notes for RF pad devices

The RF pad is a soldered antenna connection. The RF signal travels from pin 33 on the device to the antenna through an RF trace transmission line on the PCB. Any additional components between the device and antenna violates modular certification. The controlled impedance for the RF trace is 50 Ω.

We recommend using a microstrip trace, although you can also use a coplanar waveguide if you need more isolation. A microstrip generally requires less area on the PCB than a coplanar waveguide. We do not recommend using a stripline because sending the signal to different PCB layers can introduce matching and performance problems.

Following good design practices is essential when implementing the RF trace on a PCB. Consider the following points:
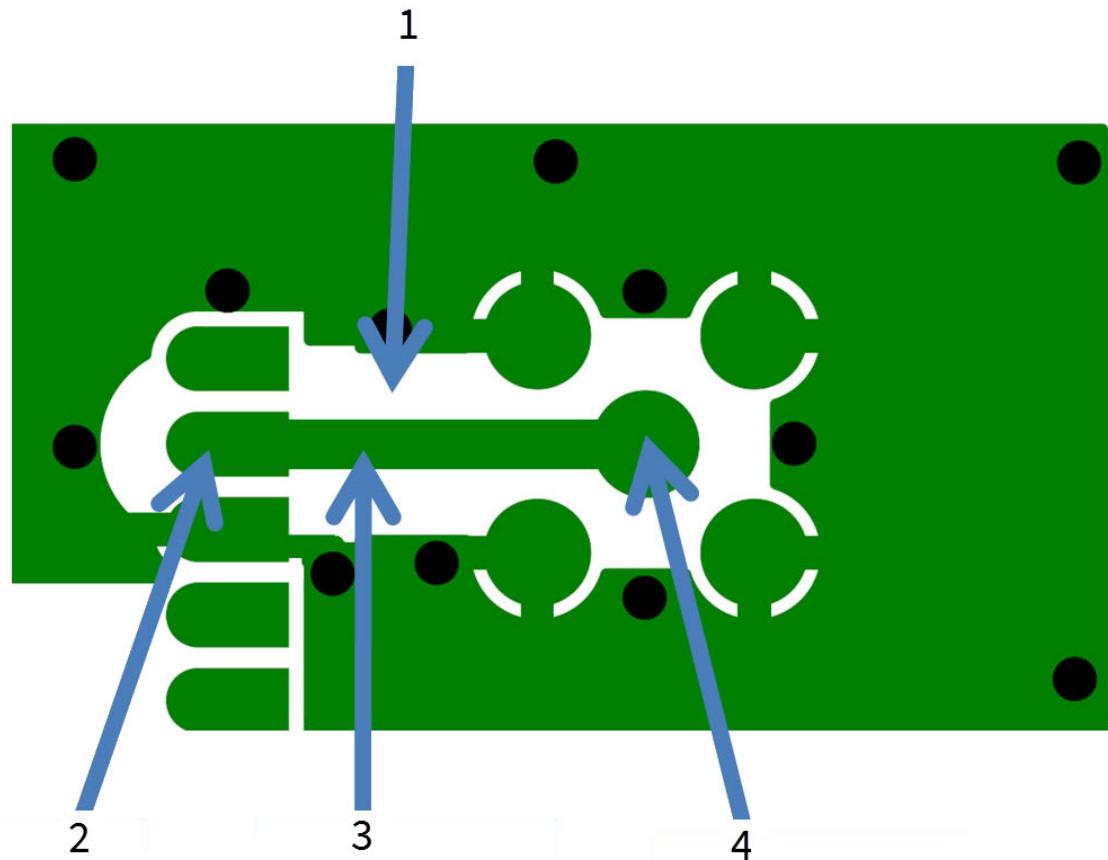
- Minimize the length of the trace by placing the RPSMA jack close to the device.
- Connect all of the grounds on the jack and the device to the ground planes directly or through closely placed vias.
- Space any ground fill on the top layer at least twice the distance **d** (in this case, at least 0.028") from the microstrip to minimize their interaction.

Additional considerations:

- The top two layers of the PCB have a controlled thickness dielectric material in between.
- The second layer has a ground plane which runs underneath the entire RF pad area. This ground plane is a distance **d**, the thickness of the dielectric, below the top layer.
- The top layer has an RF trace running from pin 33 of the device to the RF pin of the RPSMA connector.
- The RF trace width determines the impedance of the transmission line with relation to the ground plane. Many online tools can estimate this value, although you should consult the PCB manufacturer for the exact width.

Implementing these design suggestions helps ensure that the RF pad device performs to its specifications.

The following figures show a layout example of a host PCB that connects an RF pad device to a right angle, through-hole RPSMA jack.

Number	Description
1	Maintain a distance of at least 2 d between microstrip and ground fill.
2	Device pin 33.
2	RF pad pin.
3	50 Ω microstrip trace.
4	RF connection of RPSMA jack.
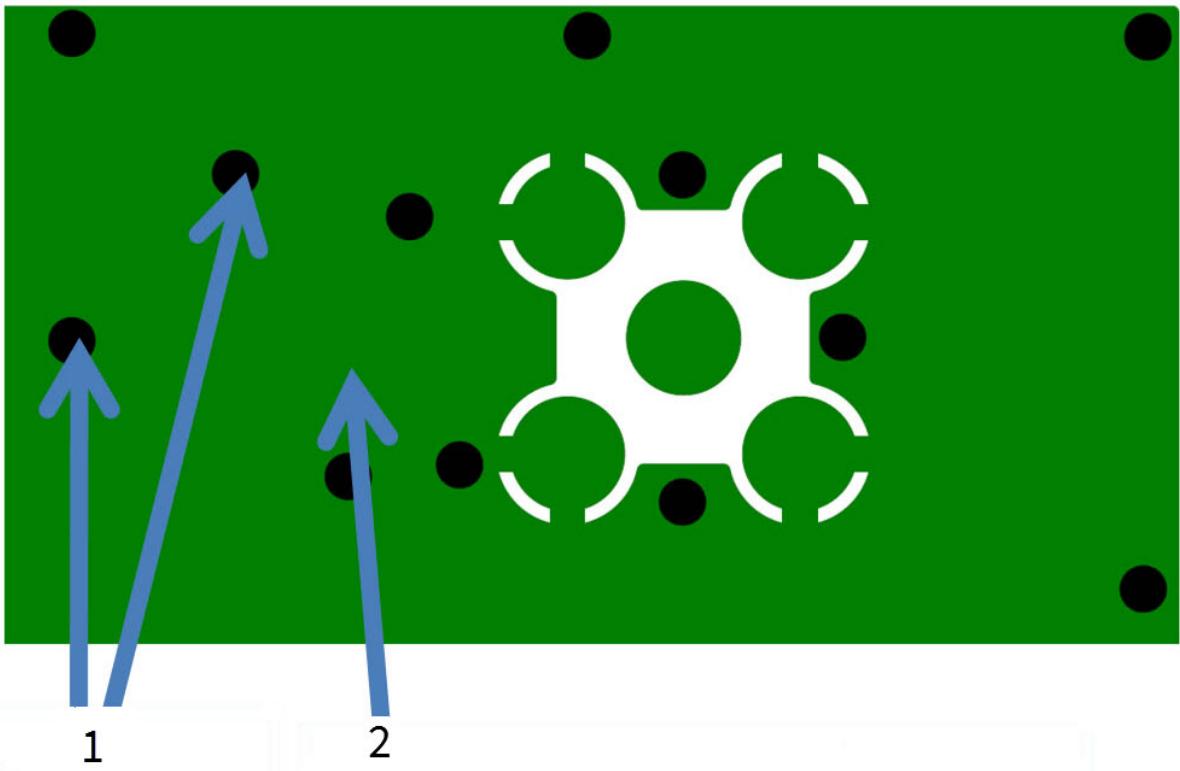
The width in this example is approximately 0.025 in for a 50 Ω trace, assuming d = 0.014 in, and that the dielectric has a relative permittivity of 4.4. This trace width is a good fit with the device footprint's 0.335" pad width.

Note We do not recommend using a trace wider than the pad width, and using a very narrow trace (under 0.010") can cause unwanted RF loss.

The following illustration shows PCB layer 2 of an example RF layout.

Number	Description
1	Use multiple vias to help eliminate ground variations.
2	Put a solid ground plane under RF trace to achieve the desired impedance.

Module operation for the programmable variant

The modules with the programmable option have a secondary processor with 32k of flash and 2k of RAM. This allows module integrators to put custom code on the XBee module to fit their own unique needs. The secondary processor intercepts the DIN, DOUT, RTS, CTS, and RESET lines to allow it to be in control of the data transmitted and received. All other lines are in parallel and can be controlled by either the EM357 or the MC9SO8QE micro. See the following block diagram for details. The Programmable XBee SDK native APIs automatically handle pin use.

For the secondary processor to sample with ADCs, the XBee VREF pin (27/SMT, 14/TH) must be connected to a reference voltage.

Digi provides a bootloader that can take care of programming the processor over-the-air or through the serial interface. This means that over-the-air updates can be supported through an XMODEM protocol. The processor can also be programmed and debugged through a one wire interface BKGD (Pin 9/SMT, Pin 8/TH).

Programmable XBee SDK

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

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

- CodeWarrior IDE: http://ftp1.digi.com/support/sampleapplications/40003004_B.exe
- Programmable XBee SDK: http://ftp1.digi.com/support/sampleapplications/40003003_D.exe
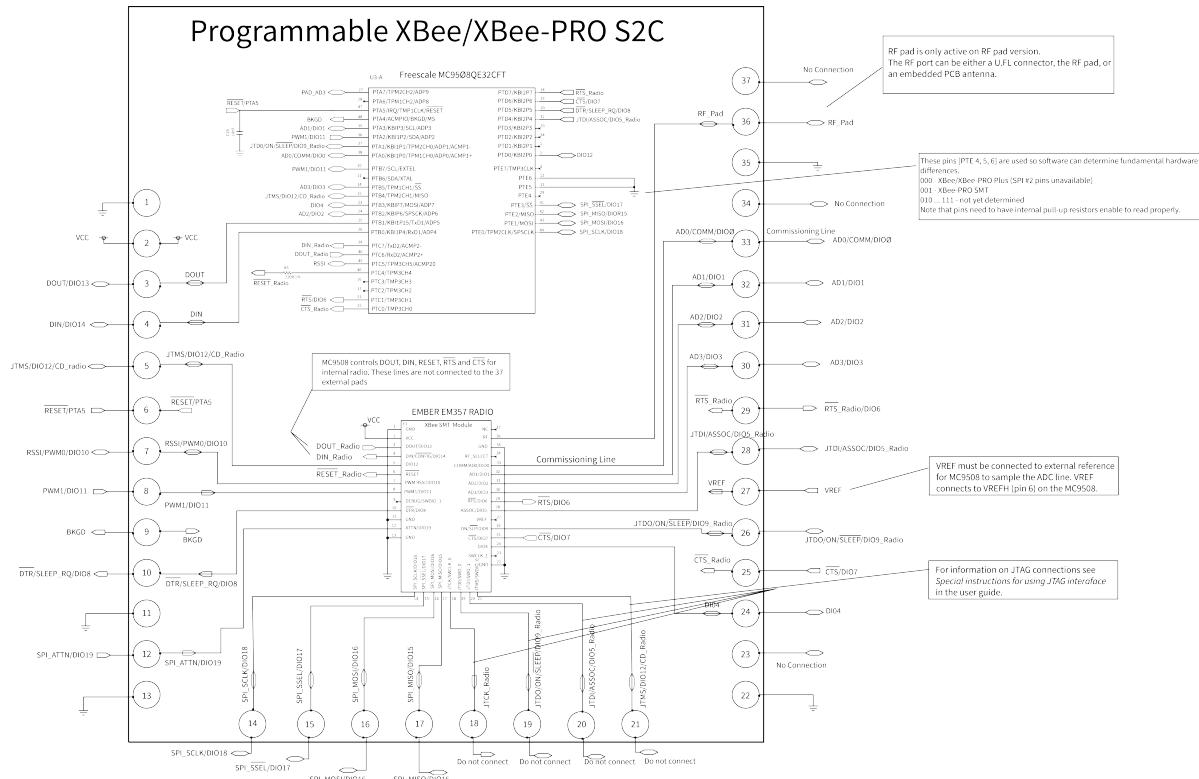
If these revisions change, search for the part number on Digi's website. For example, search for **40003003**.

Install the IDE first, and then install the SDK.
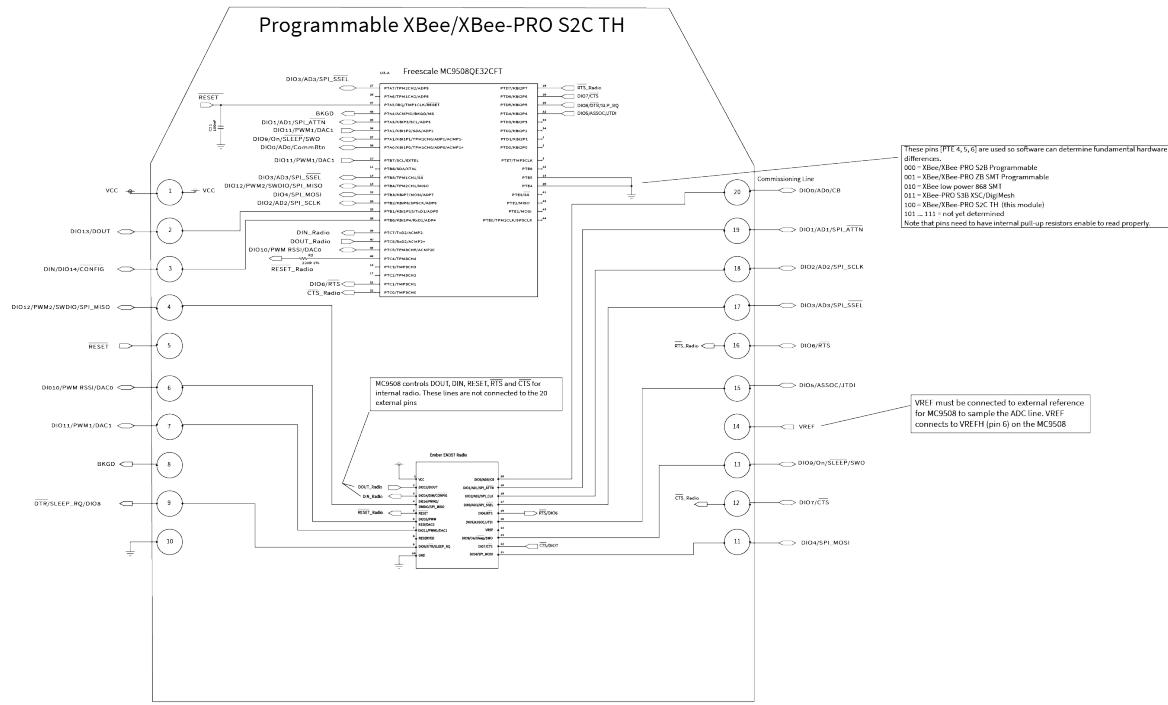
The documentation for the Programmable XBee SDK is built into the SDK, so the Getting Started guide appears when you open CodeWarrior.

Programmable connections

The following figure shows the programmable connections for the SMT.

Programmable XBee/XBee-PRO S2C

The following illustration shows the programmable connections for the TH Module.

Programmable XBee/XBee-PRO S2C TH

Operation
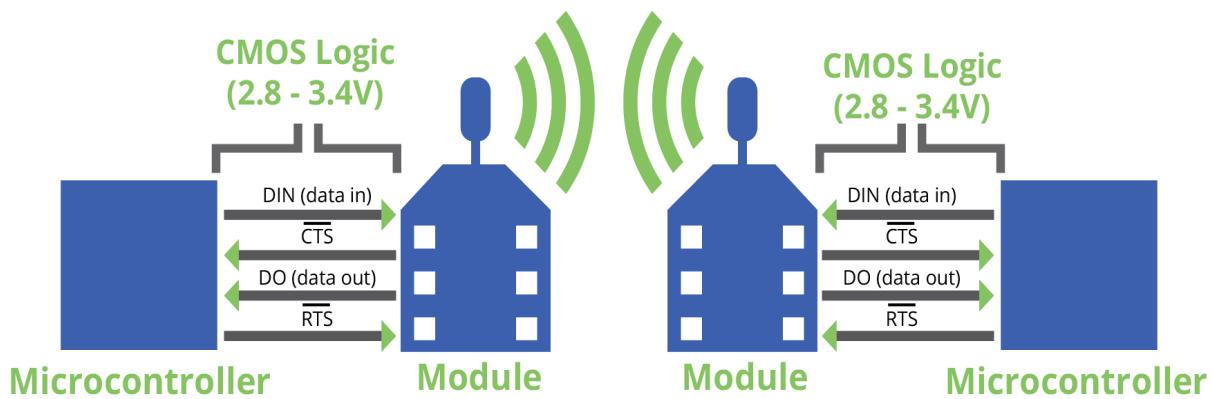
Serial interface

The XBee/XBee-PRO Zigbee RF Module interface to a host device through a serial port. The device can communicate with any logic and voltage compatible UART, through a level translator to any serial device (for example, through a RS-232 or USB interface board), or through a SPI, as described in SPI communications.

Note Two Wire serial Interface (TWI) is also available, but not supported by Digi. For information on the TWI, see the EM357 pin mappings.

UART data flow

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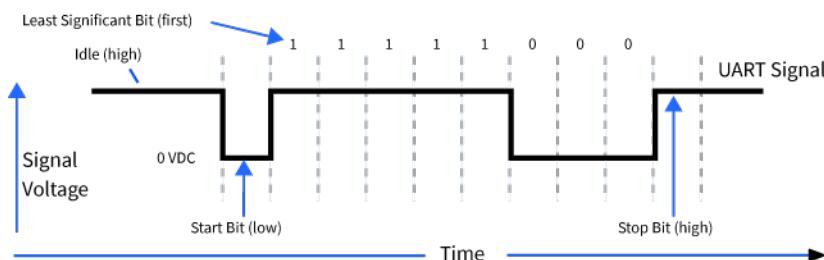
Serial data

A device sends data to the XBee/XBee-PRO Zigbee RF Module's UART through TH pin 4/SMT pin 4 DIN as an asynchronous serial signal. When the device is not transmitting data, the signals should idle high.

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

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

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

SPI communications

The XBee/XBee-PRO Zigbee RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.
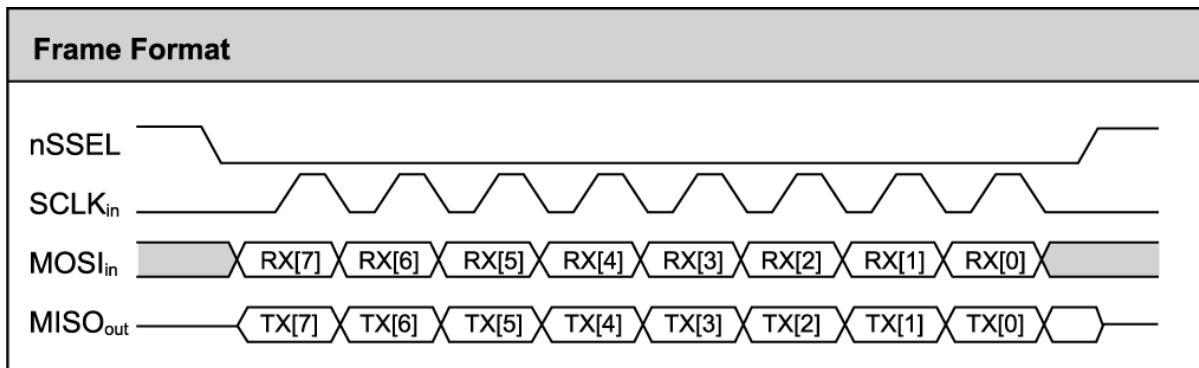
Signal	Function
SPI_MOSI (Master Out, Slave In)	Inputs serial data from the master
SPI_MISO (Master In, Slave Out)	Outputs serial data to the master
SPI_SCLK (Serial Clock)	Clocks data transfers on MOSI and MISO
SPI_SSEL (Slave Select)	Enables serial communication with the slave

The preceding four pins are standard for SPI. This devicd also supports an additional pin, which may be configured to alert the SPI master when it has data to send. This pin is called SPI_ATTN. If the master monitors this pin (through polling or interrupts), it can know when it needs to receive data from the device. SPI_ATTN asserts whenever it has data to send, and it remains asserted until all available data has been shifted out to the SPImaster.

In this mode:

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

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

SPI operation

When the master asserts the slave select (SPI_SSEL), the device drives SPI transmit data to the output pin (SPI_MISO), and receives SPI data from the input pin SPI_MOSI. The SPI_SSEL pin must be asserted to enable the transmit serializer to drive data to the output signal SPI_MISO. A rising edge on SPI_SSEL resets the SPI slave shift registers.

If the SPI_SCLK is present, the SPI_MISO line is always driven whether with or without the SPI_SSEL line driven.

If the input buffer is empty, the SPI serializer transmits a busy token (0xFF). Otherwise, all transactions on the SPI port use API operation. See API Operation for more information.

The SPI slave controller must guarantee that there is time to move new transmit data from the transmit buffer into the hardware serializer. To provide sufficient time, the SPI slave controller inserts a byte of padding at the start of every new string of transmit data. Whenever the transmit buffer is empty and the device places data into the transmit buffer, the SPI hardware inserts a byte of padding onto the front of the transmission as if this byte were placed there by software.

Serial port selection

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

If you enable only the UART, the XBee/XBee-PRO Zigbee RF Module uses only the UART, and ignores the SPI_SSEL.

If you enable only the SPI, the XBee/XBee-PRO Zigbee RF Module uses only the SPI, and ignores UART communications. If you hold DOUT low during boot, then the XBee/XBee-PRO Zigbee RF Module only uses the SPI.
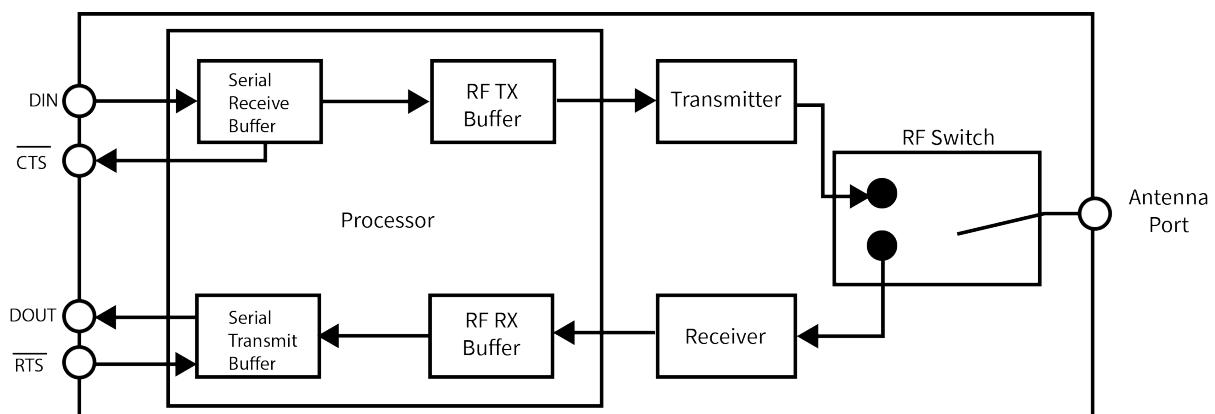
Once SPI is in use, do not attempt to apply changes (**AC**) which change the UART or SPI settings. Instead, use the following:

- 0x09 frames to reconfigure UART/SPI/other settings.
- **WR** to save the settings.
- **FR** to reset the XBee/XBee-PRO Zigbee RF Module and use the new configuration settings.

If neither serial port is enabled, then UART remains enabled, the device uses only the UART, and it ignores SPI_SSEL.

Serial buffers

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

Serial receive buffer

When serial data enters the XBee/XBee-PRO Zigbee RF Module through the serial port, the device stores the data in the serial receive buffer until it can be processed. Under certain conditions, the device may receive data when the serial receive buffer is already full. In that case, the device discards the data.

The serial receive buffer becomes full when data is streaming into the serial port faster than it can be processed and sent over the air (OTA). While the speed of receiving the data on the serial port can be much faster than the speed of transmitting data for a short period, sustained operation in that mode causes the device to drop data due to running out of places to put the data. Some things that may delay over the air transmissions are address discovery, route discovery, and retransmissions. Processing received RF data can also take away time and resources for processing incoming serial data.

If the UART is the serial port and you enable the $\overline{\text{CTS}}$ flow control, the device alerts the external data source when the receive buffer is almost full. The host delays sending data to the device until the module asserts $\overline{\text{CTS}}$ again, allowing more data to come in.

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

Serial transmit buffer

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

In situations where the serial transmit buffer may become full, resulting in dropped RF packets:

1. If the RF data rate is set higher than the interface data rate of the device, the device may receive data faster than it can send the data to the host. Even occasional transmissions from a large number of devices can quickly accumulate and overflow the transmit buffer.

2. If the host does not allow the device to transmit data out from the serial transmit buffer due to being held off by hardware flow control.

UART flow control

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

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

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

If you enable CTS flow control (**D7** command), when the serial receive buffer is 17 bytes away from being full, the device de-asserts $\overline{\text{CTS}}$ (sets it high) to signal to the host device to stop sending serial data. The device reasserts $\overline{\text{CTS}}$ after the serial receive buffer has 34 bytes of space.

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

RTS flow control

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

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

Break control

If a serial break (DIN held low) signal is sent for over five seconds, the device resets, and it boots into Command mode with default baud settings (9600 baud). If either **P3** or **P4** are not enabled, this break function is disabled.

Serial interface protocols

The XBee/XBee-PRO Zigbee RF Module supports both Transparent and Application Programming Interface (API) serial interfaces.

Transparent operating mode

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

Note Transparent operation is not available when using SPI.

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

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

API operating mode

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

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

The API operation option facilitates many operations such as:

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

Compare Transparent and API operation

The following tables compare the advantages of Transparent and API operating modes:

Transparent operation	
Simple interface	The device transmits all received serial data unless it is in Command mode.
Easy to support	It is easier for an application to support transparent operation and Command mode.

API operation	
Easy to manage data transmissions to multiple destinations	Transmitting RF data to multiple remotes only requires changing the address in the API frame. This process is much faster than in transparent operation where the application must enter Command mode, change the address, exit command mode, and then transmit data. Each API transmission can return a transmit status frame indicating the success or reason for failure.
Received data frames indicate the sender's address	All received RF data API frames indicate the source address.
Advanced Zigbee addressing support	API transmit and receive frames can expose Zigbee addressing fields including source and destination endpoints, cluster ID and profile ID. This makes it easy to support ZDO commands and public profile traffic.
Advanced networking diagnostics	API frames can provide indication of I/O samples from remote devices, and node identification messages.
Remote configuration	Set and read configuration commands can be sent to remote devices to configure them as needed using the API.

Generally, API mode is recommended when a device:

- Sends RF data to multiple destinations.
- Sends remote configuration commands to manage devices in the network.
- Receives RF data packets from multiple devices, and the application needs to know which device sent which packet.
- Must support multiple Zigbee endpoints, cluster IDs, and/or profile IDs.
- Uses the Zigbee device profile services.

API mode is required when:

- Using Smart Energy firmware.
- Using SPI for the serial port.
- Receiving I/O samples from remote devices.
- Using source routing.

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

Modes

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

- Transmit mode (Serial data in the serial receive buffer is ready to be packetized)
- Sleep mode
- Command Mode (Command mode sequence is issued)

Idle mode

When not receiving or transmitting data, the device is in Idle mode. During Idle mode, the device listens for valid data on both the RF and serial ports.
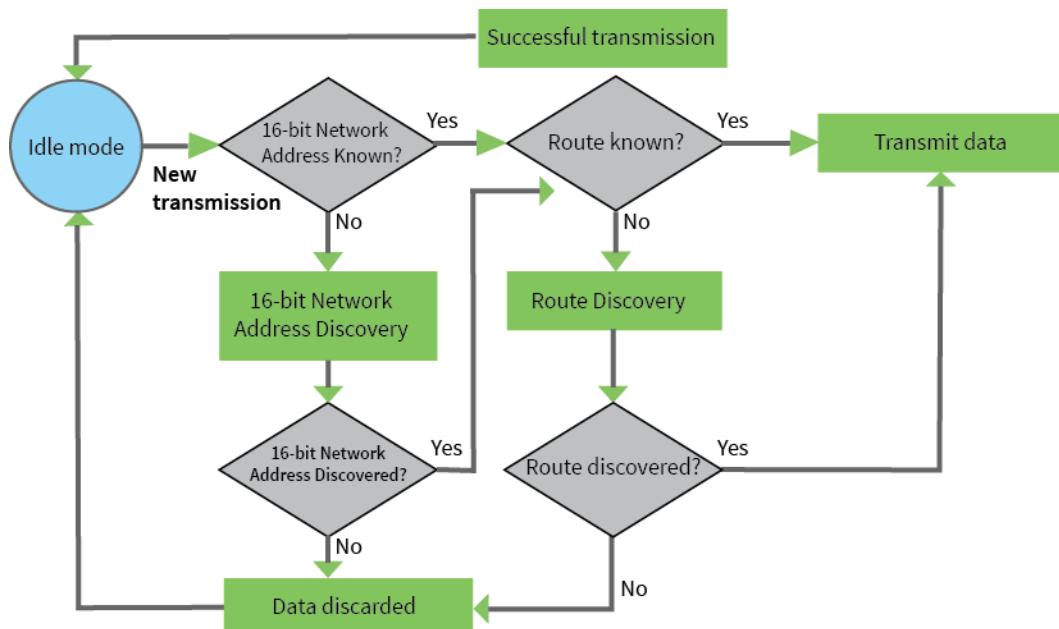
The device shifts into the other modes of operation under the following conditions:

- Transmit mode (serial data in the serial receive buffer is ready to be packetized).
- Receive mode (valid RF data received through the antenna).
- Command mode (Command mode sequence issued, not available with Smart Energy software or when using the SPI port).

Transmit mode

Prior to transmitting data, the module ensures that a 16-bit network address and route to the destination node have been established.

If you do not know the destination 16-bit network address known, network address discovery takes place. If you do not know the a route is not known, route discovery takes place for the purpose of establishing a route to the destination node. If a device with a matching network address is not discovered, it discards the packet. The device transmits the data once a route is established. If route discovery fails to establish a route, the device discards the packet. The following diagram shows the Transmit Mode sequence.

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

It is possible in rare circumstances for the destination to receive a data packet, but for the source to not receive the network acknowledgment. In this case, the source retransmits the data, which can cause the destination to receive the same data packet multiple times. The XBee modules do not filter out duplicate packets. We recommend that the application includes provisions to address this issue.

For more information, see Transmission, addressing, and routing.

Receive mode

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

Command mode

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

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

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

Enter Command mode

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

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

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

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

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

Troubleshooting

Failure to enter Command mode is often due to baud rate mismatch. Ensure that the baud rate of the connection matches the baud rate of the device. By default, BD (Interface Data Rate) = **3** (9600 b/s).
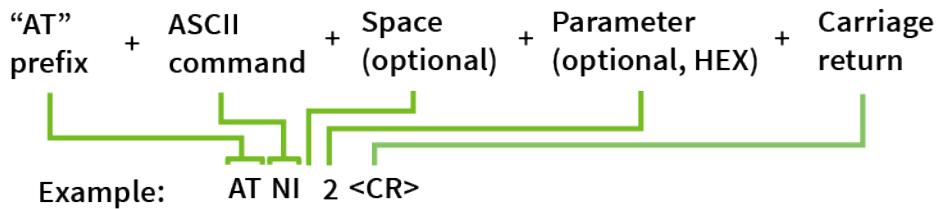
There are two alternative ways to enter Command mode:

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

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

Send AT commands

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

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

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

Multiple AT commands

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

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

Parameter format

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

Response to AT commands

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

For devices with a file system:

ATAP1<cr>

OK<cr>

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

ATAP<cr>

1<cr>

Apply command changes

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

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

Make command changes permanent

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

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

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

Exit Command mode

1. Send CN (Exit Command mode) followed by a carriage return.
   or:

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

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

Sleep mode

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

Sleep modes are discussed in detail in Manage End Devices.

Sleep modes allow the device to enter states of low power consumption when not in use. The device is almost completely off during sleep, and is incapable of sending or receiving data until it wakes up. XBee devices support pin sleep, where the device enters sleep mode upon pin transition, and cyclic sleep, where the device sleeps for a fixed time.

For more information, see Manage End Devices.

Zigbee networks

About the Zigbee specification

Zigbee is an open global standard for low-power, low-cost, low-data-rate, wireless mesh networking based on the IEEE 802.15.4 standard. It represents a network layer above the 802.15.4 layers to support advanced mesh routing capabilities. The Zigbee specification is developed by a consortium of companies that make up the Zigbee Alliance. The alliance is made up of over 300 members, including semiconductor, module, stack, and software developers. For more information, see http://www.zigbee.org/.

Definitions

This section provides definitions of the Zigbee note types and protocols.

Zigbee node types

Node	Description
Coordinator	A node that has the unique function of forming a network. The coordinator is responsible for establishing the operating channel and PAN ID for an entire network. Once established, the coordinator can form a network by allowing routers and end devices to join. Once the network is formed, the coordinator functions like a router. It can participate in routing packets and be a source or destination for data packets. ■ One coordinator per PAN ■ Establishes/Organizes PAN ■ Can route data packets to/from other nodes ■ Can be a data packet source and destination ■ Mains-powered Refer to Zigbee coordinator operation for more information.
Router	A node that creates/maintains network information and uses this information to determine the best route for a data packet. A router must join a network before it can allow other routers and end devices to join. A router can participate in routing packets and is intended to be a mains-powered node. ■ Several routers can operate in one PAN ■ Can route data packets to/from other nodes ■ Can be a data packet source and destination ■ Mains-powered Refer to Router operation for more information.

Node	Description
End device	End devices must always interact with their parent to receive or transmit data. (See the "Joining" in the following table.) They are intended to sleep periodically and therefore have no routing capacity. An end device can be a source or destination for data packets but cannot route packets. End devices can be battery-powered and offer low-power operation. ■ Several end devices can operate in one PAN ■ Can be a data packet source and destination ■ All messages are relayed through a coordinator or router ■ Lower power modes

Zigbee protocol
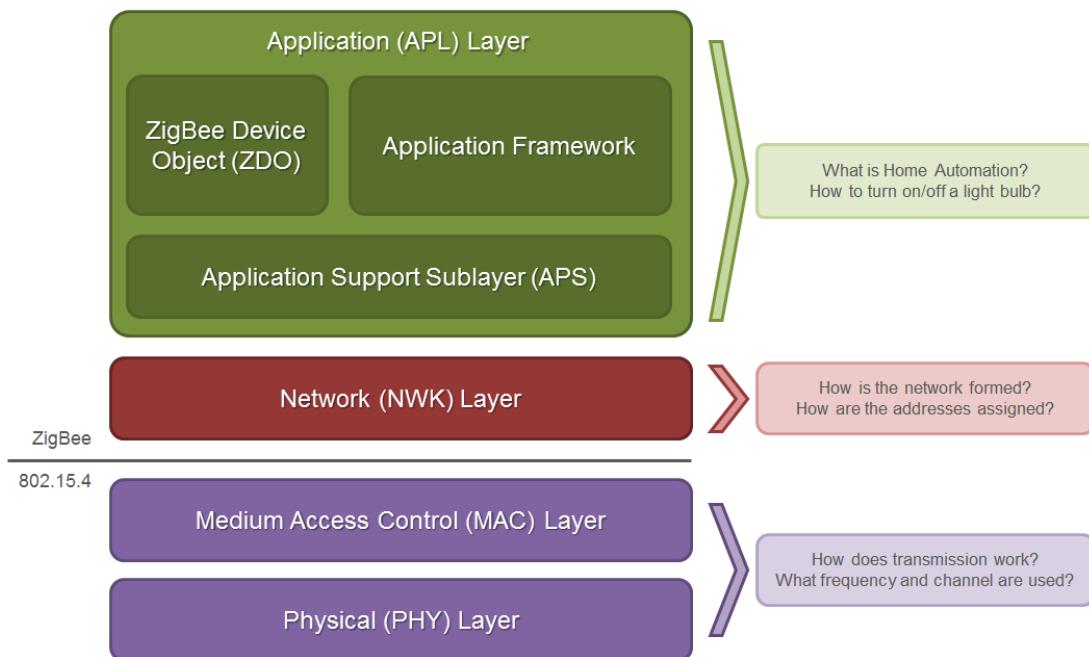
Protocol	Description
PAN	Personal Area Network - A data communication network that includes a coordinator and one or more routers/end devices.
Joining	The process of a node becoming part of a Zigbee PAN. A node becomes part of a network by joining to a coordinator or a router (that has previously joined to the network). During the process of joining, the node that allowed joining (the parent) assigns a 16-bit address to the joining node (the child).
Network address	The 16-bit address assigned to a node after it has joined to another node. The coordinator always has a network address of 0.
Operating channel	The frequency selected for data communications between nodes. The operating channel is selected by the coordinator on power-up.
Energy scan	A scan of RF channels that detects the amount of energy present on the selected channels. The coordinator uses the energy scan to determine the operating channel.
Route request	Broadcast transmission sent by a coordinator or router throughout the network in attempt to establish a route to a destination node.
Route reply	Unicast transmission sent back to the originator of the route request. It is initiated by a node when it receives a route request packet and its address matches the Destination Address in the route request packet.
Route reply	The process of establishing a route to a destination node when one does not exist in the Routing Table. It is based on the Ad-hoc On-demand Distance Vector routing (AODV) protocol.
Zigbee stack	Zigbee is a published specification set of high-level communication protocols for use with small, low- power modules. The Zigbee stack provides a layer of network functionality on top of the 802.15.4 specification. For example, the mesh and routing capabilities available to Zigbee solutions are absent in the 802.15.4 protocol.

Zigbee stack layers

Most network protocols use the concept of layers to separate different components and functions into independent modules that can be assembled in different ways.

Zigbee is built on the Physical (PHY) layer and Medium Access Control (MAC) sub-layer defined in the IEEE 802.15.4 standard. These layers handle low-level network operations such as addressing and message transmission/reception.

The Zigbee specification defines the Network (NWK) layer and the framework for the application (APL) layer. The Network layer takes care of the network structure, routing, and security. The application layer framework consists of the Application Support sub-layer (APS), the Zigbee device objects (ZDO) and user-defined applications that give the device its specific functionality.

This table describes the Zigbee layers.

Zigbee layer	Descriptions
PHY	Defines the physical operation of the Zigbee device including receive sensitivity, channel rejection, output power, number of channels, chip modulation, and transmission rate specifications. Most Zigbee applications operate on the 2.4 GHz ISM band at a 250 kb/s data rate. See the IEEE 802.15.4 specification for details.
MAC	Manages RF data transactions between neighboring devices (point to point). The MAC includes services such as transmission retry and acknowledgment management, and collision avoidance techniques (CSMA-CA).
Network	Adds routing capabilities that allows RF data packets to traverse multiple devices (multiple hops) to route data from source to destination (peer to peer).

Zigbee layer	Descriptions
APS (AF)	Application layer that defines various addressing objects including profiles, clusters, and endpoints.
ZDO	Application layer that provides device and service discovery features and advanced network management capabilities.

Zigbee networking concepts

Device types

Zigbee defines three different device types: coordinator, router, and end device.

Coordinator

Zigbee networks may only have a single coordinator device. This device:

- Starts the network, selecting the channel and PAN ID (both 64-bit and 16-bit).
- Buffers wireless data packets for sleeping end device children.
- Manages the other functions that define the network, secure it, and keep it healthy.
- Cannot sleep; the coordinator must be powered on all the time.

Router

A router is a full-featured Zigbee node. This device:

- Can join existing networks and send, receive, and route information. Routing involves acting as a messenger for communications between other devices that are too far apart to convey information on their own.
- Can buffer wireless data packets for sleeping end device children. Can allow other routers and end devices to join the network.
- Cannot sleep; router(s) must be powered on all the time.
- May have multiple router devices in a network.
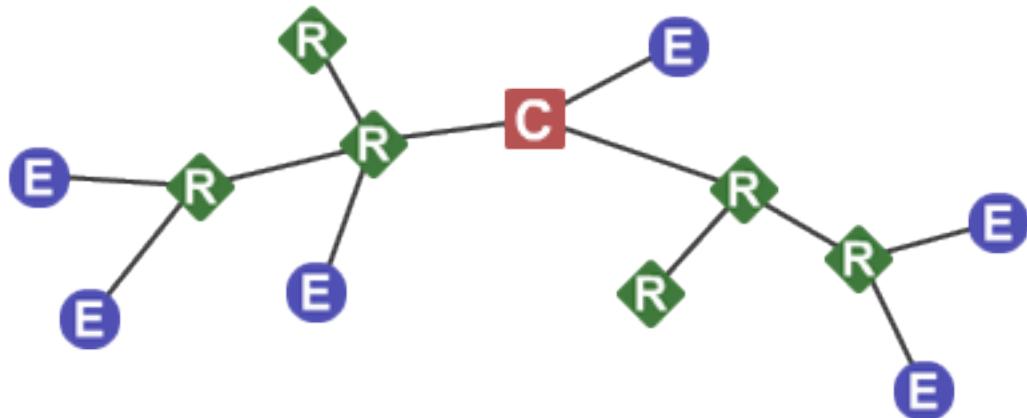
End device

An end device is essentially a reduced version of a router. This device:

- Can join existing networks and send and receive information, but cannot act as messenger between any other devices.
- Cannot allow other devices to join the network.
- Uses less expensive hardware and can power itself down intermittently, saving energy by temporarily entering a non responsive sleep mode.
- Always needs a router or the coordinator to be its parent device. The parent helps end devices join the network, and stores messages for them when they are asleep.

Zigbee networks may have any number of end devices. In fact, a network can be composed of one coordinator, multiple end devices, and zero routers.

The following diagram shows a generic Zigbee network.

Note Each Zigbee network must be formed by one, and only one, coordinator and at least one other device (router or end device).

In Zigbee networks, the coordinator must select a PAN ID (64-bit and 16-bit) and channel to start a network. After that, it behaves essentially like a router. The coordinator and routers can allow other devices to join the network and can route data.

After an end device joins a router or coordinator, it must be able to transmit or receive RF data through that router or coordinator. The router or coordinator that allowed an end device to join becomes the "parent" of the end device. Since the end device can sleep, the parent must be able to buffer or retain incoming data packets destined for the end device until the end device is able to wake and receive the data.

A device can only operate as one of the three device types. The device type is selected by configuration rather than by firmware image as was the case on earlier hardware platforms.

By default, the device operates as a router. To select coordinator operation, set **CE** to 1. To select end device operation, set **SM** to a non-zero value. To select router operation, both **CE** and **SM** must be 0.

If a device is a coordinator and it needs to be changed into an end device, you must set **CE** to 0 first. If not, the **SM** configuration will conflict with the **CE** configuration. Likewise, to change an end device into a coordinator, you must change it into a router first.

Another complication is that default parameters do not always work well for a coordinator.

For example:

- **DH**/**DL** is 0 by default, which allows routers and end devices to send transparent data to the coordinator when they first come up. If **DH**/**DL** is not changed from the default value when the device is changed to a coordinator, then the device sends data to itself, causing characters to be echoed back to the screen as they are typed. Since this is probably not the desired operation, set **DH**/**DL** to the broadcast address or some specific unicast address when the device is changed to a coordinator.

- Another example is **EO** for smart energy builds. Set this value to 08 for routers and end devices and 02 for the coordinator to designate it as the trust center. Therefore, if using authentication, which is the normal case for Smart Energy builds, change **EO** from 02 to 08 when **CE** is set to 1.

- Another example is **EO** for Zigbee builds. By default the value is 0x00. But if **EO** and **EE** are set to 0x01 on all radios in a network, then the network key will be sent in the clear (unencrypted) at association time. This may be a useful setting in development environments, but is discouraged for security reasons for product deployment.

In general, it is your responsibility to ensure that parameters are set to be compatible with the new device type when changing device types.

PAN ID

Zigbee networks are called personal area networks (PANs). Each network is defined with a unique PAN identifier (PAN ID), which is common among all devices of the same network. Zigbee devices are either preconfigured with a PAN ID to join, or they can discover nearby networks and select a PAN ID to join.

Zigbee supports both a 64-bit and a 16-bit PAN ID. Both PAN IDs are used to uniquely identify a network. Devices on the same Zigbee network must share the same 64-bit and 16-bit PAN IDs. If multiple Zigbee networks are operating within range of each other, each should have unique PAN IDs.

16-bit PAN ID

The 16-bit PAN ID is used as a MAC layer addressing field in all RF data transmissions between devices in a network. However, due to the limited addressing space of the 16-bit PAN ID (65,535 possibilities), there is a possibility that multiple Zigbee networks (within range of each other) could use the same 16-bit PAN ID. To resolve potential 16-bit PAN ID conflicts, the Zigbee Alliance created a 64-bit PAN ID.

64-bit PAN ID

The 64-bit PAN ID (also called the extended PAN ID), is intended to be a unique, non-duplicated value. When a coordinator starts a network, it can either start a network on a preconfigured 64-bit PAN ID, or it can select a random 64-bit PAN ID. Devices use a 64-bit PAN ID during joining; if a device has a preconfigured 64-bit PAN ID, it will only join a network with the same 64-bit PAN ID. Otherwise, a device could join any detected PAN and inherit the PAN ID from the network when it joins. All Zigbee beacons include the 64-bit PAN ID and is used in 16- bit PAN ID conflict resolution.

Routers and end devices

Routers and end devices are typically configured to join a network with any 16-bit PAN ID as long as the 64-bit PAN ID is valid. Coordinators typically select a random 16-bit PAN ID for their network.

Since the 16-bit PAN ID only allows up to 65,535 unique values, and the device randomly selects the 16-bit PAN ID, provisions exist in Zigbee to detect if two networks (with different 64-bit PAN IDs) are operating on the same 16- bit PAN ID. If the device detects a conflict, the Zigbee stack can perform PAN ID conflict resolution to change the 16- bit PAN ID of the network in order to resolve the conflict. See the Zigbee specification for details.

Zigbee routers and end devices should be configured with the 64-bit PAN ID of the network they want to join, and they typically acquire the 16-bit PAN ID when they join a network.

Only enable **CE** on one device to avoid PAN ID conflicts and network problems.

Operating channels

Zigbee uses direct-sequence spread spectrum modulation and operates on a fixed channel. The 802.15.4 PHY defines 16 operating channels (channels 11 to 26) in the 2.4 GHz frequency band. XBee modules support all 16 channels.

Zigbee application layers: in depth

The following topics provide a more in-depth look at the Zigbee application stack layers (APS, ZDO) including a discussion on Zigbee endpoints, clusters, and profiles. Much of the material in these topics discuss details of the Zigbee stack that are not required in many cases.

Read these topics if:

- The XBee/XBee-PRO Zigbee RF Module may talk to non-Digi Zigbee devices.
- The XBee/XBee-PRO Zigbee RF Module requires network management and discovery capabilities of the ZDO layer.
- The XBee/XBee-PRO Zigbee RF Module needs to operate in a public application profile (for example, smart energy, home automation, and so on).

Skip these topics if:

- The XBee/XBee-PRO Zigbee RF Module does not need to interoperate or talk to non-Digi Zigbee devices.
- The XBee/XBee-PRO Zigbee RF Module simply needs to send data between devices.

Application Support Sublayer (APS)

The APS layer in Zigbee adds support for application profiles, cluster IDs, and endpoints.

Application profiles

Application profiles specify various device descriptions including required functionality for various devices. The collection of device descriptions forms an application profile. Application profiles are defined as Public or Private profiles. Private profiles are defined by a manufacturer whereas public profiles are defined, developed, and maintained by the Zigbee Alliance. Each application profile has a unique profile identifier assigned by the Zigbee Alliance.

Examples of public profiles include:

- Home automation
- Smart Energy
- Commercial building automation

For example, the Smart Energy profile defines various device types including an energy service portal, load controller, thermostat, in-home display, and so on. The Smart Energy profile defines required functionality for each device type. For example, a load controller must respond to a defined command to turn a load on or off. By defining standard communication protocols and device functionality, public profiles allow interoperable Zigbee solutions to be developed by independent manufacturers.

Digi XBee Zigbee firmware operates on a private profile called the Digi Drop-In Networking profile. However, in many cases the XBee/XBee-PRO Zigbee RF Module can use API mode to talk to devices in public profiles or non-Digi private profiles. For more information, see API Operation.

Clusters

A cluster is an application message type defined within a profile. You can use clusters to specify a unique function, service, or action. The following examples are some clusters defined in the home automation profile:

- On/Off - Used to switch devices on or off (lights, thermostats, and so forth)
- Level Control - Used to control devices that can be set to a level between on and off
- Color Control - Controls the color of color capable devices

Each cluster has an associated 2-byte cluster identifier (cluster ID). All application transmissions include the cluster ID. Clusters often have associated request and response messages. For example, a smart energy gateway (service portal) might send a load control event to a load controller in order to schedule turning on or off an appliance. Upon executing the event, the load controller sends a load control report message back to the gateway.

Devices that operate in an application profile (private or public) must respond correctly to all required clusters. For example, a light switch that operates in the home automation public profile must correctly implement the On/Off and other required clusters in order to interoperate with other home automation devices. The Zigbee Alliance has defined a Zigbee cluster library (ZCL) that contains definitions or various general use clusters that could be implemented in any profile.

XBee modules implement various clusters in the Digi private profile. You can also use the API to send or receive messages on any cluster ID (and profile ID or endpoint). For more information, see Explicit Receive Indicator - 0x91.

Endpoints

The APS layer includes supports for endpoints. An endpoint can be thought of as a running application, similar to a TCP/IP port. A single device can support one or more endpoints. A 1- byte value identifies each application endpoint, ranging from 1 to 240. Each defined endpoint on a device is tied to an application profile. A device could, for example, implement one endpoint that supports a Smart Energy load controller, and another endpoint that supports other functionality on a private profile.

No TX Status frame is generated for API frames that have both **0xE6** as the destination endpoint and **0xC105** as the Profile ID as this combination is reserved for internal XBee/XBee-PRO Zigbee RF Module operations.

Zigbee device profile

Profile ID 0x0000 is reserved for the Zigbee device profile. This profile is implemented on all Zigbee devices. Device Profile defines many device and service discovery features and network management capabilities. Endpoint 0 is a reserved endpoint that supports the Zigbee device profile. This endpoint is called the Zigbee device objects (ZDO) endpoint.

Zigbee device objects

The ZDO (endpoint 0) supports the discovery and management capabilities of the Zigbee device profile. See the Zigbee specification for a complete listing of all ZDP services. Each service has an associated cluster ID.

The XBee Zigbee firmware allows applications to easily send ZDO messages to devices in the network using the API. For more information, see ZDO transmissions.

Zigbee coordinator operation

Form a network

The coordinator is responsible for selecting the channel, PAN ID, security policy, and stack profile for a network. Since a coordinator is the only device type that can start a network, each Zigbee network must have one coordinator. After the coordinator has started a network, it can allow new devices to join the network. It can also route data packets and communicate with other devices on the network.

To ensure the coordinator starts on a good channel and unused PAN ID, the coordinator performs a series of scans to discover any RF activity on different channels (energy scan) and to discover any nearby operating PANs (PAN scan). The process for selecting the channel and PAN ID are described in the following topics.

Security policy

The security policy determines which devices are allowed to join the network, and which device(s) can authenticate joining devices. See Zigbee security for a detailed discussion of various security policies.

Channel selection

When starting a network, the coordinator must select a "good" channel for the network to operate on. To do this, it performs an energy scan on multiple channels (that is, frequencies) to detect energy levels on each channel. The coordinator removes channels with excessive energy levels from its list of potential channels to start on.

PAN ID selection

After completing the energy scan, the coordinator scans its list of potential channels (remaining channels after the energy scan) to obtain a list of neighboring PANs. To do this, the coordinator sends a beacon request (broadcast) transmission on each potential channel. All nearby coordinators and routers that have already joined a Zigbee network respond to the beacon request by sending a beacon back to the coordinator. The beacon contains information about which PAN the device is on, including the PAN identifiers (16-bit and 64-bit). This scan (collecting beacons on the potential channels) is typically called an active scan or PAN scan.

After the coordinator completes the channel and PAN scan, it selects a random channel and unused 16-bit PAN ID to start on.

Persistent data

Once a coordinator starts a network, it retains the following information through power cycle or reset events:

- PAN ID
- Operating channel
- Security policy and frame counter value
- Child table (end device children that are joined to the coordinator)
- Binding table
- Group table

The coordinator retains this information indefinitely until it leaves the network. When the coordinator leaves a network and starts a new network, the previous PAN ID, operating channel, link key table, and child table data are lost.

Coordinator startup

The following table provides the network formation commands that the coordinator uses to form a network.

Command	Description
ID	Used to determine the 64-bit PAN ID. If set to 0 (default), a random 64-bit PAN ID will be selected.
SC	Determines the scan channels bitmask (up to 16 channels) used by the coordinator when forming a network. The coordinator will perform an energy scan on all enabled **SC** channels. It will then perform a PAN ID scan.
SD	Set the scan duration, or time that the router will listen for beacons on each channel.
ZS	Set the Zigbee stack profile for the network.
EE	Enable or disable security in the network.
NK	Set the network security key for the network. If set to 0 (default), the coordinator uses a random network security key.
KY	Set the trust center link key for the network. If set to 0 (default), the coordinator uses a random link key.
EO	Set the security policy for the network.

Configuration changes delay the start of network formation for five seconds after the last change.

Once the coordinator starts a network, the network configuration settings and child table data persist through power cycles as mentioned in Persistent data.

When the coordinator has successfully started a network, it:

- Allows other devices to join the network for a time; see NJ (Node Join Time)
- Sets **AI** = **0**
- Starts blinking the Associate LED
- Sends an API modem status frame ("coordinator started") out the serial port when using API mode

These behaviors are configurable using the following commands:

Command	Description
NJ	Sets the permit-join time on the coordinator, measured in seconds.
D5	Enables the Associate LED functionality.
LT	Sets the Associate LED blink time when joined. If **LT** = **0**, the default is 1 blink per 500 ms (coordinator) 250 ms (router/end device).

If any of the command values in the network formation commands table changes, the coordinator leaves its current network and starts a new network, possibly on a different channel.

Note Command changes must be applied (**AC** or **CN** command) before taking effect.

Permit joining

You can use NJ (Node Join Time) to configure the permit joining attribute on the coordinator. You can configure **NJ** to always allow joining, or to allow joining for a short time.

Joining temporarily enabled

Set **NJ** < **0xFF**, to enable joining for only a number of seconds, based on the **NJ** parameter. Once the XBee/XBee-PRO Zigbee RF Module joins a network, the timer starts. The coordinator does not re-enable joining if the device is power cycled or reset. The following actions restart the permit-joining timer:

- Changing **NJ** to a different value (and applying changes with the **AC** or **CN** commands).
- Pressing the Commissioning button twice.
- Issuing the **CB** command with a parameter of **2**.

The last two actions enable joining for one minute if **NJ** is **0x0** or **0xFF**. Otherwise, the Commissioning button and the CB2 command enable joining for **NJ** seconds.

Joining always enabled

If **NJ** = **0xFF**, joining is permanently enabled.

Use this mode carefully. Once a network has been deployed, we strongly recommend that the application consider disabling joining to prevent unwanted joins from occurring. An always-open network operates outside of the Zigbee 3.0 specifications.

Reset the coordinator

When you reset or power cycle the coordinator, it checks its PAN ID, operating channel and stack profile against the network configuration settings (**ID**, **CH**, **ZS**). It also verifies the saved security policy against the security configuration settings (**EE**, **NK**, **KY**). If the coordinator's PAN ID, operating channel, stack profile, or security policy is not valid based on its network and security configuration settings, the coordinator leaves the network and attempts to form a new network based on its network formation command values.

To prevent the coordinator from leaving an existing network, issue the **WR** command after all network formation commands have been configured in order to retain these settings through power cycle or reset events.

Leave a network

The following mechanisms cause the coordinator to leave its current PAN and start a new network based on its network formation parameter values.

- Change the **ID** command such that the current 64-bit PAN ID is invalid.
- Change the **SC** command such that the current channel (**CH**) is not included in the channel mask.
- Change the **ZS** or any of the security command values.
- Send the **NR0** command to cause the coordinator to leave.
- Send the **NR1** command to send a broadcast transmission, causing all devices in the network to leave and migrate to a different channel.
- Press the commissioning button four times or send the **CB** command with a parameter of **4**. This restores the device to a default configuration state.
- Send a network leave command.

Note Changes to **ID**, **SC**, **ZS**, and security command values only take effect when changes are applied (**AC** or **CN** commands).

Replace a coordinator (security disabled only)

On rare occasions, it may become necessary to replace an existing coordinator in a network with a new physical device. If security is not enabled in the network, you can configure a replacement XBee coordinator with the PAN ID (16-bit and 64-bit), channel, and stack profile settings of a running network in order to replace an existing coordinator.

Note Avoid having two coordinators on the same channel, stack profile, and PAN ID (16-bit and 64-bit) as it can cause problems in the network. When replacing a coordinator, turn off the old coordinator before starting the new coordinator.

To replace a coordinator, read the following commands from a device on the network:

Command	Description
OP	Read the operating 64-bit PAN ID.
OI	Read the operating 16-bit PAN ID.
CH	Read the operating channel.
ZS	Read the stack profile.

Each of the commands listed above can be read from any device on the network. These parameters will be the same on all devices in the network. After reading the commands from a device on the network, program the parameter values into the new coordinator using the following commands.

Command	Description
ID	Set the 64-bit PAN ID to match the read **OP** value.
II	Set the initial 16-bit PAN ID to match the read **OI** value.
SC	Set the scan channels bitmask to enable the read operating channel (**CH** command). For example, if the operating channel is 0x0B, set **SC** to 0x0001. If the operating channel is 0x17, set **SC** to 0x1000.
ZS	Set the stack profile to match the read **ZS** value.

II is the initial 16-bit PAN ID. Under certain conditions, the Zigbee stack can change the 16-bit PAN ID of the network. For this reason, you cannot save the **II** command using the **WR** command. Once **II** is set, the coordinator leaves the network and starts on the 16-bit PAN ID specified by **II**.

Example: start a coordinator

1. Set CE (Coordinator Enable) to 1 or load coordinator firmware on the device, and use the **WR** command to save the changes.
2. Set **SC** and **ID** to the desired scan channels and PAN ID values. The defaults are sufficient.

3. If you change **SC** or **ID** from the default, issue the **WR** command to save the changes.

4. If you change **SC** or **ID** from the default, apply changes (make **SC** and **ID** changes take effect) either by sending the **AC** command or by exiting **AT** command mode.

5. If an Associate LED has been connected, it starts blinking once the coordinator has selected a channel and PAN ID.

6. The API Modem Status frame (Coordinator Started) is sent out the serial port when using API mode.

7. Reading the **AI** command (association status) returns a value of 0, indicating a successful startup.

8. Reading the **MY** command (16-bit address) returns a value of 0, the Zigbee-defined 16-bit address of the coordinator.

After startup, the coordinator allows joining based on its **NJ** value.

Example: replace a coordinator (security disabled)

1. Read the **OP**, **OI**, **CH**, and **ZS** commands on the running coordinator.

2. Set the **CE**, **ID**, **SC**, and **ZS** parameters on the new coordinator to match the existing coordinator, followed by **WR** command to save these parameter values.

3. Turn off the running coordinator.

4. Set the II (Initial 16-bit PAN ID) parameter on the new coordinator to match the read **OI** value on the old coordinator.

5. Wait for the new coordinator to start (**AI** = **0**).

Router operation

Routers must discover and join a valid Zigbee network before they can participate in a Zigbee network. After a router has joined a network, it can allow new devices to join the network. It can also route data packets and communicate with other devices on the network.

Discover Zigbee networks

To discover nearby Zigbee networks, the router performs a PAN (or active) scan, just like the coordinator does when it starts a network. During the PAN scan, the router sends a beacon request (broadcast) transmission on the first channel in its scan channels list. All nearby coordinators and routers operating on that channel that are already part of a Zigbee network respond to the beacon request by sending a beacon back to the router.

The beacon contains information about the PAN the nearby device is on, including the PAN identifier (PAN ID), and whether or not joining is allowed. The router evaluates each beacon received on the channel to determine if it finds a valid PAN. A PAN is valid if any of the following exist:

- Has a valid 64-bit PAN ID (PAN ID matches **ID** if **ID** > 0)
- Has the correct stack profile (**ZS** command)
- Allows joining the network

If the router does not find a valid PAN, it performs the PAN scan on the next channel in its scan channels list and continues scanning until it finds a valid network, or until all channels have been scanned. If the rounter scans all channels and does not discover a valid PAN, it scans all channels again.

The Zigbee Alliance® requires that certified solutions not send beacon request messages too frequently. To meet certification requirements, the XBee firmware attempts nine scans per minute for the first five minutes, and three scans per minute thereafter. If a valid PAN is within range of a joining router, it typically discovers the PAN within a few seconds.

Join a network

Once the router discovers a valid network, it sends an association request to the device that sent a valid beacon requesting a join on the Zigbee network. The device allowing the join then sends an association response frame that either allows or denies the join.

When a router joins a network, it receives a 16-bit address from the device that allowed the join. The device that allowed the join randomly selects the 16-bit address.

Authentication

In a network where security is enabled, the router must follow an authentication process. See Zigbee security for a discussion on security and authentication.

After the router is joined (and authenticated, in a secure network), it can allow new devices to join the network.

Persistent data

Once a router joins a network, it retains the following information through power cycle or reset events:

- PAN ID
- Operating channel
- Security policy and frame counter values
- Child table (end device children that are joined to the coordinator)
- Binding table
- Group table

The router retains this information indefinitely until it leaves the network. When the router leaves a network, it loses the previous PAN ID, operating channel, and child table data.

Zigbee router joining

When the router powers on, if it is not already joined to a valid Zigbee network, it immediately attempts to find and join a valid Zigbee network.

Set DJ (Disable Joining) to **1** to disable joining. You cannot write the **DJ** parameter with the **WR** command, so a power cycle always clears the **DJ** setting.

The following commands control the router joining process.

Command	Description
ID	Sets the 64-bit PAN ID to join. Setting **ID** = **0** allows the router to join any 64-bit PAN ID.

Command	Description
SC	Set the scan channels bitmask that determines which channels a router scans to find a valid network. Set **SC** on the router to match **SC** on the coordinator. For example, setting **SC** to 0x281 enables scanning on channels 0x0B, 0x12, and 0x14, in that order.
SD	Set the scan duration, or time that the router listens for beacons on each channel.
ZS	Set the stack profile on the device.
EE	Enable or disable security in the network. This must be set to match the **EE** value (security policy) of the coordinator.
KY	Set the trust center link key. If set to 0 (default), the link key is expected to be obtained (unencrypted) during joining.

Configuration changes delay the start of joining for five seconds after the last change.

Once the router joins a network, the network configuration settings and child table data persist through power cycles as mentioned in Persistent data. If joining fails, read the status of the last join attempt in the **AI** command register.

If any of the above command values change, when command register changes are applied (**AC** or **CN** commands), the router leaves its current network and attempts to discover and join a new valid network. When a Zigbee router has successfully joined a network, it:

- Allows other devices to join the network for a time
- Sets **AI** = **0**
- Starts blinking the Associate LED
- Sends an API modem status frame (associated) out the serial port when using API mode

You can configure these behaviors using the following commands:

Command	Description
NJ	Sets the permit-join time on the router, or the time that it allows new devices to join the network, measured in seconds. Set **NJ** = **0xFF** to always enable permit joining.
D5	Enables the Associate LED functionality.
LT	Sets the Associate LED blink time when joined. The default is 2 blinks per second (router).

Permit joining

You can use NJ (Node Join Time) to configure the permit joining attribute on the coordinator. You can configure **NJ** to always allow joining, or to allow joining for a short time.

Joining temporarily enabled

Set **NJ** < **0xFF**, to enable joining for only a number of seconds, based on the **NJ** parameter. Once the XBee/XBee-PRO Zigbee RF Module joins a network, the timer starts. The coordinator does not re-enable joining if the device is power cycled or reset. The following actions restart the permit-joining timer:

- Changing **NJ** to a different value (and applying changes with the **AC** or **CN** commands).
- Pressing the Commissioning button twice.
- Issuing the **CB** command with a parameter of **2**.

The last two actions enable joining for one minute if **NJ** is **0x0** or **0xFF**. Otherwise, the Commissioning button and the CB2 command enable joining for **NJ** seconds.

Joining always enabled

If **NJ** = **0xFF**, joining is permanently enabled.

> Use this mode carefully. Once a network has been deployed, we strongly recommend that the application consider disabling joining to prevent unwanted joins from occurring. An always-open network operates outside of the Zigbee 3.0 specifications.

Router network connectivity

Once a router joins a Zigbee network, it remains connected to the network on the same channel and PAN ID unless it is forced to leave (see Leave a network). If the scan channels (**SC**), PAN ID (**ID**) and security settings (**EE**, **KY**) do not change after a power cycle, the router remains connected to the network after a power cycle.

If a router is physically moved out of range of the network it initially joined, make sure the application includes provisions to detect if the router can still communicate with the original network. If communication with the original network is lost, the application may choose to force the router to leave the network. The XBee firmware includes two provisions to automatically detect the presence of a network and leave if the check fails.

Power-On join verification

JV (Coordinator Join Verification) enables the power-on join verification check. If enabled, the XBee/XBee-PRO Zigbee RF Module attempts to discover the 64-bit address of the coordinator when it first joins a network. Once it has joined, it also attempts to discover the 64-bit address of the coordinator after a power cycle event. If 3 discovery attempts fail, the router leaves the network and try to join a new network. The default setting for Power-on join verification is disabled (**JV** defaults to 0).

Network watchdog

The NW (Network Watchdog Timeout) feature allows a powered router to verify the presence of a coordinator if no communication events with the coordinator have occurred within a timeout period. This timeout is specified in minutes using the **NW** command. The default setting for the network watchdog feature is disabled (**NW** = **0**) and can be configured for up to several days.

Anytime a router receives valid data from the coordinator or data collector, it clears the watchdog timeouts counter and restarts the watchdog timer.

The following events restart the network watchdog timer:

- RF data received from the coordinator
- RF data sent to the coordinator and an acknowledgment was received
- Many-to-one route request was received (from any device)
- Change the value of **NW**

If any of the events listed above occur during the watchdog period then no additional network traffic will be generated. If the watchdog timer does expire (no valid data received for 1 **NW** time period), the router attempts to initiate communication with the coordinator by sending an IEEE 64-bit address discovery message to the coordinator. If the router cannot discover the address, it records one watchdog timeout. After three consecutive network watchdog timeouts expire (3 * **NW**) and the coordinator has not responded to the address discovery attempts, the router will enter one of three modes based on the configuration of DC (Joining Device Controls) bit 5 and DO (Miscellaneous Device Options) bit 7:

1. **No Network Locator (Leave Network)**: If neither **DC** bit 5 or **DO** bit 7 is set, the router will immediately leave the network and begin searching for a network to join based on its networking settings. If API mode is enabled, a network disassociated modem status frame (0x03) will be emitted when the router leaves the network. If the router finds and joins a new coordinator or the original coordinator, a joined network modem status frame (0x02) will be emitted if API mode is enabled.

2. **Network Locator with Network Leave**: If **DO** bit 7 is set but **DC** bit 5 is not set, the behavior of **JV** and **NW** are modified. The router will remain on the network until a new network is found. The router starts scanning for a network across the channels of the Scan Channel mask (SC). Scanning occurs at a random interval of between 90 and 135 seconds. If API mode is enabled, a network watchdog scanning modem status frame (0x42) will be emitted when scanning begins. If the device finds a network on the old channel with the same **OI** and operating **ID**, the search mode ends. If the device finds a network with a new **OI** but satisfies the device's search for a matching **ID** and **ZS**, the device leaves the old network and joins the new network with the new **OI**. These leave and join actions will cause the router to emit a disassociated network modem status (0x03) and a joined network modem status frame (0x02) if API mode is enabled. This supports swapping or replacing a coordinator in a running network.

3. **Network Locator with Rejoin**: If **DC** bit 5 is set, the router will begin scanning the current channel indefinitely in an attempt to find the coordinator on the original network or re-join the coordinator if it has moved to a new network. If API mode is enabled, a network watchdog scanning modem status frame (0x42) will be emitted when scanning begins. Scanning occurs at a random interval of between 90 and 135 seconds. If it finds the coordinator which must have a matching **ID** (extended PAN ID) with the same PAN ID (**OI**) or a new PAN ID, the router will rejoin the coordinator even if the coordinator is configured with a **NJ** of **0** (joining disabled). If the router finds the coordinator on the original network or rejoins coordinator on a new network, a joined network modem status frame (0x02) will be emitted if API mode is enabled.

Leave a network

The following mechanisms cause the coordinator to leave its current PAN and start a new network based on its network formation parameter values.

- Change the **ID** command such that the current 64-bit PAN ID is invalid.
- Change the **SC** command such that the current channel (**CH**) is not included in the channel mask.
- Change the **ZS** or any of the security command values.
- Send the **NR0** command to cause the coordinator to leave.
- Send the **NR1** command to send a broadcast transmission, causing all devices in the network to leave and migrate to a different channel.

- Press the commissioning button four times or send the **CB** command with a parameter of **4**. This restores the device to a default configuration state.
- Send a network leave command.

Note Changes to **ID**, **SC**, **ZS**, and security command values only take effect when changes are applied (**AC** or **CN** commands).

Reset the Router

When you reset or power cycle the router, it checks its PAN ID, operating channel and stack profile against the network configuration settings (**ID**, **SC**, **ZS**). It also verifies the saved security policy is valid based on the security configuration commands (**EE**, **KY**). If the router's PAN ID, operating channel, stack profile, or security policy is invalid, the router leaves the network and attempts to join a new network based on its network joining command values.

To prevent the router from leaving an existing network, issue the **WR** command after all network joining commands have been configured; this retains the settings through power cycle or reset events.

Example: join a network

After starting a coordinator that is allowing joins, the following steps cause a router to join the network:

1. Set **ID** to the desired 64-bit PAN ID, or to 0 to join any PAN.
2. Set **SC** to the list of channels to scan to find a valid network.
3. Set the security settings to match the coordinator.
4. If you **SC** or **ID** from the default, apply changes (that is, make **SC** and **ID** changes take effect) by issuing the **AC** or **CN** command.
5. The Associate LED starts blinking once the router has joined a PAN.
6. If the Associate LED is not blinking, read the **AI** command to determine the cause of join failure.
7. Once the router joins, the **OP** and **CH** commands indicate the operating 64-bit PAN ID and channel the router joined.
8. The **MY** command reflects the 16-bit address the router received when it joined.
9. The API Modem Status frame ("Associated") is sent out the serial port when using API mode.
10. The joined router allows other devices to join for a time based on its **NJ** setting.

End device operation

Similar to routers, end devices must discover and join a valid Zigbee network before they can participate in the network. After an end device joins a network, it can communicate with other devices on the network. Because end devices are battery powered and support low power (sleep) modes, they cannot allow other devices to join or route data packets.

Discover Zigbee networks

End devices go through the same process as routers to discover networks by issuing a PAN scan. After sending the broadcast beacon request transmission, the end device listens for a short time in order to receive beacons sent by nearby routers and coordinators on the same channel. The end device evaluates each beacon received on the channel to determine if it finds a valid PAN. A PAN is valid if any of the following exist:

- Has a valid 64-bit PAN ID (PAN ID matches ID if **ID** > **0**)
- Has the correct stack profile (**ZS** command)
- Allows joining the network
- Has capacity for additional end devices

If the end device does not find a valid PAN, it performs the PAN scan on the next channel in its scan channels list and continues this process until it finds a valid network, or until all channels have been scanned. If the end device scan all channels and does not discover a valid PAN, it may enter a low power sleep state and scan again later.

If scanning all **SC** channels fails to discover a valid PAN, XBee Zigbee devices attempt to enter a low power state and retries scanning all **SC** channels after the device wakes from sleeping. If the device cannot enter a low power state, it retries scanning all channels, similar to the router. To meet Zigbee Alliance requirements, the end device attempts up to nine scans per minute for the first five minutes, and three scans per minute thereafter.

Note The XBee Zigbee end device will not enter sleep until it has completed scanning all **SC** channels for a valid network.

Join a network

Once the end device discovers a valid network, it joins the network, similar to a router, by sending an association request (to the device that sent a valid beacon) to request a join on the Zigbee network. The device allowing the join then sends an association response frame that either allows or denies the join.

When an end device joins a network, it receives a 16-bit address from the device that allowed the join. The device that allowed the join randomly selects the 16-bit address.

Parent child relationship

Since an end device may enter low power sleep modes and not be immediately responsive, the end device relies on the device that allowed the join to receive and buffer incoming messages on its behalf until it is able to wake and receive those messages. The device that allowed an end device to join becomes the parent of the end device, and the end device becomes a child of the device that allowed the join.

End device capacity

Routers and coordinators maintain a table of all child devices that have joined called the child table. This table is a finite size and determines how many end devices can join. If a router or coordinator has at least one unused entry in its child table, the device has end device capacity. In other words, it can allow one or more additional end devices to join. Zigbee networks have sufficient routers to ensure adequate end device capacity.

The initial release of software on this platform supports up to 20 end devices when configured as a coordinator or a router.

In Zigbee firmware, use the **NC** command (number of remaining end device children) to determine how many additional end devices can join a router or coordinator. If **NC** returns 0, then the router or coordinator device has no more end device capacity.

Note Because routers cannot sleep, there is no equivalent need for routers or coordinators to track joined routers. There is no limit to the number of routers that can join a given router or coordinator device and no "router capacity" metric.

Authentication

In a network where security is enabled, the end device must then go through an authentication process. For more information, see Zigbee security.

Persistent data

The end device can retain its PAN ID, operating channel, and security policy information through a power cycle. However, since end devices rely heavily on a parent, the end device does an orphan scan to try and contact its parent. If the end device does not receive an orphan scan response (coordinator realignment command), it leaves the network and tries to discover and join a new network. When the end device leaves a network, it loses the previous PAN ID and operating channel settings.

Orphan scans

When an end device comes up from a power cycle, it performs an orphan scan to verify it still has a valid parent. The device sends the orphan scan as a broadcast transmission and contains the 64-bit address of the end device. Nearby routers and coordinator devices that receive the broadcast check their child tables for an entry that contains the end device's 64-bit address. If the devices find an entry with a matching 64-bit address, they send a coordinator realignment command to the end device that includes the 16-bit address of the end device, 16-bit PAN ID, operating channel, and the parent's 64-bit and 16-bit addresses.

If the orphaned end device receives a coordinator realignment command, it joins the network. Otherwise, it attempts to discover and join a valid network.

End device joining

When you power on an end device, if it is not joined to a valid Zigbee network, or if the orphan scan fails to find a parent, the device attempts to find and join a valid Zigbee network.

Note Set the **DJ** command to 1 to disable joining. You cannot write the **DJ** parameter with **WR**, so a power cycle always clears the **DJ** setting.

The following commands control the end device joining process.

Command	Description
ID	Sets the 64-bit PAN ID to join. Setting **ID** = **0** allows the router to join any 64-bit PAN ID.
SC	Set the scan channels bitmask that determines which channels an end device will scan to find a valid network. **SC** on the end device should be set to match SC on the coordinator and routers in the desired network. For example, setting **SC** to **0x281** enables scanning on channels 0x0B, 0x12, and 0x14, in that order.
SD	Set the scan duration, or time that the end device will listen for beacons on each channel.

Command	Description
ZS	Set the stack profile on the device.
EE	Enable or disable security in the network. This must be set to match the **EE** value (security policy) of the coordinator.
KY	Set the trust center link key. If set to 0 (default), the link key is expected to be obtained (unencrypted) during joining.

Once the end device joins a network, the network configuration settings persist through power cycles as mentioned in Persistent data. If joining fails, read the status of the last join attempt in the **AI** command register.

If any of these command values change when command register changes are applied, the end device leaves its current network and attempts to discover and join a new valid network.

When a Zigbee end device has successfully started a network, it:

- Sets **AI** equal to **0**
- Starts blinking the Associate LED if one has been connected to the device's ASSC pin (TH pin 15/SMT pin 28)
- Sends an API modem status frame ("associated") out the serial port when using API mode
- Attempts to enter the sleep mode defined by the **SM** parameter

You can use the following commands to configure these behaviors:

Command	Description
D5	Enables the Associate LED functionality.
LT	Sets the Associate LED blink time when joined. Default is 2 blinks per second (end devices).
SM, **SP**, **ST**, **SN**, **SO**	Parameters that configure the sleep mode characteristics. See End Device configuration.

Parent connectivity

The XBee/XBee-PRO Zigbee RF Module end device sends regular poll transmissions to its parent when it is awake. These poll transmissions query the parent for any new received data packets. The parent always sends a MAC layer acknowledgment back to the end device. The acknowledgment indicates whether the parent has data for the end device.

If the end device does not receive an acknowledgment for three consecutive poll requests, it considers itself disconnected from its parent and attempts to discover and join a valid Zigbee network. For more information, see Manage End Devices.

Reset the end device

When the end device is reset or power cycled, if the orphan scan successfully locates a parent, the end device then checks its PAN ID, operating channel and stack profile against the network configuration settings (**ID**, **SC**, **ZS**). It also verifies the saved security policy is valid based on the security configuration commands (**EE**, **EO**, **KY**). If the end device's PAN ID, operating channel, stack profile, or

security policy is invalid, the end device will leave the network and attempt to join a new network based on its network joining command values.

To prevent the end device from leaving an existing network, the **WR** command should be issued after all network joining commands have been configured in order to retain these settings through power cycle or reset events.

Leave a network

The following mechanisms cause the coordinator to leave its current PAN and start a new network based on its network formation parameter values.

- Change the **ID** command such that the current 64-bit PAN ID is invalid.
- Change the **SC** command such that the current channel (**CH**) is not included in the channel mask.
- Change the **ZS** or any of the security command values (excluding **NK**).
- Send the **NR0** command to cause the coordinator to leave.
- Send the **NR1** command to send a broadcast transmission, causing all devices in the network to leave and migrate to a different channel.
- Press the commissioning button four times or issue the **CB** command with a parameter of 4.
- The parent of the end device parent powers down or the end device moves out of range of the parent such that the end device fails to receive poll acknowledgment messages.

Note Changes to command values only take effect when changes are applied (**AC** or **CN** commands).

Example: join a network

After starting a coordinator that is allowing joins, the following steps cause a router to join the network:

1. Set **ID** to the desired 64-bit PAN ID, or to 0 to join any PAN.
2. Set **SC** to the list of channels to scan to find a valid network.
3. If you **SC** or **ID** from the default, apply changes (that is, make **SC** and **ID** changes take effect) by issuing the **AC** or **CN** command.
4. The Associate LED starts blinking once the router has joined a PAN.
5. If the Associate LED is not blinking, read the **AI** command to determine the cause of join failure.
6. Once the router joins, the **OP** and **CH** commands indicate the operating 64-bit PAN ID and channel the end device joined.
7. The **MY** command reflects the 16-bit address the router received when it joined.
8. The API Modem Status frame ("Associated") is sent out the serial port when using API mode.
9. The joined end device attempts to enter low power sleep modes based on its sleep configuration commands (**SM**, **SP**, **SN**, **ST**, **SO**).

Zigbee channel scanning

Routers and end devices must scan one or more channels to discover a valid network to join. When a join attempt begins, the device sends a beacon request transmission on the lowest channel specified in the SC (Scan Channels) bitmask. If the device finds a valid PAN on the channel, it attempts to join

the PAN on that channel. Otherwise, if the device does not find a valid PAN on the channel, it attempts scanning on the next higher channel in the **SC** bitmask.

The device continues to scan each channel (from lowest to highest) in the **SC** bitmask until it finds a valid PAN or all channels have been scanned. Once the device scans all channels, the next join attempt starts scanning on the lowest channel specified in the **SC** bitmask.

For example, if the **SC** command is set to **0x400F**, the device starts scanning on channel 11 (**0x0B**) and scans until it finds a valid beacon, or until it scans channels 11, 12, 13, 14, and 25 have been scanned (in that order).

Once an XBee router or end device joins a network on a given channel, if the XBee device receives a network leave command (see Leave a network), it leaves the channel it joined on and continues scanning on the next higher channel in the **SC** bitmask.

For example, if the **SC** command is set to **0x400F** and the device joins a PAN on channel 12 (**0x0C**), if the XBee/XBee-PRO Zigbee RF Module leaves the channel, it starts scanning on channel 13, followed by channels 14 and 25 if it does not find a valid network. Once all channels have been scanned, the next join attempt starts scanning on the lowest channel specified in the **SC** bitmask.

Manage multiple Zigbee networks

In some applications, multiple Zigbee networks may exist in proximity of each other. The application may need provisions to ensure the device joins the desired network. There are a number of features in Zigbee to manage joining among multiple networks. These include the following:

- PAN ID filtering
- Preconfigured security keys
- Permit joining
- Application messaging

Filter PAN ID

Set ID (Extended PAN ID) to a non-zero value to configure the XBee/XBee-PRO Zigbee RF Module with a fixed PAN ID.

If you set the PAN ID to a non-zero value, the device will only join a network with the same PAN ID.

Configure security keys

Similar to PAN ID filtering, this method requires that you install a known security key on a router to ensure it joins a Zigbee network with the same security key.

1. Use EE (Encryption Enable) to enable security.
2. Use KY (Link Key) to set the preconfigured link key to a non-zero value.

Now the XBee router or end device will only join a network with the same security key.

Prevent unwanted devices from joining

You can disable the permit-joining parameter in a network to prevent unwanted devices from joining. When you need to add a new device to a network, enable permit-joining for a short time on the desired network.

In the XBee firmware:

1. Set NJ (Node Join Time) to a value less than **0xFF** on all routers and coordinator devices to restrict joining (recommended).
2. Use the Commissioning pushbutton or CB (Commissioning Pushbutton) to allow joining for a short time; for more information, see Network commissioning and diagnostics .

Application messaging framework

If none of the previous mechanisms are feasible, you can build a messaging framework between the coordinator and devices that join its network into the application. For example, the application code in joining devices could send a transmission to the coordinator after joining a network, and wait to receive a defined reply message. If the application does not receive the expected response message after joining, it could force the device to leave and continue scanning; see NR (Network Reset).

Transmission, addressing, and routing

Addressing

All Zigbee devices have two different addresses, a 64-bit and a 16-bit address. This section describes the characteristics of each.

64-bit device addresses

The 64-bit address is a device address which is unique to each physical device. It is sometimes also called the MAC address or extended address and is assigned during the manufacturing process. The first three bytes of the 64-bit address is a Organizationally Unique Identifier (OUI) assigned to the manufacturer by the IEEE. The OUI of XBee devices is 0x0013A2.

16-bit device addresses

A device receives a 16-bit address when it joins a Zigbee network. For this reason, the 16-bit address is also called the network address. The 16-bit address of 0x0000 is reserved for the coordinator. All other devices receive a randomly generated address from the router or coordinator device that allows the join. The 16-bit address can change under certain conditions:

- An address conflict is detected where two devices are found to have the same 16-bit address
- A device leaves the network and later joins (it can receive a different address)

All Zigbee transmissions are sent using the source and destination 16-bit addresses. The routing tables on Zigbee devices also use 16-bit addresses to determine how to route data packets through the network. However, since the 16-bit address is not static, it is not a reliable way to identify a device.

To solve this problem, the 64-bit destination address is often included in data transmissions to guarantee data is delivered to the correct destination. The Zigbee stack can discover the 16-bit address, if unknown, before transmitting data to a remote.

Application layer addressing

Zigbee devices support multiple application profiles, cluster IDs, and endpoints (for more information, see Zigbee application layers: in depth). Application layer addressing allows data transmissions to be addressed to specific profile IDs, cluster IDs, and endpoints. Application layer addressing is useful if an application must do any of the following:

- Interoperate with other Zigbee devices outside of the Digi application profile.
- Use service and network management capabilities of the ZDO.
- Operate on a public application profile such as Home Automation or Smart Energy.

API mode provides a simple yet powerful interface that easily sends data to any profile ID, endpoint, and cluster ID combination on any device in a Zigbee network.
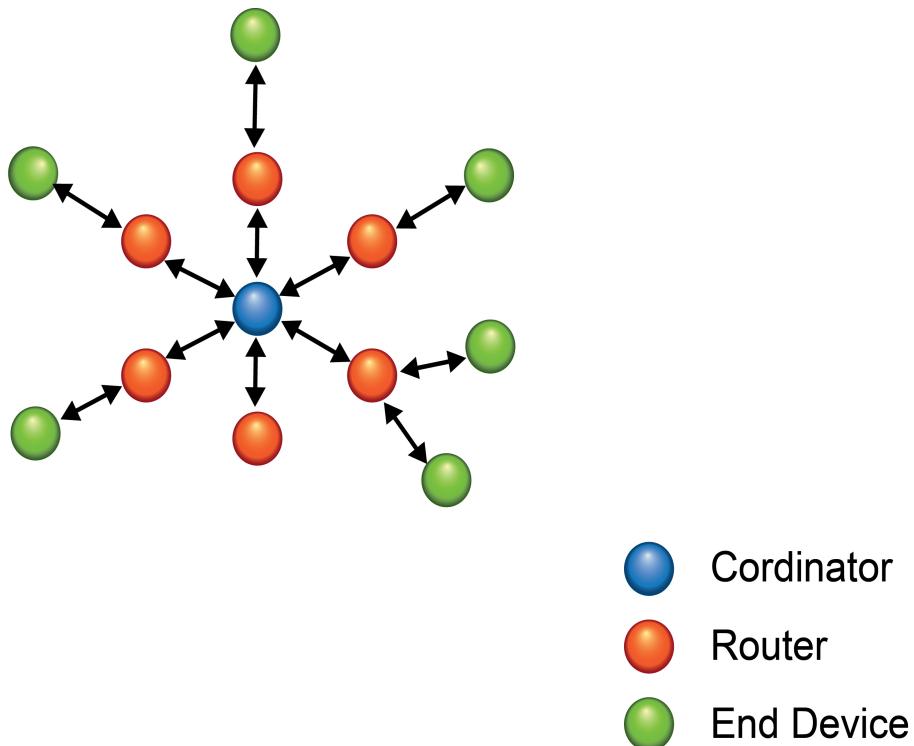
Data transmission

You can send Zigbee data packets as either unicast or broadcast transmissions. Unicast transmissions route data from one source device to one destination device, whereas broadcast transmissions are sent to many or all devices in the network.

Broadcast transmissions

Broadcast transmissions within the Zigbee protocol are intended to be propagated throughout the entire network such that all nodes receive the transmission. To accomplish this, the coordinator and all routers that receive a broadcast transmission retransmits the packet three times.

Note When a router or coordinator delivers a broadcast transmission to an end device child, the transmission is only sent once (immediately after the end device wakes and polls the parent for any new data). For more information, see Parent operation.

● Cordinator

● Router

● End Device

Each node that transmits the broadcast also creates an entry in a local broadcast transmission table. This entry to keeps track of each received broadcast packet to ensure the packets are not transmitted endlessly. Each entry persists for 8 seconds, and the broadcast transmission table holds 8 entries, effectively limiting network broadcast transmissions to once per second.

For each broadcast transmission, the Zigbee stack reserves buffer space for a copy of the data packet that retransmits the packet as needed. Large broadcast packets require more buffer space. Users cannot change any buffer spacing; information on buffer space is for general knowledge only. The XBee/XBee-PRO Zigbee RF Module handles buffer spacing automatically.

Since each device in the network retransmits broadcast transmissions, use broadcast messages sparingly to avoid network congestion.

Unicast transmissions

Unicast transmissions are sent from one source device to another destination device. The destination device could be an immediate neighbor of the source, or it could be several hops away. Unicast transmissions sent along a multiple hop path require some means of establishing a route to the destination device. For more information, see RF packet routing.

Address resolution

Each device in a Zigbee network has both a 16-bit (network) address and a 64-bit (extended) address. The 64-bit address is unique and assigned to the device during manufacturing, and the 16-bit address is obtained after joining a network. The 16-bit address can also change under certain conditions.

When sending a unicast transmission, the Zigbee network layer uses the 16-bit address of the destination and each hop to route the data packet. If you do not know the 16-bit address of the destination, the Zigbee stack includes a discovery provision to automatically discover the destination 16-bit address of the device before routing the data.

To discover a 16-bit address of a remote, the device initiating the discovery sends a broadcast address discovery transmission. The address discovery broadcast includes the 64-bit address of the remote device with the 16-bit address being requested. All nodes that receive this transmission check the 64-bit address in the payload and compare it to their own 64-bit address. If the addresses match, the device sends a response packet back to the initiator. This response includes the remote's 16-bit address. When the device receives the discovery response, the initiator transmits the data.

You can address frames using either the extended or the network address. If you use the extended address form, set the network address field to 0xFFFE (unknown). If you use the network address form, set the extended address field to 0xFFFFFFFFFFFFFFFF (unknown).

If you use an invalid 16-bit address as a destination address, and the 64-bit address is unknown (0xFFFFFFFFFFFFFFFF), the modem status message shows a delivery status code of 0x21 (network ack failure) and a discovery status of 0x00 (no discovery overhead). If you use a non-existent 64-bit address as a destination address, and the 16-bit address is unknown (0xFFFE), the device attempts address discovery and the modem status message shows a delivery status code of 0x24 (address not found) and a discovery status code of 0x01 (address discovery was attempted).

Address table

Each Zigbee device maintains an address table that maps a 64-bit address to a 16-bit address. When a transmission is addressed to a 64-bit address, the Zigbee stack searches the address table for an entry with a matching 64-bit address to determining the destination's 16-bit address. If the Zigbee stack does not find a known 16-bit address, it performs address discovery to discover the device's current 16-bit address.

64-bit address	16-bit address
0013 A200 4000 0001	0x4414
0013 A200 400A 3568	0x1234
0013 A200 4004 1122	0xC200
0013 A200 4002 1123	0xFFFE (unknown)

For the Smart Energy profiles and related firmware, the XBee devices can store up to 10 address table entries. For the standard firmware versions, the module supports up to 20 address table entries. For applications where a single device (for example, coordinator) sends unicast transmissions to more than 10 devices, the application implements an address table to store the 16-bit and 64-bit addresses for each remote device. Use API mode for any XBee device that sends data to more than 10 remotes. The application can then send both the 16-bit and 64-bit addresses to the XBee device in the API transmit frames which significantly reduces the number of 16-bit address discoveries and greatly improves data throughput.

If an application supports an address table, the size should be larger than the maximum number of destination addresses the device communicates with. Each entry in the address table should contain a 64-bit destination address and its last known 16-bit address.

When sending a transmission to a destination 64-bit address, the application searches the address table for a matching 64-bit address. If it finds a match, the application populates the 16-bit address into the 16-bit address field of the API frame. If it does not find a match, set the 16-bit address to 0xFFFE (unknown) in the API transmit frame. The API provides indication of a remote device's 16-bit address in the following frames:

- All receive data frames
- Rx Data (0x90)
- Rx Explicit Data (0x91)
- I/O Sample Data (0x92)
- Node Identification Indicator (0x95)
- Route Record Indicator (0xA1) and so forth
- Transmit status frame (0x8B)

Group table

Each router and the coordinator maintain a persistent group table. Each entry contains the following:

- Endpoint value
- Two byte group ID
- Optional name string of zero to 16 ASCII characters
- Index into the binding table

More than one endpoint may be associated with a group ID, and more than one group ID may be associated with a given endpoint. The capacity of the group table is 16 entries.

The application always updates the 16-bit address in the address table when it receives one of the frames to ensure the table has the most recently known 16-bit address. If a transmission failure occurs, the application sets the 16-bit address in the table to 0xFFFE (unknown).

Binding transmissions

Binding transmissions use indirect addressing to send one or more messages to other destination devices. The device handles an Explicit Addressing Command Request - 0x11 using the Indirect Tx Option (0x04) as a binding transmission request.

Address resolution

The XBee/XBee-PRO Zigbee RF Module use the source endpoint and cluster ID values of a binding transmission as keys to lookup matching binding table entries. For each matching binding table entry, the type field of the entry indicates whether to send a unicast or a multicast message. In the case of a unicast entry, the transmission request is updated with the Destination Endpoint and MAC Address, and unicast to its destination. In the case of a multicast entry, the device updates the message using the two least significant bytes of the Destination MAC Address as the groupID, and multicast to its destinations.

Binding table

Each router and coordinator maintain a persistent binding table to map source endpoint and cluster ID values into 64 bit destination address and endpoint values. The capacity of the binding table is 16 entries.

Multicast transmissions
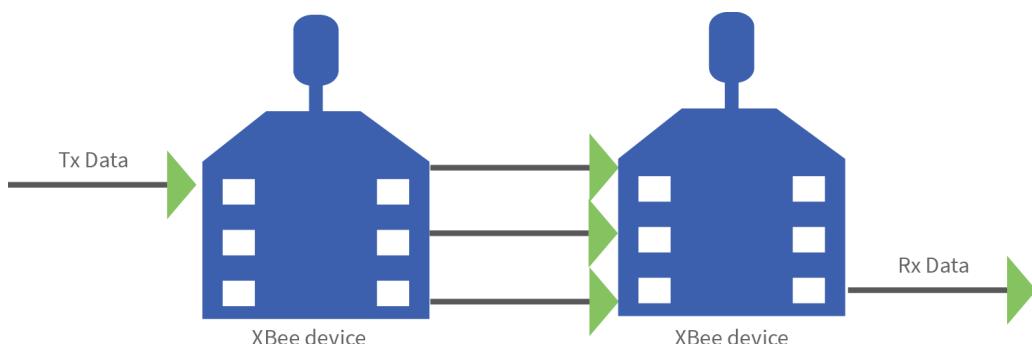
XBee modules use multicast transmissions to broadcast a message to destination devices that have active endpoints associated with a common group ID. The device handles an Explicit Addressing Command Request - 0x11 using the Multicast Tx Option (0x08) as a multicast transmission request.

Address resolution

The 64 bit destination address value does not matter and we recommend that it be set to 0xFFFFFFFFFFFFFFFF. Set the 16 bit destination address value to the destination groupID.

Fragmentation

Each unicast transmission may support up to 84 bytes of RF payload, although enabling security or using source routing can reduce this number. For more information, see NP (Maximum Packet Payload Bytes). However, the XBee Zigbee firmware supports a Zigbee feature called fragmentation that allows a single large data packet to be broken up into multiple RF transmissions and reassembled by the receiver before sending data out its serial port.

The transmit frame can include up to 255 bytes of data broken up into multiple transmissions and reassembled on the receiving side. If one or more of the fragmented messages are not received by the receiving device, it drops the entire message, and the sender indicates a transmission failure in Extended Transmit Status - 0x8B.

Applications that do not wish to use fragmentation should avoid sending more than the maximum number of bytes in a single RF transmission (see Maximum RF payload size.

If you enable RTS flow control on the receiving device (using the **D6** command) it receives a fragmented message, it ignores RTS flow control.

Note Broadcast transmissions do not support fragmentation. Maximum payload size = up to 84 bytes.

Data transmission examples

This section provides examples for data transmission using AT and API firmware.

Transparent mode

To send a data packet in Transparent mode (AP=0), set the **DH** and **DL** commands to match the 64- bit address of the destination device. **DH** must match the upper 4-bytes, and **DL** must match the lower 4 bytes. Since the coordinator always receives a 16-bit address of 0x0000, a 64-bit address of 0x0000000000000000 is the coordinator's address (in ZB firmware). The default values of **DH** and **DL** are 0x00, which sends data to the coordinator.

Example 1: Send a transmission to the coordinator.

In this example, a '\r' refers to a carriage return character.

A router or end device can send data in two ways. First, set the destination address (**DH** and **DL** commands) to 0x00.

1. Enter command mode ('+++').
2. After receiving an OK\r, issue the following commands:
    - ATDH0\r
    - ATDL0\r
    - ATCN\r
3. Verify that each of the three commands returned an OK\r response.
4. After setting these command values, all serial characters received on the UART are sent as a unicast transmission to the coordinator.

Alternatively, if the coordinator's 64-bit address is known, you can set **DH** and **DL** to the coordinator's 64-bit address. Suppose the coordinator's address is 0x0013A200404A2244.

1. Enter command mode ('+++')
2. After receiving an OK\r, issue the following commands:
    a. ATDH13A200\r
    b. ATDL404A2244\r
    c. ATCN\r
3. Verify that each of the three commands returned an OK\r response.
4. After setting these command values, all serial characters received on the UART are sent as a unicast transmission to the coordinator.

API mode

Use the transmit request, or explicit transmit request frame (0x10 and 0x11 respectively) to send data to the coordinator. The 64-bit address can either be set to 0x0000000000000000, or to the 64-bit address of the coordinator. The 16-bit address should be set to 0xFFFE when using the 64-bit address of all 0x00s.

To send an ASCII "1" to the coordinator's 0x00 address, use the following API frame:

7E 00 0F 10 01 0000 0000 0000 0000 FFFE 00 00 31 C0

If you use the explicit transmit frame, set the the cluster ID to 0x0011, the profile ID to 0xC105, and the source and destination endpoints to 0xE8. These are the recommended defaults for data transmissions in the Digi profile.

You can send the same transmission using the following explicit transmit frame:

7E 00 15 11 01 0000 0000 0000 0000 FFFE E8 E8 0011 C105 00 00 31 18

The 16-bit address is set to 0xFFFE. This is required when sending to a 64-bit address of 0x00s.

Suppose the coordinator's 64-bit address is 0x0013A200404A2244. The following transmit request API frame (0x10) sends an ASCII "1" to the coordinator:

7E 00 0F 10 01 0013 A200 404A 2244 0000 0000 31 18

Example 2: Send a broadcast transmission

In this example, a '\r' refers to a carriage return character.

Perform the following steps to configure a broadcast transmission:

1. Enter command mode ('+++')
2. After receiving an OK\r, issue the following commands:
    - ATDH0\r
    - ATDLffff\r
    - ATCN\r
3. Verify that each of the three commands returned an OK\r response.
4. After setting these command values, all serial characters are sent as a broadcast transmission.

API frame

A transmit request API frame (0x10) can send an ASCII "1" in a broadcast transmission using the following API frame:

7E 00 0F 10 01 0000 0000 0000 FFFF FFFE 00 00 31 C2

The destination 16-bit address is set to 0xFFFE for broadcast transmissions.

Example 3: Send an indirect (binding) transmission.

This example uses the explicit transmit request frame (0x11) to send a transmission using indirect addressing through the binding table. It assumes the binding table has already been set up to map a source endpoint of 0xE7 and cluster ID of 0x0011 to a destination endpoint and 64 bit destination address. The message data is a manufacturing specific profile message using profile ID 0xC105, command ID 0x00, a ZCL Header of 151E10, transaction number EE, and a ZCL payload of 000102030405:

7E 001E 11 e4 FFFFFFFFFFFFFFFF FFFE E7 FF 0011 C105 00 04 151E10EE000102030405 14

Note The 64 bit destination address has been set to all 0xFF values, and the destination endpoint set to 0xFF. The Tx Option 0x04 indicates indirect addressing. The 64 bit destination address and destination endpoint are completed by looking up data associated with binding table entries. This matches the following example.

Example 4: Send a multicast (group ID) broadcast.

This example uses the explicit transmit request frame (0x11) to send a transmission using multicasting. It assumes the destination devices already have their group tables set up to associate an active endpoint with the group ID (0x1234) of the multicast transmission. The message data is a manufacturing specific profile message using profile ID 0xC105command ID 0x00, a ZCL Header of 151E10, transaction number EE, and a ZCL payload of 000102030405:

7E 001E 11 01 FFFFFFFFFFFFFFFF 1234 E6 FE 0001 C105 00 08 151E10EE000102030405 BC

Note The 64 bit destination address has been set to all 0xFF values, and the destination endpoint set to 0xFE. The Tx Option 0x08 indicates use of multicast (group) addressing.

RF packet routing

Unicast transmissions may require some type of routing. Zigbee includes several different methods to route data, each with its own advantages and disadvantages as summarized in the following table.

Routing approach	Description	When to use
Ad hoc On-demand Distance Vector (AODV) Mesh Routing	Routing paths are created between source and destination, possibly traversing multiple nodes ("hops"). Each device knows where to send data next to eventually reach the destination.	Use in networks that will not scale beyond about 40 destination devices.
Many-to-One Routing	A single broadcast transmission configures reverse routes on all devices into the device that sends the broadcast.	Useful when many remote devices must send data to a single gateway or collector device.
Source Routing	Data packets include the entire route the packet should traverse to get from source to destination.	Improves routing efficiency in large networks (over 40 remote devices).

Note End devices do not make use of these routing protocols. Rather, an end device sends a unicast transmission to its parent and allows the parent to route the data packet in its behalf.

Note To revert from Many-to-One routing to AODV routing, a network must first do a network reset (**NR**).
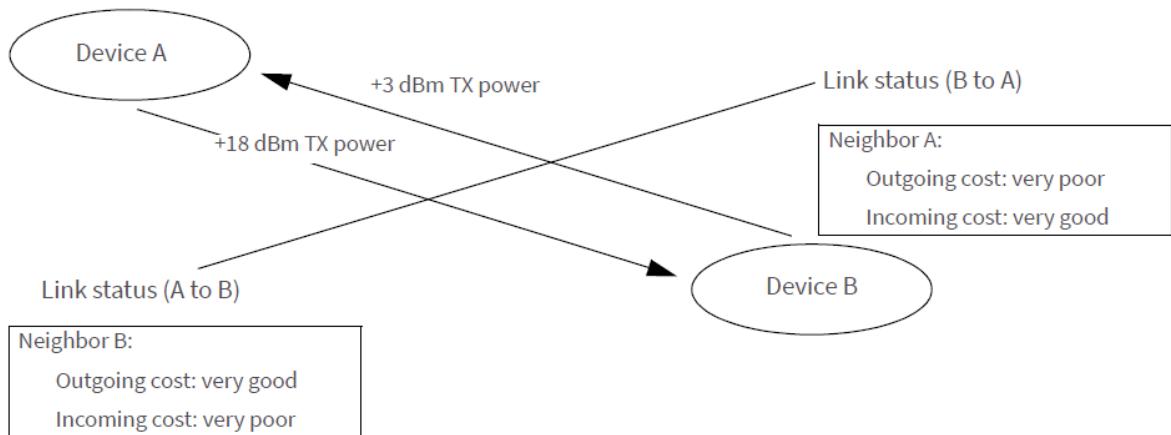
Link status transmission

Before discussing the various routing protocols, it is worth understanding the primary mechanism in Zigbee for establishing reliable bi-directional links. This mechanism is especially useful in networks that may have a mixture of devices with varying output power and/or receiver sensitivity levels.

Each coordinator or router device periodically sends a link status message as a 1-hop broadcast transmission, received only by one-hop neighbors. The link status message contains a list of neighboring devices and incoming and outgoing link qualities for each neighbor. Using these messages, neighboring devices determines the quality of a bi-directional link with each neighbor and uses that information to select a route that works well in both directions.

For example, consider a network of two neighboring devices that send periodic link status messages. Suppose that the output power of device A is +18 dBm, and the output power of device B is +3 dBm (considerably less than the output power of device A). The link status messages might indicate the following:
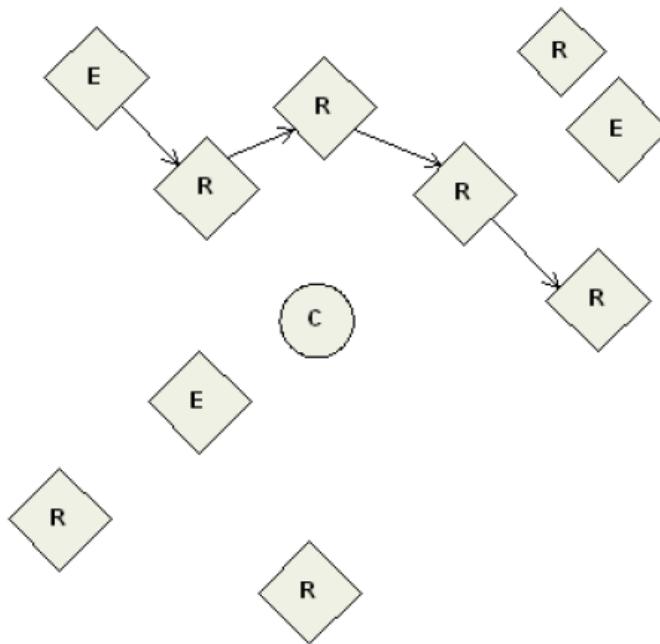
This mechanism enables devices A and B to recognize that the link is not reliable in both directions and select a different neighbor when establishing routes. Such links are called asymmetric links, meaning the link quality is not similar in both directions.

When a router or coordinator device powers on, it sends link status messages every couple seconds to attempt to discover link qualities with its neighbors quickly. After being powered on for some time, the link status messages are sent at a much slower rate, about every 3-4 times per minute.

AODV mesh routing

Zigbee employs mesh routing to establish a route between the source device and the destination. Mesh routing allows data packets to traverse multiple nodes (hops) in a network to route data from a source to a destination. Routers and coordinators can participate in establishing routes between source and destination devices using a process called route discovery. The Route discovery process is based on the Ad-hoc On-demand Distance Vector routing (AODV) protocol.

Sample transmission through a mesh network:
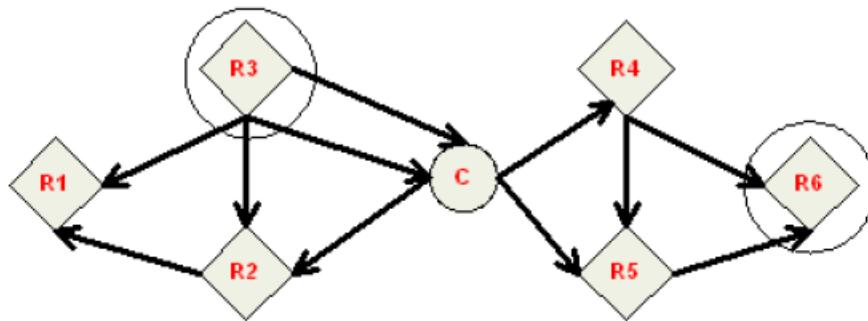
AODV routing algorithm

Routing under the AODV protocol uses tables in each node that store the next hop (intermediary node between source and destination nodes) for a destination node. If a next hop is unknown, route discovery takes place to find a path. Since only a limited number of routes can be stored on a router, route discovery takes place more often on a large network with communication between many different nodes.

Node	Destination address	Next hop address
R3	Router 6	Coordinator
C	Router 6	Router 5
R5	Router 6	Router 6
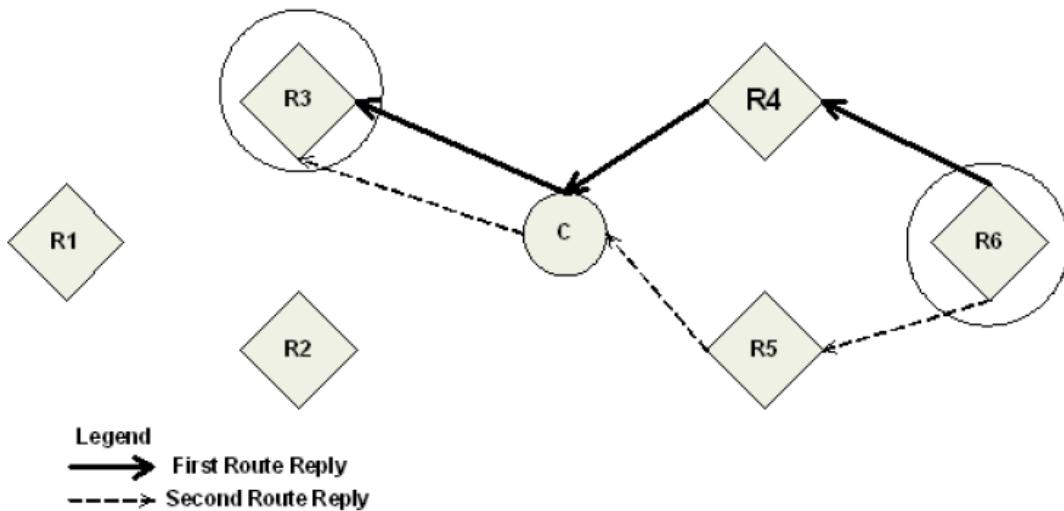
When a source node discovers a route to a destination node, it sends a broadcast route request command. The route request command contains the source network address, the destination network address and a path cost field (a metric for measuring route quality). As the route request command propagates through the network (refer to Broadcast transmissions), each node that re-broadcasts the message updates the path cost field and creates a temporary entry in its route discovery table.

The following graphic is a sample route request (broadcast) transmission where R3 is trying to discover a route to R6:

When the destination node receives a route request, it compares the 'path cost' field against previously received route request commands. If the path cost stored in the route request is better than any previously received, the destination node transmits a route reply packet to the node that originated the route request. Intermediate nodes receive and forward the route reply packet to the source node (the node that originated route request).

The following graphic is a sample route reply (unicast) where R6 sends a route reply to R3:

Note R6 could send multiple replies if it identifies a better route.

Retries and acknowledgments

Zigbee includes acknowledgment packets at both the Mac and Application Support (APS) layers. When data is transmitted to a remote device, it may traverse multiple hops to reach the destination. As the device transmits data from one node to its neighbor, it transmits an acknowledgment packet (Ack) in the opposite direction to indicate that the transmission was successfully received. If the Ack is not received, the transmitting device retransmits the data, up to 4 times.

This Ack is called the Mac layer acknowledgment. In addition, the device that originated the transmission expects to receive an acknowledgment packet (Ack) from the destination device. This Ack traverses the same path the data traversed, but in the opposite direction. If the originator fails to

receive this Ack, it retransmits the data, up to 2 times until it receives an Ack. This Ack is called the Zigbee APS layer acknowledgment.

Note Refer to the Zigbee specification for more details.

Many-to-One routing

In networks where many devices must send data to a central collector or gateway device, AODV mesh routing requires significant overhead. If every device in the network had to discover a route before it could send data to the data collector, the network could easily become inundated with broadcast route discovery messages.

Many-to-one routing is an optimization for these kinds of networks. Rather than require each device to do its own route discovery, the device sends a single many-to-one broadcast transmission from the data collector to establish reverse routes on all devices.

The many-to-one broadcast is a route request message with the target discovery address set to the address of the data collector. Devices that receive this route request create a reverse many-to-one routing table entry to create a path back to the data collector. The Zigbee stack on a device uses historical link quality information about each neighbor to select a reliable neighbor for the reverse route.

When a device sends data to a data collector, and it finds a many-to-one route in its routing table, it transmits the data without performing a route discovery. Send the many-to-one route request periodically to update and refresh the reverse routes in the network.

Applications that require multiple data collectors can also use many-to-one routing. If more than one data collector device sends a many-to-one broadcast, devices create one reverse routing table entry for each collector.

The ZB firmware uses AR (Aggregate Routing Notification) to enable many-to-one broadcasting on a device. **AR** sets a time interval (measured in 10 second units) for sending the many to one broadcast transmission.

High/Low RAM Concentrator mode

When Many to One (MTO) requests are broadcast, **DO** = 40 (bit 6) determines if the concentrator is operating in high or low RAM mode. High RAM mode indicates the concentrator has enough memory to store source routes for the whole network, and remote nodes may stop sending route records after the concentrator has successfully received one. Low RAM mode indicates the concentrator lacks RAM to store route records, and that route records be sent to the concentrator to precede every inbound APS unicast message. By default the device uses low RAM mode.
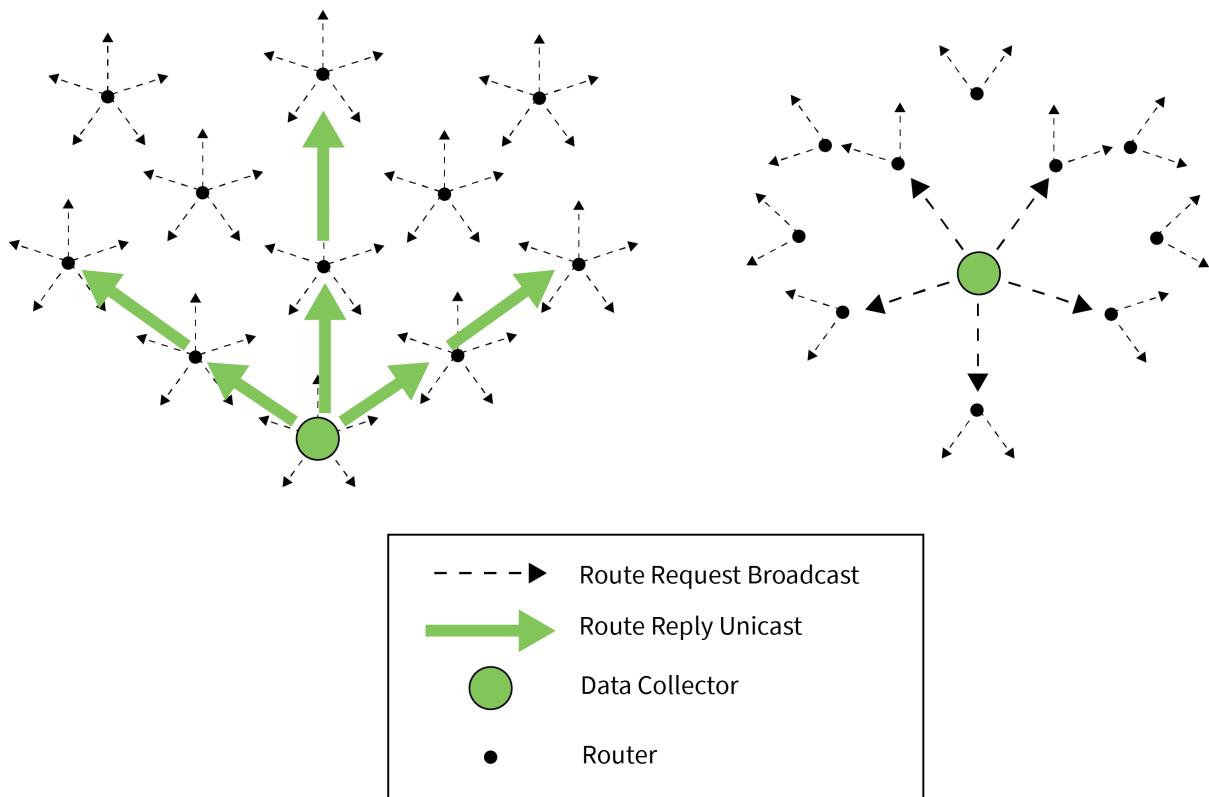
Source routing

In applications where a device must transmit data to many remotes, AODV routing requires performing one route discovery for each destination device to establish a route. If there are more destination devices than there are routing table entries, new routes overwrite established AODV routes, causing route discoveries to occur more regularly. This can result in larger packet delays and poor network performance.
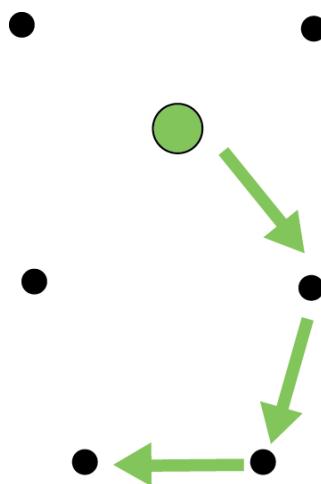
Zigbee source routing helps solve these problems. In contrast to many-to-one routing that establishes routing paths from many devices to one data collector, source routing allows the collector to store and specify routes for many remotes.

To use source routing, a device must use the API mode, and it must send periodic many-to-one route request broadcasts (AR command) to create a many-to-one route to it on all devices. When remote devices send RF data using a many-to-one route, they first send a route record transmission. The
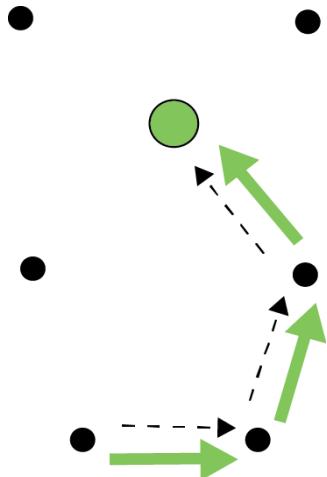
route record transmission is unicast along the many-to-one route until it reaches the data collector. As the route record traverses the many-to-one route, it appends the 16-bit address of each device in the route into the RF payload. When the route record reaches the data collector, it contains the address of the sender, and the 16-bit address of each hop in the route. The data collector can store the routing information and retrieve it later to send a source routed packet to the remote as shown in the following images.

The data collector sends a many-to-one route request broadcast to create reverse routes on all devices.

A remote device sends an RF data packet to the data collector. This is prefaced by a route record transmission to the data collector.

After obtaining a source route, the data collector sends a source routed transmission to the remote device.

Acquiring source routes

Acquiring source routes requires the remote devices to send a unicast to a data collector (device that sends many-to-one route request broadcasts). There are several ways to force remotes to send route record transmissions.

1. If the application on remote devices periodically sends data to the data collector, each transmission forces a route record to occur.

2. The data collector can send a network discovery command (**ND** command) to force all XBee devices to send a network discovery response. A route record prefaces each network discovery response.

3. You can enable periodic I/O sampling on remotes to force them to send data at a regular rate. A route record prefaces each I/O sample. For more information, see Analog and digital I/O lines.

4. If the **NI** string of the remote device is known, the **DN** command can be sent with the **NI** string of the remote in the payload. The remote device with a matching **NI** string would send a route record and a DN response.

Store source routes

When a data collector receives a route record, it sends it out the serial port as a Route Record Indicator - 0xA1. To use source routing, the application receives these frames and stores the source route information.

Send a source routed transmission
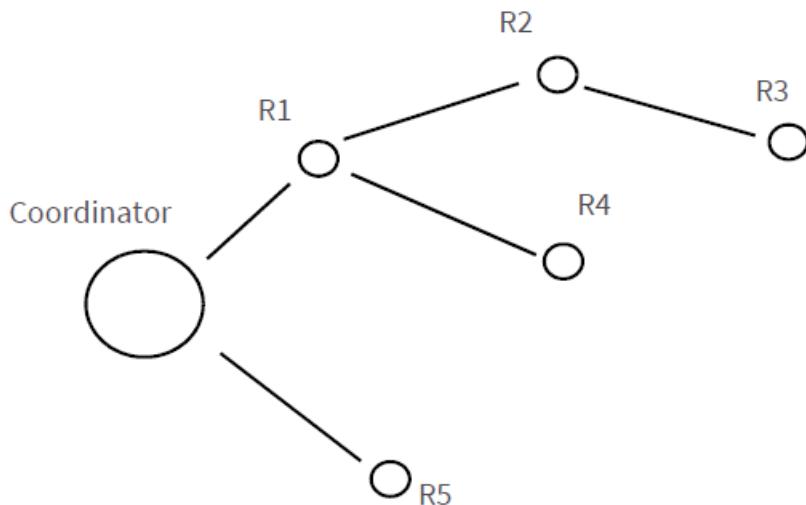
To send a source routed transmission, the application must send a Create Source Route - 0x21 to the XBee/XBee-PRO Zigbee RF Module to create a source route in its internal source route table. After sending the Create Source Route frame, the application can send data transmission or remote command request frames as needed to the same destination, or any destination in the source route. Once data must be sent to a new destination (a destination not included in the last source route), the application must first send a new Create Source Route - 0x21.

Note If a Create Source Route API frame does not precede the data frames, you may encounter data loss.

The XBee/XBee-PRO Zigbee RF Module can buffer one source route that includes up to 11 hops (excluding source and destination). For example, suppose a network exists with a coordinator and 5 routers (R1, R2, R3, R4, R5) with known source routes as shown in the following image.

To send a source-routed packet to R3, the application sends a Create Source Route API frame (0x21) to the XBee, with a destination of R3, and 2 hops (R1 and R2). If the 64- bit address of R3 is 0x0013A200 404a1234 and the 16-bit addresses of R1, R2, and R3 are:

Device	16-bit address
R1	0xAABB
R2	0xCCDD
R3	0xEEFF

The Create Source Route API frame would be:

7E 0012 21 00 0013A200 404A1234 EEFF 00 02 CCDD AABB 5C

Field composition

0x0012	length
0x21	API ID (create source route)
0x00	frame ID (set to 0 always)
0x0013A200 404A1234	64-bit address of R3 (destination)
0xEEFF	16-bit address of R3 (destination)

0x00	Route options (set to 0)
0x02	Number of intermediate devices in the source route
0xCCDD	Address of furthest device (1-hop from target)
0xAABB	Address of next-closer device
0x5C	Checksum (0xFF - SUM (all bytes after length))

Repair source routes

It is possible for a network to have an existing source route fail (for example, a device in the route moves or goes down). If a device goes down in a source routed network, all routes that used the device will be broken.

As mentioned previously, source routing must be used with many-to-one routing. A device that uses source routing must also send a periodic many-to-one broadcast in order to keep routes fresh. If a source route breaks, remote devices send in new route record transmissions to the data collector to provide it with a new source route. This requires that remote devices periodically send data transmissions into the data collector. For more information, see Acquiring source routes.

Retries and acknowledgments

Zigbee includes acknowledgment packets at both the Mac and Application Support (APS) layers. When data transmits to a remote device, it may traverse multiple hops to reach the destination. As data transmits from one node to its neighbor, an acknowledgment packet (Ack) transmits in the opposite direction to indicate that the transmission was successfully received. If the transmitting device does not receive the Ack, it retransmits the data up to four times. This Ack is called the Mac layer acknowledgment.

In addition, the device that originated the transmission expects to receive an acknowledgment packet (Ack) from the destination device. This Ack traverses the same path that the data traversed, but in the opposite direction. If the originator fails to receive this Ack, it retransmits the data, up to two times until an Ack is received. This Ack is called the Zigbee APS layer acknowledgment.

Note Refer to the Zigbee specification for more details.

Disable MTO routing

To disable MTO (many-to-one) routing in a network, first reconfigure the **AR** setting on the aggregator and then broadcast a network wide power reset—0x08 of RE (Restore Defaults)—to rebuild the routing tables.

1. Set **AR** on the aggregator to **0xFF**.
2. Complete an **AC** command to enact the change.
3. Complete a **WR** command if the saved configuration setting value for **AR** is not 0xFF.

This ends the periodic broadcast of aggregator messages if the previous setting was 0x01 - 0xFE, and prevents a single broadcast after a power reset if the previous setting was 0x00. Broadcast a **FR** remote command to the network and wait for the network to reform. This removes the aggregator's status as an aggregator from the network's routing tables so that no more route records will be sent to the aggregator.

Disable route records

If an aggregator collects route records from the nodes of the network and no longer needs route records sent (which consume network throughput) :

1. Set Bit 6 of **DO** to Enable High RAM Concentrator mode. High RAM mode means the aggregator has sufficient memory to hold route records for its potential destinations.

2. Set **AR** to 0x00 for a one-time broadcast (which some nodes might miss), or a value in the range of 0x01 to 0xFE (in units of 10 seconds) to periodically send a broadcast to inform the network that the aggregator is operating in High RAM Concentrator mode and no longer needs to receive route records.

3. Use Create Source Route - 0x21 to load the route record for a destination into the local device's source route table.

4. Send a unicast to the destination. The route record embeds in the payload and determines the sequence of routers to use in transmitting the unicast to the destination. After receiving the unicast, the destination no longer sends route records to the aggregator, now that it has confirmed the High RAM Concentrator aggregator 'knows' its route record.

Clear the source route table

To clear the source route table, change the **AR** setting from a non-0xFF setting to 0xFF and complete an **AC** command. To re-establish periodic aggregator broadcasts, change the **AR** setting to a non-0xFF setting and complete an **AC** command.

Encrypted transmissions

Encrypted transmissions are routed similar to non-encrypted transmissions with one exception. As an encrypted packet propagates from one device to another, each device decrypts the packet using the network key and authenticates the packet by verifying packet integrity. It then re-encrypts the packet with its own source address and frame counter values and sends the message to the next hop. This process adds some overhead latency to unicast transmissions, but it helps prevent replay attacks. For more information see Zigbee security.

Maximum RF payload size

The **NP** command returns the maximum payload size in bytes. The actual maximum payload is a function of the following:

- message type (broadcast or unicast)
- AP setting
- APS encryption option
- Source-routing

Broadcasts, which cannot be encrypted with APS or fragmented have a maximum payload of 0x54 bytes (84 bytes). Unicasts where **AP** is 0 also have a maximum payload of 0x54 bytes. A non-zero **AP** means **NP** will be 0xFF or 255 bytes.

For broadcast messages and unicast messages when **AP**==0, the maximum payload is 0x54 bytes.

For unicast messages when **AP** is nonzero (API mode), the maximum payload is 0xFF (255 bytes) bytes. If the combination of payload and optional APS encryption overhead (EE1, TxOption 0x20) is too high, the message fragments into a maximum of five fragments. The firmware encrypts and transmits each fragment separately. The destination radio reassembles the fragments into a full message.

For Smart Energy firmware revision 5x32 and earlier, **NP**==0x80. As of 5x56, **NP**==0xFF.

The maximum payload is calculated to estimate for aggregator source-routing. To reduce the maximum payload, when an aggregator sends a source-routed message it embeds the route into the message as overhead, or into each fragment of the message, if fragmentation is necessary. If you use APS encryption (EE1, Tx Option 0x20), it reduces the number further.

The route overhead is 2 bytes plus 2 bytes per hop. The bytes are:

- One byte is the number of hops
- One byte is an index into the route list that increments in value at each hop
- Other data is a list of the 16-bit network addresses of the routing radios

Firmware revisions before 4x58 support a maximum of 11 aggregator source-routed hops. Firmware revisions 4x58 and following support a maximum of 25 aggregator source-routed hops.

Aggregator source-routed payload maximums do not apply to messages that are sourced by non-aggregator nodes, which send route records ahead of their messages to aggregators. Aggregators are either Coordinators or Routers which have the following:

- Source routing enabled or
- **AR** setting which is not 0xFF

The following table shows the aggregator source-routed payload maximums as a function of hops and APS encryption:

Hops	Maximum encrypted payload	Maximum unencrypted payload
1	255	255
2	255	255
3	255	255
4	255	255
5	255	255
6	215	255
7	205	250
8	195	240
9	185	230
10	175	220
11	165	210
12	155	200
13	145	190
14	135	180
15	125	170
16	115	160

Hops	Maximum encrypted payload	Maximum unencrypted payload
17	105	150
18	95	140
19	85	130
20	75	120
21	65	110
22	55	100
23	45	90
24	35	80
25	25	70

Throughput

Throughput in a Zigbee network can differ by a number of variables, including:

- Number of hops
- Encryption enabled/disabled
- Sleeping end devices
- Failures/route discoveries.

Our empirical testing showed the following throughput performance in a robust operating environment (low interference).

Configuration	Data throughput
1 hop, **RR**, **SD**	58 kb/s
1 hop, **RR**, **SE**	34 kb/s
1 hop, **RE**, **SD**	Not yet available
1 hop, **RR**, **SE**	Not yet available
1 hop, **ER**, **SD**	Not yet available
1 hop, **ER**, **SE**	Not yet available
4 hops, **RR**, **SD**	Not yet available
4 hops, **RR**, **SE**	Not yet available
RR = router to router **RE** = router to end device (non-sleeping) **ER** = end device (non-sleeping) to router **SD** = security disabled **SE** = security enabled 4 hops = 5 nodes total, 3 intermediate router nodes	

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

Latency timing specifications

Network Depth	10 Node Network	200 Node Network
1	1-byte packet: 32-byte packet:	1-byte packet: 32-byte packet:
2	1-byte packet: 32-byte packet:	1-byte packet: 32-byte packet:
4	1-byte packet: 32-byte packet:	1-byte packet: 32-byte packet:

ZDO transmissions

Zigbee defines a Zigbee device objects layer (ZDO) that provides device and service discovery and network management capabilities.

Cluster name	Cluster ID	Description
Network Address Request	0x0000	Request a 16-bit address of the radio with a matching 64-bit address (required parameter).
Active Endpoints Request	0x0005	Request a list of endpoints from a remote device.
LQI Request	0x0031	Request data from a neighbor table of a remote device.
Routing Table Request	0x0032	Request to retrieve routing table entries from a remote device.
Network Address Response	0x8000	Response that includes the 16-bit address of a device.
LQI Response	0x8031	Response that includes neighbor table data from a remote device.
Routing Table Response	0x8032	Response that includes routing table entry data from a remote device.

Refer to the Zigbee specification for a detailed description of all Zigbee device profile services.

Send a ZDO command

You must use an explicit transmit API frame (0x11) to send a ZDO command, and you must format it correctly.

1. Set the source and destination endpoints and profile ID to 0.

2. Set the cluster ID to match the cluster ID of the appropriate service. For example, to send an active endpoints request, set the cluster ID to 0x0005.

3. The first byte of payload in the API frame is an application sequence number (transaction sequence number) that can be set to any single byte value. The first byte of the ZDO response uses this same value.

4. All remaining payload bytes must be set as required by the ZDO. All multi-byte values must be sent in little endian byte order.

Receiving ZDO command and responses

In XBee ZB firmware, you can easily send ZDO commands using the API. To receive incoming ZDO commands, enable receiver application addressingwith the **AO** command. See the examples later in this section. Not all incoming ZDO commands are passed up to the application.

When a ZDO message is received on endpoint 0 and profile ID 0, the cluster ID indicates the type of ZDO message received. The first byte of payload is generally a sequence number that corresponds to a sequence number of a request. The remaining bytes are set as defined by the ZDO. Similar to a ZDO request, all multi-byte values in the response are in little endian byte order.

Example 1: Send a ZDO LQI request to read the neighbor table contents of a remote

Looking at the Zigbee specification, the cluster ID for an LQI Request is 0x0031, and the payload only requires a single byte (start index). This example sends an LQI request to a remote device with a 64-bit address of 0x0013A200 40401234. The start index is set to 0, and the transaction sequence number is set to 0x76.

API Frame

7E 0016 11 01 0013A200 40401234 FFFE 00 00 0031 0000 00 00 76 00 CE

Field composition

0x0016	length
0x11	Explicit transmit request
0x01	Frame ID (set to a non-zero value to enable the transmit status message, or set to 0 to disable)
0x0013A200 40401234	64-bit address of the remote
0xFFFE	16-bit address of the remote (0xFFFE = unknown). Optionally, set to the 16-bit address of the destination if known.
0x00	Source endpoint
0x00	Destination endpoint
0x0031	Cluster ID (LQI Request, or Neighbor table request)
0x0000	Profile ID (Zigbee device profile)

0x00	Broadcast radius
0x00	Tx Options
0x76	Transaction sequence number
0x00	Required payload for LQI request command
0xCE	Checksum (0xFF - SUM (all bytes after length))

Description

This API frame sends a ZDO LQI request (neighbor table request) to a remote device to obtain data from its neighbor table. You must set the **AO** command correctly on an API device to enable the explicit API receive frames to receive the ZDO response.

Example 2: Send a ZDO network Address Request to discover the 16-bit address of a remote

Looking at the Zigbee specification, the cluster ID for a network Address Request is 0x0000, and the payload only requires the following:

[64-bit address] + [Request Type] + [Start Index]

This example sends a Network Address Request as a broadcast transmission to discover the 16-bit address of the device with a 64-bit address of 0x0013A200 40401234. The request type and start index are set to 0, and the transaction sequence number is set to 0x44.

API frame

7E 001F 11 01 00000000 0000FFFF FFFE 00 00 0000 0000 00 00 44 34124040 00A21300 00 00 33

Field composition

0x001F	length
0x11	Explicit transmit request
0x01	Frame ID (set to a non-zero value to enable the transmit status message, or set to 0 to disable)
0x00000000 0000FFFF	64-bit address for a broadcast transmission
0xFFFE	Set to this value for a broadcast transmission
0x00	Source endpoint
0x00	Destination endpoint
0x0000	Cluster ID (Network Address Request)
0x0000	Profile ID (Zigbee device profile)
0x00	Broadcast radius
0x00	Tx Options
0x44	Transaction sequence number

0x34124040 00A21300 00 00	Required payload for Network Address Request command
0x33	Checksum (0xFF - SUM (all bytes after length))

Description

This API frame sends a broadcast ZDO Network Address Request to obtain the 16-bit address of a device with a 64-bit address of 0x0013A200 40401234. We inserted the bytes for the 64-bit address in little endian byte order. You must insert data for all multi-byte fields in the API payload of a ZDO command in little endian byte order. You must set the **AO** command correctly on an API device to enable the explicit API receive frames to receive the ZDO response.

Transmission timeouts

The Zigbee stack includes two kinds of transmission timeouts, depending on the nature of the destination device. Destination devices such as routers with receivers always on use a unicast timeout. The unicast timeout estimates a timeout based on the number of unicast hops the packet should traverse to get data to the destination device. For transmissions destined for end devices, the Zigbee stack uses an extended timeout that includes the unicast timeout (to route data to the end device's parent), and it includes a timeout for the end device to finish sleeping, wake, and poll the parent for data.

The Zigbee stack includes some provisions for a device to detect if the destination is an end device. The Zigbee stack uses the unicast timeout unless it knows the destination is an end device.

The XBee API includes a transmit options bit that you can set to specify the extended timeout used for a given transmission. If you set this bit, the extended timeout will be used when sending RF data to the specified destination. To improve routing reliability, applications set the extended timeout bit when sending data to end devices if:

- The application sends data to 10 or more remote devices, some of which are end devices.
- The end devices may sleep longer than the unicast timeout.

Equations for these timeouts are computed in the following sections.

Note The timeouts in this section are worst-case timeouts and should be padded by a few hundred milliseconds. These worst-case timeouts apply when an existing route breaks down (for example, intermediate hop or destination device moved).

Unicast timeout

Set the unicast timeout with the **NH** command. The actual unicast timeout is computed as ((50 * NH) + 100). The default **NH** value is 30 which equates to a 1.6 second timeout.

The unicast timeout includes 3 transmission attempts (1 attempt and 2 retries).

The maximum total timeout is approximately:

3 * ((50 * NH) + 100)

For example, if NH=30 (0x1E), the unicast timeout is approximately 3 * ((50 * 30) + 100) or one of the following:

- 3 * (1500 + 100)
- 3 * (1600)

- 4800 ms
- 4.8 seconds

Extended timeout

The worst-case transmission timeout when you are sending data to an end device is a larger issue than when transmitting to a router or coordinator. As described in Parent operation, RF data packets are sent to the parent of the end device, which buffers the packet until the end device wakes to receive it. The parent buffers an RF data packet for up to (1.2 * **SP**) time.

To ensure the end device has adequate time to wake and receive the data, the extended transmission timeout to an end device is:

(50 * **NH**) + (1.2 * **SP**)

This timeout includes the packet buffering timeout (1.2 * **SP**) and time to account for routing through the mesh network (50 * **NH**).

If no acknowledgment is received within this time, the sender resends the transmission up to two more times. With retries included, the longest transmission timeout when sending data to an end device is:

3 * ((50 * **NH**) + (1.2 * **SP**))

The **SP** value in both equations must be entered in millisecond units. The **SP** command setting uses 10 ms units and must be converted to milliseconds to be used in this equation.

For example, suppose a router is configured with **NH**=30 (0x1E) and **SP**=0x3E8 (10,000 ms), and that it is either trying to send data to one of its end device children, or to a remote end device. The total extended timeout to the end device is approximately:

3 * ((50 * **NH**) + (1.2 * **SP**)) or one of the following:

- 3 * (1500 + 12000)
- 3 * (13500)
- 40500 ms
- 40.5 seconds

Transmission examples

Example 1: Send a unicast API data transmission to the coordinator using 64-bit address 0, with payload "TxData".

API frame

7E 0014 10 01 00000000 00000000 FFFE 00 00 54 78 44 61 74 61 AB

Field composition

0x0014	length
0x10	API ID (TX data)
0x01	Frame ID (set greater than 0 to enable the TX-status response)
0x00000000 00000000	64-bit address of coordinator (ZB definition)

0xFFFE	Required 16-bit address if sending data to 64-bit address of 0
0x00	Broadcast radius (0 = max hops)
0x00	Tx options
0x54 78 44 61 74 61	ASCII representation of "TxData" string
0xAB	Checksum (0xFF - SUM (all bytes after length))

Description

This transmission sends the string "TxData" to the coordinator, without knowing the 64-bit address of the coordinator device. ZB firmware defines a 64-bit address of 0 as the coordinator. If the coordinator's 64-bit address was known, the 64-bit address of 0 could be replaced with the coordinator's 64-bit address, and the 16-bit address could be set to 0.

Example 2: Send a broadcast API data transmission that all devices can receive (including sleeping end devices), with payload "TxData".

API frame

7E 0014 10 01 00000000 0000FFFF FFFE 00 00 54 78 44 61 74 61 AD

Field composition

0x0014	length
0x10	API ID (TX data)
0x01	Frame ID (set to a non-zero value to enable the TX-status response)
0x00000000 0000FFFF	Broadcast definition (including sleeping end devices)
0xFFFE	Required 16-bit address to send broadcast transmission
0x00	Broadcast radius (0 = max hops)
0x00	Tx options
0x54 78 44 61 74 61	ASCII representation of "TxData" string
0xAD	Checksum (0xFF - SUM (all bytes after length))

Description

This transmission sends the string "TxData" as a broadcast transmission. Since the destination address is set to 0xFFFF, all devices, including sleeping end devices can receive this broadcast.

If receiver application addressing is enabled, the XBee/XBee-PRO Zigbee RF Module reports all received data frames in the explicit format (0x91) to indicate the source and destination endpoints, cluster ID, and profile ID where each packet was received. Status messages like modem status and route record indicators are not affected.

To enable receiver application addressing, set the **AO** command to 1 using the Local AT Command Request - 0x08 as follows:

API frame

7E 0005 08 01 414F 01 65

Field composition

0x0005	length
0x08	API ID (AT command)
0x01	Frame ID (set to a non-zero value to enable AT command response frames)
0x414F	ASCII representation of 'A','O' (the command being issued)
0x01	Parameter value
0x65	Checksum (0xFF - SUM (all bytes after length))

Description

Setting **AO** = **1** is required for the XBee/XBee-PRO Zigbee RF Module to use the Explicit Receive Indicator - 0x91 when receiving RF data packets. This is required if the application needs indication of source or destination endpoint, cluster ID, or profile ID values used in received Zigbee data packets. ZDO messages can only be received if **AO** = **1**.

Zigbee security

Zigbee supports various levels of security that you can configure depending on the needs of the application. Security provisions include:

- 128-bit AES encryption
- Two security keys that can be preconfigured or obtained during joining
- Support for a trust center
- Provisions to ensure message integrity, confidentiality, and authentication

This section describes various security features defined in the Zigbee specification and illustrates how you can configure theXBee/XBee-PRO Zigbee RF Modules to support these features.

Security modes

The Zigbee standard supports three security modes: residential, standard, and high security.

- **Residential security** requires a network key be shared among devices.
- **Standard security** adds a number of optional security enhancements over residential security, including an APS layer link key.
- **High security** adds entity authentication and a number of other features not widely supported.
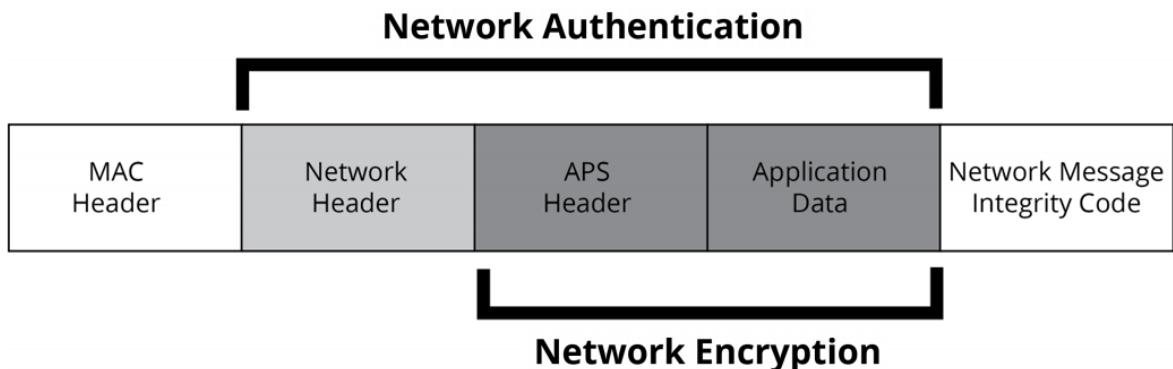
XBee ZB modules primarily support standard security, although end devices that support residential security can join and interoperate with standard security devices. This section focuses on material that is relevant to standard security.

Zigbee security model

Zigbee security is applied to the Network and APS layers. Packets are encrypted with 128-bit AES encryption. A network key and optional link key can be used to encrypt data. Only devices with the same keys are able to communicate together in a network. Routers and end devices that will communicate on a secure network must obtain the correct security keys.

Network layer security

The network key is used to encrypt the APS layer and application data. In addition to encrypting application messages, network security is also applied to route request and reply messages, APS commands, and ZDO commands. Network encryption is not applied to MAC layer transmissions such as beacon transmissions. If you enable security on a network, all data packets are encrypted with the network key.

Network Authentication

Frame counter

The network header of encrypted packets includes a 32-bit frame counter. Each device in the network maintains a 32-bit frame counter that increments for every transmission. In addition, devices track the last known 32-bit frame counter for each of its neighbors. If a device receives a packet from a neighbor with a smaller frame counter than previously seen, it discards the packet. The device uses the frame counter to protect against replay attacks.

If the frame counter reaches a maximum value of 0xFFFFFFFF, it does not wrap to 0 and cannot send any more transmissions. Due to the size of the frame counters, reaching the maximum value is uncommon for most applications. The following table shows the required time for the frame counter to reach its maximum value.

Average Transmission Rate	Time until 32-bit frame counter expires
1 / second	136 years
10 / second	13.6 years

To clear the frame counters without compromising security, you can change the network key in the network. When the network key is updated, the frame counters on all devices reset to 0. See Network key updates for details.

Message integrity code

The network header, APS header, and application data are all authenticated with 128-bit AES. The device performs a hash on these fields and is appended as a 4-byte message integrity code (MIC) to the end of the packet. The MIC allows receiving devices to ensure the message has not been changed. The MIC provides message integrity in the Zigbee security model. If a device receives a packet and the MIC does not match the device's own hash of the data, it drops the packet.

Network layer encryption and decryption

Packets with network layer encryption are encrypted and decrypted by each hop in a route. When a device receives a packet with network encryption, it decrypts the packet and authenticates the packet. If the device is not the destination, it then encrypts and authenticates the packet, using its own frame counter and source address in the network header section.

Since the device performs network encryption at each hop, packet latency is slightly longer in an encrypted network than in a non-encrypted network. Also, security requires 18 bytes of overhead to

include a 32-bit frame counter, an 8-byte source address, 4-byte MIC, and 2 other bytes. This reduces the number of payload bytes that can be sent in a data packet.
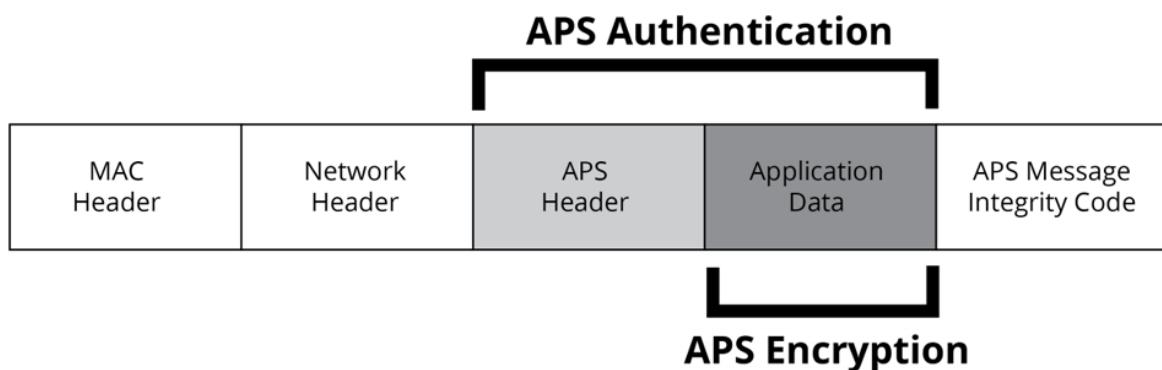
Network key updates

Zigbee supports a mechanism for changing the network key in a network. When the network key is changed, the frame counters in all devices reset to 0.

APS layer security

APS layer security can be used to encrypt application data using a key that is shared between source and destination devices. Where network layer security is applied to all data transmissions and is decrypted and reencrypted on a hop-by-hop basis, APS security is optional and provides end-to-end security using an APS link key known by only the source and destination device. APS security cannot be applied to broadcast transmissions.

If you enable APS security, the APS header and data payload are authenticated with 128-bit AES as shown in the following image:

Message integrity code

If you enable APS security, the APS header and data payload are authenticated with 128-bit AES. The device performs a hash on these fields and appends as a 4-byte message integrity code (MIC) to the end of the packet. This MIC is different than the MIC appended by the network layer. The MIC allows the destination device to ensure the message has not been changed. If the destination device receives a packet and the MIC does not match the destination device's own hash of the data, it drops the packet.

APS link keys
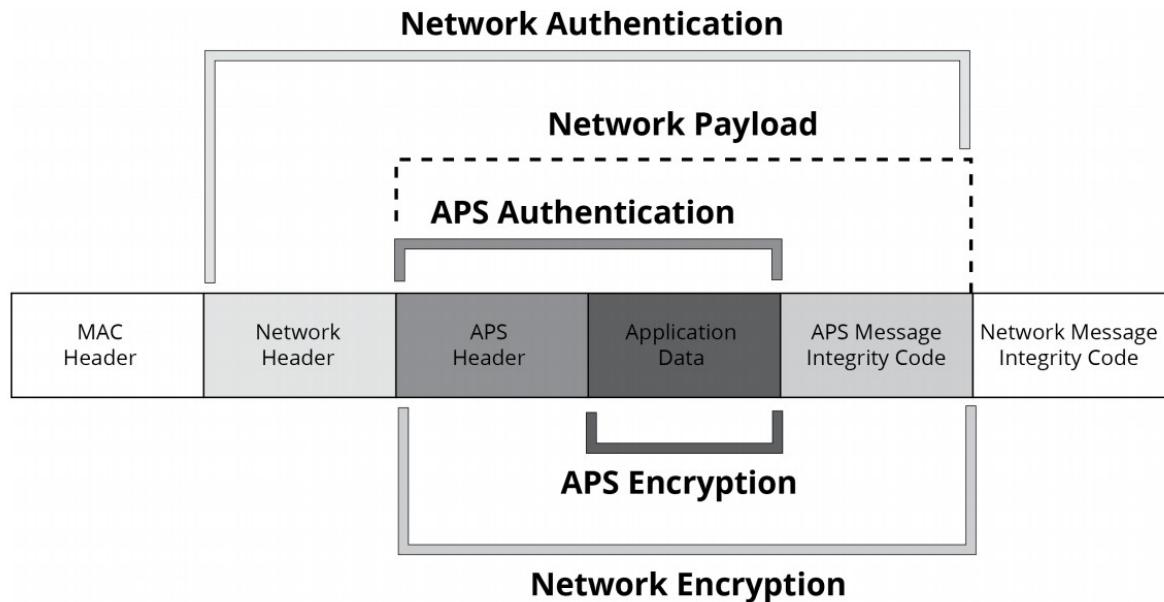
There are two kinds of APS link keys: trust center link keys and application link keys. A trust center link key is established between a device and the trust center, where an application link key is established between a device and another device in the network where neither device is the trust center.

APS layer encryption and decryption

Packets with APS layer encryption are encrypted at the source and only decrypted by the destination. Since APS encryption requires a 5-byte header and a 4-byte MIC, the maximum data payload is reduced by 9 bytes when APS encryption is used.

Network and APS layer encryption

Network and APS layer encryption can both be applied to data. The following figure demonstrates the authentication and encryption performed on the final Zigbee packet when both are applied.

Trust center

Zigbee defines a trust center device that is responsible for authenticating devices that join the network. The trust center also manages link key distribution in the network.

Forming or joining a secure network

The coordinator is responsible for selecting a network encryption key. This key can either be preconfigured or randomly selected. In addition, the coordinator generally operates as a trust center and must select the trust center link key. The trust center link key can also be preconfigured or randomly selected.

Devices that join the network must obtain the network key when they join. When a device joins a secure network, the network and link keys can be sent to the joining device. If the joining device has a preconfigured trust center link key, the network key will be sent to the joining device encrypted by the link key. Otherwise, if the joining device is not preconfigured with the link key, the device could only join the network if the network key is sent unencrypted ("in the clear").

The trust center must decide whether or not to send the network key unencrypted to joining devices that are not preconfigured with the link key. We do not recommend sending the network key unencrypted as it can open a security hole in the network. To maximize security, preconfigure devices with the correct link key.

Implement security on the XBee/XBee-PRO Zigbee RF Module

If you enable security in the XBee Zigbee firmware, devices acquire the network key when they join a network. Data transmissions are always encrypted with the network key, and can optionally be end-to-end encrypted with the APS link key.

Enabling security

To enable security on a device, the Encryption Enable (**EE**) parameter must be set to 1. When the parameter value changes, the XBee module leaves the network (PAN ID and channel) it was operating on and attempt to form or join a new network.

If you set **EE** to 1, all data transmissions are encrypted with the network key. When you enable security, the maximum number of bytes in a single RF transmission will be reduced. For more information, see NP (Maximum Packet Payload Bytes).

Note The **EE** parameter must be set the same on all devices in a network. Changes to the **EE** command should be written to non-volatile memory (to be preserved through power cycle or reset events) using the **WR** command.

Setting the network security key

The coordinator selects the network security key for the network using the Network Encryption Key (**NK**) parameter (write-only). If **NK** = **0** (default), the coordinator will selects a random network key. Otherwise, you set **NK** to a non-zero value, it uses this value as network security key.

NK is only supported on the coordinator. Routers and end devices with security enabled (**EE** = 1) acquire the network key when they join a network. They receive the network key encrypted with the link key if they share a preconfigured link key with the coordinator.

Note In Zigbee, if **EE** and **EO** are set to 0x01, then the device sends the network key in the clear (unencrypted) with the link key at association time. This may be a useful setting in development environments, but we discourage it for product deployment for security reasons.

Set the APS trust center link key

The coordinator must also select the trust center link key, using KY (Link Key). If **KY** = **0** (default), the coordinator selects a random trust center link key (not recommended). Otherwise, if **KY** is set greater than 0, the device uses this value as the pre-configured trust center link key. **KY** is write-only and cannot be read.

Note Application link keys sent between two devices where neither device is the coordinator are not supported in Zigbee firmware.

Random trust center link keys

If the coordinator selects a random trust center link key (**KY** = **0**, default), then it allows devices to join the network without having a pre-configured link key. However, this sends the network key unencrypted over-the-air to joining devices and is not recommended.

Pre-configured trust center link keys

If the coordinator uses a pre-configured link key (**KY** > 0), then it will not send the network key unencrypted to joining devices. Only devices with the correct pre-configured link key can to join and communicate on the network.

Enable APS encryption

APS encryption is an optional layer of security that uses the link key to encrypt the data payload. Unlike network encryption that is decrypted and encrypted on a hop-by-hop basis, APS encryption is

only decrypted by the destination device. The XBee/XBee-PRO Zigbee RF Module must be configured with security enabled (**EE** set to 1) to use APS encryption.

APS encryption can be enabled in API firmware on a per-packet basis. To enable APS encryption for a given transmission, set the "enable APS encryption" transmit options bit in the API transmit frame. Enabling APS encryption decreases the maximum payload size by nine bytes.

Use a trust center

Use the Encryption Options (**EO**) parameter define the coordinator as a trust center. If the coordinator is a trust center, it received alerts to all new join attempts in the network. The trust center also has the ability to update or change the network key on the network.

In ZB firmware, you can establish a secure network with or without a trust center. Network and APS layer encryption are supported regardless of whether you use a trust center.

Updating the network key with a trust center.

If the trust center has started a network and the **NK** value changes, the coordinator updates the network key on all devices in the network. Changes to **NK** will not force the device to leave the network. The network continues to operate on the same channel and PAN ID, but the devices in the network update their network key, increment their network key sequence number, and restore their frame counters to 0.

Updating the network key without a trust center.

If the coordinator is not running as a trust center, the Network Reset (**NR1**) command can be used to force all devices in the network to leave the current network and rejoin the network on another channel. When devices leave and reform then network, the frame counters are reset to 0. This approach causes the coordinator to form a new network that the remaining devices should join. Resetting the network in this manner brings the coordinator and routers in the network down for about ten seconds, and causes the 16-bit PAN ID and 16-bit addresses of the devices to change.

In Zigbee firmware, a secure network can be established with or without a trust center. Network and APS layer encryption are supported regardless of whether a trust center is used.

Security examples

This section covers some sample XBee device configurations to support different security modes and lists several **AT** commands with suggested parameter values.

In **AT** command mode, issue each command with a leading 'AT' and no '=' sign: For example, EE1. In the API, the two byte command is used in the command field, and parameters are populated as binary values in the parameter field.

Example 1: Forming a network with security (pre-configured link keys)

1. Start a coordinator with the following settings:
   a. **ID** = 2234 (arbitrarily selected)
   b. **EE** = 1
   c. **NK** = 0
   d. **KY** = 4455
   e. **WR** (save networking parameters to preserve them through power cycle)

2. Configure one or more routers or end devices with the following settings:
   a. **ID** = 2234
   b. **EE** = 1
   c. **KY** = 4455
   d. **WR** (save networking parameters to preserve them through power cycle)
3. Read the **AI** setting on the coordinator and joining devices until they return 0 (formed or joined a network).

This example sets the **EE**, **ID**, and **KY** commands the same on all devices. After successfully joining the secure network, the network key encrypts all application data transmissions. Since **NK** was set to 0 on the coordinator, the device selects a random network key. Because the link key (**KY**) was configured to a non- zero value on all devices, the pre-configured link key (**KY**) sent the network key encrypted when the devices joined.

Example 2: Forming a network with security (obtaining keys during joining)

1. Start a coordinator with the following settings:
   a. **ID** = 2235
   b. **EE** = 1
   c. **NK** = 0
   d. **KY** = 0
   e. **WR** (save networking parameters to persist through power cycle)
2. Configure one or more routers or end devices with the following settings:
   a. **ID** = 2235
   b. **EE** = 1
   c. **KY** = 0
   d. **WR** (save networking parameters to persist through power cycle)
3. Read the **AI** setting on the coordinator and joining devices until they return 0 (formed or joined a network).

This example sets the **EE**, **ID**, and **KY** commands the same on all devices. Since **NK** was set to 0 on the coordinator, the device selects a random network key. Because **KY** was set to 0 on all devices, the network key was sent unencrypted ("in the clear") when the devices joined.

⚠️ This approach introduces a security vulnerability into the network and is not recommended.

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.
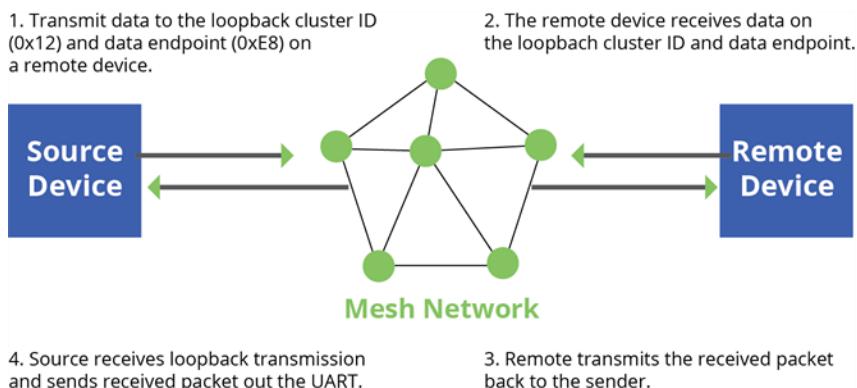
Place devices

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

Test links in a network - loopback cluster

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

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

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

Transparent operating mode configuration (AP = 0)

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

1. Set the **CI** command to **0x12**.

2. Set the **SE** and **DE** commands to **0xE8** (default value).

3. Set the **DH** and **DL** commands to the address of the remote (**0** for the coordinator, or the 64-bit address of the remote).

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

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

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

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

RSSI indicators

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

Determine the **DB** value in hardware using the RSSI/PWM device pin (TH pin 6/SMT pin 7). If you enable the RSSI PWM functionality (**P0** command), when the device receives data, it sets the RSSI PWM to a value based on the RSSI of the received packet (this value only indicates the quality of the last hop). You could connect this pin to an LED to indicate if the link is stable or not.

Device discovery

Network discovery

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

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

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

ZDO discovery

The Zigbee device profile includes provisions to discover devices in a network that are supported on all Zigbee devices (including non-Digi products). These include the LQI Request (cluster ID 0x0031) and the Network Update Request (cluster ID 0x0038). You can use the LQI Request to read the devices in the neighbor table of a remote device, and the Network Update Request for a remote device to complete an active scan to discover all nearby Zigbee devices. Note that the LQI of End Devices is not tracked and will always return 0xFF (255). You can send both of these ZDO commands using the XBee Explicit API transmit frame (0x11). For more information, see API Operation. Refer to the Zigbee specification for formatting details of these two ZDO frames.

Joining Announce

All Zigbee devices send a ZDO Device Announce broadcast transmission when they join a Zigbee network (ZDO cluster ID 0x0013). These frames are sent out the device's serial port as an Explicit Rx Indicator API frame (0x91) if **AO** is set to 1. The device announce payload includes the following information:

[Sequence Number] + [16-bit address] + [64-bit address] + [Capability]

The 16-bit and 64-bit addresses are received in little-endian byte order (LSB first). See the Zigbee specification for details.

Commissioning pushbutton and associate LED

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

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

Commissioning pushbutton

The commissioning pushbutton definitions provide a variety of simple functions to help with deploying devices in a network. Enable the commissioning button functionality on pin 20 by setting D0 (AD0/DIO0 Configuration) to **1** (enabled by default).

Button presses	Description
1	Start Joining. Wakes a sleeping end device for 30 seconds, regardless of the **ST**/**SN** setting. It also sends node identification broadcast if joined to a network. A Zigbee device blinks a numeric error code on the Associate pin indicating the cause of join failure for (**AI** - 32) times. A **SE** router or **SE** end device which is associated but not authenticated to a network leaves its network; then attempt to join.
2	Enable Joining. Broadcast a Mgmt_Permit_Joining_req (ZDO ClusterID 0x0036) with TC_ Significance set to **0x00**. If **NJ** is **0x00** or **0xFF**, PermitDuration is set to one minute, otherwise PermitDuration is set to **NJ**.
4	Restore configuration to default values and leave the network. Equivalent to issuing **NR**, **RE**, and **AC** commands.

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

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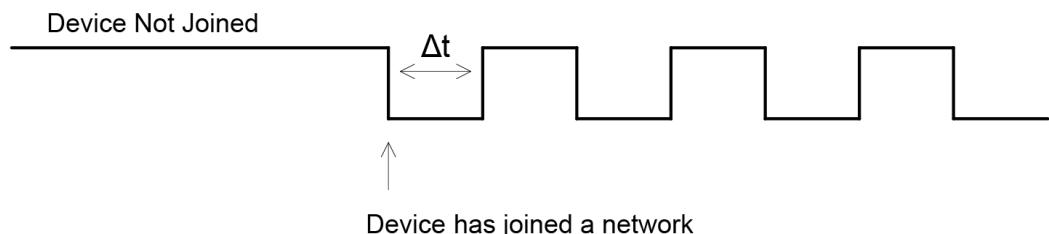
Associate LED

The Associate pin (pin 28/SMT, pin 33/TH) provides an indication of the device's network status and diagnostics information. Connect an LED to the Associate pin as shown in the figure in Commissioning pushbutton and associate LED. Enable the Associate LED functionality the **D5** command to 1 (enabled by default). If the Associate pin is enabled, it configured as an output.

Joined indication

The Associate pin indicates the network status of a device. If the device is not joined to a network, the Associate pin is set high. Once the device successfully joins a network, the Associate pin blinks at a regular time interval. The following figure shows the joined status of a device.

Associate

The associate pin can indicate the joined status of a device. Once the device has joined a network, the associate pin toggles state at a regular interval (Δt). Use the **LT** command to set the time.
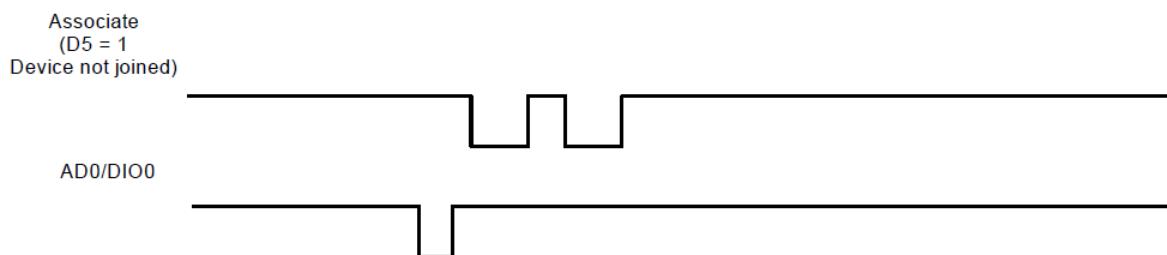
The **LT** command defines the blink time of the Associate pin. If it is set to 0, the device uses the default blink time (500 ms for coordinator, 250 ms for routers and end devices).

Diagnostics support
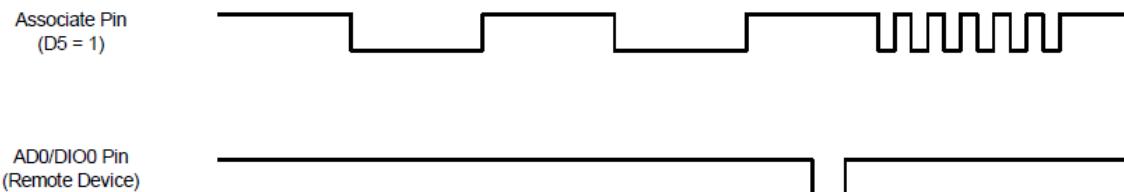
The Associate pin works with the commissioning pushbutton to provide additional diagnostics behaviors to aid in deploying and testing a network. If the commissioning push button is pressed once, and the device has not joined a network, the Associate pin blinks a numeric error code to indicate the cause of join failure. The number of blinks is equal to (**AI** value – 0x20). For example, if **AI** = 0x22, 2 blinks occur.

If the commissioning push button is pressed once and the device has joined a network, the device transmits a broadcast node identification packet. If the Associate LED functionality is enabled (**D5** command), a device that receives this transmission will blink its Associate pin rapidly for 1 second.

The following image illustrates the behavior pressing the commissioning button press once when the device has not joined a network, causing the associate pin to blink to indicate the **AI** Code where: **AI** = # blinks + 0x20. In this example, **AI** = 0x22.

The following image illustrates the behavior pressing the button once on a remote device, causing a broadcast node identification transmission to be sent. All devices that receive this transmission blink their associate pin rapidly for one second if the associate LED functionality is enabled (**D5** = 1).

Binding

The Digi XBee firmware supports three binding request messages:

- End Device Bind
- Bind
- Unbind

End_Device_Bind_req

The End Device Bind request (ZDO cluster 0x0020) is described in the Zigbee Specification.

During a deployment, an installer may need to bind a switch to a light. After pressing a commissioning button sequence on each device, this causes them to send End_Device_Bind_req messages to the Coordinator within a time window (60 s). The payload of each message is a simple descriptor which lists input and output clusterIDs. The Coordinator matches the requests by pairing complementary clusterIDs. After a match has been made, it sends messages to bind the devices together. When the process is over, both devices will have entries in their binding tables which support indirect addressing of messages between their bound endpoints.

R1->C End_Device_Bind_req

R2->C End_Device_Bind_req

> R1, R2 send End_Device_Bind_req within 60 s of each other to C
>
> C matches the requests.
>
> C tests one to see if binding is already in place:

R2<-C Unbind_req

R2->C Unbind-rsp (status code - NO_ENTRY)

> C proceeds to create binding table entries on the two devices.

R1<-C Bind_req

R1->C Bind_rsp

R2<-C Bind_req

R2->C Bind_rsp

> C sends responses to the original End_Device_Bind_req messages.

R1-<C End_Device_Bind_rsp

R2-<C End_Device_Bind_rsp

End Device binding sequence (binding)

This message has a toggle action. If the same two devices were to subsequently send End_Device_Bind_req messages to the Coordinator, the Coordinator would detect they were already bound, and then send Unbind_req messages to remove the binding.

An installer can use this to remove a binding which was made incorrectly, say from a switch to the wrong lamp, by repeating the commissioning button sequence used beforehand.

R1->C End_Device_Bind_req

R2->C End_Device_Bind_req

> R1, R2 send End_Device_Bind_req within 60 s of each other to C
>
> C matches the requests.
>
> C tests one to see if binding is already in place:

R2<-C Unbind_req

R2->C Unbind-rsp (status code - SUCCESS)

> C proceeds to remove binding table entries from the two devices.

R1<-C Unbind_req

R1->C Unbind_rsp

R2<-C Unbind_req

R2->C Unbind_rsp

> C sends responses to the original End_Device_Bind_req messages.

R1-<C End_Device_Bind_rsp

R2-<C End_Device_Bind_rsp

End Device binding sequence (removal)

This example shows a correctly formatted End_Device_Bind_req (ZDO cluster 0x0020) using a Digi 0x11 Explicit API Frame:

The frame as a bytelist:

7e00281101000000000000000fffe000000020000000000001f2995cb5474000a21300e605c1010100010200046

Same frame broken into labeled fields.

Note Multibyte fields are represented in big-endian format.

7e	Frame Delimiter
0028	Frame Length
11	API Frame Type (Explicit Frame)
01	Frame Identifier (for response matching)
0000000000000000	Coordinator address
fffe	Code for unknown network address
00	Source Endpoint (need not be 0x00)
00	Destination Endpoint (ZDO endpoint)
0020	Cluster 0x0020 (End_Device_Bind_req)
0000	ProfileID (ZDO)

00	Radius (default, maximum hops)
00	Transmit Options
01f2995cb5474000a21300e605c1010100010200	RFData (ZDO payload)
46	Checksum

Here is the RFData (the ZDO payload) broken into labeled fields. Note the multi-byte fields of a ZDO payload are represented in little-endian format.

01	Transaction Sequence Number
f299	Binding Target (16 bit network address of sending device)
5cb5474000a21300	(64 bit address of sending device)
e6	Source Endpoint on sending device
05c1	ProfileID (0xC105) - used when matching End_Device_Bind_requests
01	Number of input clusters
0100	Input cluster ID list (0x0100)
01	Number of output clusters
0200	Output cluster ID list (0x0200)

Example of a End_Device_Bind_req

Bind_req

The Bind request (ZDO cluster 0x0021) is described in the Zigbee Specification. A binding may be coded for either a unicast or a multicast/groupID message.

Unbind_req

The Unbind request (ZDO cluster 0x0022) is described in the Zigbee Specification.

Group Table API

Unlike the Binding Table that is managed with ZDO commands, a Zigbee group table is managed by the Zigbee cluster library Groups Cluster (0x0006) with ZCL commands.

The Digi Zigbee XBee firmware is intended to work with an external processor where a Public Application Profile with endpoints and clusters is implemented, including a Groups Cluster. Configure the Zigbee XBee firmware to forward all ZCL commands addressed to this Group Cluster out the UART (see ATAO3). The XBee Zigbee will not use remote Groups Cluster commands to manage its own Group Table.

But to implement multicast (group) addressing within the XBee, the external processor must keep the XBee device's group table state in sync with its own. For this reason, a Group Table API has been defined where the external processor can manage the state of the XBee/XBee-PRO Zigbee RF Module group table.

The design of the Group Table API of the XBee firmware derives from the ZCL Group Cluster 0x0006. Use the Explicit Addressing Command Request - 0x11 addressed to the Digi Device Object endpoint (0xE6) with the Digi XBee ProfileID (0xC105) to send commands and requests to the local device.

The Zigbee home automation public application profile says groups should only be used for sets of more than five devices. This implies sets of five or fewer devices should be managed with multiple binding table entries.

There are five commands implemented in the API:

- Add Group command
- View group
- Get Group Membership
- Remove Group
- Remove All Groups

There is a sixth command of the Group Cluster described in the ZCL: Add Group If Identifying. This command is not supported in this API, because its implementation requires access to the Identify Cluster, which is not maintained on the XBee. The external processor needs to implement that server command while using the Group Table API to keep the XBee device's group table in sync using the five command primitives.

Add Group command

The purpose of the Add Group command is to add a group table entry to associate an active endpoint with a groupID and optionally a groupName. The groupID is a two byte value. The groupName consists of zero to 16 ASCII characters.

The following example adds a group table entry which associates endpoint E7 with groupID 1234 and groupName "ABCD".

The example packet is given in three parts, the preamble, ZCL Header, and ZCL payload:

>   Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterID of 0x0006, and profileID of 0xC105. The destination endpoint E7 holds the endpoint parameter for the "Add Group" command.

>   ZCL_header = "01 ee 00"

The first field (byte) is a frame control field which specifies a Cluster Specific command (0x01) using a Client->Server direction(0x00). The second field is a transaction sequence number used to associate the response with the command request. The third field is the command identifier for "Add Group" (0x00).

>   ZCL_payload = "3412 04 41 42 43 44"

The first two bytes is the group Id to add in little endian representation. The next byte is the string name length (00 if there is no string). The other bytes are the descriptive ASCII string name ("ABCD") for the group table entry. The string is represented with its length in the first byte, and the other bytes containing the ASCII characters.

The example packet in raw hex byte form:

>   7e001e11010013a2004047b55cfffee6e70006c105000001ee0034120441424344c7

The response in raw hex byte form, consisting of two packets:

>   7e0018910013a2004047b55cfffee7e68006c1050009ee0000341238

>   7e00078b01fffe00000076

The response in decoded form:

> Zigbee Explicit Rx Indicator
>
> API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE SrcEP 0xE7 DestEP 0xE6
>
> ClusterID 0x8006 ProfileID 0xC105 Options 0x00
>
> RF_Data 0x09EE00003412

The response in terms of Preamble, ZCL Header, and ZCL payload:

> Preamble = "910013a2004047b55cfffee7e68006c10500"

The packet has its endpoint values reversed from the request, and the clusterID is 0x8006 indicating a Group cluster response.

> ZCL_header = "09 ee 00"

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server-> Client direction. The second field is a transaction sequence number used to associate the response with the command request. The third field is the command identifier "Add Group" (0x00).

> ZCL_payload = "00 3412"

The first byte is a status byte (SUCCESS=0x00). The next two bytes hold the group ID (0x1234) in little endian form.

This is the decoded second message, which is a Tx Status for the original command request. If the FrameId value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message occurs.

> Zigbee Tx Status
>
> API 0x8B FrameID 0x01 16DestAddr 0xFFFE
>
> Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success

View group

The purpose of the View Group command is to get the name string which is associated with a particular endpoint and groupID.

The following example gets the name string associated with the endpoint E7 and groupID 1234.

The packet:

```
Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"
```

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterID of 0x0006, and profileID of 0xC105. The destination endpoint E7 is the endpoint parameter for the "View Group" command.

```
ZCL_header = "01 ee 01"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Client->Server direction(0x00). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "View Group" (0x01) .

```
ZCL_payload = "3412"
```

The two byte value is the groupID in little-endian representation.

The packet in raw hex byte form:

```
7e001911010013a2004047b55cfffee6e70006c105000001ee013412d4
```

The response in raw hex byte form, consisting of two packets:

```
7e001d910013a2004047b55cffffee7e68006c1050009ee01003412044142434424
7e00078b01fffe00000076
```

The command response in decoded form:

```
Zigbee Explicit Rx Indicator
API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE SrcEP 0xE7 DestEP 0xE6
ClusterID 0x8006 ProfileID 0xC105 Options 0x00
RF_Data 0x09EE010034120441424344
```

The response in terms of Preamble, ZCL Header, and ZCL payload:

```
Preamble = "910013a2004047b55cffffee7e68006c10500"
```

The packet has its endpoint values reversed from the request, and the clusterID is 0x8006 indicating a Group cluster response.

```
ZCL_header = "09 ee 01"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server->Client direction (0x08). The second field is a transaction sequence number which associates the response with the command request. The third field is the command identifier "View Group" (0x01) .

```
ZCL_payload = "00 3412 0441424344"
```

The first byte is a status byte (SUCCESS=0x00). The next two bytes hold the groupID (0x1234) in little-endian form. The next byte is the name string length (0x04). The remaining bytes are the ASCII name string characters ("ABCD").

The following is the decoded second message, which is a Tx Status for the original command request. If the FrameId value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message would occur.

```
Zigbee Tx Status
API 0x8B FrameID 0x01 16DestAddr 0xFFF
Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success
```

Get Group Membership

Get Group Membership (1 of 2)

The purpose of this first form of the Get Group Membership command is to get all the groupIDs associated with a particular endpoint.

The intent of the example is to get all the groupIDs associated with endpoint E7.

The example packet is given in three parts, the preamble, ZCL Header, and ZCL payload:

```
Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"
```

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterID of 0x0006, and profileID of 0xC105. The destination endpoint E7 holds the endpoint parameter for the "Get Group Membership" command.

```
ZCL_header = "01 ee 02"
```

The first field (byte) is a frame control field which specifies a Cluster Specific command (0x02) using a Client->Server direction(0x00). The second field is a transaction sequence number which is used to

associate the response with the command request. The third field is the command identifier for "Get Group Membership" (0x02) .

```
ZCL_payload = "00"
```

The first byte is the group count. If it is zero, then all groupIDs with an endpoint value which matches the given endpoint parameter will be returned in the response.

The example packet in raw hex byte form:

```
7e001811010013a2004047b55cffffee6e70006c105000001ee020019
```

The response in raw hex byte form, consisting of two packets:

```
7e0019910013a2004047b55cffffee7e68006c1050009ee02ff01341235
```

```
7e00078b01fffe00000076
```

The response in decoded form:

```
Zigbee Explicit Rx Indicator
API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE SrcEP 0xE7 DestEP 0xE6
ClusterID 0x8006 ProfileID 0xC105 Options 0x00
RF_Data 0x09EE02FF013412
```

The response in terms of Preamble, ZCL Header, and ZCL Payload:

```
Preamble = "910013a2004047b55cffffee7e68006c10500"
```

The packet has the endpoints reversed from the request, and the clusterID is 0x8006 indicating a Group cluster response.

```
ZCL_header = "09 ee 02"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server->Client direction (0x08). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Get Group Membership" (0x02) .

```
ZCL_payload = "FF 01 3412"
```

The first byte is the remaining capacity of the group table. 0xFF means unknown. The XBee returns this value because the capacity of the group table is dependent on the remaining capacity of the binding table, thus the capacity of the group table is unknown. The second byte is the group count (0x01). The remaining bytes are the groupIDs in little-endian representation.

The following is the decoded second message, which is a Tx Status for the original command request. If the FrameId value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message would occur.

```
Zigbee Tx Status
API 0x8B FrameID 0x01 16DestAddr 0xFFFE
Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success
```

Get Group Membership (2 of 2)

The purpose of this second form of the Get Group Membership command is to get the set of groupIDs associated with a particular endpoint which are a subset of a list of given groupIDs.

The following example gets the groupIDs associated with endpoint E7 which are a subset of a given list of groupIDs (0x1234, 0x5678).

The example packet is given in three parts, the preamble, ZCL Header, and ZCL payload:

```
Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"
```

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterID of 0x0006, and profileID of 0xC105. The destination endpoint E7 is the endpoint parameter for the "Get Group Membership" command.

```
ZCL_header = "01 ee 02"
```

The first field (byte) is a frame control field which specifies a Cluster Specific command (0x02) using a Client->Server direction(0x00). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier for "Get Group Membership" (0x02) .

```
ZCL_payload = "02 34127856"
```

The first byte is the group count. The remaining bytes are a groupIDs which use little-endian representation.

The example packet in raw hex byte form:

```
  7e001c11010013a2004047b55cfffee6e70006c105000001ee02023412785603
```

The response in raw hex byte form, consisting of two packets:

```
7e0019910013a2004047b55cfffee7e68006c1050009ee02ff01341235
7e00078b01fffe00000076
```

The response in decoded form:
```
Zigbee Explicit Rx Indicator
```

```
API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE        SrcEP 0xE7     DestEP
0xE6
ClusterID 0x8006 ProfileID 0xC105     Options 0x00
RF_Data 0x09EE02FF013412
```

The response in terms of Preamble, ZCL Header, and ZCL Payload:

```
Preamble = "910013a2004047b55cfffee7e68006c10500"
```

The packet has the endpoints reversed from the request, the clusterID is 0x8006 indicating a Group cluster response.

```
ZCL_header = "09 ee 02"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server->Client direction (0x08). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Get Group Membership" (0x02) .

```
ZCL_payload = "FF 01 3412"
```

The first byte is the remaining capacity of the group table. 0xFF means unknown. The XBee returns this value because the capacity of the group table is dependent on the remaining capacity of the

binding table, thus the capacity of the group table is unknown. The second byte is the group count (0x01). The remaining bytes are the groupIDs in little-endian representation.

The following is the decoded second message, which is a Tx Status for the original command request. If the FrameId value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message occurs.

```
Zigbee Tx Status
API 0x8B        FrameID 0x01    16DestAddr 0xFFFE
Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success
```

Remove Group

The purpose of the Remote Group command is to remove a Group Table entry which associates a given endpoint with a given groupID.

The intent of the example is to remove the association of groupID [TBD] with endpoint E7.

The example packet is given in three parts: the preamble, ZCL Header, and ZCL payload.

```
Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"
```

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterID of 0x0006, and profileID of 0xC105. The destination endpoint E7 is the endpoint parameter for the "Remove Group" command.

```
ZCL_header = "01 ee 03"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Client->Server direction(0x00). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Remove Group" (0x03) .

```
ZCL_payload = "3412"
```

The two bytes value is the groupID to be removed in little-endian representation.

The packet in raw hex byte form:

```
7e001911010013a2004047b55cffffee6e70006c105000001ee033412d2
```

The response in raw hex byte form, consisting of two packets:

```
 7e0018910013a2004047b55cffffee7e68006c1050009ee0300341235
7e00078b01fffe00000076
```

The command response in decoded form:

```
Zigbee Explicit Rx Indicator
API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE        SrcE 0xE DestEP 0xE6
ClusterID 0x8006 ProfileID 0xC105    Options 0x00
RF_Data 0x09EE03003412
```

The response in terms of Preamble, ZCL Header, and ZCL payload:

```
 Preamble = "910013a2004047b55cffffee7e68006c10500"
```

The packet has its endpoint values reversed from the request, and the clusterID is 0x8006 indicating a Group cluster response.

```
ZCL_header = "09 ee 03"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server->Client direction (0x08). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Remove Group" (0x03) .

```
ZCL_payload = "00 3412"
```

The first byte is a status byte (SUCCESS=0x00). The next two bytes is the groupID (0x1234) value in little- endian form.

The following is the decoded second message, which is a Tx Status for the original command request. If the FrameId value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message would occur.

```
Zigbee Tx Status
API 0x8B        FrameID 0x01    16DestAddr 0xFFFE
Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success
```

Remove All Groups

The purpose of the Remove All Groups command is to clear all entries from the group table which are associated with a target endpoint.

The following example removes all groups associated with endpoint E7.

The packet:

```
Preamble = "11 01 "+LocalDevice64Addr+"FFFE E6 E7 0006 C105 00 00"
```

The packet is addressed to the local node, using a source endpoint of 0xE6, clusterId of 0x0006, and profileID of 0xC105. The destination endpoint E7 is the endpoint parameter for the "Remove All Groups" command.

```
ZCL_header = "01 ee 04"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Client->Server direction(0x00). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Remove All Groups" (0x04) .

```
ZCL_payload = ""
```

No payload is needed for this command.

The packet in raw hex byte form:

```
7e001711010013a2004047b55cffffee6e70006c105000001ee0417
```

The response in raw hex byte form, consisting of two packets:

```
7e0016910013a2004047b55cffffee7e68006c1050009ee04007c
7e00078b01fffe00000076
```

The command response in decoded form:

```
Zigbee Explicit Rx Indicator
API 0x91 64DestAddr 0x0013A2004047B55C 16DestAddr 0xFFFE        SrcEP 0xE7     DestEP
0xE6
ClusterID 0x8006 ProfileID 0xC105     Options 0x00
RF_Data 0x09ee0400
```

The response in terms of Preamble, ZCL Header, and ZCL payload.

```
Preamble = "910013a2004047b55cffffee7e68006c10500"
```

The packet has its endpoints values reversed from the request, and the clusterID is 0x8006 indicating a Group cluster response.

```
ZCL_header = "09 ee 04"
```

The first field is a frame control field which specifies a Cluster Specific command (0x01) using a Server->Client direction (0x08). The second field is a transaction sequence number which is used to associate the response with the command request. The third field is the command identifier "Remove All Groups" (0x04) .

```
ZCL_payload = "00"
```

The first byte is a status byte (SUCCESS=0x00)[4].

And here is the decoded second message, which is a Tx Status for the original command request. If the FrameID value in the original command request had been zero, or if no space was available in the transmit UART buffer, then no Tx Status message would occur.

```
Zigbee Tx Status
API 0x8B       FrameID 0x01    16DestAddr 0xFFFE
Transmit Retries 0x00 Delivery Status 0x00 Discovery Status 0x00 Success
```

Default responses

Many errors are returned as a default response. For example, an RFData payload of a response containing 08010b788b would be decoded as:

```
ZCL_header = "08 01 03" – general command/server-to-client, transseqnum=1,
default_response_command(0x03)
ZCL_payload = "78 8b" – original cmdID, status code (0x8b)
EMBER_ZCL_STATUS_NOT_FOUND
```

Common status codes

This section lists some of the more frequently occurring status codes.

```
0x00 EMBER_ZCL_STATUS_SUCCESS: Command request was successful
0x01 EMBER_ZCL_STATUS_FAILURE: Command request failed - for example,
a call to remove an entry from the group table returned an error
0x80 EMBER_ZCL_STATUS_MALFORMED_COMMAND: no RFData in the API frame;
ZCL Payload appears truncated from what is expected
0x81 EMBER_ZCL_STATUS_UNSUP_CLUSTER_COMMAND: unexpected direction
in the Frame Control Field of the ZCL Header; unexpected command identifier code
value
in the ZCL header
0x82 EMBER_ZCL_STATUS_UNSUP_GENERAL_COMMAND: unexpected frametype
in the Frame Control Field of the ZCL Header
0x84 EMBER_ZCL_STATUS_UNSUP_MANUF_GENERAL_COMMAND: unexpected
manufacturer specific indication in the Frame Control Field of the ZCL Header
0x8b EMBER_ZCL_STATUS_NOT_FOUND: An attempt at Get Group Membership or
 Remove Group could not find a matching entry in the group table
```

Manage End Devices
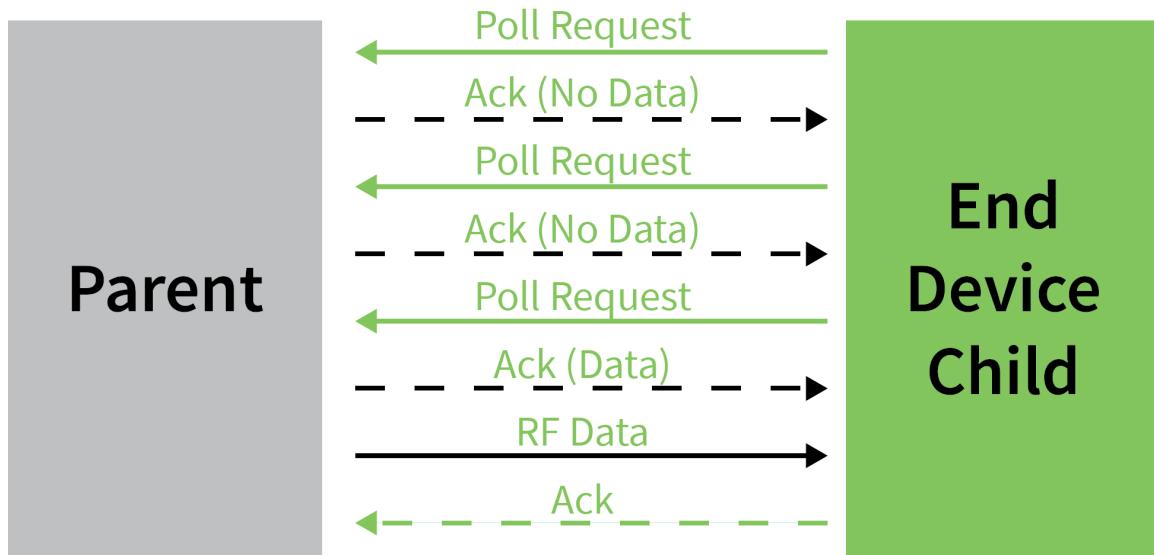
Zigbee end devices are intended to be battery-powered devices capable of sleeping for extended periods of time. Since end devices may not be awake to receive RF data at a given time, routers and coordinators are equipped with additional capabilities (including packet buffering and extended transmission timeouts) to ensure reliable data delivery to end devices.

End device operation

When an end device joins a Zigbee network, it must find a router or coordinator device that is allowing end devices to join. Once the end device joins a network, it forms a parent-child relationship with the end device and the router or coordinator that allowed it to join. For more information, see Zigbee networks.

When the end device is awake, it sends poll request messages to its parent. When the parent receives a poll request, it checks a packet queue to see if it has any buffered messages for the end device. It then sends a MAC layer acknowledgment back to the end device that indicates if it has data to send to the end device or not.

Parent		End Device Child
	← Poll Request	
	Ack (No Data) →	
	← Poll Request	
	Ack (No Data) →	
	← Poll Request	
	Ack (Data) →	
	RF Data →	
	← Ack	

If the end device receives the acknowledgment and finds that the parent has no data for it, the end device can return to idle mode or sleep. Otherwise, it remains awake to receive the data. This polling mechanism allows the end device to enter idle mode and turn its receiver off when RF data is not expected in order to reduce current consumption and conserve battery life.

The end device can only send data directly to its parent. If an end device must send a broadcast or a unicast transmission to other devices in the network, it sends the message directly to its parent and the parent performs any necessary route or address discoveries to route the packet to the final destination.

The parent of the receiving device does not send the network ACK back to the originator until the sleeping end device wakes and polls the data or until the timeout occurs.

Parent operation

Each router or coordinator maintains a child table that contains the addresses of its end device children. A router or coordinator that has unused entries in its child table has end device capacity, or the ability to allow new end devices to join. If the child table is completely filled (such that the number of its end device children matches the number of child table entries), the device cannot allow any more end devices to join.

Since the end device children are not guaranteed to be awake at a given time, the parent is responsible for managing incoming data packets of its end device children. If a parent receives an RF data transmission destined for one of its end device children, and if the parent has enough unused buffer space, it buffers the packet. The data packet remains buffered until a timeout expires, or until the end device sends a poll request to retrieve the data.

The parent can buffer one broadcast transmission for all of its end device children. When the parent receives and buffers a broadcast transmission, it sets a flag in its child table when each child polls and retrieves the packet. Once all children have received the broadcast packet, the parent discards the buffered broadcast packet. If all children have not received a buffered broadcast packet and the parent receives a new broadcast, it discards the old broadcast packet, clears the child table flags, and buffers the new broadcast packet for the end device children as shown in the following figure.

End Device Child Table

	Address	Received Broadcast
Buffered Broadcast Data Packet	Ox2120	T
	OxF220	F
	OxC100	F
	Ox5750	T

When an end device sends data to its parent that is destined for a remote device in the network, the parent buffers the data packet until it can establish a route to the destination. The parent may perform a route or 16-bit address discovery of its end device children. Once a route is established, the parent sends the data transmission to the remote device.

End Device poll timeouts

To better support mobile end devices (end devices that can move within a network), parent router and coordinator devices have a poll timeout for each end device child. If an end device does not send a poll request to its parent within the poll timeout, the parent removes the end device from its child table. This allows the child table on a router or coordinator to better accommodate mobile end devices in the network.

Packet buffer usage

Packet buffer usage on a router or coordinator varies depending on the application. The following activities can require use of packet buffers for up to several seconds:

- Route and address discoveries
- Application broadcast transmissions
- Stack broadcasts (for example ZDO "Device Announce" messages when devices join a network)
- Unicast transmissions buffered until acknowledgment is received from destination or retries exhausted
- Unicast messages waiting for end device to wake

Applications that use regular broadcasting or that require regular address or route discoveries use up a significant number of buffers, reducing the buffer availability for managing packets for end device children. Applications can reduce the number of required application broadcasts, and consider implementing an external address table or many-to-one and source routing if necessary to improve routing efficiency.

Non-Parent device operation

Devices in the Zigbee network treat data transmissions to end devices differently than transmissions to other routers and coordinators. When a device sends a unicast transmission, if it does not receive a network acknowledgment within a timeout, the device resends the transmission. When transmitting data to remote coordinator or router devices, the transmission timeout is relatively short since these devices are powered and responsive.

However, since end devices may sleep for some time, unicast transmissions to end devices use an extended timeout mechanism in order to allow enough time for the end device to wake and receive the data transmission from its parent.

If a non-parent device does not know the destination is an end device, it uses the standard unicast timeout for the transmission. However, provisions exist in the Ember Zigbee stack for the parent to inform the message sender that the destination is an end device. Once the sender discovers the destination device is an end device, future transmissions will use the extended timeout. For more information see Router/Coordinator configuration.

End Device configuration

XBee end devices support three different sleep modes:

- Pin sleep
- Cyclic sleep
- Cyclic sleep with pin wake-up

Pin sleep allows an external microcontroller to determine when the XBee/XBee-PRO Zigbee RF Module sleeps and when it wakes by controlling the Sleep_RQ pin. In contrast, cyclic sleep allows the sleep period and wake times to be configured through the use of AT commands. Cyclic sleep with pin wake-up is the same as cyclic sleep except the device can be awakened before the sleep period expires by lowering the SLEEP_RQ line. The **SM** command configures the sleep mode.

In both pin and cyclic sleep modes, XBee end devices poll their parent every 100 ms while they are awake to retrieve buffered data. When the end device sends a poll request, it enables the receiver until it receives an acknowledgment from the parent. It typically takes less than 10 ms between sending the poll request to receiving the acknowledgment. The acknowledgment indicates if the parent has buffered data for the end device child. If the acknowledgment indicates the parent has pending data, the end device leaves the receiver on to receive the data. Otherwise, the end device turns off the receiver and enter idle mode (until it sends the next poll request) to reduce current consumption (and improve battery life).

Once the device enters sleep mode, the On/Sleep pin (TH pin 13/SMT pin 26) it de-asserts (low) to indicate the device is entering sleep mode. If the device enables CTS hardware flow control (**D7** command), it de-asserts (high) the CTS pin (TH pin 12/SMT pin 25) when entering sleep to indicate that serial data should not be sent to the device.

If the Associate LED pin is configured (**D5** command), the associate pin is driven low to avoid using power to light the LED. The Sleep_Rq pin is configured as a pulled-down input so that an external device must drive it low to wake the device. All other pins are left unmodified during sleep so that they can operate as previously configured by the user. The device does not respond to serial or RF data when it is sleeping.

Applications that must communicate serially to sleeping end devices are encouraged to observe CTS flow control.

When the device wakes from sleep, it asserts (high) the On/Sleep pin, and if it enables flow control, it also asserts (low) the CTS pin. The associate LED and all other pins resume their former configured
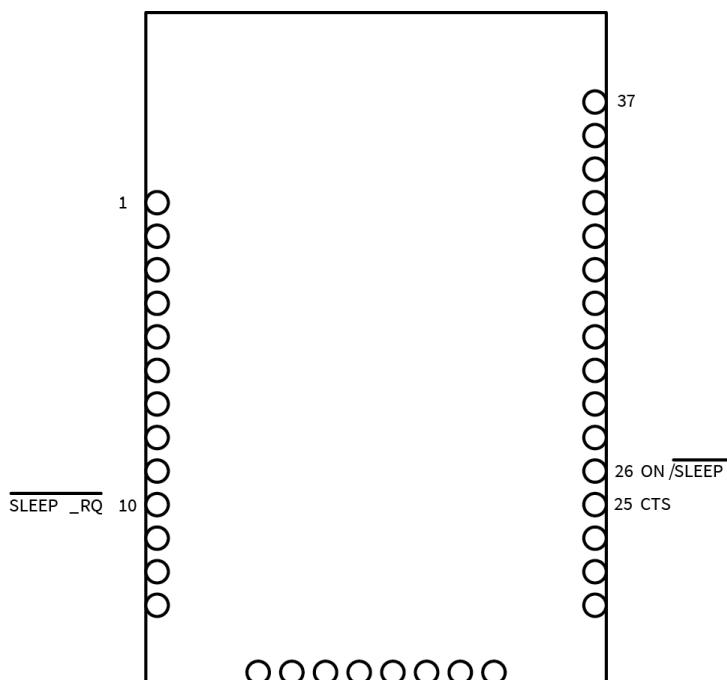
operation. If the device has not joined a network, it scans all **SC** channels after waking to try and find a valid network to join.

Pin sleep
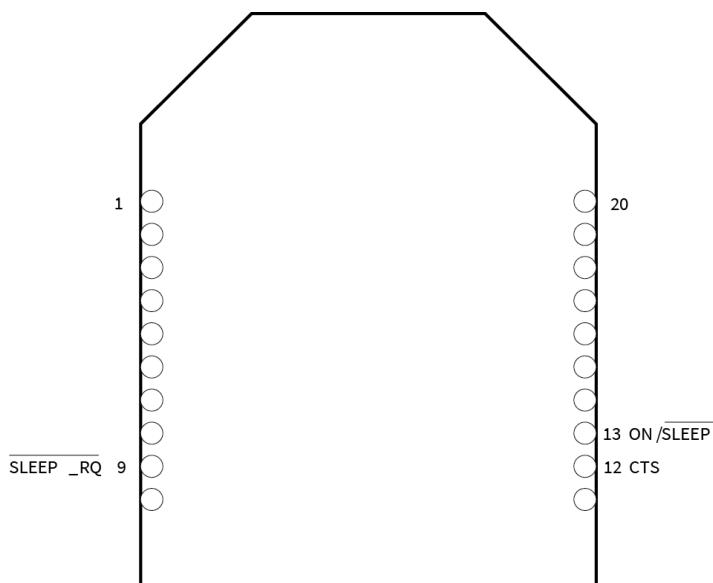
Pin sleep allows the module to sleep and wake according to the state of the SLEEP_RQ pin (TH pin 9/SMT pin 10). Pin sleep mode is enabled by setting the **SM** command to 1.

When the device asserts (high) SLEEP_RQ, it finishes any transmit or receive operations for the current packet that is processing and enters a low power state. For example, if the device has not joined a network and SLEEP_RQ is asserted (high), it sleeps once the current join attempt completes (that is, when scanning for a valid network completes). The device wakes from pin sleep when the SLEEP_RQ pin is de-asserted (low). The following figures show the device's sleep pins.
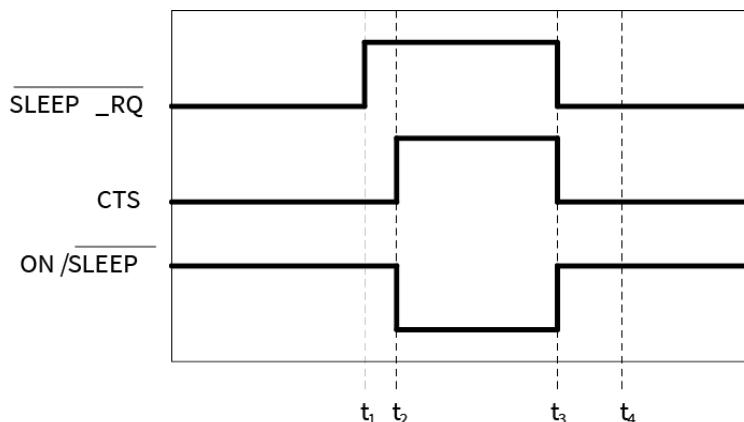
Surface-mount sleep pins

Through-hole sleep pins

The following figure show the pin sleep waveforms:

In the previous figure, t1, t2, t3 and t4 represent the following events:

- t1 - Time when Sleep_RQ is asserted (high)
- t2 - Time when the device enters sleep ($\overline{CTS}$ state change only if hardware flow control is enabled)
- t3 - Time when Sleep_RQ is de-asserted (low) and the device wakes
- t4 - Time when the module sends a poll request to its parent

The time between t1 and t2 varies depending on the state of the module. In the worst case scenario, if the end device is trying to join a network, or if it is waiting for an acknowledgment from a data transmission, the delay could be up to a few seconds. The time between t3 and t4 is 1-2 ms for a regular device and about 6 ms for a PRO device.

When the XBee/XBee-PRO Zigbee RF Module is awake and is joined to a network, it sends a poll request to its parent to see if the parent has any buffered data. The end device continues to send poll requests every 100 ms while it is awake.

Demonstration of pin sleep

Parent and remote devices must be configured to buffer data correctly and to use adequate transmission timeouts. For more information, see Router/Coordinator configuration.

Cyclic sleep
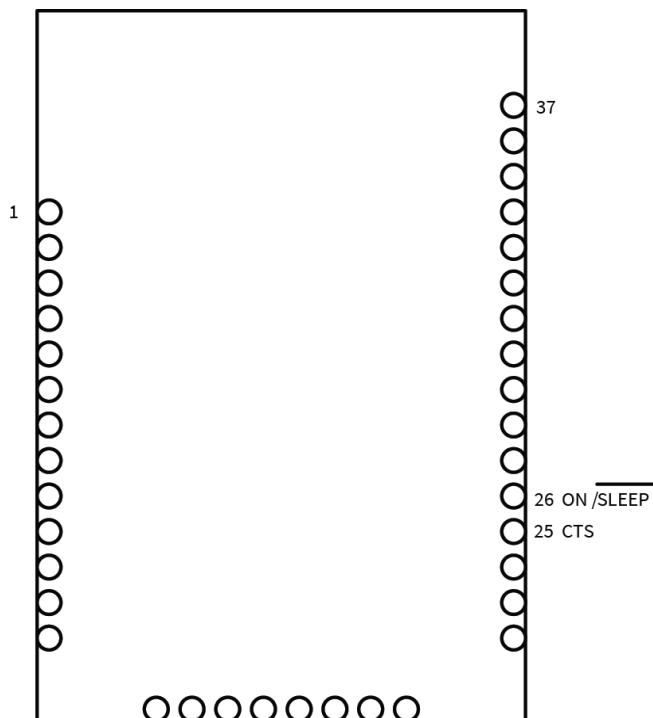
Cyclic sleep allows the device to sleep for a specified time and wake for a short time to poll its parent for any buffered data messages before returning to sleep again. Enable cyclic sleep mode by setting the **SM** command to 4 or 5. **SM**5 is a slight variation of **SM**4 that allows the device to wake up prematurely by asserting the Sleep_RQ pin(pin 10/SMT, pin 9/TH). In **SM**5, the XBee device can wake after the sleep period expires, or if a high-to- low transition occurs on the Sleep_RQ pin. Setting **SM** to 4 disables the pin wake option.

In cyclic sleep, the device sleeps for a specified time, and then wakes and sends a poll request to its parent to discover if the parent has any pending data for the end device. If the parent has buffered data for the end device, or if it receives serial data, the device remains awake for a time. Otherwise, it enters sleep mode immediately.
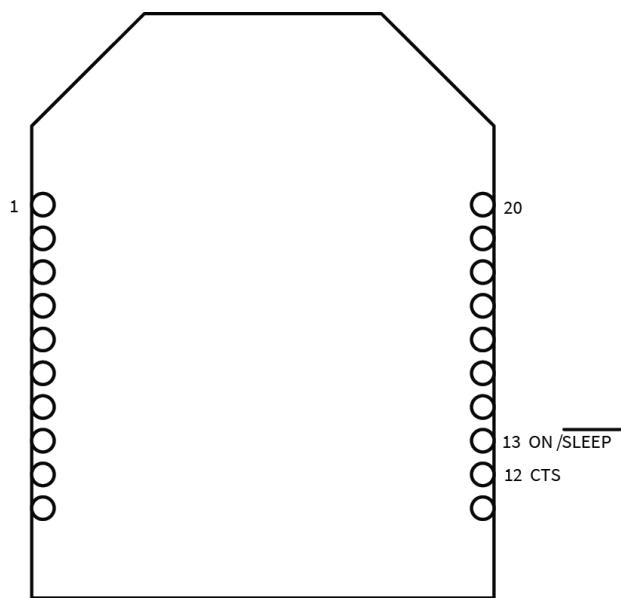
When the device wakes, it asserts (high) the On/Sleep line, and de-asserted (low) when the device sleeps. If you enable hardware flow control (**D7** command), the CTS pin asserts (low) when the device wakes and can receive serial data, and de-assert (high) when the device sleeps.

The following figure shows the XBee sleep pins.
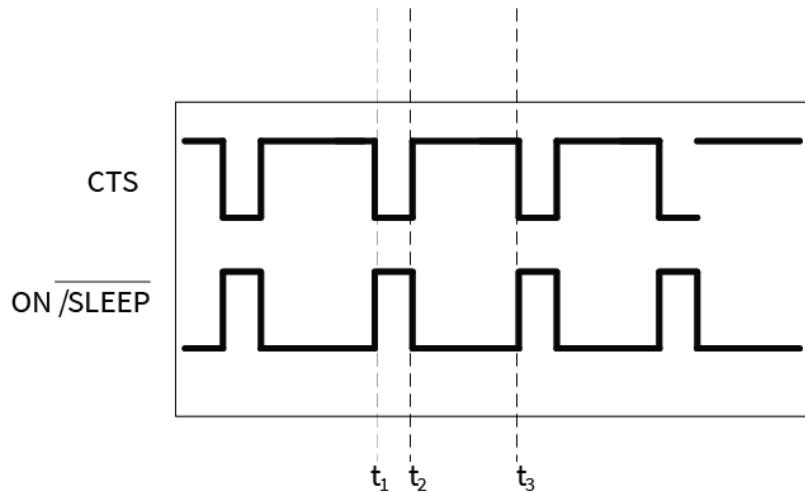
Surface-mount cyclic sleep pins

S2C Through-hole cyclic sleep pins

The following figure shows the cyclic sleep waveforms.
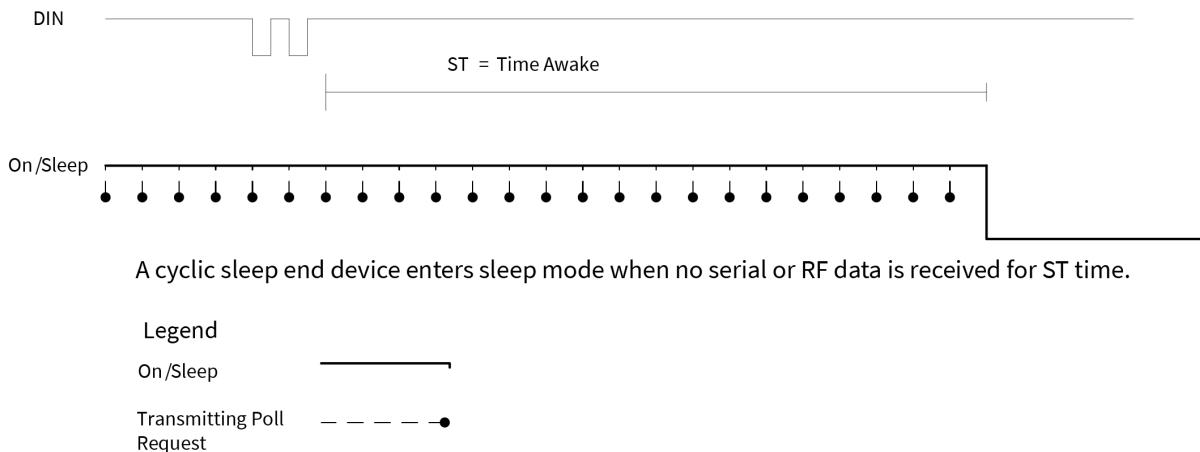
In the figure above, t1, t2, and t3 represent the following events:

- t1 - Time when the device wakes from cyclic sleep
- t2 - Time when the device returns to sleep
- t3 - Later time when the device wakes from cyclic sleep

The wake time and sleep time are configurable with software commands.

Wake time (until sleep)

In cyclic sleep mode (**SM** = 4 or 5), if the device receives serial or RF data, it starts a sleep timer (time until sleep). Any data received serially or over the RF link restarts the timer. Set the sleep timer value with ST (Time before Sleep). While the device is awake, it sends poll request transmissions every 100 ms to check its parent for buffered data messages. The device returns to sleep when the sleep timer expires, or if it receives SI (Sleep Immediately) as shown in the following image.

A cyclic sleep end device enters sleep mode when no serial or RF data is received for ST time.

Legend

On /Sleep

Transmitting Poll
Request

Sleep period

Configure the sleep period based on the **SP**, **SN**, and **SO** commands. The following table lists the behavior of these commands.

Command	Range	Description
SP	0x20 - 0xAF0 (x 10 ms) (320 - 28,000 ms)	Configures the sleep period of the device.
SN	1 - 0xFFFF	Configures the number of sleep periods multiplier.
SO	0 - 0xFF	Defines options for sleep mode behavior. 0x02 - Always wake for full **ST** time 0x04 - Enable extended sleep (sleep for full (**SP** * **SN**) time)

The device supports both a Short cyclic sleep and an Extended cyclic sleep that make use of these commands. These two modes allow the sleep period to be configured according to the application requirements.
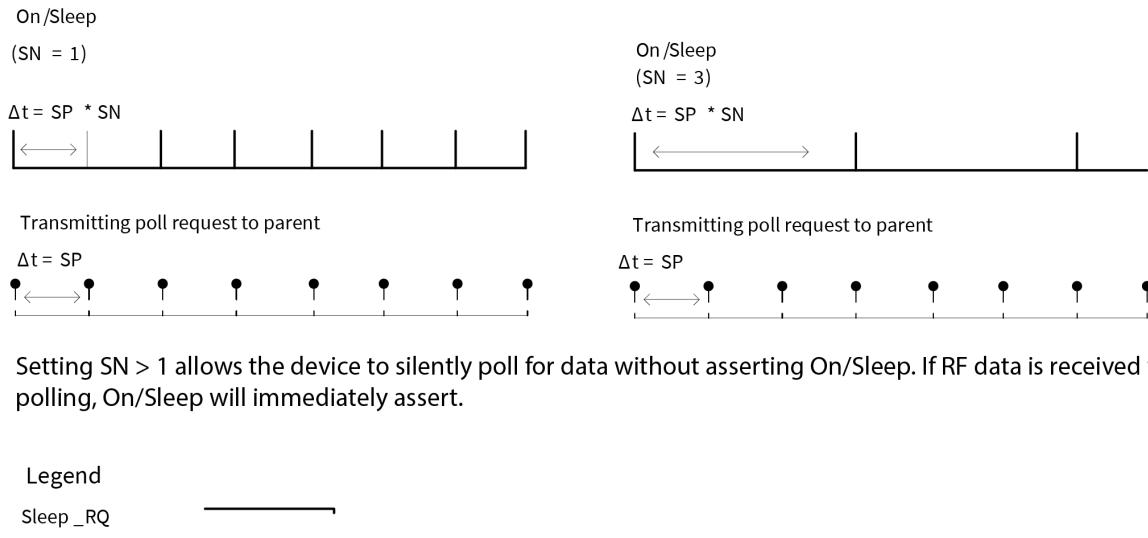
Short cyclic sleep

In short cyclic sleep mode, define the sleep behavior of the device by the **SP** and **SN** commands, and the **SO** command must be set to 0x00 (default) or 0x02. In short cyclic sleep mode, the **SP** command defines the sleep period and you can set it for up to 28 seconds. When the device enters short cyclic sleep, it remains in a low power state until the **SP** time has expired.

After the sleep period expires, the XBee/XBee-PRO Zigbee RF Module sends a poll request transmission to its parent to determine if the parent has any buffered data waiting for the end device. Since router and coordinator devices can buffer data for end device children up to 30 seconds, the **SP** range (up to 28 seconds) allows the end device to poll regularly enough to receive buffered data. If the parent has data for the end device, the end device starts its sleep timer (**ST**) and continues polling every 100 ms to receive data. If the end device wakes and finds that its parent has no data for it, the end device can return to sleep immediately.

Use the **SN** command to control when the On/Sleep line is asserted (high). If you **SN** to 1 (default), the On/Sleep line sets high each time the device wakes from sleep. Otherwise, if **SN** is greater than 1, the On/ Sleep line only sets high if RF data is received, or after **SN** wake cycles occur. This allows an external device to remain powered off until it receives RF data, or until a number of sleep periods have expired (**SN** sleep periods). This mechanism allows the device to wake at regular intervals to poll its

parent for data without waking an external device for an extended time (**SP** * **SN** time) as shown in the following figure.

On/Sleep
(SN = 1)

Δt = SP * SN

Transmitting poll request to parent

Δt = SP

On/Sleep
(SN = 3)

Δt = SP * SN

Transmitting poll request to parent

Δt = SP

Setting SN > 1 allows the device to silently poll for data without asserting On/Sleep. If RF data is received when polling, On/Sleep will immediately assert.

Legend

Sleep_RQ

Transmitting Poll Request

Note **SP** controls the packet buffer time on routers and coordinators. Set **SP** on all router and coordinator devices to match the longest end device **SP** time. For more information, see Router/Coordinator configuration.

Extended cyclic sleep

In extended cyclic sleep operation, an end device can sleep for a multiple of **SP** time which can extend the sleep time up to several days. Configure the sleep period using the **SP** and **SN** commands. The total sleep period is equal to (**SP** * **SN**) where **SP** is measured in 10ms units. The **SO** command must be set correctly to enable extended sleep.

Since routers and coordinators can only buffer incoming RF data for their end device children for up to 30 seconds, if an end device sleeps longer than 30 seconds, devices in the network need some indication when an end device is awake before they can send data to it. End devices that use extended cyclic sleep should send a transmission (such as an **I/O** sample) when they wake to inform other devices that they are awake and can receive data. We recommended that extended sleep end devices set **SO** to wake for the full **ST** time to provide other devices with enough time to send messages to the end device.

Similar to short cyclic sleep, end devices running in this mode return to sleep when the sleep timer expires, or when they receive the **SI** command.

Deep sleep

The following are preconditions for maintaining low current draw during sleep:

- You must maintain the supply voltage within a valid operating range (2.1 to 3.6 V for the XBee, 3.0 to 3.6 V for the XBee-PRO (S2), 2.7 to 3. V for the XBee-PRO S2B).
- Each GPIO input line with a pullup resistor which is driven low draws about 100 uA current through the internal pullup resistor.

- If circuitry external to the XBee drives such input lines low, then the current draw rises above expected deep sleep levels.
- Each GPIO input line that has no pullup or pull-down resistor (is floating) has an indeterminate voltage which can change over time and temperature in an indeterminate manner.

Recommended sleep current measurements

Properly measuring the sleep current helps to accurately estimate battery life requirements. To ensure that you take proper measurements without upsetting the normal operation of the unit under test, read the following steps.
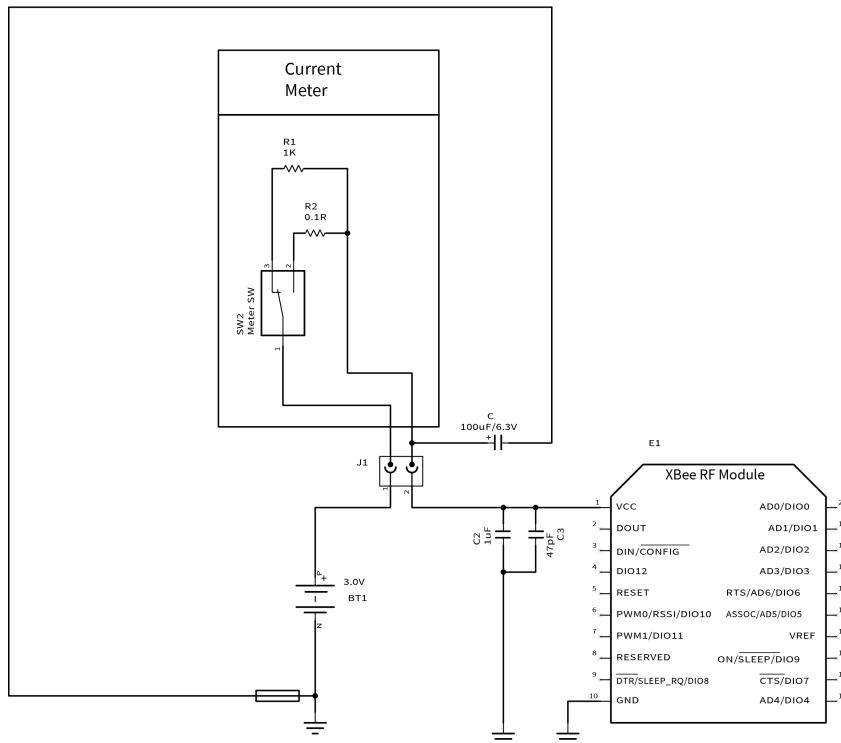
When you measure sleep currents, it can cause problems with the devices because the equipment that measures very low currents accurately requires a large resistor in series with the power supply. This large resistor starves current from the device during a momentary wake cycle, forcing the voltage to drop to brownout levels rapidly. This voltage drop places the device in a state that may require a reset to resolve the problem.

Achieve the lowest sleep current

To achieve the lowest sleep current, you must disable brownout detectors during sleep modes. Even if the measurement equipment automatically changes current ranges, it is often too slow and cannot keep up with the necessary sudden short bursts. During long cyclic sleep periods, the device can wake every 10 to 30 seconds to reset timers and perform other necessary steps. These wake times are small and you may not notice them when measuring sleep currents.

Compensate for switching time

To compensate for the switching time of the equipment you must temporarily add an additional large cap when you need measurements to allow for short pulses of current draw (see the following schematic for details). A cap of 100 uF is enough to handle 1.5 milliseconds with 20 mA of current. You can increase or decrease the capacitor based on the switching time of the measurement circuitry and the momentary on time of the unit. Measure the leakage current of the additional cap to verify that it does not skew the low current reading. The capacitor averages the spike in current draw. The actual magnitude of the current spike is no longer visible, but you can account for the total energy consumed by integrating the current over time and multiplying by the voltage.

Internal pin pull-ups

Internal pin pull-ups can pull excess current and cause the sleep current readings to be higher than desired if you drive or float the pull-ups.

- Disable all pull-ups for input lines that have a low driven state during sleep.
- Enable pull-ups for floating lines or inputs that do not connect to other circuitry.

If you use an analog-to-digital converter (ADC) to read the analog voltage of a pin, it may not be possible to stop all leakage current unless you can disconnect the voltage during sleep. Each floating input that is not at a valid high or low level can cause leakage depending on the temperature and charge buildup that you may not observe at room temperature.

Transmit RF data

An end device may transmit data when it wakes from sleep and has joined a network. End devices transmit directly to their parent and then wait for an acknowledgment to be received. The parent performs any required address and route discoveries to help ensure the packet reaches the intended destination before reporting the transmission status to the end device.

Receiving RF data

After waking from sleep, an end device sends a poll request to its parent to determine if the parent has any buffered data for it. In pin sleep mode, the end device polls every 100 ms while the Sleep_RQ pin is de-asserted (low). In cyclic sleep mode, the end device will only poll once before returning to sleep unless the sleep timer (**ST**) is started (serial or RF data is received). If the sleep timer is started, the end device will continue to poll every 100 ms until the sleep timer expires.

This firmware includes an adaptive polling enhancement where, if an end device receives RF data from its parent, it sends another poll after a very short delay to check for more data. The end device continues to poll at a faster rate as long as it receives data from its parent. This feature greatly improves data throughput to end devices. When the end device no longer receives data from its parent, it resumes polling every 100 ms.

I/O sampling

End devices can be configured to send one or more I/O samples when they wake from sleep. To enable I/O sampling on an end device, the IR command must be set to a non-zero value, and at least one analog or digital I/O pin must be enabled for sampling (**D0** - **D9**, **P0** - **P4** commands). If I/O sampling is enabled, an end device sends an I/O sample when it wakes and starts the **ST** timer. It will continue sampling at the **IR** rate until the sleep timer (**ST**) has expired. For more information, see Analog and digital I/O lines.

Wake end devices with the Commissioning Pushbutton

If you use D0 (AD0/DIO0 Configuration) to enable the Commissioning Pushbutton functionality, a high-to-low transition on the AD0/DIO0 pin (TH pin 20/SMT pin 33) causes an end device to wake for 30 seconds. For more information, see Commissioning pushbutton and associate LED.

Parent verification

Since an end device relies on its parent to maintain connectivity with other devices in the network, XBee end devices include provisions to verify the connection with its parent. End devices monitor the link with their parent when sending poll messages and after a power cycle or reset event as described below.

When an end device wakes from sleep, it sends a poll request to its parent. In cyclic sleep, if the end device does not receive RF or serial data and the sleep timer is not started, it polls one time and returns to sleep for another sleep period. Otherwise, the end device continues polling every 100ms. If the parent does not send an acknowledgment response to three consecutive poll request transmissions, the end device assumes the parent is out of range, and attempts to find a new parent.

After a power-up or reset event, the end device does an orphan scan to locate its parent. If the parent does not send a response to the orphan scan, the end device attempts to find a new parent.

Rejoining

Once all devices have joined a Zigbee network, disable the permit-joining attribute disabled such that new devices are no longer allowed to join the network. You can enable permit-joining later as needed for short times. This provides some protection in preventing other devices from joining a live network.

If an end device cannot communicate with its parent, the end device must be able to join a new parent to maintain network connectivity. However, if permit-joining is disabled in the network, the end device will not find a device that is allowing new joins.

To overcome this problem, Zigbee supports rejoining, where an end device can obtain a new parent in the same network even if joining is not enabled. When an end device joins using rejoining, it performs a PAN ID scan to discover nearby networks. If a network is discovered that has the same 64-bit PAN ID as the end device, it joins the network by sending a rejoin request to one of the discovered devices. The device that receives the rejoin request sends a rejoin response if it can allow the device to join the network (that is, the child table is not full). You can use the rejoin mechanism to allow a device to join the same network even if permit-joining is disabled.

To enable rejoining, set **NJ** to less than 0xFF on the device joining. If **NJ** < 0xFF, the device assumes the network is not allowing joining and first tries to join a network using rejoining. If multiple rejoining attempts fail, or if **NJ** = 0xFF, the device attempts to join using association.

Router/Coordinator configuration

XBee routers and coordinators may require some configuration to ensure the following are set correctly.

- RF Packet buffering timeout
- Child poll timeout
- Transmission timeout

The value of these timeouts depends on the sleep time used by the end devices.

RF packet buffering timeout

When a router or coordinator receives an RF data packet intended for one of its end device children, it buffers the packet until the end device wakes and polls for the data, or until a packet buffering timeout occurs. Use the **SP** command to set the timeout . The actual timeout is (1.2 * SP), with a minimum timeout of 1.2 seconds and a maximum of 30 seconds. Since the packet buffering timeout is set slightly larger than the **SP** setting, set **SP** the same on routers and coordinators as it is on cyclic sleep end devices. For pin sleep devices, set **SP** as long as the pin sleep device can sleep, up to 30 seconds.

Note In pin sleep and extended cyclic sleep, end devices can sleep longer than 30 seconds. If end devices sleep longer than 30 seconds, parent and non-parent devices must know when the end device is awake in order to reliably send data. For applications that require sleeping longer than 30 seconds, end devices should transmit an **I/O** sample or other data when they wake to alert other devices that they can send data to the end device.

Child poll timeout

Router and coordinator devices maintain a timestamp for each end device child indicating when the end device sent its last poll request to check for buffered data packets. If an end device does not send a poll request to its parent for a certain period of time, the parent assumes the end device has moved out of range and removes the end device from its child table. This allows routers and coordinators to be responsive to changing network conditions. You can send the **NC** command at any time to read the number of remaining (unused) child table entries on a router or coordinator.

Set the child poll timeout with the **SP** and **SN** commands. **SP** and **SN** should be set such that **SP** * **SN** matches the longest expected sleep time of any end devices in the network. The device calculates the actual timeout as (3* **SP** * **SN**), with a minimum of 5 seconds. For networks consisting of pin sleep end devices, set the **SP** and **SN** values on the coordinator and routers so the **SP** * **SN** matches the longest expected sleep period of any pin sleep device. The 3 multiplier ensures the end device will not be removed unless 3 sleep cycles pass without receiving a poll request. You can set the poll timeout up to two months.

Adaptive polling

The PO command determines the regular polling rate. However, if RF data has been recently received by an end device, it is likely that more RF data could be on the way. Therefore, the end device polls at

a faster rate, gradually decreasing its adaptive poll rate until polling resumes at the regular rate as defined by the **PO** command.

Transmission timeout

When you are sending RF data to a remote router, because routers are always on, the timeout is based on the number of hops the transmission may traverse. Set the timeout using the **NH** command. For more information, see Transmission, addressing, and routing.

Since end devices may sleep for lengthy periods of time, the transmission timeout to end devices also allows for the sleep period of the end device. When sending data to a remote end device, the transmission timeout is calculated using the **SP** and **NH** commands. If the timeout occurs with no acknowledgment received, the source device re-sends the transmission until it receives an acknowledgment, up to two more times.

The transmission timeout per attempt is:

> 3 * ((unicast router timeout) + (end device sleep time))
>
> 3 * ((50 * **NH**) + (1.2 * **SP**)), where **SP** is measured in 10 ms units.

Short sleep periods

Pin and cyclic sleep devices that sleep less than 30 seconds can receive data transmissions at any time since their parent devices are able to buffer data long enough for the end devices to wake and poll to receive the data. Set **SP** the same on all devices in the network. If end devices in a network have more than one **SP** setting, set **SP** on the routers and coordinators to match the largest **SP** setting of any end device. This ensure the RF packet buffering, poll timeout, and transmission timeouts are set correctly.

Extended sleep periods

Pin and cyclic sleep devices that might sleep longer than 30 seconds cannot receive data transmissions reliably unless you take certain design approaches. Specifically, the end devices should use I/O sampling or another mechanism to transmit data when they wake to inform the network they can receive data. **SP** and **SN** should be set on routers and coordinators such that (**SP** * **SN**) matches the longest expected sleep time. This configures the poll timeout so end devices are not expired from the child table unless routers and coordinators do not receive a poll request for 3 consecutive sleep periods.

As a general rule, **SP** and **SN** should be set the same on all devices in almost all cases.

Sleep examples

Some sample XBee configurations to support different sleep modes follow. In Command mode, issue each command with a leading **AT** and no **=** sign, for example, **ATSM4**. In the API, the two byte command is used in the command field, and parameters are populated as binary values in the parameter field.

Example 1: Configure a device to sleep for 20 seconds, but set SN such that the On/sleep line will remain de-asserted for up to 1 minute

The following settings should be configured on the end device.

- **SM** = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake).
- **SP** = 0x7D0 (2000 decimal). This causes the end device to sleep for 20 seconds since **SP** is measured in units of 10 ms.
- **SN** = 3. (With this setting, the On/Sleep pin asserts once every 3 sleep cycles, or when it receives RF data) **SO** = 0.

Set all router and coordinator devices on the network **SP** to match **SP** on the end device. This set the RF packet buffering times and transmission timeouts correctly.

Since the end device wakes after each sleep period (**SP**), you can set the **SN** command to 1 on all routers and the coordinator.

Example 2: Configure an end device to sleep for 20 seconds, send 4 I/O samples in 2 seconds, and return to sleep

Because **SP** is measured in 10 ms units, and **ST** and **IR** are measured in 1 ms units, configure an end device with the following settings:

- **SM** = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake).
- **SP** = 0x7D0 (2000 decimal). This causes the end device to sleep for 20 seconds.
- **SN** = 1.
- **SO** = 0.
- **ST** = 0x7D0 (2000 decimal). This sets the sleep timer to 2 seconds.
- **IR** = 0x258 (600 decimal). Set **IR** to a value greater than (2 seconds / 4) to get 4 samples in 2 seconds. The end device sends an I/O sample at the **IR** rate until the sleep timer has expired.

You must enable at least one analog or digital I/O line for I/O sampling to work. To enable AD1/DIO1 (TH pin 19/SMT pin 32) as a digital input line, you must set the following:

D1 = 3

Set all router and coordinator devices on the network **SP** to match **SP** on the end device. This ensures that RF packet buffering times and transmission timeouts are set correctly.

Example 3: configure a device for extended sleep: to sleep for 4 minutes

- **SP** and **SN** must be set such that **SP** * **SN** = 4 minutes. Since **SP** is measured in 10 ms units, use the following settings to obtain 4 minute sleep.
- **SM** = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake) **SP** = 0x7D0 (2000 decimal, or 20 seconds).
- **SN** = 0x0C (12 decimal).
- **SO** = 0x04 (enable extended sleep).

With these settings, the module sleeps for **SP** * **SN** time, or (20 seconds * 12) = 240 seconds = 4 minutes.

For best results, the end device should send a transmission when it wakes to inform the coordinator (or network) when it wakes. It should also remain awake for a short time to allow devices to send data to it. The following are recommended settings.

- **ST** = 0x7D0 (2 second wake time)
- **SO** = 0x06 (enable extended sleep and wake for ST time)

- **IR** = 0x800 (send 1 I/O sample after waking). Enable at least one analog or digital I/O sample enabled for I/O sampling.

With these settings, the end device wakes after 4 minutes and sends 1 I/O sample. It then remains awake for 2 seconds before returning to sleep.

Set **SP** and **SN** to the same values on all routers and coordinators that could potentially allow the end device to join. This ensures the parent does not timeout the end device from its child table too quickly.

The **SI** command can optionally be sent to the end device to cause it to sleep before the sleep timer expires.

Analog and digital I/O lines

XBee ZB firmware supports a number of analog and digital I/O pins that are configured through software commands. Analog and digital I/O lines can be set or queried.

Configurable I/O pins and configuration commands

The following tables list the configurable I/O pins and the corresponding configuration commands.

Module Pin Names	Module Pin	AT Command	Command Range
DOUT/DIO13	3	**P3**	0, 1, 3-5
DIN/CONFIG/DIO14	4	**P4**	0, 1, 3-5
RSSI PWM/DIO10	7	**P0**	0, 1, 3-5
PWM1/DIO11	8	**P1**	0, 1, 3-5
DTR/SLEEP_RQ /DIO8	10	**D8**	0, 1, 3-5
SPI_ATTN/ BOOTMODE/DIO19	12	**P9**	0, 1, 6
SPI_SClk/DIO18	14	**P8**	0, 1
SPI_SSEl/DIO17	15	**P7**	0, 1
SPI_MOSI/DIO16	16	**P6**	0, 1
SPI_MISO/DIO15	17	**P5**	0, 1
[reserved]*	21	**P2**	0, 3-5
DIO4	24	**D4**	0, 3-5
CTS/DIO7	25	**D7**	0, 1, 3-7
ON/SLEEP/DIO9	26	**D9**	0, 1, 3-5
ASSOCIATE/DIO5	28	**D5**	0, 1, 3-5
RTS/DIO6	29	**D6**	0, 1, 3-5
AD3/DIO3	30	**D3**	0, 2-5
AD2/DIO2	31	**D2**	0, 2-5
AD1/DIO1	32	**D1**	0, 2-6
AD0/DIO0	33	**D0**	0-5

XBee ZB through-hole RF module

Module Pin Names	Module Pin	AT Command	Command Range
DIO13/DOUT	2	**P3**	0, 1, 3-5
DIO14/DIN/CONFIG	3	**P4**	0, 1, 3-5
DIO12/PWM2/SWDIO/SPI_MISO	4	**P2**	
DIO10/PWM RSSI/DAC0	6	**P0**	0, 1, 3-5

Module Pin Names	Module Pin	AT Command	Command Range
DIO11/PWM1/DAC1	7	**P1**	0, 1, 3-5
DIO8/DTR/SLP_RQ	9	**D8**	0, 1, 3-5
DIO4/SPI_MOSI	11	**D4**	0, 1, 3-5
DIO7/CTS	12	**D7**	0, 1, 3-7
DIO9/On/SLEEP/SWO	13	**D9**	0, 1, 3-5
DIO5/ASSOC/JTDI	15	**D5**	0, 1, 3-5
DIO6/RTS	16	**D6**	0, 1, 3-5
DIO3/AD3/SPI_SSEL	17	**D3**	0-5
DIO2/AD2/SPI_SCLK	18	**D2**	0-5
DIO1/AD1/SPI_ATTN	19	**D1**	0-6
DIO0/AD0/CommBtn	20	**D0**	0-5

I/O configuration

To enable an analog or digital I/O function on one or more XBee/XBee-PRO Zigbee RF Module pin, you must issue the appropriate configuration command with the correct parameter. After issuing the configuration command, you must apply changes on the device for the I/O settings to take effect.

Pin command parameter	Description
0	Disabled
1	Peripheral control
2	Analog
3	Data in monitored (see the information following the table)
4	Data out default low
5	Data out default high
6	RS-485 enable low/packet trace interface
7	RS-485 enable high
>7	Unsupported

When the pin command parameter is a 0 or a 3, it operates the same on this platform, except the device does not monitor the pin by I/O sampling if the parameter is 0.

Inputs have three variations:

- Floating
- Pulled-up
- Pulled-down

A floating input is appropriate if the pin is attached to an output that always drives the line. In this case, a pull-up or pull-down resistor draws more current.

A pulled-up input is useful where there might not always be an external source to drive the pin and it is desirable to have the line read high in the absence of an external driver.

Likewise, a pulled-down input is useful when there is not always an external source to drive the pin and it is desirable to have the line read low in the absence of an external driver.

Two commands are available to configure the input type:

- **PR** determines whether or not to pull an input. If the corresponding bit in **PR** is set, the signal pulls. If it is clear, then the signal floats.
- **PD** determines the pull direction. It only applies when the corresponding bit in **PR** is set. Set the bit in PD to enable an internal pull-up resistor; clear it to enable an internal pull-down resistor.

I/O sampling

The XBee/XBee-PRO Zigbee RF Modules have the ability to monitor and sample analog and digital I/O lines. I/O samples can be read locally or transmitted to a remote device to provide an indication of the current I/O line states. You must enable API mode on the receiving device to send I/O samples out the serial port. If you do not enable this mode, the device discards the remote I/O samples.

There are three ways to obtain I/O samples, either locally or remotely:

- Queried Sampling (IS)
- Periodic Sampling (IR)
- Change Detection Sampling (IC)

I/O sample data is formatted as shown in the following table:

Bytes	Name	Description
1	Sample Sets	Number of sample sets in the packet (always set to 1).
2	Digital Channel Mask	Indicates which digital I/O lines have sampling enabled. Each bit corresponds to one digital I/O line on the device. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 4 = DIO4 bit 5 = ASSOC/DIO5 bit 6 = RTS/DIO6 bit 7 = CTS/DIO7 bit 8 = SLP_RQ/DIO8 bit 9 = ON_SLP/DIO9 bit 10 = RSSI/DIO10 bit 11 = PWM/DIO11 bit 12 = CD/DIO12 bit 13 = DOUT/DIO13 bit 14 = DIN/DIO14 For example, a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital I/O.

Bytes	Name	Description
1	Analog Channel Mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 7 = Supply Voltage
Variable	Sampled Data Set	A sample set consisting of 1 sample for each enabled ADC and/or DIO channel, which has voltage inputs of 1143.75 and 342.1875 mV. If any digital I/O lines are enabled, the first two bytes of the data set indicate the state of all enabled digital I/O. Only digital channels that are enabled in the Digital Channel Mask bytes have any meaning in the sample set. If no digital I/O are enabled on the device, these 2 bytes will be omitted. Following the digital I/O data (if any), each enabled analog channel returns 2 bytes. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3, and the supply voltage (if enabled) at the end.

The sampled data set includes 2 bytes of digital I/O data only if one or more I/O lines on the device are configured as digital I/O. If no pins are configured as digital I/O, these 2 bytes are omitted. Pins are configured as digital I/O by setting them to a value of 3, 4, or 5.

The digital I/O data is only relevant if the same bit is enabled in the digital I/O mask.

Analog samples are returned as 10-bit values. The device scales the analog reading such that 0x0000 represents 0 V, and 0x3FF = 1.2 V (The analog inputs on the device cannot read more than 1.2 V.). The device returns analog samples in order starting with AIN0 and finishing with AIN3, and the supply voltage. Only enabled analog input channels return data.

To convert the A/D reading to mV, do the following:

$$AD(mV) = (A/D \text{ reading} * 1200mV) / 1023$$

The reading in the sample frame represents voltage inputs of 1143.75 and 342.1875 mV for AD0 and AD1 respectively.

Queried sampling

You can send the **IS** command to a device locally, or to a remote device using the API remote command frame (for more information, see API Operation). When you send the IS command, the receiving device samples all enable digital I/O and analog input channels and return an I/O sample. If you send the **IS** locally, the device sends the I/O sample out the serial port. If the **IS** command was received as a remote command, the I/O sample is sent over-the-air to the device that sent the **IS** command.

If you issue the **IS** command in command mode, the device returns a carriage return-delimited list containing the fields listed in I/O sampling. If you issue the **IS** command in API mode, an API command response contains the same information.

The following table shows an example of the fields in an **IS** response.

Example	Sample AT Response
0x01	[1 sample set]
0x0C0C	[Digital Inputs: DIO 2, 3, 10, 11 enabled]
0x03	[Analog Inputs: A/D 0, 1 enabled]
0x0408	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0	[Analog input ADIO 0= 0x3D0]
0x0124	[Analog input ADIO 1=0x120]

Periodic I/O sampling

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

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

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

You can set **DH** and **DL** to **0** to transmit to the coordinator, or to the 64-bit address of the remote device (**SH** and **SL**).

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

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

Change detection sampling

You can configure devices to transmit a data sample immediately whenever a monitored digital I/O pin changes state. The **IC** command is a bitmask used to set which digital I/O lines to monitor for a state change. If one or more bits in **IC** is set, the device transmits an I/O sample as soon as it observes a state change in one of the monitored digital I/O lines. Use **DH** and **DL** to specify the 64-bit address to transmit change detection samples.

RSSI PWM

The XBee/XBee-PRO Zigbee RF Module features an RSSI/PWM pin (TH pin 6/SMT pin 7) that, if enabled, adjusts the PWM output to indicate the signal strength of the last received packet. Use P0 (RSSI/PWM0 Configuration) to enable the RSSI pulse width modulation (PWM) output on the pin. If **P0** is set to 1 (and **P1** is not set to 1), the RSSI/PWM pin outputs a pulse width modulated signal where the frequency adjusts based on the received signal strength of the last packet. Otherwise, for all other **P0** settings, use the pin for general purpose **IO**.

When a data packet is received, if you set **P0** to enable the RSSI/PWM feature, the RSSI PWM output adjusts based on the RSSI of the last packet. The RSSI/PWM output is enabled for a time based on the **RP** command. Each time the device receives an RF packet, the RSSI/PWM output adjusts based on the RSSI of the new packet, and resets the RSSI timer. If the RSSI timer expires, the RSSI/PWM pin drives low. **RP** is measured in 100 ms units and defaults to a value of 40 (4 seconds).

The RSSI PWM runs at 12 MHz and has 2400 total counts (200 μs period). RSSI (in dBm) is converted to PWM counts using the following equation:

PWM counts = (41 * RSSI_Unsigned) - 5928

I/O examples

Example 1: Configure the following I/O settings on the XBee

Configure AD1/DIO1 as a digital input with pullup resistor enabled Configure AD2/DIO2 as an analog input

Configure DIO4 as a digital output, driving high.

To configure AD1/DIO1 as an input, issue the D1 command with a parameter of 3 ("ATD13"). To enable pull-up resistors on the same pin, issues the **PR** command with bit 3 set (for example, PR8, PR1FFF, and so on).

Issue the **D2** command with a parameter of 2 to enable the analog input ("D22"). Finally, set DIO4 as an output, driving high by issuing the **D4** command with a parameter value of 5 ("D45").

After issuing these commands, apply the changes before the module I/O pins update to the new states. Issue the **AC** or **CN** commands to apply changes (for example, AC).

Example 2: Calculate the PWM counts for a packet received with an RSSI of -84 dBm

- RSSI = -84 = 0xAC = 172 decimal (unsigned)
- PWM counts = (41 * 172) - 5928
- PWM counts = 1124

With a total of 2400 counts, this yields an ON time of (1124 / 2400) = 46.8%

Example 3: Configure the RSSI/PWM pin to operate for 2 seconds after each received RF packet

First, make sure the RSSI/PWM functionality is enabled by reading the **P0** (P-zero) command. It should be set to 1 (default).

To configure the duration of the RSSI/PWM output, set the **RP** command. To achieve a 2 second PWM output, set **RP** to 0x14 (20 decimal, or 2 seconds) and apply changes using the **AC** command.

After applying changes, all received RF data packets set the RSSI timer for 2 seconds.

PWM1

When you configure **P1** for peripheral operation by setting the value to 1, it outputs a 50% duty cycle PWM with a clock rate of 32,787 Hz, which is a period of 30.5μs. The PWM output provides a clock for the PLUS processor, although it may also be used for other purposes.

When you enable this feature, the RSSI PWM output is automatically disabled, even if it is configured.

API Operation

An alternative to Transparent Operation are Application Programming Interface (API) Operations. API operation requires that the device communicate through a structured interface (that is, data is communicated in frames in a defined order). The API specifies how the device sends and receives commands, command responses, and module status messages using a serial port Data Frame.

API frame format

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

- **AP** = 1: API operation.

- **AP** = 2: API operation (with escaped characters—only possible on UART).

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

API operation (AP parameter = 1)

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

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

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

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

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

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

Escape characters

When you are sending or receiving a UART data frame, specific data values must be escaped (flagged) so they do not interfere with the data frame sequencing. To escape an interfering data byte, insert 0x7D and follow it with the byte to be escaped XOR'd with 0x20.

Data bytes that need to be escaped:

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

Example: Raw serial data before escaping interfering bytes:

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

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

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

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

Length

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

Frame data

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

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

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

The XBee/XBee-PRO Zigbee RF Module supports the following API frames:

API frame names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09

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

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 08 08 01 4E 49 58 42 45 45 3B**

Byte(s)	Description
7E	Start delimiter

Byte(s)	Description
00 08	Length bytes
08	API identifier
01	API frame ID
4E 49	AT Command
58 42 45 45	Parameter value
3B	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 08 08 **01 4E 49 58 42 45** 45 3B

Add these hex bytes:

0x08 + 0x01 + 0x4E + 0x49 + 0x58 + 0x42 + 0x45 + 0x45 = 0x01C4

Now take the result of 0x01C4 and keep only the lowest 8 bits which in this example is 0xC4 (the two far right digits). Subtract 0xC4 from 0xFF and you get 0x3B (0xFF - 0xC4 = 0x3B). 0x3B is the checksum for this data packet.

If an API data packet is composed with an incorrect checksum, the XBee/XBee-PRO Zigbee 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.

0x08 + 0x01 + 0x4E + 0x49 + 0x58 + 0x42 + 0x45 + 0x45 + 0x3B = 0x01FF

API examples

Example: Create an API AT command frame to configure a device to allow joining (set **NJ** to 0xFF).

The frame should look like:

0x7E 0x00 0x05 0x08 0x01 0x4E 0x4A 0xFF 5F

Where

- 0x0005 = length
- 0x08 = AT Command API frame type
- 0x01 = Frame ID (set to non-zero value)
- 0x4E4A = AT Command (**NJ**)
- 0xFF = value to set command to
- 0x5F = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x4A + 0xFF)]

Example: Send an **ND** command to discover the devices in the PAN.

The frame should look like:

0x7E 0x00 0x04 0x08 0x01 0x4E 0x44 0x64

Where:

- 0x0004 = length
- 0x08 = AT Command API frame type
- 0x01 = Frame ID (set to non-zero value)
- 0x4E44 = AT command (**ND**)
- 0x64 = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x44)]

Example: Send a remote command to the coordinator to set AD1/DIO1 as a digital input (**D1**=3) and apply changes to force the I/O update.

The API remote command frame should look like:

0x7E 0x00 0x10 0x17 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFE 0x02 0x44 0x31 0x03 0x70

Where:
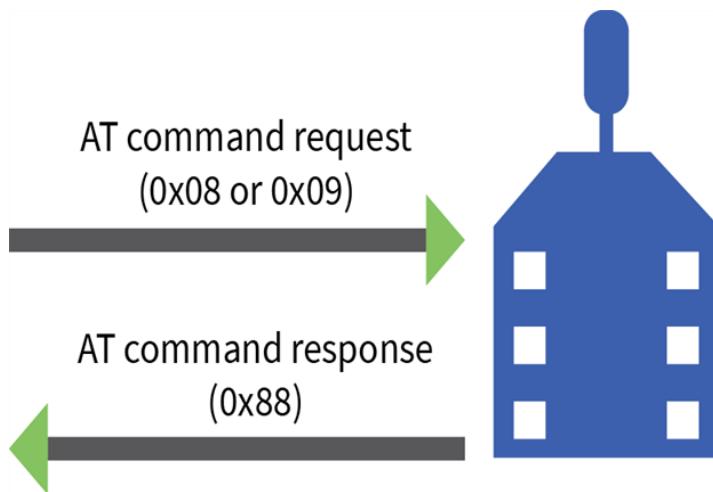
- 0x10 = length (16 bytes excluding checksum)
- 0x17 = Remote Command API frame type
- 0x01 = Frame ID
- 0x0000000000000000 = Coordinator's address (can be replaced with coordinator's actual 64-bit address if known)
- 0xFFFE = 16- bit Destination Address
- 0x02 = Apply Changes (Remote Command Options)
- 0x4431 = AT command (**D1**)
- 0x03 = Command Parameter (the parameter could also be sent as 0x0003 or 0x00000003)
  0x70 = Checksum

API serial exchanges

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

AT commands

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

Transmit and Receive RF data
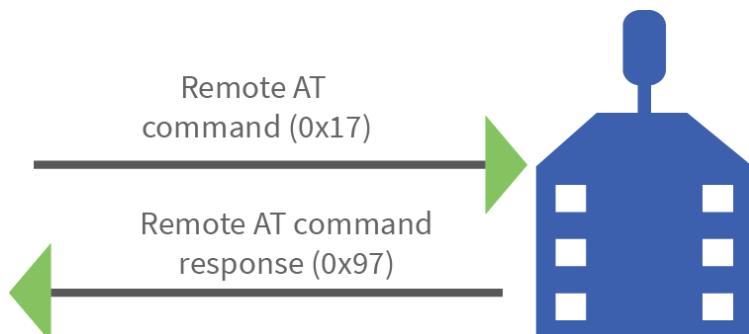
The following image shows the API frames exchange that take place at the UART interface when sending RF data to another device. The transmit status frame is always sent at the end of a data transmission unless the frame ID is set to 0 in the TX request. If the packet cannot be delivered to the destination, the transmit status frame indicates the cause of failure.

The received data frame type (0x90 or 0x91) is determined by the **AO** command.
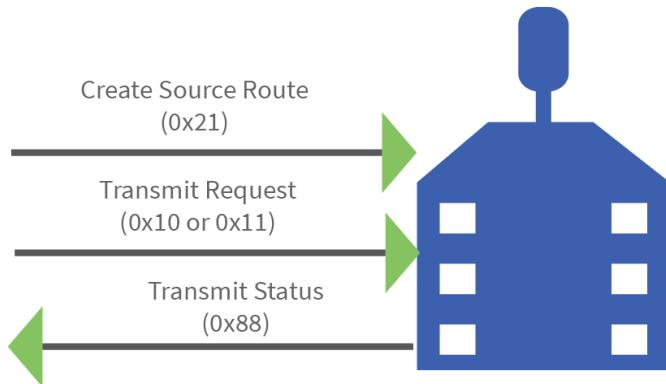
Remote AT commands

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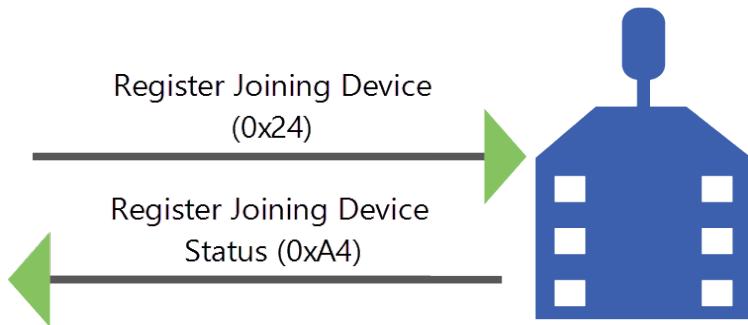
Source routing

The following image shows the API frame exchanges that take place at the serial port when sending a source routed transmission.

Device Registration

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

Send ZDO commands with the API

Zigbee specifications define Zigbee device objects (ZDOs) as part of the Zigbee device profile. These objects provide functionality to manage and map out the Zigbee network and to discover services on Zigbee devices. ZDOs are typically required when developing a Zigbee product that interoperates in a public profile such as home automation or smart energy, or when communicating with Zigbee devices from other vendors. You can also use the ZDO to perform several management functions such as frequency agility (energy detect and channel changes - Mgmt Network Update Request), discovering routes (Mgmt Routing Request) and neighbors (Mgmt LQI Request), and managing device connectivity (Mgmt Leave and Permit Join Request).

The following table shows some of the more prominent ZDOs with their respective cluster identifier. Each ZDO command has a defined payload. See the *Zigbee device profile* section of the Zigbee specification for details.

ZDO command	Cluster ID
Network Address Request	0x0000
IEEE Address Request	0x0001
Node Descriptor Request	0x0002
Simple Descriptor Request	0x0004
Active Endpoints Request	0x0005
Match Descriptor Request	0x0006
Mgmt LQI Request	0x0031
Mgmt Routing Request	0x0032
Mgmt Leave Request	0x0034
Mgmt Permit Joining Request	0x0036
Mgmt Network Update Request	0x0038

Use the Explicit Addressing Command Request - 0x11 to send Zigbee device objects commands to devices in the network. Sending ZDO commands with the Explicit Transmit API frame requires some formatting of the data payload field.

When sending a ZDO command with the API, all multiple byte values in the ZDO command (API payload), for example, u16, u32, and 64-bit addresses, must be sent in little endian byte order for the command to be executed correctly on a remote device.

For an API XBee to receive ZDO responses, set AO (API Options) to **1** to enable the explicit receive API frame.

The following table shows how you can use the Explicit API frame to send an "Active Endpoints" request to discover the active endpoints on a device with a 16-bit address of 0x1234.

Frame data fields	Offset	Description
Frame type	3	0x11
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent transmit status. If set to **0**, the device does not send a response out the serial port.
64-bit destination address	5-12	MSB first, LSB last. The 64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
16-bit destination network address	13	MSB first, LSB last. The 16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
	14	
Source endpoint	15	Set to 0x00 for ZDO transmissions (endpoint 0 is the ZDO endpoint).

Frame data fields	Offset	Description
Destination endpoint	16	Set to 0x00 for ZDO transmissions (endpoint 0 is the ZDO endpoint).
Cluster ID	17	Set to the cluster ID that corresponds to the ZDO command being sent.
	18	0x0005 = Active Endpoints Request
Profile ID	19-20	Set to 0x0000 for ZDO transmissions (Profile ID 0x0000 is the Zigbee device profile that supports ZDOs).
Broadcast radius	21	Sets the maximum number of hops a broadcast transmission can traverse. If set to **0**, the device sets the transmission radius to the network maximum hops value.
Transmission options	22	All bits must be set to 0.
Data payload	23	The required payload for a ZDO command. All multi-byte ZDO parameter values (u16, u32, 64- bit address) must be sent in little endian byte order.
	24	The Active Endpoints Request includes the following payload: [16-bit NwkAddrOfInterest]
	25	**Note** The 16-bit address in the API example (0x1234) is sent in little endian byte order (0x3412).

Example

The following example shows how you can use the Explicit API frame to send an "Active Endpoints" request to discover the active endpoints on a device with a 16-bit address of 0x1234.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x17
Frame type	3	0x11
Frame ID	4	0x01

Frame data fields	Offset	Example
64-bit destination address	MSB 5	0x00
	6	0x00
	7	0x00
	8	0x00
	9	0x00
	10	0x00
	11	0xFF
	LSB12	0xFF
16-bit Destination Network Address	MSB 13	0xFF
	LSB 14	0xFE
Source endpoint	15	0x00
Destination endpoint	16	0x00
Cluster ID	17	0x00
	18	0x05
Profile ID	19	0x00
	20	0x00
Broadcast radius	21	0x00
Transmit options	22	0x00
Data payload - transaction sequence number	23	0x01
Data payload - ZDO payload	24	0x34
	25	0x12
Checksum	29	0xA6

Send Zigbee cluster library (ZCL) commands with the API

The Zigbee cluster library defines a set of attributes and commands (clusters) that can be supported in multiple Zigbee profiles. The ZCL commands are typically required when developing a Zigbee product that will interoperate in a public profile such as home automation or smart energy, or when communicating with Zigbee devices from other vendors. Applications that are not designed for a public profile or for interoperability applications can skip this section.

The following table shows some prominent clusters with their respective attributes and commands.

Cluster (Cluster ID)	Attributes (Attribute ID)	Cluster ID
Basic (0x0000)	Application Version (0x0001) Hardware Version (0x0003) Model Identifier (0x0005)	Reset to defaults (0x00)
Identify (0x0003)	Identify Time (0x0000)	Identify (0x00) Identify Query (0x01)
Time (0x000A)	Time (0x0000) Time Status (0x0001) Time Zone (0x0002)	
Thermostat (0x0201)	Local Temperature (0x0000) Occupancy (0x0002)	Setpoint raise / lower (0x00)

The ZCL defines a number of profile-wide commands that can be supported on any profile, also known as general commands. These commands include the following.

Command (Command ID)	Description
Read Attributes (0x00)	Used to read one or more attributes on a remote device.
Read Attributes Response (0x01)	Generated in response to a read attributes command.
Write Attributes (0x02)	Used to change one or more attributes on a remote device.
Write Attributes Response (0x04)	Sent in response to a write attributes command.
Configure Reporting (0x06)	Used to configure a device to automatically report on the values of one or more of its attributes.
Report Attributes (0x0A)	Used to report attributes when report conditions have been satisfied.
Discover Attributes (0x0C)	Used to discover the attribute identifiers on a remote device.
Discover Attributes Response (0x0D)	Sent in response to a discover attributes command.

Use the Explicit Addressing Command Request - 0x11 to send ZCL commands to devices in the network. Sending ZCL commands with the Explicit Transmit API frame requires some formatting of the data payload field.

When sending a ZCL command with the API, all multiple byte values in the ZCL command (API Payload) (for example, u16, u32, 64-bit addresses) must be sent in little endian byte order for the command to be executed correctly on a remote device.

Note When sending ZCL commands, set the AO command to 1 to enable the explicit receive API frame. This provides indication of the source 64- and 16-bit addresses, cluster ID, profile ID, and endpoint information for each received packet. This information is required to properly decode received data.

The following table shows how the Explicit API frame can be used to read the hardware version attribute from a device with a 64-bit address of 0x0013A200 40401234 (unknown 16-bit address). This example uses arbitrary source and destination endpoints. The hardware version attribute (attribute

ID 0x0003) is part of the basic cluster (cluster ID 0x0000). The Read Attribute general command ID is 0x00.

Frame fields			Offset	Description
Frame type			3	
Frame ID			4	Identifies the serial port data frame for the host to correlate with a subsequent transmit status. If set to 0, no transmit status frame will be sent out the serial port.
64-bit destination address			MSB 5	The 64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
			6	
			7	
			8	
			9	
			10	
			11	
			LSB 12	
16-bit destination network address			MSB 13	The 16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
			LSB 14	
Source endpoint			15	Set to the source endpoint on the sending device (0x41 arbitrarily selected).
Destination endpoint			16	Set to the destination endpoint on the remote device (0x42 arbitrarily selected).
Cluster ID			MSB 17	Set to the cluster ID that corresponds to the ZCL command being sent. 0x0000 = Basic Cluster.
			LSB 18	
Profile ID			MSB 19	Set to the profile ID supported on the device (0xD123 arbitrarily selected).
			LSB 20	
Broadcast radius			21	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value.

Frame fields			Offset	Description
Transmit options			22	All bits must be set to 0.
Data payload	ZCL frame header	Frame control	23	Bitfield that defines the command type and other relevant information in the ZCL command. For more information, see the ZCL specification.
		Transaction sequence number	24	A sequence number used to correlate a ZCL command with a ZCL response. (The hardware version response will include this byte as a sequence number in the response.) The value 0x01 was arbitrarily selected.
		Command ID	25	Since the frame control "frame type" bits are 00, this byte specifies a general command. Command ID 0x00 is a Read Attributes command.
	ZCL payload	Attribute ID	26	The payload for a "Read Attributes" command is a list of Attribute Identifiers that are being read. **Note** The 16-bit Attribute ID (0x0003) is sent in little endian byte order (0x0300). All multi- byte ZCL header and payload values must be sent in little endian byte order.
			27	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Example

In this example, the Frame Control field (offset 23) is constructed as follows:

Name	Bits	Example Value Description
Frame Type	0-1	00 - Command acts across the entire profile.
Manufacturer Specific	2	0 - The manufacturer code field is omitted from the ZCL Frame Header.
Direction	3	0 - The command is being sent from the client side to the server side.
Disable Default Response	4	0 - Default response not disabled.
Reserved	5-7	Set to 0.

For more information, see the *Zigbee Cluster Library* specification.

Frame data fields			Offset	Example
Start delimiter			0	0x7E

Frame data fields			Offset	Example
Length			MSB 1	0x00
			LSB 2	0x19
Frame type			3	0x11
Frame ID			4	0x01
64-bit destination address			MSB 5	0x00
			6	0x13
			7	0xA2
			8	0x00
			9	0x40
			10	0x40
			11	0x12
			LSB12	0x34
16-bit destination network address			MSB 13	0xFF
			LSB 14	0xFE
Source endpoint			15	0x41
Destination endpoint			16	0x42
Cluster ID			MSB 17	0x00
			LSB 18	0x00
Profile ID			MSB 19	0xD1
			LSB 20	0x23
Broadcast radius			21	0x00
Transmit options			22	0x00
Data payload	ZCL frame header	Frame control	23	0x00
		Transaction sequence number	24	0x01
		Command ID	25	0x00
	ZCL payload	Attribute ID	26	0x03
			27	0x00
Checksum			28	0xFA

Send Public Profile Commands with the API

You can use the XBee API using the Explicit Transmit API frame (0x11) to send commands in public profiles such as Smart Energy and Home Automation. Sending public profile commands with the Explicit Transmit API frame requires some formatting of the data payload field. Most of the public profile commands fit into the Zigbee cluster library (ZCL) architecture as described in Send Zigbee cluster library (ZCL) commands with the API.

The following table shows how you can use the Explicit API frame to send a demand response and load control message (cluster ID 0x701) in the smart energy profile (profile ID 0x0109) in the revision 14 Smart Energy specification. The device sends a "Load Control Event"message (command ID 0x00) and to a device with 64- bit address of 0x0013A200 40401234 with a 16-bit address of 0x5678. The event starts a load control event for water heaters and smart appliances for a duration of 1 minute, starting immediately.

Note When sending public profile commands, set the **AO** command to 1 to enable the explicit receive API frame. This provides indication of the source 64- and 16-bit addresses, cluster ID, profile ID, and endpoint information for each received packet. This information is required to properly decode received data.

Frame specific data

Frame Fields			Offset	Description
Frame type			3	
Frame ID			4	Identifies the serial port data frame for the host to correlate with a subsequent transmit status. If set to 0, no transmit status frame will be sent out the serial port.
64-bit destination address			MSB 5	The 64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
			6	
			7	
			8	
			9	
			10	
			11	
			LSB 12	
16-bit destination network address			MSB 13	The 16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
			LSB 14	
Source endpoint			15	Set to the source endpoint on the sending device. (0x41 arbitrarily selected).

Frame Fields			Offset	Description
Destination endpoint			16	Set to the destination endpoint on the remote device. (0x42 arbitrarily selected).
Cluster ID			MSB 17	Set to the cluster ID that corresponds to the ZCL command being sent. 0x0701 = Demand response and load control cluster ID
			LSB 18	
Profile ID			MSB 19	Set to the profile ID supported on the device. 0x0109 = Smart Energy profile ID.
			LSB 20	
Broadcast radius			21	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value.
Transmit options			22	All bits must be set to 0.
Data payload	ZCL frame header	Frame control	23	Bitfield that defines the command type and other relevant information in the ZCL command. For more information, see the ZCL specification.
		Transaction sequence number	24	A sequence number used to correlate a ZCL command with a ZCL response. (The hardware version response will include this byte as a sequence number in the response.) The value 0x01 was arbitrarily selected.
			25	Since the frame control "frame type" bits are 01, this byte specifies a cluster-specific command. Command ID 0x00 in the Demand Response and Load Control cluster is a Load Control Event command. For more information, see the Smart Energy specification.
	ZCL payload - load control event data	Issuer event ID	26	The 4-byte unique identifier. <hr> **Note** The 4-byte ID is sent in little endian byte order (0x78563412). <hr> The event ID in this example (0x12345678) is arbitrarily selected.
			27	
			28	
			29	

Frame Fields			Offset	Description
		Device class	30	This bit encoded field represents the Device Class associated with the Load Control Event. A bit value of 0x0014 enables smart appliances and water heaters. **Note** The 2-byte bit field value is sent in little endian byte order.
			31	
		Utility enrollment group	32	Used to identify sub-groups of devices in the device-class. 0x00 addresses all groups.
		Start time	33	
			34	
			35	
			36	
		Duration in minutes	37	This 2-byte value must be sent in little endian byte order.
			38	
		Criticality level	39	Indicates the criticality level of the event. In this example, the level is "voluntary".
		Cooling temperature	40	Requested offset to apply to the normal cooling set point. A value of 0xFF indicates the temperature offset value is not used.
		Heating temperature offset	41	Requested offset to apply to the normal heating set point. A value of 0xFF indicates the temperature offset value is not used.
		Cooling temperature set point	42	Requested cooling set point in 0.01 degrees Celsius. A value of 0x8000 means the set point field is not used in this event. **Note** The 0x80000 is sent in little endian byte order.
			43	

Frame Fields			Offset	Description
		Heating temperature set point	44	Requested heating set point in 0.01 degrees Celsius. A value of 0x8000 means the set point field is not used in this event.
			45	**Note** The 0x80000 is sent in little endian byte order.
		Average load adjustment percentage	46	Maximum energy usage limit. A value of 0x80 indicates the field is not used.
		Duty cycle	47	Defines the maximum "On" duty cycle. A value of 0xFF indicates the duty cycle is not used in this event.
		Duty cycle event control	48	A bitmap describing event options.

Example

In this example, the Frame Control field (offset 23) is constructed as follows:

Name	Bits	Example Value Description
Frame Type	0-1	01 - Command is specific to a cluster
Manufacturer Specific	2	0 - The manufacturer code field is omitted from the ZCL Frame Header.
Direction	3	1 - The command is being sent from the server side to the client side.
Disable Default Response	4	0 - Default response not disabled
Reserved	5-7	Set to 0.

For more information, see the Zigbee cluster library specification.

Frame fields			Offset	Example
Start delimiter			0	0x7E
Length			MSB 1	0x00
			LSB 2	0x19
Frame type			3	0x11
Frame ID			4	0x01

Frame fields			Offset	Example
64-bit destination address			MSB 5	0x00
			6	0x13
			7	0xA2
			8	0x00
			9	0x40
			10	0x40
			11	0x12
			LSB 12	0x34
16-bit destination network address			MSB 13	0x56
			LSB 14	0x78
Source endpoint			15	0x41
Destination endpoint			16	0x42
Cluster ID			MSB 17	0x07
			LSB 18	0x01
Profile ID			MSB 19	0x01
			LSB 20	0x09
Broadcast radius			21	0x00
Transmit options			22	0x00

Frame fields			Offset	Example
Data payload	ZCL frame header	Frame control	23	0x09
		Transaction sequence number	24	0x01
			25	0x00
	ZCL payload - load control event data	Issuer event ID	26	0x78
			27	0x56
			28	0x34
			29	0x12
		Device class	30	0x14
			31	0x00
		Utility enrollment group	32	0x00
		Start time	33	0x00
			34	0x00
			35	0x00
			36	0x00
		Duration in Minutes	37	0x01
			38	0x00
		Criticality level	39	0x04
		Cooling temperature	40	0xFF
		Heating temperature offset	41	0xFF
		Cooling temperature set point	42	0x00
			43	0x80
		Heating temperature set point	44	0x00
			45	0x80
		Average load adjustment percentage	46	0x80
		Duty cycle	47	0xFF
		Duty cycle event control	48	0x00
Checksum			49	0x5B

Frame descriptions

The following sections describe the API frames.

Local AT Command Request - 0x08

Response frame: Local AT Command Response - 0x88

Description

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

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

Format

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

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

Examples

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

Set the local command parameter

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

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

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

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

Query local command parameter

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

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

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

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

Queue Local AT Command Request - 0x09

Response frame: Local AT Command Response - 0x88

Description

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

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

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

Format

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

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

Examples

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

Queue setting local command parameter

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

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

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

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

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

Query local command parameter

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

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

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

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

Transmit Request - 0x10

Response frame: Extended Transmit Status - 0x8B

Description

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

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

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

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node
- If transmitting to a 64-bit destination, set the 16-bit address field to **0xFFFE**

16-bit addressing

- DigiMesh does not support 16-bit addressing. The 16-bit address field is considered reserved and should be set to **0xFFFE**
- For unicast transmissions, set the 16-bit address field to the address of the desired destination node
- To use 16-bit addressing, set the 64-bit address field to**0xFFFFFFFFFFFFFFFF**

Zigbee-specific addressing information

- A Zigbee coordinator can be addressed in one of two ways:
  - Set the 64-bit address to all 0x00s and the 16-bit address to**0xFFFE**
  - Set the 64-bit address to the coordinator's 64-bit address and the 16-bit address to**0x0000**
- When using 64-bit addressing, populating the correct 16-bit address of the destination helps improve performance when transmitting to multiple devices. If you do not know a 16-bit address, set this field to**0xFFFE**(unknown). If the transmission is successful, the Transmit Status frame - 0x8B indicates the discovered 16-bit address.
- When using 16-bit addressing, the following addresses are reserved:
  - **0xFFFC** = Broadcast to all routers
  - **0xFFFD** = Broadcast to all non-sleepy devices
  - **0xFFFF** = Broadcast to all devices, including sleepy end devices

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Request - **0x10**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response frame. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**. Zigbee coordinator address is **0x0000000000000000**. When using 16-bit addressing, set this field to **0xFFFFFFFFFFFFFFFF**.
13	16-bit	**16-bit destination address**	Set to the 16-bit network address of the destination device, if known. If transmitting to a 64-bit address, sending a broadcast, or the 16-bit address is unknown, set this field to **0xFFFE**.
15	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to **0**—recommended—the value of **NH** specifies the broadcast radius.
16	8-bit	**Transmit options**	See the Transmit options bit field table below for available options. If set to **0**, the value of **TO** specifies the transmit options.
17-n	variable	**Payload data**	Data to be sent to the destination device. Up to **NP** bytes per packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

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

Zigbee

Bit	Meaning	Description
0	Disable ACK [**0x01**]	Disable retries and route repair
1	Reserved	<set this bit to 0>
2	Indirect Transmission [**0x04**]	Used for Binding transmissions.
3	Multicast [**0x08**]	See Multicast transmissions for more information.

Bit	Meaning	Description
4	Secure Session Encryption [**0x10**]	Encrypt payload for transmission across a Secure Session Reduces maximum payload size by 4 bytes.
5	Enable APS encryption [**0x20**]	APS encrypt the payload using the link key set by **KY** Reduces maximum payload size by 4 bytes.
6	Use extended timeout [**0x40**]	See Extended timeout for more information.

Examples

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

64-bit unicast

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Bcast radius	Options	RF data
0x10	0x52	0x0013A200 12345678	0xFFFE	0x00	0x00	0x547844617461
Request	*Matches response*	*Destination*	*Unknown*	*N/A*	*Will use* ***TO***	*"TxData"*

64-bit broadcast

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Bcast radius	Tx Options	RF data
0x10	0x00	0x00000000 0000FFFF	0xFFFE	0x01	0x00	0x42726F616463617374
Request	*Suppress response*	*Broadcast address*	*Reserved*	*Single hop broadcast*	*Will use* ***TO***	*"Broadcast"*

16-bit unicast

Sending a unicast transmission to a device with the 16-bit address of **1234** with the serial data "**TxData**". Disable retries and acknowledgments to prioritize performance over reliability. The

corresponding Legacy TX Status frame - 0x89 response with a matching Frame ID can be used to verify that the transmission was sent.

```
7E 00 14 10 8D FF FF FF FF FF FF FF FF 12 34 00 01 54 78 44 61 74 61 DD
```

Frame type	Frame ID	64-bit dest	16-bit dest	Bcast radius	Tx Options	RF data
0x10	0x8D	0xFFFFFFFF FFFFFFFF	0x1234	0x00	0x01	0x547844617461
Request	*Matches response*	*Use 16-bit addressing*	*Destination*	N/A	*Disable retries*	*"TxData"*

Explicit Addressing Command Request - 0x11

Response frame: Extended Transmit Status - 0x8B

Description

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

This frame type is typically used for OTA updates, serial data transmissions, ZDO command execution, third-party Zigbee interfacing, and advanced Zigbee operations.

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

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node
- If transmitting to a 64-bit destination, set the 16-bit address field to **0xFFFE**

16-bit addressing

- DigiMesh does not support 16-bit addressing. The 16-bit address field is considered reserved and should be set to **0xFFFE**
- For unicast transmissions, set the 16-bit address field to the address of the desired destination node
- To use 16-bit addressing, set the 64-bit address field to **0xFFFFFFFFFFFFFFFF**

Zigbee-specific addressing information

- A Zigbee coordinator can be addressed in one of two ways:
  - Set the 64-bit address to all 0x00s and the 16-bit address to **0xFFFE**
  - Set the 64-bit address to the coordinator's 64-bit address and the 16-bit address to **0x0000**
- When using 64-bit addressing, populating the correct 16-bit address of the destination helps improve performance when transmitting to multiple devices. If you do not know a 16-bit address, set this field to **0xFFFE** (unknown). If the transmission is successful, the Extended Transmit Status - 0x8B indicates the discovered 16-bit address.
- When using 16-bit addressing, the following addresses are reserved:
  - **0xFFFC** = Broadcast to all routers
  - **0xFFFD** = Broadcast to all non-sleepy devices
  - **0xFFFF** = Broadcast to all devices, including sleepy end devices

- To send a ZDO command (ZCL/ZDP):
  - Enter the ZDO command in the command data field (payload).
  - Each field in the ZDO command frame is represented in little endian format.
  - For information on the command formatting, refer to the ZCL and ZDP specifications.

Reserved endpoints

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

The active Digi endpoints are:

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

Reserved cluster IDs

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

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

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

Reserved profile IDs

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

The following profile IDs are handled by the XBee natively, all others—such as Smart Energy and Home Automation—can be passed through to a host:

- **0xC105** - Digi profile ID
- **0x0000** - Zigbee device profile ID (ZDP)

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Addressing Command Request - **0x11**

Offset	Size	Frame Field	Description
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**. Zigbee coordinator address is **0x0000000000000000**. When using 16-bit addressing, set this field to **0xFFFFFFFFFFFFFFFF**.
13	16-bit	**16-bit destination address**	Set to the 16-bit network address of the destination device if known. If transmitting to a 64-bit address, sending a broadcast, or the 16-bit address is unknown, set this field to **0xFFFE**.
15	8-bit	**Source Endpoint**	Source endpoint for the transmission. Serial data transmissions should use **0xE8**.
16	8-bit	**Destination Endpoint**	Destination endpoint for the transmission. Serial data transmissions should use **0xE8**.
17	16-bit	**Cluster ID**	The Cluster ID that the host uses in the transmission. Serial data transmissions should use **0x11**.
19	16-bit	**Profile ID**	The Profile ID that the host uses in the transmission. Serial data transmissions between XBee devices should use **0xC105**.
21	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to **0** (recommended), the value of **NH** specifies the broadcast radius.
22	8-bit	**Transmit options**	See the Transmit options bit field table below for available options. If set to **0**, the value of **TO** specifies the transmit options.
23-n	variable	**Command data**	Data to be sent to the destination device. Up to **NP** bytes per packet. For ZDO and ZCL commands, the command frame is inserted here. The fields in this nested command frame are represented in little-endian.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

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

Zigbee

Bit	Meaning	Description
0	Disable ACK [**0x01**]	Disable retries and route repair
1	Reserved	\<set this bit to 0>
2	Indirect Transmission [**0x04**]	Used for Binding transmissions.
3	Multicast [**0x08**]	See Multicast transmissions for more information.
4	Secure Session Encryption [**0x10**]	Encrypt payload for transmission across a Secure Session Reduces maximum payload size by 4 bytes.
5	Enable APS encryption [**0x20**]	APS encrypt the payload using the link key set by **KY** Reduces maximum payload size by 4 bytes.
6	Use extended timeout [**0x40**]	See Extended timeout for more information.

Examples

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

64-bit unicast

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0x87	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0011	0xC105	0x00	0x00	0x5478446174 61
Explicit request	*Matches response*	*Destination*	*Unknown*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Use **TO***	*"TxData"*

Loopback Packet

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

disabled.

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0xF8	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0012	0xC105	0x00	0x01	0x5478446174 61
Explicit request	*Matches response*	*Destination*	*Unknown*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Disable retries*	*"TxData"*

ZDO command - ZDP Management Leave Request

Request a Zigbee device with the 64-bit address of **0013A20012345678** leave the network via a ZDO command. The ZDP request is sent as a broadcast with the destination defined in the ZDO command frame. Each field in the ZDO frame is in little-endian, the rest of the Digi API frame is big-endian.

In order to output the response to the ZDO command request, the sender must be configured to emit explicit receive frames by setting bit 0 of AO (API Options)—**AO** = 1. See Receiving ZDO command and responses for more information.

The corresponding Extended Transmit Status - 0x8B response with a matching Frame ID will indicate whether the transmission succeeded. The destination will handle the request and return a response to the sender, which will be emitted as a Explicit Receive Indicator - 0x91 if enabled.

```
7E 00 1E 11 01 00 00 00 00 00 00 FF FF FF FE 00 00 00 34 00 00 00 00A1 78 56 34
21 00 A2 13 00 00 45
```

Frame type	Frame ID	64-bit dest	16-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
11	DE	00000000 00 0000FF FF	FFFE	00	00	0000	0000	00	00	*A1 78563421 00A21300 00*

Frame type	Frame ID	64-bit dest	16-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0xDE	0x00000000 0000FFFF	0xFFFE	0x00	0x00	0x0034	0x0000	0x00	0x00	■ 0xA1 ■ 0x0013A200 12345678 ■ 0x00
Explicit request	*Matches response*	*ZDO commands should be broadcasted*	*Reserved*	*ZDO*	*ZDO*	*Management Leave Request Cluster*	*Zigbee Device Profile (ZDP)*	*Use **BH***	*Use **TO***	■ *Sequence num* ■ *64-bit dest* ■ *Options*

Remote AT Command Request - 0x17

Response frame: Remote AT Command Response- 0x97

Description

This frame type is used to query or set AT command parameters on a remote device.

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

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

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

Format

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

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

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

Examples

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

Set remote command parameter

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Command options	AT command	Parameter value
0x17	0x27	0x0013A200 12345678	0xFFFE	0x02	0x4E49	0x52656D6F7465
Request	*Matches response*		*Unknown*	*Apply Change*	*"NI"*	*"Remote"*

Queue remote command parameter change

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Command options	AT command	Parameter value
0x17	0x68	0x0013A200 12345678	0xFFFE	0x00	0x4944	0x0451
Request	*Matches response*		*Unknown*	*Queue Change*	*"ID"*	

Query remote command parameter

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

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

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

Frame type	Frame ID	64-bit dest	16-bit dest	Command options	AT command	Parameter value
0x17	0xFA	0x0013A200 12345678	0xFFFE	0x00	0x5450	(omitted)
Request	*Matches response*		*Unknown*	*N/A*	*"TP"*	*Query the parameter*

Create Source Route - 0x21

Description

This frame type is used to create an entry in the source route table of a local device. A source route specifies the complete route a packet traverses to get from source to destination. For best results, use source routing with many-to-one routing. See Source routing for more information.

In most cases, this frametype is used in combination with routing information received from a corresponding Route Record Indicator - 0xA1 frame. Route indicators are generated when a network device sends data to a concentrator. The order in which addresses are entered into the 0x21 frame are the same as provided by the 0xA1 frame—destination to source.

There is no response frame for this frame type. Take care when generating source routes as an incorrectly formatted frame will be silently rejected or may cause unexpected results.

Note Both the 64-bit and 16-bit destination addresses are required when creating a source route.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Create Source Route - **0x21**
4	8-bit	**Frame ID (reserved)**	This frame type generates no response, so the Frame ID field is not used. Set this field to 0.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device (required).
13	16-bit	**16-bit destination address**	Set to the 16-bit network address of the destination device (required).
15	8-bit	**Options (reserved)**	Source routing options are not available yet. This bit field is reserved for future functionality. Set this field to 0.

Offset	Size	Frame Field	Description
16	8-bit	**Number of addresses**	The number of addresses in the source route (excluding source and destination). A route can only traverse across a maximum of 30 hops. If this number is **0** or exceeds the maximum hop count, the frame is silently discarded and a route will not be created.
17-n	16-bit variable	**Address**	The 16-bit network address(es) of the devices along the source route, excluding the source and destination. The addresses should be entered in reverse order (from destination to source) to match the order provided in Route Record Indicator - 0xA1.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

4-hop route

A concentrator needs to send data to a device with a 64-bit address of **0013A20012345678** that is 4-hops away. Due to the size of the network, route discoveries need to be be minimized. The concentrator had previously received a Route Record Indicator - 0xA1 with the source route to this device and had stored this information.

The route looks like this:

Source (concentrator) <> **Router A** <> **Router B** <> **Router C** <> **Destination (remote)**

In this example, the network addresses are simplified.

```
7E 00 14 21 00 00 13 A2 00 12 34 56 78 DD DD 00 03 CC CC BB BB AA AA F6
```

Frame type	Frame ID	64-bit dest	16-bit dest	Options	Num of addresses	Address 1	Address 2	Address 3
0x21	0x00	0x0013A200 12345678	0xDDDD	0x00	0x03	0xCCCC	0xBBBB	0xAAAA
Route request	*Not used*	*Destination IEEE address*	*Destination NWK address*	*Not used*		*Neighbor of dest*	*Intermediate hop*	*Neighbor of source*

Local AT Command Response - 0x88

Request frames:

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

Description

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

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

Format

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

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

Examples

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

Set local command parameter

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

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

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

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

Query local command parameter

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

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

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

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

Modem Status - 0x8A

Description

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

Format

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

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

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

Modem status codes

Statuses for specific modem types are listed here.

XBee Zigbee

0x00 = Hardware reset or power up

0x01 = Watchdog timer reset

0x02 = Joined network (routers and end devices)

0x03 = Disassociated

0x06 = Coordinator started

0x07 = Network security key was updated

0x0D = Voltage supply limit exceeded

0x11 = Modem configuration changed while join in progress

0x3B = XBee 3 - Secure session successfully established

0x3C = XBee 3 - Secure session ended

0x3D = XBee 3 - Secure session authentication failed

0x3E = XBee 3 - Coordinator detected a PAN ID conflict but because **CR** = **0**, no action will be taken.

0x3F = Coordinator changed PAN ID due to a conflict

0x32 = XBee 3 - BLE Connect

0x33 = XBee 3 - BLE Disconnect

0x34 = XBee 3 - No Secure Session Connection

0x40 = Router PAN ID was changed by coordinator due to a conflict

0x42 = Network Watchdog timeout expired three times

0x80 - **0xFF** = Stack error

Examples

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

Boot status

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

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

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

Extended Transmit Status - 0x8B

Request frames:

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

Description

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

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

Zigbee transmissions to an unknown network address of **0xFFFE** will return the discovered 16-bit network address in this response frame. This network address should be used in subsequent transmissions to the specific destination.

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x8B**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	16-bit	**16-bit destination address**	The 16-bit network address where the packet was delivered (if successful). If not successful, this address is **0xFFFD** (destination address unknown). **0xFFFE** indicates 16-bit addressing was not used.
7	8-bit	**Transmit retry count**	The number of application transmission retries that occur.

Offset	Size	Frame Field	Description

8	8-bit	**Delivery status**	Complete list of delivery statuses:
			    0x00 = Success
			    0x01 = MAC ACK failure
			    0x02 = CCA/LBT failure
			    0x03 = Indirect message unrequested / no spectrum available
			    0x15 = Invalid destination endpoint
			    0x21 = Network ACK failure
			    0x22 = Not joined to network
			    0x23 = Self-addressed
			    0x24 = Address not found
			    0x25 = Route not found
			    0x26 = Broadcast source failed to hear a neighbor relay the message
			    0x2B = Invalid binding table index
			    0x2C = Resource error - lack of free buffers, timers, etc.
			    0x2D = Attempted broadcast with APS transmission
			    0x2E = Attempted unicast with APS transmission, but **EE** = **0**
			    0x31 = Internal resource error
			    0x32 = Resource error lack of free buffers, timers, etc.
			    0x34 = No Secure Session connection
			    0x35 = Encryption failure
			    0x74 = Data payload too large
			    0x75 = Indirect message unrequested
			Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.

9	8-bit	**Discovery status**	Complete list of delivery statuses:
			    0x00 = No discovery overhead
			    0x01 = Zigbee address discovery
			    0x02 = Route discovery
			    0x03 = Zigbee address and route discovery
			    0x40 = Zigbee end device extended timeout

EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

Successful transmission

Host sent a unicast transmission to a remote Zigbee device using a Transmit Request - 0x10 frame. The transmission was sent using the destination's IEEE 64-bit address with a 16-bit network address of 0xFFFE (unknown).

The corresponding Extended Transmit Status - 0x8B with a matching Frame ID is emitted as a response to the request:

```
7E 00 07 8B 52 12 34 02 00 01 D9
```

Frame type	Frame ID	16-bit dest address	Tx retries	Delivery status	Discovery status
0x8B	0x52	0x1234	0x02	0x00	0x01
Response	*Matches request*	*Discovered NWK address*	*2 retries*	*Success*	*Address discovery*

To reduce discovery overhead, the host can retrieve the discovered 16-bit network address from this response frame to use in subsequent transmissions.

Receive Packet - 0x90

Request frames:

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

Description

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Receive Packet - **0x90**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**16-bit source address**	The sender's 16-bit network address.

Offset	Size	Frame Field	Description

14	8-bit	**Receive options**	Bit field of options that apply to the received message:
			
			 ■ **Bit 0**: Packet was Acknowledged [**0x01**]
			 ■ **Bit 1**: Packet was sent as a broadcast [**0x02**]
			 ■ **Bit 2**: Reserved
			 ■ **Bit 3**: Reserved
			 ■ **Bit 4**: Packet was sent across a secure session [0x10]
			 ■ **Bit 5**: Packet encrypted with Zigbee APS security [**0x20**]
			 ■ **Bit 6**: Packet was sent from an End Device [**0x40**]
			 ■ **Bit 6, 7**: DigiMesh delivery method
			   • b'00 = \<invalid option\>
			   • b'01 = Point-multipoint [**0x40**]
			   • b'10 = Directed Broadcast [**0x80**]
			   • b'11 = DigiMesh [**0xC0**]
			
			 Note Option values may be combined.

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

Examples

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

64-bit unicast

A device with the 64-bit address of **0013A20087654321** sent a unicast transmission to a specific device with the payload of "**TxData**". The following frame is emitted if the destination is configured with **AO** = **0**.

```
7E 00 12 90 00 13 A2 00 87 65 43 21 56 14 01 54 78 44 61 74 61 B9
```

Frame type	64-bit source	16-bit source	Rx options	Received data
0x90	0x0013A200 87654321	0x5614	0x01	0x547844617461
Output		*Network address*	*ACK was sent*	*"TxData"*

Explicit Receive Indicator - 0x91

Request frames:

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

Description

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

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

This frame is also emitted as a response to ZDO command requests, see Receiving ZDO command and responses for more information. The Cluster ID and endpoints are 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.

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

Offset	Size	Frame Field	Description
20	8-bit	**Receive options**	Bit field of options that apply to the received message for packets sent using Digi endpoints (0xDC-0xEE): ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: Reserved ■ **Bit 3**: Reserved ■ **Bit 4**: Packet was sent across a secure session [**0x10**] ■ **Bit 5**: Packet encrypted with Zigbee APS security [**0x20**] ■ **Bit 6**: Packet was sent from an End Device [**0x40**] ■ **Bit 6, 7**: DigiMesh delivery method • b'00 = \<invalid option\> • b'01 = Point-multipoint [**0x40**] • b'10 = Directed Broadcast [**0x80**] • b'11 = DigiMesh [**0xC0**] **Note** Option values may be combined.
21-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

64-bit unicast

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

```
7E 00 18 91 00 13 A2 00 87 65 43 21 87 BD E8 E8 00 11 C1 05 01 54 78 44 61 74 61
37
```

Frame type	64-bit source	16-bit source	Source EP	Dest EP	Cluster	Profile	Rx options	Received data
0x91	0x0013A200 87654321	0x87BD	0xE8	0xE8	0x0011	0xC105	0x01	0x547844617461
Explicit output		*Network address*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*ACK was sent*	*"TxData"*

ZDO command - ZDP IEEE Address Response

A ZDP IEEE address request is issued in order to identify the 64-bit address of a Zigbee device with the 16-bit network address of 0x046D. The following response is emitted out of the device that issued the request if configured to do so. In order to output the response to the ZDO command request, the sender must be configured to emit explicit receive frames by setting bit 0 of AO (API Options) (**AO** = **1**). See Receiving ZDO command and responses for more information.

Note Each field in the ZDO frame is in little-endian, the rest of the Digi API frame is big-endian.

```
7E 00 1E 91 00 13 A2 00 12 34 56 78 04 6D 00 00 80 01 00 00 01 B5 00 78 56 34 12
00 A2 13 00 6D 04 C3
```

Frame type	64-bit source	16-bit source	Source EP	Dest EP	Cluster	Profile	Rx options	Received data
91	0013A200 12345678	046D	00	00	8001	0000	01	B5 00 78563412 00A21300 6D04
0x91	0x0013A200 87654321	0x046D	0x00	0x00	0x8001	0xC105	0x01	■ 0xB5 ■ 0x00 ■ 0x0013A200 12345678 ■ 0x046D
Explicit output		*Network address*	*ZDO*	*ZDO*	*IEEE Address Response*	*ZDO*	*ACK was sent*	■ *Sequence Num* ■ *Status* ■ *IEEE Address* ■ *NWK Address*

I/O Sample Indicator - 0x92

Description

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

Format

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

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

Offset	Size	Frame Field	Description

16	16-bit	**Digital sample mask**	Bit field that indicates which I/O lines on the remote are configured as digital inputs or outputs, if any:
			    bit 0: DIO0
			    bit 1: DIO1
			    bit 2: DIO2
			    bit 3: DIO3
			    bit 4: DIO4
			    bit 5: DIO5
			    bit 6: DIO6
			    bit 7: DIO7
			    bit 8: DIO8
			    bit 9: DIO9
			    bit 10: DIO10
			    bit 11: DIO11
			    bit 12: DIO12
			    bit 13: DIO13
			    bit 14: DIO14
			    bit 15: N/A
			For example, a digital channel mask of **0x002F** means DIO **0**, **1**, **2**, **3**, and **5** are enabled as digital I/O.

18	8-bit	**Analog sample mask**	Bit field that indicates which I/O lines on the remote are configured as analog input, if any:
			    bit 0: AD0
			    bit 1: AD1
			    bit 2: AD2
			    bit 3: AD3
			    bit 7: Supply Voltage (enabled with **V+** command)

19	16-bit	**Digital samples (if included)**	If the sample set includes any digital I/O lines (**Digital channel mask > 0**), this field contain samples for all enabled digital I/O lines. If no digital lines are configured as inputs or outputs, this field will be omitted. DIO lines that do not have sampling enabled return 0. Bits in this field are arranged the same as they are in the Digital channel mask field.
22	16-bit variable	**Analog samples (if included)**	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 16-bit value indicating the ADC measurement of that input. Analog samples are ordered sequentially from AD0 to AD3.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

I/O sample

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

I/O, and AD1 and AD2 configured as an analog input.

The destination will emit the following frame:

7E 00 16 **92 00 13 A2 00 12 34 56 78 87 AC 01 01 00 38 06 00 28 02 25 00 F8** EA

Frame type	64-bit source	16-bit source	Rx options	Num samples	Digital channel mask	Analog channel mask	Digital samples	Analog sample 1	Analog sample 2
0x92	0x0013A200 12345678	0x87AC	0x01	0x01	0x0038	0x06	0x0028	0x0225	0x00F8
Sample		*Network address*	*ACK was sent*	*Single sample (typical)*	*b'00**111**000 DIO3, DIO4, and DIO5 enabled*	*b'0**110** AD1 and AD2 enabled*	*b'00**101**000 DIO3 and DIO5 are HIGH; DIO4 is LOW*	*AD1 data*	*AD2 data*

XBee Sensor Read Indicator - 0x94

Description

When the device receives a sensor sample (from a Digi 1-wire sensor adapter), it is sent out the serial port using this message type (when **AO**=0).

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	0x94
64-bit source address	4-11	MSB first, LSB last. The sender's 64-bit address.
16-bit Source network address	12-13	MSB first, LSB last. The sender's 16-bit address.
Receive options	14	0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet 0x20 = Packet encrypted with APS encryption
1-Wire Sensors	15	0x01 = A/D Sensor Read 0x02 = Temperature Sensor Read 0x60 = Water present (module CD pin low)
A/D Values	16	Indicates a two-byte value for each of four A/D sensors (A, B, C, D). Set to 0xFFFFFFFFFFFFFFFF if no A/Ds are found.
	17	
	18	
	19	
	20	
	21	
	22	
	23	
Temperature Read	24	Indicates the two-byte value read from a digital thermometer if present. Set to 0xFFFF if not found.
	25	

Example

Suppose a device receives a 1-wire sensor sample from a device with a 64-bit address of 0x0013A200 40522BAA and a 16-bit address of 0xDD6C. If the sensor sample comes from a 1-wire humidity sensor, the API frame could look like this (if AO=0):

For convenience, let's label the A/D and temperature readings as AD0, AD1, AD2, AD3, and T. Using the data in this example:

AD0 = 0x0002

AD1 = 0x00CE

AD2 = 0x00EA

AD3 = 0x0052

T = 0x016A

To convert these to temperature and humidity values, the following equations should be used.

Temperature (℃) = (T / 16), for T < 2048

= - (T & 0x7FF) / 16, for T >= 2048

Vsupply = (AD2 * 5.1) / 255

Voutput = (AD3 * 5.1) / 255

Relative Humidity = ((Voutput / Vsupply) - 0.16) / (0.0062)

True Humidity = Relative Humidity / (1.0546 - (0.00216 * Temperature (℃))) Looking at the sample data, we have:

Vsupply = (234 * 5.1 / 255) = 4.68

Voutput = (82 * 5.1 / 255) = 1.64 Temperature = (362 / 16) = 22.625℃

Relative H = (161.2903 * ((1.64/4.68) - 0.16)) = 161.2903 * (0.19043) = 30.71%

True H = (30.71 / (1.0546 - (0.00216 * 22.625))) = (30.71 / 1.00573) = 30.54%

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x17
Frame type	3	0x94
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
16-bit source network address	MSB 12	0xDD
	LSB 13	0x6C
Receive options	14	0x01

Frame fields	Offset	Example
1-Wire sensors	15	0x03
A/D Values	16	0x00
	17	0x02
	18	0x00
	19	0xCE
	20	0x00
	21	0xEA
	22	0x00
	23	0x52
Temperature Read	24	0x01
	25	0x6A
Checksum	26	0x8B

Node Identification Indicator - 0x95

Description

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

A node identifies itself to the network under these conditions:

- The commissioning button is pressed once.
- A **CB 1** command is issued.
- A device with JN (Join Notification) enabled successfully associates with a Zigbee network.
- A device that is associated with a Zigbee network that has JN (Join Notification) enabled is power cycled.

See ND (Node Discovery) for information on the payload formatting.

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Node Identification Indicator - **0x95**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**16-bit source address**	The sender's 16-bit network address.

Offset	Size	Frame Field	Description

14	8-bit	**Options**	Bit field of options that apply to the received message:
			
			■ Bit 0: Reserved
			■ **Bit 1**: Packet was sent as a broadcast [**0x02**]
			■ **Bit 2**: Reserved
			■ Bit 4: Reserved
			■ Bit 5: Reserved
			■ **Bit 6, 7**: DigiMesh delivery method
			    ● b'00 = \<invalid option\>
			    ● b'01 = Point-multipoint [**0x40**]
			    ● b'10 = Directed Broadcast [**0x80**]
			    ● b'11 = DigiMesh [**0xC0**]
			
			Note Option values may be combined.

15	16-bit	**16-bit remote address**	The 16-bit network address of the device that sent the Node Identification.
17	64-bit	**64-bit remote address**	The 64-bit address of the device that sent the Node Identification.
25	variable (2-byte minimum)	**Node identification string**	Node identification string on the remote device set by NI (Node Identifier). The identification string is terminated with a NULL byte (0x00).
27+NI	16-bit	**Zigbee 16-bit parent address**	Indicates the 16-bit address of the remote's parent or **0xFFFE** if the remote has no parent. Equivalent to MP (16-bit Parent Network Address).
29+NI	8-bit	**Network device type**	What type of network device the remote identifies as: 0 = Coordinator 1 = Router 2 = End Device
30+NI	8-bit	**Source event**	The event that caused the node identification broadcast to be sent. 0 = Reserved 1 = Frame sent by node identification pushbutton event—see D0 (AD0/DIO0 Configuration). 2 = Frame sent after joining a Zigbee network—see JN (Join Notification). 3 = Frame sent after a power cycle event occurred while associated with a Zigbee network—see JN (Join Notification).
31+NI	16-bit	**Digi Profile ID**	The Digi application Profile ID—**0xC105**.

Offset	Size	Frame Field	Description
33+NI	16-bit	**Digi Manufacturer ID**	The Digi Manufacturer ID—**0x101E**.
35+NI	32-bit	**Device type identifier (optional)**	The user-defined device type on the remote device set by DD (Device Type Identifier). Only included if the receiving device has the appropriate NO (Network Discovery Options) bit set.
EOF-1	8-bit	**RSSI (optional)**	The RSSI of the last hop that relayed the message. Only included if the receiving device has the appropriate NO (Network Discovery Options) bit set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte—between length and checksum.

Examples

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

Identify remote device

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

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

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

Frame type	64-bit source	16-bit source	Options	16-bit remote	64-bit remote	NI String	Parent	Device type	Event	Profile ID	MFG ID
0x95	0x0013 A200 123456 78	0xFF FE	0xC2	0xFF FE	0x0013 A200 123456 78	0x4C4837 35 00	0xFF FE	0x01	0x01	0xC 105	0x101E
Identification		*Unknown*	*DigiMesh broadcast*	*Unknown*		*"LH75" + null*	*Unknown*	*Router*	*Button press*	*Digi*	*Digi*

Identify joining device

A Zigbee end device has join notification enabled by setting **JN** to **1**. When the joining device successfully associates with a Zigbee network, it will broadcast a node identification message.

The network has a variety of devices that are assigned identifier strings after association; a unique **DD** value is set to identify this type of device. The gateway that manages the network has the **NO** command set to **1** to display this information.

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

```
7E 00 24 95 00 13 A2 00 87 65 43 21 77 92 02 77 92 00 13 A2 00 87 65 43 21 20 00
45 A3 02 02 C1 05 10 1E 00 12 00 27 13
```

	Frame type	64-bit source	16-bit source	Options	16-bit remote	64-bit remote	NI String	Parent	Device type	Event	Profile ID	DD value
0x95	0x0013 A200 876543 21	0x7 792	0x02	0x77 92	0x0013 A200 123456 78	0x2 0 00	0x4 5A3	0x0 2	0x0 2	0xC 105	0x1 01E	0x0012 0027
Identific ation			*Broad cast*			*No NI strin g set*		*End devi ce*	*Join ed*	*Digi*	*Digi*	*Zigbee + User-defined*

Remote AT Command Response- 0x97

Request frame: Remote AT Command Request - 0x17

Description

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Response - **0x97**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	64-bit	**64-bit source address**	The sender's 64-bit address.
13	16-bit	**16-bit source address**	The sender's 16-bit network address.
15	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
17	8-bit	**Command status**	Status code for the host's request: 　**0x00** = OK 　**0x01** = ERROR 　**0x02** = Invalid command 　**0x03** = Invalid parameter 　**0x04** = Transmission failure Statuses for Secured remote AT commands: 　**0x0B** = No Secure Session - Remote command access requires a secure session be established first 　**0x0C** = Encryption error 　**0x0D** = Command was sent insecurely - A Secure Session exists, but the request needs to have the appropriate command option set (bit 4).

Offset	Size	Frame Field	Description
18-n	variable	**Parameter value (optional)**	If the host requested a command parameter change, this field will be omitted. If the host queried a command by omitting the parameter value in the request, this field will return the value currently set on the device.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

Set remote command parameter

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

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

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

Frame type	Frame ID	64-bit source	16-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x127E	0x4E49	0x00	(omitted)
Response	*Matches request*		*Network address*	*"NI"*	*Success*	*Parameter changes return no data*

Transmission failure

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

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

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

Frame type	Frame ID	64-bit source	16-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0xFFFE	0x4944	0x04	(omitted)
Response	*Matches request*		*Unknown*	*"ID"*	*Transmission failure*	*Parameter changes return no data*

Query remote command parameter

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

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

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

Frame type	Frame ID	64-bit source	16-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x127E	0x4944	0x00	0x002F
Response	*Matches request*		*Network address*	*"TP"*	*Success*	*+47 ℃*

Extended Modem Status - 0x98

Description

The Extended Modem Status - 0x98 frame is intended to provide additional in-frame diagnostic information over the traditional Modem Status - 0x8A frame.

This frame type can be used to troubleshoot Zigbee network association. To enable verbose join information, use DC (Joining Device Controls).

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Extended Modem Status - **0x98**
4	8-bit	**Status code**	Refer to the tables below for appropriate status codes
n	variable	**Status data (optional)**	Additional fields that provide information about the status
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Secure Session status codes

When **AZ** (Extended API Options) is configured to output extended secure session statuses, whenever Secure Session API Frames are emitted, the extended modem status will provide additional details about the event.

Status code	Description	Status data	Size	Description
0x3B	A Secure Session was established with this node	Address	64-bit	The address of the client in the session.
		Options	8-bit	Session options set by the client.
		Timeout	16-bit	Session timeout set by the client.

Status code	Description	Status data	Size	Description
0x3C	A Secure Session ended	Address	64-bit	The address of the other node in this session.
		Reason	8-bit	The reason the session was ended: **0x00** - Session was terminated by the other node **0x01** - Session Timed out **0x02** - Received a transmission with an invalid encryption counter **0x03** - Encryption counter overflow - the maximum number of transmissions for a single session has been reached **0x04** - Remote node out of memory
0x3D	A Secure Session authentication attempt failed	Address	64-bit	Address of the client node.
		Error	8-bit	Error that caused the authentication to fail.

Examples

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

Secure Session established

A device has established a secure session with the local node that has **AZ** (Extended API Options) configured to output extended secure session information. The following frame is emitted that announces the secure session establishment.

```
7E 00 0D 98 3B 00 13 A2 00 12 34 56 78 00 46 50 CD
```

Frame type	Status code	Status data
0x98	0x3B	■ 0x0013A20012345678 ■ 0x00 ■ 0x4650
Extended status	*Secure Session established*	■ Address ■ Options ■ Timeout (30 min)

Zigbee Verbose join messages

The following example shows a successful association of a device that has DC (Joining Device Controls) configured to enable Verbose Join messages. The device is operating in Transparent mode—**AP** = **0**—to allow a human-friendly way to troubleshoot association issues, if set for API mode—**AP** = **1**—equivalent 0x98 Extended Modem Status frames would be emitted.

Message	Description
V AI -SearchingforParent:FF	...search has started
V Scanning:03FFF800	...channels 11 through 25 are enabled by the SC setting for the Active Search.
V BeaconRsp:0000000000000042A6010B949AC8FF	**ZS** = 0x00 - extendedPanId = 00000000000042A6 - allowingJoin 0x01 (yes) - radiochannel 0x0B - panid 0x949A - rssi 0xC8 - lqi = 0xFF
V Reject ID	...beacon response's extendedPanId does not match this radio's ID setting of 3151
V BeaconRsp:0200000000000002AB010C55D2B2DB	**ZS** = 0x02 - extendedPanId = 0x00000000000002AB - allowingJoin = 0x01 (yes) - radiochannel = 0x0C - panid = 0x55D2 - rssi = 0xB2 - lqi = 0xDB
V Reject ZS	...beacon response's **ZS** does not match this radio's **ZS** setting
V BeaconRsp:00000000000000003151010EE29FDFFF	
V BeaconSaved:0E05E29F0000000000003151	...this beacon response is acceptable as a candidate for association
V Joining:0E05E29F0000000000003151	...sending association request
V StackStatus: joined, network up 0290	...we are joined, the network is up, we can send and transmit
V Joined unsecured network:	
V AI -AssociationSucceeded:00	

Status code descriptions

The following table describes the various Verbose Join trace messages in Status Code order. The Transparent mode string column shows the string which appears if you run Verbose Join in Command Mode. The Description column gives a more detailed explanation of each particular message. When a message accompanies Status Data, the Status Data column shows how to parse the hexadecimal string into fields. The number of bytes per field appears in parentheses "()".

Status code	Transparent mode string	Description	Status data	Description
0x00	Rejoin	A join attempt is being started.	rejoinState(1)	The rejoinState is a count of join attempts.
0x01	Stack Status	Shows status and state.	EmberStatus(1)	0x00 - no network 0x01 - joining 0x02 - joined 0x03 - joined (no parent) 0x04 - leaving
			emberNetworkState(1)	0x90 - Network is up and ready to receive/transmit. 0x91 - Network is down and cannot receive/transmit. 0x94 - Join attempt failed. 0x96 - A node's attempt to re-establish contact with the network after moving failed. 0x98 - A join attempt as a router failed due to a Zigbee 2006 versus Zigbee PRO 2007 incompatibility. Try to join as an end device. 0x99 - The network ID has changed. 0x9A - The PAN ID has changed. 0x9B - The channel has changed. 0xAB - No beacons were received in response to a beacon request.

Status code	Transparent mode string	Description	Status data	Description
0x02	Joining	An association request is being made.	radioChannel(1)	channel number ranging from 11 to 26 (0x0B to 0x1A)
			radioTxPower(1)	low level signed byte value for transmit power, values range from 0xC9 to 0x05 inclusive
			panid(2)	16 bit 4med' a network, or a Router/End Device has 'joined' a network.
			extendedPanId(8)	64 bit PAN Identifier for network
0x03	Joined	Joined - Coordinator "Formed:", Router/End Device "Joined"		
0x04	Beacon Response	Data received from a neighboring node in response to a beacon request	ZS[stackProfile](1)	See ZS (Zigbee Stack Profile).
			extendedPanId(8)	64 bit PAN Identifier for network
			allowingJoin(1)	0x00 - not permitting joins to its network 0x01 - permitting joins to its network
			radioChannel(1)	channel number ranging from 11 to 26 (0x0B to 0x1A)
			panid(2)	16 bit PAN Identifier for network
			rssi(1)	maximum relative signal strength indicator value measured in units of dBm
			lqi	link quality indicator
0x05	Reject **ZS**	Not an association candidate because **ZS** does not match that given in the beacon response.		
0x06	Reject **ID**	Not an association candidate because configured pan **ID** does not match that given in the beacon response.		
0x07	Reject **NJ**	Not an association candidate because it is not allowing joins.		

Status code	Transparent mode string	Description	Status data	Description
0x08	panID Match	**JV**/**NW** with search option (DO80) has found a matching network.	panId(2)	16 bit PAN Identifier for network
0x09	Reject LQIRSSI	**JV**/**NW** with search option (DO80) candidate rejected because this beacon response is weaker than an earlier beacon response.		
0x0A	Beacon Saved	This beacon response is a suitable candidate for an association request.	radioChannel(1)	channel number ranging from 11 to 26 (0x0B to 0x1A)
			radioTxPower(1)	low level signed byte value for transmit power, values range from 0xC9 to 0x05 inclusive
			panid(2)	16 bit PAN Identifier for network
			extendedPanId(8)	64 bit PAN Identifier for network
0x0B	**AI**	**AI** value has changed.	AIStatusCode(1)	See a description of AI (Association Indication)
0x0C	Permit Join	**NJ** setting (Permit Join Duration) has changed	value(1)	See a description of the NJ (Node Join Time) command.
0x0D	Scanning	Active scanning has begun.	ChannelMask(4)	A 32 bit value driven by the **SC** setting where bit positions 11 through 26 show which channels are enabled for the upcoming Active Scan. See a description of SC (Scan Channels).
0x0E	Scan Error	An error occurred during active scan.	StatusCode(1)	
0x0F	Join Request	High level request for a form/join.		
0x10	Reject LQI	Reject because LQI is worse than an already saved beacon	lqi(1)	link quality indicator
0x11	Reject RSSI	Rejected because RSSI is worse than an already saved beacon	rssi(1)	relative signal strength indicator

Status code	Transparent mode string	Description	Status data	Description
0x12	Rejected (cmdL ast)	Rejected because it matches the last associated network.		
0x13	Rejected (cmdS ave)	Rejected because it matches an already saved beacon response.		
0x14	Reject strength	During first/best phase, response is weaker than an already saved beacon response.		
0x16	Reset for DC80	With DC80 enabled, reset if no joinable beacon responses are received within 60s of joining.		
0x18	ScanCh	Scanning on Channel	radioChannel(1)	channel number ranging from 11 to 26 (0x0B to 0x1A)
0x19	Scan Mode	Shows phase of Ordered Association.	mode(1)	0: first/bestcandidate 1: ordered association by extpanid, then by channel
0x1A	Scan Init	Starting a scan	channel(1) TxPower(1)	channel being scanned low level radio transmit power setting
0x1D	Energy Scan - channel mask	Starting energy scan	**SC** mask(4)	Scan channel mask
0x1E	Energy Scan - energies	Channel Energies observed	Energies(16)	Energy Levels per channel in **SC**
0x1F	PanIdScan - radio channel	Pan Id Scan starting on channel	channel(1)	Radio Channel
0x20	FormNetwork - parameters	Forming a network	radioChannel(1)	channel number ranging form 11 to 26
			radioTxPower(1)	low level radio transmit power setting
			panid(2)	16 bit PAN identifier for network
			extendedpanid(8)	64 bit PAN identifier for network

Status code	Transparent mode string	Description	Status data	Description
0x21	Discovering **KE** Endpoint	Looking for Key Establishment Endpoint		
0x22	**KE** Endpoint	Found Key Establishment Endpoint	Endpoint(1)	Endpoint number

The following example shows a successful association with Verbose Join enabled in AT Command Mode.

Note Comments are included with the trace messages to explain the content and are preceded by an ellipsis "...".

```
+++OK
atid3151 OK
```

...configured pan identifier has been changed atdc10

```
OK
```

...and verbose join enabled atac

```
OK
```

...applying changes to the configuration V AI - Searching for Parent:FF
...search has started

```
V AI – Searching for Parent:FF
```

...and started again

```
V Scanning:03FFF800
```

...Channels 11 through 25 are enabled by the SC setting for the Active Search.

```
V Beacon Rsp:0000000000000042A6010B949AC8FF
```

...ZS(0), extendedPanId(00000000000042A6), allowingJoin(1), radiochannel(0x0B), panid(949A), rssi(C8), lqi(FF)

```
V Reject ID
```

...beacon response's extendedPanId does not match this radio's ID setting of 3151

```
V Beacon Rsp:0200000000000002AB010C55D2B2DB
```

...ZS(2), extendedPanId(00000000000002AB), allowingJoin(1), radiochannel(0x0C), panid(55D2), rssi(B2), lqi(DB)
```
V Reject ZS
```

...beacon response's ZS does not match this radio's ZS setting of 0x00

```
V Beacon Rsp:000000000000003151010EE29FDFFF
V Beacon Saved:0E05E29F0000000000003151
```

...this beacon response is acceptable as a candidate for association

```
V Joining:0E05E29F0000000000003151
```

...sending association request

```
V Stack Status: joined, network up 0290
```

...we are joined, the network is up, we can send and transmit

```
V Joined:
V AI - Association Succeeded:00
```

Over-the-Air Firmware Update Status - 0xA0

Description

The Over-the-Air Firmware Update Status frame provides an indication of the status of a firmware update transmission attempt. A query command (0x01 0x51) sent to a target with a 64-bit address of 0x0013A200 40522BAA through an updater with 64-bit address 0x0013A200403E0750 and 16-bit address 0x0000, generates the following expected response.

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	0xA0
64-bit Source (remote) address	4-11	MSB first, LSB last. The address of the remote radio returning this response.
16-bit destination address	12-13	The 16-bit address of the updater device.
Receive options	14	0x01 - Packet Acknowledged. 0x02 - Packet was a broadcast.
Bootloader message type	15	0x06 - ACK 0x15 - NACK 0x40 - No Mac ACK 0x51 - Query (received if the bootloader is not active on the target) 0x52 - Query Response
Block number	16	Block number used in the update request. Set to 0 if not applicable.
64-bit target address	17-n	The 64-bit Address of remote device that is being updated (target)

Example

If a query request returns a 0x15 (NACK) status, the target is likely waiting for a firmware update image. If no messages are sent to it for about 75 seconds, the target will timeout and accept new query messages.

If a query returns a 0x51 (QUERY) status, then the target's bootloader is not active and will not respond to query messages.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x16
Frame type	3	0xA0
64-bit source (remote) address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x3E
	10	0x07
	11	0x50
16-bit destination address	12	0x00
	13	0x00
Receive options	14	0x01
Bootloader message type	15	0x52
Block number	16	0x00
64-bit target address	17	0x00
	18	0x13
	19	0xA2
	20	0x00
	21	0x40
	22	0x52
	23	0x2B
	24	0xAA
Checksum	25	0x66

Route Record Indicator - 0xA1

Description

This frame type contains the routing information for a remote device on the network. This route information should be stored in external memory and used in a Create Source Route - 0x21 frame to provide a return route for subsequent data transmissions; this eliminates the need to perform a route discovery.

This frame type is emitted when a network concentrator receives a route record from a remote device. The type of concentrator determines how often this frame type is emitted: a high RAM concentrator (the default) will emit this frame type when a unicast data transmission is received for the first time. If a previously established route fails, a new 0xA1 Route Record Indicator will be generated. A low RAM concentrator will emit this frame for every received transmission. Concentrator type is determined by DO (Miscellaneous Device Options) bit 6.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Route Record Indicator - **0xA1**
5	64-bit	**64-bit source address**	The 64-bit IEEE address of the device that initiated the route record.
13	16-bit	**16-bit source address**	The 16-bit network address of the device that initiated the route record.
15	8-bit	**Receive Options**	Bit field of options that apply to the received message: ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**]
16	8-bit	**Number of addresses**	The number of addresses in the source route—excluding source and destination.

Offset	Size	Frame Field	Description
17-n	16-bit variable	**Address**	The 16-bit network address(es) of the devices along the source route, excluding the source and destination. The addresses are in order from destination to source and match the order to be entered into the Create Source Route - 0x21 frame.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

4-hop route

A remote device sends a unicast transmission to a concentrator that is 4-hops away. The concentrator emits a route record that can be stored for use in a subsequent Create Source Route - 0x21 frame prior to sending data back to the remote.

The route looks like this:

Destination (concentrator) <> **Router A** <> **Router B** <> **Router C** <> **Source (remote)**

```
7E 00 13 A1 00 13 A2 00 12 34 56 78 DD DD 01 03 CC CC BB BB AA AA 75
```

Frame type	64-bit source	16-bit source	Options	Num of addresses	Address 1	Address 2	Address 3
0xA1	0x0013A200 12345678	0xDDDD	0x01	0x03	0xCCCC	0xBBBB	0xAAAA
Route	*Source IEEE address*	*Source NWK address*		*3*	*Neighbor of source*	*Intermediate hop*	*Neighbor of destination*

Many-to-One Route Request Indicator - 0xA3

Description

This frame type is emitted on devices that receive a many-to-one route request from a network concentrator. Typically, a device that emits this frame type should send a unicast message to the sender so a route record can be generated.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Many-to-One Route Request Indicator - **0xA3**
4	64-bit	**64-bit source address**	The 64-bit IEEE address of the device that sent the many-to-one route request.
12	16-bit	**16-bit source address**	The 16-bit network address of the device that sent the many-to-one route request.
14	8-bit	**Receive options (reserved)**	Options are not available yet. This bit field is reserved for future functionality. This field returns 0.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

Many-to-one route request

A gateway is configured for many-to-one routing by setting AR (Aggregate Routing Notification) to send a many-to-one route request every 10 minutes—**AR** = **0x3C**.

Whenever an aggregator broadcast is sent, the following frame is emitted on all devices:

```
7E 00 0C A3 00 13 A2 00 87 65 43 21 00 00 00 57
```

Frame type	64-bit source	16-bit source	Reserved
0xA3	0x0013A200 87654321	0x0000	0x00
MTORR		*NWK address*	

AT commands

Network commands

ID (Extended PAN ID)

Set or read the 64-bit extended PAN ID. If set to **0**, the coordinator selects a random extended PAN ID, and the router/end device joins any extended PAN ID.

Write changes to **ID** to non-volatile memory using the **WR** command to preserve the **ID** setting if a power cycle occurs.

Parameter range

0 - 0xFFFFFFFFFFFFFFFF

Default

0

SC (Scan Channels)

Set or read the list of channels to scan.

Coordinator - Bit field list of channels to choose from prior to starting network.

Router/End Device - Bit field list of channels scanned to find a Coordinator/Router to join.

Write changes to **SC** using the **WR** command to preserve the **SC** setting if a power cycle occurs.

Parameter range

0 - 0xFFFF (bit field)

Bit field mask:

Bit	Parameter
0	0x0B
1	0x0C
2	0x0D
3	0x0E
4	0x0F
5	0x10
6	0x11
7	0x12
8	0x13
9	0x14
10	0x15
11	0x16
12	0x17

Bit	Parameter
13	0x18
14	0x19
15	0x1A

Note When you set **SC** to **0xFFFF** on the device, Channel 26 is not allowed to transmit at more than 3 dBm. If Channel 26 is present in the search mask (**SC**), active search (beaconing) for network formation by a Coordinator is limited to no more than 3 dBm on all channels.
Other communication by a Coordinator/Router/EndDevice, or active search for network joining (association) by Routers and End Devices is limited to no more than 3 dBm on Channel 26. **PL** and **PM** configuration settings control the transmit power on other channels.

  - For the XBee-PRO SMT module, Channel 26 is not allowed to transmit at more than 6 dBm.
  - For the XBee-PRO TH module, Channel 26 is not allowed to transmit at more than 2 dBm.

Default

0x7FFF

SD (Scan Duration)

Sets or displays the scan duration exponent. Write changes to **SD** using the **WR** command.

Note If you enable channel 26 (0x8000) in the search channel mask (**SC**), the device caps transmit power on all channels at 3 dBm during network formation or joining.

Coordinator - Duration of the Active and Energy Scans (on each channel) used to determine an acceptable channel and PAN ID for the coordinator to startup on.

Router/End Device - Duration of Active Scan (on each channel) used to locate an available coordinator/router to join during Association.

Scan Time is measured as:

   ([# of channels to scan] * (2 ^**SD**) * 15.36 ms) + (38 ms * [# of channels to scan]) + 20 ms

Use the **SC** (Scan Channels) command to set the number of channels to scan. The XBee can scan up to 16 channels (**SC** = 0xFFFF). The XBee-PRO can scan up to 13 channels (**SC**= 0x1FFE).

SD influences the time the MAC listens for beacons or runs an energy scan on a given channel.

Example

The following table shows the results for a thirteen channel scan.

SD setting	Time
0	0.200 s
2	0.799 s
4	3.190 s
6	12.780 s

Note **SD** influences the time the MAC listens for beacons or runs an energy scan on a given channel. The **SD** time is not a accurate estimate of the router/end device joining time requirements. Zigbee joining adds additional overhead including beacon processing on each channel, and sending a join request that extend the actual joining time.

Parameter range

0 - 7 (exponent)

Default

3

ZS (Zigbee Stack Profile)

Set or read the Zigbee stack profile value. This must be the same on all devices that will join the same network. Effective with release 4x5E, changing **ZS** to a different value causes all current parameters to be written to persistent storage. While the stack profile is changing, CTS is de-asserted to prevent serial input. We recommend CTS flow control.

Parameter range

0 - 2

Default

0

NJ (Node Join Time)

Set or read the time that a Coordinator/Router allows nodes to join. You can changes this value at run time without requiring a Coordinator or Router to restart. The time starts once the Coordinator or Router starts, and the timer resets on power-cycle or when **NJ** changes.

For an end device to enable rejoining, set **NJ** less than 0xFF on the device that joins. If **NJ** < 0xFF, the device assumes the network is not allowing joining and first tries to join a network using rejoining. If multiple rejoining attempts fail, or if **NJ** = **0xFF**, the device attempts to join using association.

Parameter range

0 - 0xFF (x1 sec)

Default

0xFF (always allows joining)

NW (Network Watchdog Timeout)

Set or read the network watchdog timeout value.

If **NW** is set > 0, the router monitors communication from the coordinator (or data collector) and leaves the network if it cannot communicate with the coordinator for 3 **NW** periods. The device resets the timer each time it receives or sends data to a coordinator, or if it receives a many-to-one broadcast.

Parameter range

0 - 0x64FF [x 1 minute](up to over 17 days)

Default

0 (disabled)

JV (Coordinator Join Verification)

Set or read the channel verification parameter.

If **JV** = **1**, a router or end device verifies the coordinator is on its operating channel when joining or coming up from a power cycle. If a coordinator is not detected, the router or end device leaves its current channel and attempts to join a new PAN. If **JV** = **0**, the router or end device continues operating on its current channel even if a coordinator is not detected.

Parameter range

0 - 1

Value	Description
0	Channel verification disabled
1	Channel verification enabled

Default

0

JN (Join Notification)

Set or read the join notification setting.

If enabled, the device transmits a broadcast node identification packet on power up and when joining. This action blinks the Associate LED rapidly on all devices that receive the transmission, and sends an API frame out the serial port of API devices.

Digi recommends you disable this feature for large networks to prevent excessive broadcasts.

Parameter range

0 - 1

Default

0

OP (Operating Extended PAN ID)

Read the 64-bit extended PAN ID. The **OP** value reflects the operating extended PAN ID where the device is running. If **ID** > 0, **OP** equals **ID**.

Parameter range

0x01 - 0xFFFFFFFFFFFFFFFF

Default

N/A

OI (Operating 16-bit PAN ID)

Read the 16-bit PAN ID. The **OI** value reflects the actual 16-bit PAN ID where the device is running.

Parameter range

0 - 0xFFFF

Default

[read-only]

CH (Operating Channel)

Read the channel number used to transmit and receive data between RF devices and uses 802.15.4 channel numbers.

A value of 0 means the device has not joined a PAN and is not operating on any channel.

Parameter range

0, 0x0B - 0x1A (XBee)

0, 0x0B - 0x19 (XBee-PRO, Channels 11-25)

Default

[read-only]

NC (Number of Remaining Children)

Read the number of remaining end device children that can join the device. If **NC** returns 0, the device is at capacity and cannot allow any more end device children to join.

Parameter range

0 - MAX_CHILDREN (maximum varies) [read-only]

Default

N/A

CE (Coordinator Enable)

Sets or displays whether the device is a coordinator.

Parameter range

Parameter	Description
0	Not a Coordinator
1	Coordinator (**SM** must be **0** to set **CE** to **1**)

Default

0

DO (Miscellaneous Device Options)

Sets or displays the device options.

Bit field:

Bit	Description
0	Reserved
1	Reserved for Smart Energy devices.
2	0/1 = First or Best Join. First join means the device joins the network through the first acceptable Beacon response it receives. Best join means the device joins the network through the strongest Beacon response it receives after searching all search mask channels.
3	Disable NULL transport key (coordinator only).
4	Disable Tx packet extended timeout.
5	Disable ACK for end device I/O sampling.
6	Enable high ram concentrator.
7	Enable **NW** to find new network before leaving the network. See Network watchdog for more information.

Parameter range

0x00 - 0xFF

Default

0x00

DC (Joining Device Controls)

Bit field:

Bit settings to enable or disable certain behaviors.

Bit	Description
0	Generate a preconfigured link key using the device's install code (KY will be ignored).
1	Network Leave Request Not Allowed. Indicates if a router node discards or accepts network leave commands.
2	Reserved
3	Reserved
4	Verbose Joining Mode. See Extended Modem Status - 0x98 for a full description.
5	An orphaned router will not leave the network but will attempt to rejoin the coordinator indefinitely. This functionality also requires the network watchdog to be enabled (**NW** > **0**). See Network watchdog for more information.
7	Enable **FR** after 60 seconds of no beacon responses during join.

Parameter range

0 - 0xFFFF

Default

0x00

II (Initial 16-bit PAN ID)

The preconfigured 16-bit PAN ID used when forming a network. Use this command to replace a coordinator node on an existing network.

When you set **II** to the default value (recommended) the module forms a network on a random 16-bit PAN ID.

Range

0 - 0xFFFF

Default

0xFFFF

ED (Energy Detect)

Start an Energy Detect scan. This command accepts an argument to specify the time in milliseconds to scan IEEE 802.15.4 channels 11 through 26. The device loops through all 16 channels until the time elapses and returns the maximal energy on each channel. In Transparent mode, a comma must follow each value with the list ending with a carriage return. The values returned reflect the detected energy level in units of -dBm. Convert an **ED** response of 49, 3A, and so on, to decimal to become -73 dBm, -58 dBm, and so on.

Parameter range

1 - 0xFF (x1 ms)

Default

0x10 (16 ms)

Addressing commands

The following AT commands are addressing commands.

SH (Serial Number High)

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

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

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

SL (Serial Number Low)

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

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

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

MY (16-bit Network Address)

Reads the 16-bit network address of the device.

A value of 0xFFFE means the device has not joined a Zigbee network.

Parameter range

0 - 0xFFFF [read-only]

Default

0 - 0xFFFE

MP (16-bit Parent Network Address)

Read the 16-bit network address of the device's parent. Coordinators and Routers will always show **0xFFFE**. An End Device will show its parent's address, or **0xFFFE** if it has not yet joined a PAN.

Parameter range

0 - 0xFFFE [read-only]

Default

0xFFFE

DH (Destination Address High)

Set or read the upper 32 bits of the 64-bit destination address. When you combine **DH** with **DL**, it defines the 64-bit destination address that the device uses for data transmission.

Special definitions for **DH** and **DL** include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).

Parameter range

0 - 0xFFFFFFFF

Default

0

DL (Destination Address Low)

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

Special definitions for **DH** and **DL** include **0x000000000000FFFF** (broadcast) and **0x0000000000000000** (coordinator).

Parameter range

0 - 0xFFFFFFFF

Default

0xFFFF (Coordinator)

0 (Router/End Device)

NI (Node Identifier)

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

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

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

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

Parameter range

A string of case-sensitive ASCII printable characters from 0 to 20 bytes in length. A carriage return or a comma automatically ends the command.

Default

0x20 (an ASCII space character)

NH (Maximum Unicast Hops)

Sets or displays the maximum number of hops across the network. This limit sets the maximum broadcast hops value (**BH**) and determines the unicast timeout.

The timeout is computed as (50 * **NH**) + 100 ms.

The default unicast timeout of 1.6 seconds (**NH**=0x1E) is enough time for data and the acknowledgment to traverse approximately 8 hops.

Parameter range

0 - 0xFF

Default

0x1E

BH (Broadcast Hops)

The maximum transmission hops for broadcast data transmissions.

Parameter range

0 - 0x1E

Default

    0

AR (Aggregate Routing Notification)

Set or read the periodic time for broadcasting aggregate route messages. If used, these messages enable many-to-one routing to the broadcasting device. Set **AR** to **0x00** to send only one broadcast, to **0xFF** to disable broadcasts, or to other values for periodic broadcasts in 10 second units.

Parameter range

    0 - 0xFF (x10 sec)

Default

    0xFF (disabled)

DD (Device Type Identifier)

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

Digi reserves the range 0 - 0xFFFFFF. For the XBee ZB SMT device, the device type is 0xA0000.

Parameter range

    0 - 0xFFFFFFFF

Default

    0xA0000

NT (Node Discover Timeout)

Sets or displays the amount of time a base node waits for responses from other nodes when using the **ND** (Node Discover) command.

When you issue the **ND** command, the transmission includes the **NT** value to provide all remote devices with a response timeout. Remote devices wait a random time, less than **NT**, before sending their response.

Parameter range

    0x20 - 0xFF (x 100 ms)

Default

    0x3C (6 seconds)

NO (Network Discovery Options)

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

Parameter range

    0 - 0x03 (bit field)

Bit field

Option	Description
0x01	Append the **DD** (Digi Device Identifier) value to **ND** responses and node identification frames.
0x02	Local device sends **ND** response when the **ND** is issued.

Default

0x0

NP (Maximum Packet Payload Bytes)

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

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

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

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

Parameter range

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

Default

[read-only]

CR (Conflict Report)

The number of PAN ID conflict reports that must be received by the network manager within one minute to trigger a PAN ID change.

A corrupt beacon can cause a report of a false PAN ID conflict.

A higher value reduces the chance of a false PAN ID change.

Setting **CR** to **0** sets the threshold value to the default configuration value (3).

Parameter range

1 - 0x3F

Default

3

Zigbee addressing commands

The following AT commands adjust the advanced communication settings that affect outgoing data transmissions.

SE (Source Endpoint)

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

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

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

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

Parameter range

> 0 - 0xFF

Default

> 0xE8

DE (Destination Endpoint)

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

Parameter range

> 0 - 0xFF

Default

> 0xE8

CI (Cluster ID)

Sets or displays the application layer ID value. Use this value as the cluster ID for all data transmissions. **CI** is only used in Transparent mode.

Parameter range

> 0 - 0xFFFF

Default

> 0x11 (Transparent data cluster ID)

TO (Transmit Options)

The bitfield that configures the transmit options for Transparent mode.

Sets or displays the Zigbee application layer source transmit options. Use this value as the transmit options for all data transmissions in transparent mode.

Parameter range

> 0 - 0xFF

Bit field:

Unused bits must be set to 0. These bits may be logically ORed together:

Parameter	Description
0x01	Disable retries and route repair.

Parameter	Description
0x20	Enable APS Encryption—if **EE** = **1**. **Note** This decreases the maximum RF payload by 4 bytes below the value reported by **NP**.
0x40	Use the extended timeout for this destination.

Default

0x00

RF interfacing commands

The following AT commands are RF interfacing commands.

PL (TX Power Level)

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

For XBee-PRO, **PL**= **4** is calibrated and the remaining power levels are approximate. The device recalibrates its power setting every 15 seconds based on factory calibration settings, the current temperature and the amount of voltage over the typical 3.3 V. If the input voltage is too high, the device resets.

For XBee, **PL** = **4**, **PM** = **1** is tested at the time of manufacturing. Other power levels are approximate. On channel 26, transmitter power will not exceed +3 dBm output.

Parameter range

These parameters equate to the following settings for the XBee RF module (boost mode disabled):

Setting	Power level
0	-5 dBm
1	-1 dBm
2	+1 dBm
3	+3 dBm
4	+5 dBm

These parameters equate to the following settings for the XBee-PRO RF module (boost mode enabled):

Setting	Power level
0	0 dBm (approximate)
1	+12 dBm (approximate)

Setting	Power level
2	+14 dBm (approximate)
3	+16 dBm (approximate)
4	+18 dBM

Default
    4

PP (Power at PL4)

Reads the dBm output when you select maximum power (PL4).

Note Read the maximum allowed power level when device is configured with **PL** = **4**. Use this command to determine if the device is a Pro or non-Pro variant. The value the command returns will be in hex representation: Pro = 0x14 and Non-Pro = 0x8.

Parameter range
    0x08 - 0x14

Default
    [read-only]

PM (Power Mode)

Set or read the power mode of the device. Enabling boost mode improves the receive sensitivity by 2dB and increase the transmit power by 3dB.

Parameter range
    0 - 1

Setting	Meaning
0	Boost mode disabled
1	Boost mode enabled

Default
    1

Security commands

The following AT commands are security commands.

EE (Encryption Enable)

Set or read the encryption enable setting.

Parameter range

0 - 1

Parameter	Description
0	Encryption Disabled
1	Encryption Enabled

Default

0

EO (Encryption Options)

Configure options for encryption when **EE** = **1**. Set unused option bits to **0**. Options include:

Options	Description
0x01	(Zigbee) Send the network key in the clear (unencrypted) over- the-air during a join
0x04	Use EUI64-hashed link keys (used on trust center only).
0x02	(Smart Energy) Enable as a trust center (Coordinator only)
0x08	(Smart Energy) Authenticate during joining (End Device and Router only)

Parameter range

0 - 0xFF

Default

N/A

KY (Link Key)

Sets the 128-bit AES link key value that the device uses for encryption and decryption. This command is write-only. If you set **KY** to **0** the coordinator transmits the network key in the clear to joining devices, and joining devices acquire the network key in the clear when joining.

When queried, **KY** will return **0** if no key has been set; otherwise, it will give an **OK** response to indicate a key is present.

Parameter range

128-bit value

Default

0

NK (Trust Center Network Key)

Set the 128-bit AES network encryption key. This command is write-only. If set to 0 (default), the device selects a random network key.

When queried, **KY** will return **0** if no key has been set; otherwise, it will give an **OK** response to indicate that a key is present.

Parameter range

128-bit value

Default

0

Serial interfacing commands

The following AT commands are serial interfacing commands.

BD (Interface Data Rate)

The device interprets any value above 0x0A as an actual baud rate. Standard baud rates up to 115200 are supported. Non-standard baud rates above 115200 are permitted but their performance is not guaranteed.

Parameter range

Standard baud rates: 0x0 - 0x0A

Value	Description
0x1	2,400 b/s
0x2	4,800 b/s
0x3	9,600 b/s
0x4	19,200 b/s
0x5	38,400 b/s
0x6	57,600 b/s
0x7	115,200 b/s
0x8	230,400 b/s
0x9	460,800 b/s
0xA	921,600 b/s

Default

0x03 (9600 b/s)

NB (Parity)

Set or read the serial parity settings for UART communications.

Parameter range

Parameter	Description
0x00	No parity
0x01	Even parity
0x02	Odd parity
0x03	Mark parity

Default

0x00

SB (Stop Bits)

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

Parameter range

0 - 1

Parameter	Configuration
0	One stop bit
1	Two stop bits

Default

0

RO (Packetization Timeout)

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

Set **RO** to **0** to transmit characters as they arrive instead of buffering them into one RF packet. The **RO** command is only supported when operating in Transparent mode.

Parameter range

0 - 0xFF (x character times)

Default

3

D6 (DIO6/RTS)

Sets or displays the DIO6/$\overline{RTS}$ configuration (TH pin 16/SMT pin 29).

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	$\overline{\text{RTS}}$ flow control
3	Digital input
4	Digital output, low
5	Digital output, high

Default

   0

D7 (DIO7/CTS)

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration (TH pin 12/SMT pin 25).

Parameter range

   0, 1, 3 - 7

Parameter	Description
0	Unmonitored digital input
1	$\overline{\text{CTS}}$ flow control
3	Digital input
4	Digital output, low
5	Digital output, high
6	RS-485 Tx enable, low Tx
7	RS-485 Tx enable high, high Tx

Default

   0x1

AP (API Enable)

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

Parameter range

   0 - 2

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

Default

    0

AO (API Options)

Configure the options for API. The current options select the type of receive API frame to send out the UART for received RF data packets.

Parameter range

    0x00 - 0x0B

Parameter	Description
0x00	Default API Rx Indicator enabled
0x01	Default API Explicit Rx Indicator - 0x91, this is for Explicit Addressing data frames.
0x03	Enable ZDO passthrough of ZDO requests to the serial port that are not supported by the stack, as well as Simple_Desc_req, Active_EP_req, and Match_Desc_req.
0x07	Same as 0x03, plus supported ZDO request passthrough.
0x0B	Same as 0x03, plus Binding Request passthrough.

Default

    0

Command mode options

The following commands are Command mode option commands.

CT (Command Mode Timeout)

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

Parameter range

    2 - 0x28F

Default

    0x64 (10 seconds)

GT (Guard Times)

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

Parameter range

    0x1 - 0x0CE4 (x 1 ms) (max of 3.3 decimal sec)

Default

>   0x3E8 (one second)

CC (Command Character)

Sets or displays the ASCII character value the device uses between Guard Times of the Command mode sequence (**GT** + **CC** + **GT**). The Command mode sequence enters the device into Command mode.

For more information about Command mode sequence, see Command mode options.

Parameter range

>   0 - 0xFF

Default

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

CN (Exit Command mode)

Immediately exits Command Mode and applies pending changes.

Parameter range

>   N/A

Default

>   N/A

Sleep commands

The following AT commands are sleep commands.

SP (Sleep Period)

Sets the duration of sleep time for the end device, up to 28 seconds. Use the **SN** command to extend the sleep time past 28 seconds.

On the parent, this value determines how long the parent buffers a message for the sleeping end device. Set the value to at least equal to the longest **SP** time of any child end device.

Parameter range

>   0x20 - 0xAF0 x 10ms (Quarter second resolution)

Default

>   0x20

SN (Number of Cycles Between ON_SLEEP)

Set or read the number of sleep periods value. This command controls the number of sleep periods that must elapse between assertions of the ON_SLEEP line during the wake time if no RF data is waiting for the end device. This command allows a host application to sleep for an extended time if no RF data is present.

Parameter range

> 1 - 0xFFFF

Default

> 1

SM (Sleep Mode)

Sets or displays the sleep mode of the device.

When **SM** > **0**, the device operates as an end device. However, **CE** must be **0** before **SM** can be set to a value greater than **0** to change the device to an end device. Changing a device from a router to an end device (or vice versa) forces the device to leave the network and attempt to join as the new device type when changes are applied.

Parameter range

0, 1, 4, 5

Parameter	Description
0	Sleep disabled (router)
1	Pin sleep enabled
4	Cyclic sleep enabled
5	Cyclic sleep, pin wake

Default

> 0 - Router
>
> 4 - End device

ST (Time before Sleep)

Sets or displays the wake time of the device.

The timer resets each time the device receives serial or RF data. Once the timer expires, an end device may enter low power operation. This applies to cyclic sleep end devices only.

Parameter range

> 1 - 0xFFFF (x 1 ms)

Default

> 0x1388 (5 seconds)

SO (Sleep Options)

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

Set unused option bits to 0. Sleep options should not be used for most applications. See Manage End Devices.

Parameter range

> 0 - 0xFF

Bit	Option
0x02	Always wake for **ST** time
0x04	Sleep entire **SN** * **SP** time

Default

> 0

WH (Wake Host Delay)

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

If you set **WH** to a non-zero value, this timer specifies a time in milliseconds that the device delays after waking from sleep before sending data out the UART or transmitting an I/O sample. If the device receives serial characters, the **WH** timer stops immediately.

Parameter range

> 0 - 0xFFFF (x 1 ms)

Default

> 0

PO (Polling Rate)

Set or read the end device poll rate.

Setting this to 0 (default) enables polling at 100 ms (default rate), advancing in 10 ms increments. Adaptive polling may allow the end device to poll more rapidly for a short time when receiving RF data.

Parameter range

0 - 0x3E8

Default

0

I/O settings commands

The following AT commands are I/O settings commands.

D0 (AD0/DIO0 Configuration)

Sets or displays the DIO0/AD0 configuration (TH pin 20/SMT pin 33).

Parameter range

0 - 5

Parameter	Description
0	Unmonitored digital input
1	Commissioning Pushbutton
2	Analog input, single ended
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

D1 (AD1/DIO1/PTI_En Configuration)

Sets or displays the AD1/DIO1/PTI_En configuration TH pin 19/SMT pin 32.

Parameter range

0 - 6

Parameter	Description
0	Unmonitored digital input
1	SPI_ATTN - Analog input, single ended for the through-hole device
2	Analog input, single ended
3	Digital input
4	Digital output, low

Parameter	Description
5	Digital output, high
6	Packet trace interface enable. Must be set along with **P9**= 6 to output traces for OTA sniffing.

Default

    0

D2 (AD2/DIO2 Configuration)

Sets or displays the DIO2/AD2 configuration (TH pin 18/SMT pin 31).

Parameter range

    0 - 5

    0 - 1

Parameter	Description
0	Unmonitored digital input
1	SPI_CLK for through-hole devices
2	Analog input, single ended
3	Digital input
4	Digital output, low
5	Digital output, high

Default

    0

D3 (AD3/DIO3 Configuration)

Sets or displays the DIO3/AD3 configuration (TH pin 17/SMT pin 30).

Parameter range

    0 - 5

Parameter	Description
0	Unmonitored digital input
1	SPI_$\overline{SSEL}$ for the through-hole device
2	Analog input, single ended
3	Digital input

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

Default

0

D4 (DIO4 Configuration)

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

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	SPI_MOSI for the through-hole device
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D5 (DIO5/Associate Configuration)

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

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	Associate LED indicator - blinks when associated
3	Digital input
4	Digital output, default low
5	Digital output, default high

Default

1

D8 (DIO8/DTR/SLP_RQ)

Sets or displays the DIO8/DTR/SLP_RQ configuration (TH pin 9/SMT pin 10).

Parameter range

   0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	Input to sleep and wake device
3	Digital input
4	Digital output, low
5	Digital output, high

Default

   N/A

D9 (DIO9/ON_SLEEP)

Sets or displays the DIO9/ON_SLEEP configuration (TH pin 13/SMT pin 26).

Parameter range

   0, 1, 3 - 5

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

Default

   1

P0 (RSSI/PWM0 Configuration)

Parameter range

   0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	RSSI PWM0
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high

Default

  1

P1 (DIO11/PWM1 Configuration)

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

Parameter range

  0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	Output 50% duty cycle clock at 32.787 kHz
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high

Default

  0

P2 (DIO12 Configuration)

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

Parameter range

Parameter	Description
0	Unmonitored digital input
1	SPI_MISO
3	3- Digital input, monitored
4	4- Digital output, default low
5	Digital output, default high

Default
> 0

P3 (DIO13/DOUT Configuration)

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

Parameter range
> 0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	Data out for UART
3	Monitored digital input
4	Digital output low
5	Digital output high

Default
> 1

P4 (DIO14/DIN)

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

Parameter range
> 0, 1, 3 - 5

Parameter	Description
0	Unmonitored digital input
1	Data in for UART
3	Digital input
4	Digital output low
5	Digital output high

Default
> 1

P5 (DIO15/SPI_MISO)

Sets or displays the DIO15/SPI_MISO configuration.
This only applies to surface-mount devices.

Parameter range

0, 1

Parameter	Description
0	Unmonitored digital input
1	Output from SPI port

Default

1

P6 (SPI_MOSI Configuration)

Sets or displays the DIO16/SPI_MOSI configuration.

This only applies to surface-mount 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 (DIO17/SPI_SSEL)

Sets or displays the DIO17/SPI_SSEL configuration.

This only applies to surface-mount devices.

Parameter range

0, 1

Parameter	Description
0	Disabled
1	SPI_SSEL

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

Default

    1

P8 (DIO18/SPI_SCLK)

Sets or displays the DIO18/SPI_CLK configuration.

This only applies to surface-mount devices.

Parameter range

    0, 1

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

Default

    1

P9 (DIO19/SPI_ATTN/PTI_DATA)

Sets or displays the DIO19 configuration.

This only applies to surface-mount devices.

Parameter range

    0, 1, 6

Parameter	Description
0	Disabled
1	$\overline{\text{SPI\_ATTN}}$

Parameter	Description
2	N/A
3	N/A
4	N/A
5	N/A
6	PTI_DATA

Default

   1

PR (Pull-up/Down Resistor Enable)

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

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

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

Bit	I/O line	Module pin
0	DIO4	24/SMT, 11/TH
1	AD3/DIO3	30/SMT, 17/TH
2	AD2/DIO2	31/SMT, 18/TH
3	AD1/DIO1	32/SMT, 19/TH
4	AD0/DIO0	33/SMT, 20/TH
5	$\overline{RTS}$/DIO6	SMT, 16/TH
6	DTR/SLEEP_REQUEST	10/SMT, 9/TH
7	DIN/$\overline{CONFIG}$	4/SMT, 3/TH
8	ASSOCIATE/DIO5	28/SMT, 15/TH
9	On/$\overline{SLEEP}$/DIO9	26/SMT, 13/TH
10	DIO12	5/SMT, 4/TH
11	PWM0/RSSI/DIO10/	7/SMT, 6/TH
12	PWM1/DIO11	8/SMT, 7/TH
13	$\overline{CTS}$/DIO7	25/SMT, 12/TH
14	DOUT/DIO13	3/SMT, 2/TH

Parameter range

   0 - 0x7FFF (bit field)

Default

0x1FFF

PD (Pull Up/Down Direction)

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

If the bit is set, the device uses an internal pull-up resistor. If it is clear, the device uses an internal pull-down resistor. See the **PR** command for the bit order.

Parameter range

0x0 - 0x7FFF

Default

0x1FBF

LT (Associate LED Blink Time)

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

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

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

Parameter range

0, 0x0A - 0xFF (100 - 2550 ms)

Default

0

RP (RSSI PWM Timer)

The PWM timer expiration in 0.1 seconds. **RP** sets the duration of pulse width modulation (PWM) signal output on the RSSI pin. The signal duty cycle updates with each received packet and shuts off when the timer expires.

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

Parameter range

0 - 0xFF (x 100 ms)

Default

0x28 (four seconds)

I/O sampling commands

The following AT commands configure I/O sampling parameters.

IR (I/O Sample Rate)

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

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

If you set the I/O sample rate to greater than **0**, the device samples and transmits all enabled digital I/O and analog inputs every **IR** milliseconds. I/O Samples transmit to the address specified by **DT**.

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

For more information, see the following commands:

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

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

Parameter range

0, 0x32 - 0xFFFF (ms)

Default

0

IC (Digital Change Detection)

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

IC works with the individual pin configuration commands (**D0** - **D9**, **P0** - **P4**) . If you enable a pin as a digital I/O, you can use the **IC** command to force an immediate I/O sample transmission when the DIO state changes. IC is a bitmask that you can use to enable or disable edge detection on individual channels.

Set unused bits to 0.

Bit	I/O line	Module pin
0	DIO0	18
1	DIO1	20
2	DIO2	22
3	DIO3	24
4	DIO4	26
5	DIO5	28
6	DIO6	41
7	DIO7	39

Bit	I/O line	Module pin
8	DIO8	30
9	DIO9	32
10	DIO10	36
11	DIO11	38
12	DIO12	40

Parameter range

0 - 0xFFFF (bit field)

Default

0

V+ (Voltage Supply Monitoring)

Set the voltage supply threshold with the **V+** command. If the measured supply voltage falls below or equal to this threshold, the supply voltage appends to the I/O sample set, and sets bit 7 of the Analog Channel Mask. Set **V+** to **0** by default (do not include the supply voltage). The units of this command are mV. For example, to include a measurement of the supply voltage when it falls below 2.7 V, set **V+** to **2700** = **0xA8A**.

Parameter range

0 - 0xFFFF

Default

0

Diagnostic commands

The following commands are diagnostic commands.

VR (Firmware Version)

Reads the firmware version on a device as a 4-digit hex number.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in the factory

HV (Hardware Version)

Display the hardware version number of the device.

Read the device's hardware version. Use this command to distinguish between different hardware platforms. The upper byte returns a value that is unique to each device type. The lower byte indicates the hardware revision.

Note The XBee returns a value of **0x22xx** for this command. The XBee-PRO returns a value of **0x21xx**.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in firmware

AI (Association Indication)

Read information regarding last node join request.

Status code	Meaning
0x00	Successfully formed or joined a network. (Coordinators form a network, routers and end devices join a network).
0x21	Scan found no PANs.
0x22	Scan found no valid PANs based on current SC and ID settings.
0x23	Valid Coordinator or Routers found, but they are not allowing joining (NJ expired).
0x24	No joinable beacons were found.
0x25	Unexpected state, node should not be attempting to join at this time.
0x27	Node Joining attempt failed (typically due to incompatible security settings).
0x2A	Coordinator Start attempt failed.
0x2B	Checking for an existing coordinator.
0x2C	Attempt to leave the network failed.
0xAB	Attempted to join a device that did not respond.
0xAD	Secure join error - network security key not received.
0xAF	Secure join error - joining device does not have the right preconfigured link key.
0xFF	Initialization time; no association status has been determined yet.

Applications should read **AI** until it returns 0x00, indicating a successful startup (coordinator) or join (routers and end devices).

Parameter range

0 - 0xFF [read-only]

Default

N/A

%V (Voltage Supply Monitoring)

Reads the voltage on the Vcc pin in mV.

Parameter range

0 - 0xFFFF [read only]

Default

N/A

DB (Received Signal Strength)

This command reports the received signal strength of the last received RF data packet or APS acknowledgment. The **DB** command only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link.

The **DB** command value is measured in -dBm. For example, if **DB** returns 0x50, then the RSSI of the last packet received was -80 dBm. Set **DB** to 0 to clear the current value, and it will be updated with the next valid packet received.

Parameter range

0 - 0xFF

Default

Read-only

TP (Temperature)

The current module temperature in degrees Celsius. The accuracy is ± 7 degrees. 1 ℃ = 0x0001 and -1 ℃ = 0xFFFF.

Note This command is only available on the XBee-PRO device.

Parameter range

0x0 - 0xFFFF

Default

N/A

VL (Version Long)

Shows detailed version information, device type, time stamp for the build, Ember stack version, and bootloader version.

Parameter range

N/A

Default

N/A

Execution commands

The location where most AT commands set or query register values, execution commands execute an action on the device. Execution commands are executed immediately and do not require changes to be applied.

AC (Apply Changes)

Immediately applies new settings without exiting Command mode.

Applies changes to all command registers and applies queued command register values. For example, changing the serial interface rate with the **BD** command does not change the UART interface rate until changes are applied with the **AC** command. The **CN** command and 0x08 API command frame also apply changes.

Parameter range

> N/A

Default

> N/A

AS (Active Scan)

Scans the neighborhood for beacon responses. The **AS** command is only valid as a local command. Response frames are structured as:

AS_type – unsigned byte = 2 - ZB firmware uses a different format than XBee Wi-Fi , which is type 1

Channel – unsigned byte

PAN – unsigned word in big endian format

Extended PAN – eight unsigned bytes in bit endian format

Allow Join – unsigned byte – 1 indicates join is enabled, 0 that it is disabled

Stack Profile – unsigned byte

LQI – unsigned byte, higher values are better

RSSI – signed byte, lower values are better

Parameter range

> 0 - 6

Default

> N/A

WR (Write)

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

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

Parameter range

> N/A

Default

> N/A

RE (Restore Defaults)

This command sets all parameters except **ZS** and **KY** to their default values. To change **ZS** and **KY**, you must explicitly set them. In order for the default parameters to persist through subsequent resets, send a separate **WR** command after **RE**. Read-only parameters are not directly affected by **RE** and reflect the current state of the device.

Parameter range

N/A

Default

N/A

FR (Software Reset)

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

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

Parameter range

N/A

Default

N/A

NR (Network Reset)

Resets network layer parameters on one or more modules within a PAN. Responds immediately with an **OK** then causes a network restart. The device loses all network configuration and routing information.

If **NR** = 0: Resets network layer parameters on the node issuing the command.

If **NR** = 1: Sends broadcast transmission to reset network layer parameters on all nodes in the PAN.

Note **NR** and **NR0** both perform the same function and may be used interchangeably.

Parameter range

0 - 1

Default

N/A

SI (Sleep Immediately)

Causes a cyclic sleep device to sleep immediately rather than wait for the **ST** timer to expire.

Note This command only has effect in API mode (sleeps immediately whether given as a 0x08 or 0x09 API frame), and no effect in AT command mode. AT command mode is exited only by the **CN** command or by timeout.

Note If you issue this command in Command Mode, the module remains in command mode until the **CT** timer expires or you issue a **CN** command.

Parameter

N/A

Default

N/A

CB (Commissioning Pushbutton)

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

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

See Commissioning pushbutton.

Parameter range

1, 2, 4

Default

N/A

&X (Clear Binding and Group Tables)

Resets the binding and group tables.

Parameter range

N/A

Default

N/A

ND (Node Discovery)

Discovers and reports all devices found in the network after a jittered time delay. For each discovered device, the following information is returned:

MY<CR> (2 bytes) (always 0xFFFE)

SH<CR> (4 bytes)

SL<CR> (4 bytes)

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

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

PARENT_NETWORK ADDRESS<CR> (2 bytes)

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

STATUS<CR> (1 byte: reserved)

PROFILE_ID<CR> (2 bytes)

MANUFACTURER_ID<CR> (2 bytes)

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

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

After (**NT** * 100) milliseconds, the command ends by returning a <CR>. **ND** also accepts a NI (Node Identifier) as a parameter (optional). In this case, only a device that matches the supplied identifier

responds after a jittered time delay. If there are no matching devices, the command returns an
ERROR.

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

The jittered time delay is based on the **NT** setting.

The radius of the **ND** command is set by BH (Broadcast Hops).

A status code of **1** = **ERROR** is returned if the transmit queue is full. That means there are already four
messages queued for transmission. The application is trying to send messages faster than the device
can process the requests. The application may either try again later, be redesigned to send messages
at a slower rate, or wait for a Tx Status response for a prior message before attempting to send
another.

For more information about the options that affect the behavior of the **ND** command, see NO
(Network Discovery Options).

Parameter range

20-byte printable ASCII string

Default

ASCII space character (0x20)

DN (Destination Node)

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

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

When **DN** is sent in Command mode (AT firmware):

1. The device sets **DL** and **DH** to the address of the device with the matching **NI** string.
   The address selected (either 16-bit short address or 64-bit extended address) is chosen based
   on the destination device's **MY** command configuration.

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 (API firmware):

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.

Parameter range

Up to 20-byte printable ASCII string

Default

N/A

DJ (Disable Joining)

Prevent a local device from joining a network.

Note This parameter is not written to flash with the **WR** command and reverts to default after a power cycle.

Parameter range

0 - 1

Value	Description
0	Join enabled
1	Join disabled

Default

0

IS (Force Sample)

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

Parameter range

N/A

Default

N/A

Module support

This section provides customization information for the XBee/XBee-PRO Zigbee RF Module. You can customize default parameters, or write or load custom firmware for the Ember EM357 chip.

Configure the device using XCTU

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

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

Click **Discover devices** and follow the instructions. XCTU should discover the connected XBee/XBee-PRO Zigbee RF Modules using the provided settings.

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

Software libraries

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

Customize XBee Zigbee firmware

Once device parameters are tested in an application and finalized, we can manufacture devices with specific, customer-defined configurations. These custom configurations can lock in a firmware version or set command values when the devices are manufactured, eliminating the need for customers to adjust device parameters on arrival. Alternatively, we can program custom firmware, including Ember's EZSP UART image, into the devices during manufacturing. Contact Digi to create a custom configuration.

XBee bootloader

XBee devices use a modified version of Ember's bootloader. This bootloader version supports a custom entry mechanism that uses module pins DIN (TH pin 4/SMT pin 4), DTR / SLEEP_RQ (TH pin 9/SMT pin 10), and RTS (TH pin 16/SMT pin 29).

To invoke the bootloader, do the following:

1. Set $\overline{\text{DTR}}$/SLEEP_RQ low (TTL 0V) and RTS high.

2. Send a serial break to the DIN pin and power cycle or reset the module.

3. When the device powers up, set $\overline{\text{DTR}}$/SLEEP_RQ and DIN to low (TTL 0V) and $\overline{\text{RTS}}$ should be high.

4. Terminate the serial break and send a carriage return at 115200 baud to the device.

5. If successful, the device sends the Ember bootloader menu out the DOUT pin at 115200 baud.

6. You can send commands to the bootloader at 115200 b/s.

Note Disable hardware flow control when entering and communicating with the Ember 357 bootloader.

All serial communications with the module use 8 data bits, no parity bit, and 1 stop bit.

You can update firmware on the XBee/XBee-PRO Zigbee RF Module serially. This is done by invoking the bootloader and transferring the firmware image using XMODEM.

Serial firmware updates

You can update firmware on the XBee/XBee-PRO Zigbee RF Module serially.

Serial firmware updates use the XBee custom bootloader which ships in all units. This bootloader is based on Ember's standalone bootloader, but with a modified entry mechanism. The modified entry mechanism uses device pins 4, 10, and 29 (DIN, DTR, and RTS respectively) on the SMT, and pins 3, 9, 16 on the TH.

XCTU can update firmware serially on the XBee/XBee-PRO Zigbee RF Module. Contact Digi support for details.

If an application requires custom firmware to update the XBee firmware serially, proceed to Invoke the XBee bootloader.

Invoke the XBee bootloader

See XBee bootloader for steps to invoke the bootloader using RS-232 signals. You can also invoke the bootloader by issuing a command using XCTU. The application makes an explicit call to the bootloader, which does not return.

If there is no valid application, the bootloader always runs.

Send a firmware image

After invoking the bootloader, the Ember bootloader sends the bootloader menu characters out the serial port, which may be the UART at 115200 b/s or the SPI, where the attached SPI master provides the clock rate. To upload a firmware image:

1. Look for the bootloader prompt **BL** > to ensure the bootloader is active.
2. Send an ASCII **1** character to initiate a firmware update.
3. After sending a **1**, the device waits for an XModem CRC upload of a .ebl image over the serial line at 115200 b/s. Send the .ebl file to the device in order.

If the firmware image is successfully loaded, the bootloader outputs a "complete" string. Invoke the newly loaded firmware by sending a **2** to the device.

If the firmware image is not successfully loaded, the bootloader outputs an "aborted string". It return to the main bootloader menu. Some causes for failure are:

- Over 1 minute passes after the command to send the firmware image and the first block of the image has not yet been sent.
- A power cycle or reset event occurs during the firmware load.
- A file error or a flash error occurs during the firmware load.

Writing custom firmware

You can use the XBee module as a hardware development platform for the EM357. You can develop custom firmware images around the EmberZNet 4.2.xx mesh stacks (for the EM357) and uploaded to the XBee.

⚠️ **CAUTION!** If you are programming firmware through the JTAG interface, you can potentially erase the XBee bootloader. If this occurs, serial firmware updates will not work.

Regulatory compliance

XBee modules are FCC and European certified for operation on all 16 channels. You can configure the EM357 output power up to 8 dBm with boost mode enabled on channels 11 through 25. On channel 26 you must reduce the power to 3 dBm.

XBee-PRO devices are FCC certified for operation on all 16 channels. The XBee-PRO contains a power compensation method to adjust the output power near 18 dBm on channels 11 through 25. You must configure the EM357 with an output power so the device outputs 18 dBm or less on channels 11 through 25. On channel 26, you must reduce the power to no more than 0 dBm. The end product is responsible to adhere to these requirements.

This section describes how to configure GPIOs to function correctly in custom applications that run on the XBee modules.

Enable GPIO 1 and 2

For GPIO pins to be configurable, the application must set the GPIO_PxCFG registers to enable the appropriate GPIO. The following table lists values for configuring the GPIO pins. Other functionality is affected by these settings. See the *EM357 datasheet* from Ember for a complete listing of functionality.

GPIO mode	GPIO_ PxCFGH/L	Description
Analog	0x0	Analog input or output. When in analog mode, the digital input (GPIO_PxIN) always reads 1.
Input (floating)	0x4	Digital input without an internal pull-up or pull-down. Output is disabled.
Input (pull-up or pull-down)	0x8	Digital input with an internal pull-up or pull-down. A set bit in GPIO_PxOUT selects pull-up and a cleared bit selects pull-down. Output is disabled.
Output (push-pull)	0x1	Push-pull output. GPIO_PxOUT controls the output.
Output (open-drain)	0x5	Open-drain output. GPIO_PxOUT controls the output. If a pull-up is required, it must be external.
Alternate Output (push-pull)	0x9	Push-pull output. An on-board peripheral controls the output.
Alternate Output (open-drain)	0xD	Open-drain output. An on-board peripheral controls the output. If a pull-up is required, it must be external.

For more information on configuring and setting GPIOs, consult the EM357 specification.

Detect XBee versus XBee-PRO

For some applications, you may need to determine if the code is running on an XBee or an XBee-PRO device.

1. Use the PC7 pin on the EM357 to identify the device type.

2. Connect PC7 to ground on the XBee module.

3. Use the following code to determine if a device is an XBee or an XBee-PRO:

```
GPIO_PCSET = 0x80; // Enable pullup resistor
GPIO_PCCFGH &= 0x0fff; // Clear PC7 config
GPIO_PCCFGH |= 0x8000;// Set PC7 as input with pullup/pulldown
if (GPIO_PCIN & 0x80) {
    ModuleIsXBeePro = true;
} else {
    ModuleIsXBeePro = false;
}
```

Special instructions for using the JTAG interface

There are four JTAG programming pins on the XBee/XBee-PRO Zigbee RF Module through which firmware can be loaded onto the EM357 processor. Three of these pins are also connected to a second pin on the device and are used for separate functions. The following table indicates the JTAG signal name, the primary connection pin on the device, the secondary connection pin, and the secondary signal name.

CAUTION! Do not load the secondary pins with circuitry that might interfere with JTAG programming (for example, an LED tied directly to the ASSOCIATE / DIO5 line). Any loading circuitry should be buffered to avoid conflicts (for example, connecting ASSOCIATE / DIO5 to the gate of a MOSFET which drives the LED).

JTAG pin name	Primary XBee pin	Secondary XBee pin	Secondary pin name
JTCK	18	N/A	N/A
JTDO	19	26	ON / SLEEP / DIO9
JTDI	20	28	ASSOCIATE / DIO5
JTMS	21	5	DIO12

Regulatory information

United States (FCC)

XBee/XBee-PRO Zigbee RF Modules comply with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

To fulfill FCC Certification, the OEM must comply with the following regulations:

1. The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.

2. RF Modules may only be used with antennas that have been tested and approved for use with the modules.

OEM labeling requirements

WARNING! As an Original Equipment Manufacturer (OEM) you must ensure that FCC labeling requirements are met. You must include a clearly visible label on the outside of the final product enclosure that displays the following content:

Required FCC Label for OEM products containing the XBee S2D SMT RF Module

Contains FCC ID: MCQ-S2DSM

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1.) this device may not cause harmful interference and (2.) this device must accept any interference received, including interference that may cause undesired operation.

Required FCC Label for OEM products containing the XBee S2C SMT RF Module

Contains FCC ID: MCQ-XBS2C

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1.) this device may not cause harmful interference and (2.) this device must accept any interference received, including interference that may cause undesired operation.

Required FCC Label for OEM products containing the XBee-PRO S2C SMT RF Module

Contains FCC ID: MCQ-PS2CSM

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1.) this device may not cause harmful interference and (2.) this device must accept any interference received, including interference that may cause undesired operation.

Note Legacy XBee-PRO SMT (Model: PRO S2C; hardware version 21xx) has FCC ID: MCQ-XBPS2C.

Required FCC Label for OEM products containing the XBee S2C TH RF Module

Contains FCC ID: MCQ-S2CTH

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1.) this device may not cause harmful interference and (2.) this device must accept any interference received, including interference that may cause undesired operation.

Required FCC Label for OEM products containing the XBee-PRO S2C TH RF Module

Contains FCC ID: MCQ-PS2CTH

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1.) this device may not cause harmful interference and (2.) this device must accept any interference received, including interference that may cause undesired operation.

FCC notices

IMPORTANT: XBee/XBee-PRO Zigbee RF Modules have been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

IMPORTANT: OEMs must test final product to comply with unintentional radiators (FCC section 15.107 & 15.109) before declaring compliance of their final product to Part 15 of the FCC Rules.

IMPORTANT: The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, the device must undergo SAR testing.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Re-orient or relocate the receiving antenna, Increase the separation between the equipment and receiver, Connect equipment and receiver to outlets on different circuits, or Consult the dealer or an experienced radio/TV technician for help.

FCC-approved antennas (2.4 GHz)

The XBee/XBee-PRO Zigbee RF Module can be installed using antennas and cables constructed with non-standard connectors (RPSMA, RPTNC, etc.) An adapter cable may be necessary to attach the XBee connector to the antenna connector.

The modules are FCC approved for fixed base station and mobile applications for the channels indicated in the tables below. If the antenna is mounted at least 25 cm (10 in) from nearby persons, the application is considered a mobile application. Antennas not listed in the table must be tested to comply with FCC Section 15.203 (Unique Antenna Connectors) and Section 15.247 (Emissions).

The antennas in the tables below have been approved for use with this module. Cable loss is required when using gain antennas as shown in the tables.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

1. If using the RF module in a portable application (for example, if the module is used in a hand-held device and the antenna is less than 25 cm from the human body when the device is in operation), The integrator is responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1091 and FCC Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields, OET Bulletin and Supplement C. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person.

2. Zigbee firmware will be limited to 32% duty cycle on channel 26.

3. If you are using 802.15.4 firmware with up to a 66% duty cycle on this channel.

XBee Zigbee SMT RF module

The following table shows the antennas approved for use with the XBee Zigbee SMT RF module.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Minimum cable loss/power reduction/attenuation required	
Integral antennas						
29000313	Integral PCB antenna	0.0	Fixed/Mobile	20 cm	N/A	N/A
A24-QI	Monopole (Integrated whip)	1.5	Fixed/Mobile	20 cm	N/A	N/A
Dipole antennas						
A24-HASM-450	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed	20 cm	N/A	N/A
A24-HABSM*	Dipole (Articulated RPSMA)	2.1	Fixed	20 cm	N/A	N/A
29000095	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave articulated bulkhead mount U.FL. w/ 5" pigtail)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA - 5.25")	2.1	Fixed	20 cm	N/A	N/A
Omni-directional antennas						
A24-F2NF	Omni-directional (Fiberglass base station)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-F3NF	Omni-directional (Fiberglass base station)	3.0	Fixed/Mobile	20 cm	N/A	N/A
A24-F5NF	Omni-directional (Fiberglass base station)	5.0	Fixed	20 cm	N/A	N/A
A24-F8NF	Omni-directional (Fiberglass base station)	8.0	Fixed	2 m	N/A	N/A
A24-F9NF	Omni-directional (Fiberglass base station)	9.5	Fixed	2 m	N/A	N/A

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Minimum cable loss/power reduction/attenuation required	
A24-F10NF	Omni-directional (Fiberglass base station)	10.0	Fixed	2 m	N/A	N/A
A24-F12NF	Omni-directional (Fiberglass base station)	12.0	Fixed	2 m	N/A	2.0
A24-W7NF	Omni-directional (Fiberglass base station)	7.2	Fixed	2 m	N/A	N/A
A24-M7NF	Omni-directional (Mag-mount base station)	7.2	Fixed	2 m	N/A	N/A
A24-F15NF	Omni-directional (Fiberglass base station)	15.0	Fixed	2 m	N/A	5.0
Panel antennas						
A24-P8SF	Flat Panel	8.5	Fixed	2 m	N/A	3.0
A24-P8NF	Flat Panel	8.5	Fixed	2 m	N/A	3.0
A24-P13NF	Flat Panel	13.0	Fixed	2 m	N/A	7.5
A24-P14NF	Flat Panel	14.0	Fixed	2 m	N/A	8.5
A24-P15NF	Flat Panel	15.0	Fixed	2 m	N/A	9.5
A24-P16NF	Flat Panel	16.0	Fixed	2 m	N/A	10.5
A24-P19NF	Flat Panel	19.0	Fixed	2 m	N/A	13.5
Yagi antennas						
A24-Y6NF	Yagi (6-element)	8.8	Fixed	2 m	N/A	2.8
A24-Y7NF	Yagi (7-element)	9.0	Fixed	2 m	N/A	3.0
A24-Y9NF	Yagi (9-element)	10.0	Fixed	2 m	N/A	4.0
A24-Y10NF	Yagi (10-element)	11.0	Fixed	2 m	N/A	5.0
A24-Y12NF	Yagi (12-element)	12.0	Fixed	2 m	N/A	6.0
A24-Y13NF	Yagi (13-element)	12.0	Fixed	2 m	N/A	6.0

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Minimum cable loss/power reduction/attenuation required	
A24-Y15NF	Yagi (15-element)	12.5	Fixed	2 m	N/A	6.5
A24-Y16NF	Yagi (16-element)	13.5	Fixed	2 m	N/A	7.5
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5	Fixed	2 m	N/A	7.5
A24-Y18NF	Yagi (18-element)	15.0	Fixed	2 m	N/A	9.0

XBee Zigbee TH RF module

The following table shows the antennas approved for use with the XBee Zigbee TH RF Module.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
Integral antennas						
29000294	Integral PCB antenna	-0.5	Fixed/Mobile	20 cm	N/A	N/A
A24-QI	Monopole (Integrated whip)	1.5	Fixed/Mobile	20 cm	N/A	N/A
Dipole antennas						
A24-HASM-450	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed	20 cm	N/A	N/A
A24-HABSM*	Dipole (Articulated RPSMA)	2.1	Fixed	20 cm	N/A	N/A
29000095	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave articulated bulkhead mount U.FL. w/ 5" pigtail)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA - 5.25")	2.1	Fixed	20 cm	N/A	N/A
Omni-directional antennas						
A24-F2NF	Omni-directional (Fiberglass base station)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-F3NF	Omni-directional (Fiberglass base station)	3.0	Fixed/Mobile	20 cm	N/A	N/A
A24-F5NF	Omni-directional (Fiberglass base station)	5.0	Fixed	20 cm	N/A	N/A

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-F8NF	Omni-directional (Fiberglass base station)	8.0	Fixed	2 m	N/A	2.0
A24-F9NF	Omni-directional (Fiberglass base station)	9.5	Fixed	2 m	N/A	3.5
A24-F10NF	Omni-directional (Fiberglass base station)	10.0	Fixed	2 m	N/A	4.0
A24-F12NF	Omni-directional (Fiberglass base station)	12.0	Fixed	2 m	N/A	6.0
A24-W7NF	Omni-directional (Fiberglass base station)	7.2	Fixed	2 m	N/A	1.2
A24-M7NF	Omni-directional (Mag-mount base station)	7.2	Fixed	2 m	N/A	1.2
A24-F15NF	Omni-directional (Fiberglass base station)	15.0	Fixed	2 m	N/A	9.0
Panel antennas						
A24-P8SF	Flat Panel	8.5	Fixed	2 m	N/A	2.5
A24-P8NF	Flat Panel	8.5	Fixed	2 m	N/A	2.5
A24-P13NF	Flat Panel	13.0	Fixed	2 m	N/A	7.0
A24-P14NF	Flat Panel	14.0	Fixed	2 m	N/A	8.0
A24-P15NF	Flat Panel	15.0	Fixed	2 m	N/A	9.0
A24-P16NF	Flat Panel	16.0	Fixed	2 m	N/A	10.0
A24-P19NF	Flat Panel	19.0	Fixed	2 m	N/A	13.0
Yagi antennas						
A24-Y6NF	Yagi (6-element)	8.8	Fixed	2 m	N/A	2.8
A24-Y7NF	Yagi (7-element)	9.0	Fixed	2 m	N/A	3
A24-Y9NF	Yagi (9-element)	10.0	Fixed	2 m	N/A	4

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-Y10NF	Yagi (10-element)	11.0	Fixed	2 m	N/A	5
A24-Y12NF	Yagi (12-element)	12.0	Fixed	2 m	N/A	6.5
A24-Y13NF	Yagi (13-element)	12.0	Fixed	2 m	N/A	6.5
A24-Y15NF	Yagi (15-element)	12.5	Fixed	2 m	N/A	6.5
A24-Y16NF	Yagi (16-element)	13.5	Fixed	2 m	N/A	7.5
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5	Fixed	2 m	N/A	7.5
A24-Y18NF	Yagi (18-element)	15.0	Fixed	2 m	0.4	9.0

.

XBee-PRO Zigbee SMT RF module

The following table shows the antennas approved for use with the XBee-PRO Zigbee SMT RF Module.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type (description)	Gain (dBi)	Application*	Min separation	Required antenna cable loss (dB) Channels 11-25	Channel 26
Internal antennas						
29000313	Integral PCB antenna	0.0	Fixed/Mobile	20 cm	N/A	N/A
A24-QI	Monopole (Integrated whip)	1.5	Fixed/Mobile	20 cm	N/A	N/A
Dipole antennas						
A24-HASM-450	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed	20 cm	N/A	N/A
A24-HABSM*	Dipole (Articulated RPSMA)	2.1	Fixed	20 cm	N/A	N/A
29000095	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave articulated bulkhead mount U.FL. w/ 5" pigtail)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA - 5.25")	2.1	Fixed	20 cm	N/A	N/A
Omni-directional antennas						
A24-F2NF	Omni-directional (Fiberglass base station)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-F3NF	Omni-directional (Fiberglass base station)	3.0	Fixed/Mobile	20 cm	N/A	N/A
A24-F5NF	Omni-directional (Fiberglass base station)	5.0	Fixed	20 cm	N/A	N/A

Part number	Type (description)	Gain (dBi)	Application*	Min separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-F8NF	Omni-directional (Fiberglass base station)	8.0	Fixed	2 m	N/A	1.3
A24-F9NF	Omni-directional (Fiberglass base station)	9.5	Fixed	2 m	N/A	2.8
A24-F10NF	Omni-directional (Fiberglass base station)	10	Fixed	2 m	N/A	3.3
A24-F12NF	Omni-directional (Fiberglass base station)	12	Fixed	2 m	N/A	5.3
A24-W7NF	Omni-directional (Fiberglass base station)	7.2	Fixed	2 m	N/A	.5
A24-M7NF	Omni-directional (Mag-mount base station)	7.2	Fixed	2 m	N/A	.5
A24-F15NF	Omni-directional (Fiberglass base station)	15.0	Fixed	2 m	1.1	8.3
Panel antennas						
A24-P8SF	Flat Panel	8.5	Fixed	2 m	2.8	4.5
A24-P8NF	Flat Panel	8.5	Fixed	2 m	2.8	4.5
A24-P13NF	Flat Panel	13.0	Fixed	2 m	7.3	9.0
A24-P14NF	Flat Panel	14.0	Fixed	2 m	8.3	10.0
A24-P15NF	Flat Panel	15.0	Fixed	2 m	9.3	11.0
A24-P16NF	Flat Panel	16.0	Fixed	2 m	10.3	12.0
A24-P19NF	Flat Panel	19.0	Fixed	2 m	13.3	15.0
Yagi antennas						
A24-Y6NF	Yagi (6-element)	8.8	Fixed	2 m	2.4	4.2
A24-Y7NF	Yagi (7-element)	9.0	Fixed	2 m	2.6	4.4
A24-Y9NF	Yagi (9-element)	10.0	Fixed	2 m	3.6	5.4

Part number	Type (description)	Gain (dBi)	Application*	Min separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-Y10NF	Yagi (10-element)	11.0	Fixed	2 m	4.6	6.4
A24-Y12NF	Yagi (12-element)	12.0	Fixed	2 m	5.6	7.4
A24-Y13NF	Yagi (13-element)	12.0	Fixed	2 m	5.6	7.4
A24-Y15NF	Yagi (15-element)	12.5	Fixed	2 m	6.1	7.9
A24-Y16NF	Yagi (16-element)	13.5	Fixed	2 m	7.1	8.9
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5	Fixed	2 m	7.1	8.9
A24-Y18NF	Yagi (18-element)	15.0	Fixed	2 m	8.6	10.4

XBee-PRO Zigbee TH RF module

The following table shows the antennas approved for use with the XBee-PRO Zigbee TH RF Module.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB) Channels 11-25	Channel 26
Integral antennas						
29000294	Integral PCB antenna	-0.5	Fixed/Mobile	20 cm	N/A	N/A
A24-QI	Monopole (Integrated whip)	1.5	Fixed/Mobile	20 cm	N/A	N/A
Dipole antennas						
A24-HASM-450	Dipole (Half-wave articulated RPSMA - 4.5”)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HABSM*	Dipole (Articulated RPSMA)	2.1	Fixed	20 cm	N/A	N/A
29000095	Dipole (Half-wave articulated RPSMA - 4.5”)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave articulated bulkhead mount U.FL. w/ 5” pigtail)	2.1	Fixed	20 cm	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA - 5.25")	2.1	Fixed/ Mobile	20 cm	N/A	N/A
Omni-directional antennas						
A24-F2NF	Omni-directional (Fiberglass base station)	2.1	Fixed/Mobile	20 cm	N/A	N/A
A24-F3NF	Omni-directional (Fiberglass base station)	3.0	Fixed/Mobile	20 cm	N/A	N/A
A24-F5NF	Omni-directional (Fiberglass base station)	5.0	Fixed	20 cm	N/A	N/A

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-F8NF	Omni-directional (Fiberglass base station)	8.0	Fixed	2 m	N/A	N/A
A24-F9NF	Omni-directional (Fiberglass base station)	9.5	Fixed	2 m	N/A	N/A
A24-F10NF	Omni-directional (Fiberglass base station)	10.0	Fixed	2 m	N/A	N/A
A24-F12NF	Omni-directional (Fiberglass base station)	12.0	Fixed	2 m	N/A	.9
A24-W7NF	Omni-directional (base station)	7.2	Fixed	2 m	N/A	N/A
A24-M7NF	Omni-directional (Mag-mount base station)	7.2	Fixed	2 m	N/A	N/A
A24-F15NF	Omni-directional (Fiberglass base station)	15.0	Fixed	2 m	2.5	3.9
Panel antennas						
A24-P8SF	Flat Panel	8.5	Fixed	2 m	1	1.6
A24-P8NF	Flat Panel	8.5	Fixed	2 m	1	1.6
A24-P13NF	Flat Panel	13	Fixed	2 m	5.5	6.1
A24-P14NF	Flat Panel	14	Fixed	2 m	6.5	7.1
A24-P15NF	Flat Panel	15.0	Fixed	2 m	7.5	8.1
A24-P16NF	Flat Panel	16.0	Fixed	2 m	8.5	9.1
A24-19NF	Flat Panel	19.0	Fixed	2 m	11.5	12.1
Yagi antennas						
A24-Y6NF	Yagi (6-element)	8.8	Fixed	2 m	.3	N/A
A24-Y7NF	Yagi (7-element)	9.0	Fixed	2 m	.5	N/A
A24-Y9NF	Yagi (9-element)	10.0	Fixed	2 m	1.5	1.0

Part number	Type (description)	Gain (dBi)	Application*	Min. separation	Required antenna cable loss (dB)	
					Channels 11-25	Channel 26
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed	2 m	2.5	2.0
A24-Y12NF	Yagi (12-element)	12.0	Fixed	2 m	3.5	3.0
A24-Y13NF	Yagi (13-element)	12.0	Fixed	2 m	3.5	3.0
A24-Y15NF	Yagi (15-element)	12.5	Fixed	2 m	4.0	3.5
A24-Y16NF	Yagi (16-element)	13.5	Fixed	2 m	5.0	4.5
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5	Fixed	2 m	5.0	4.5
A24-Y18NF	Yagi (18-element)	15.0	Fixed	2 m	6.5	6.0

XBee S2D SMT module

The following table shows the antennas approved for use with the XBee S2D SMT surface-mount module. See Associated antenna descriptions for additional cable loss required beyond the antennas listed below.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type (description)	Gain (dBi)	Application 1	Min. separation	Required cable-loss or power reduction from +8 dBm channels 11 to 24	Required cable-loss or power reduction from +8 dBm channel 25	Required cable-loss or power reduction from +1 dBm channel 26
Yagi Class Antennas							
A24-Y6NF	Yagi (6-element)	8.8	Fixed	20 cm	0	0	5
A24-Y7NF	Yagi (7-element)	9.0	Fixed	20 cm	0	0	5
A24-Y9NF	Yagi (9-element)	10.0	Fixed	20 cm	0	0	6
A24-Y10NF	Yagi (10-element)	11.0	Fixed	20 cm	0	1	7
A24-Y12NF	Yagi (12-element)	12.0	Fixed	20 cm	0	2	8
A24-Y13NF	Yagi (13-element)	12.0	Fixed	20 cm	0	2	8
A24-Y15NF	Yagi (15-element)	12.5	Fixed	20 cm	0	2.5	8.5
A24-Y16NF	Yagi (16-element)	13.5	Fixed	20 cm	0	3.5	9.5
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5	Fixed	20 cm	0	3.5	9.5
A24-Y18NF	Yagi (18-element)	15.0	Fixed	20 cm	0	5	11
Omni-directional antennas							
A24-F2NF	Omni-directional (Fiberglass base station)	2.1	Fixed/Mobile	20 cm	0	0	0

Part number	Type (description)	Gain (dBi)	Application [1]	Min. separation	Required cable-loss or power reduction from +8 dBm channels 11 to 24	Required cable-loss or power reduction from +8 dBm channel 25	Required cable-loss or power reduction from +1 dBm channel 26
A24-F3NF	Omni-directional (Fiberglass base station)	3.0	Fixed/Mobile	20 cm	0	0	0
A24-F5NF	Omni-directional (Fiberglass base station)	5.0	Fixed/Mobile	20 cm	0	0	0
A24-F8NF	Omni-directional (Fiberglass base station)	8.0	Fixed	20 cm	0	0	0
A24-F9NF	Omni-directional (Fiberglass base station)	9.5	Fixed	20 cm	0	0	1.5
A24-F10NF	Omni-directional (Fiberglass base station)	10.0	Fixed	20 cm	0	0	2.5
A24-F12NF	Omni-directional (Fiberglass base station)	12.0	Fixed	20 cm	0	0	4.5
A24-F15NF	Omni-directional (Fiberglass base station)	15.0	Fixed	20 cm	0	0	7.5
A24-W7NF	Omni-directional (Base station)	7.2	Fixed	20 cm	0	0	0
A24-M7NF	Omni-directional (Mag-mount base station)	7.2	Fixed	20 cm	0	0	0

Part number	Type (description)	Gain (dBi)	Application [1]	Min. separation	Required cable-loss or power reduction from +8 dBm channels 11 to 24	Required cable-loss or power reduction from +8 dBm channel 25	Required cable-loss or power reduction from +1 dBm channel 26
Dipole Antennas							
A24-HASM-450	Dipole (Half-wave articulated RPSMA - 4.5")	2.1	Fixed/Mobile	20 cm	0	0	0
A24-HABSM*	Dipole (Articulated RPSMA)	2.1	Fixed	20 cm	0	0	0
A24-HABUF-P5I	Dipole (Half-wave articulated bulkhead mount U.FL. w/ 5" pigtail)	2.1	Fixed	20 cm	0	0	0
A24-HASM-525	Dipole (Half-wave articulated RPSMA - 5.25")	2.1	Fixed	20 cm	0	0	0
Panel class antennas							
A24-P8SF	Flat Panel	8.5	Fixed	20 cm	0	0	5.5
A24-P8NF	Flat Panel	8.5	Fixed	20 cm	0	0	5.5
A24-P13NF	Flat Panel	13.0	Fixed	20 cm	0	2	10
A24-P14NF	Flat Panel	14.0	Fixed	20 cm	0	3	11
A24-P15NF	Flat Panel	15.0	Fixed	20 cm	0	4	12
A24-P16NF	Flat Panel	16.0	Fixed	20 cm	0	5	13
A24-P19NF	Flat Panel	19.0	Fixed	20 cm	0	8	16
Integral antennas							
29000313	Integral PCB antenna	0.0	Fixed	20 cm	0	0	0

Associated antenna descriptions

The following table shows the associated antenna descriptions.

Antenna type	Maximum gain allowed	Required minimum assembly and cable loss for maximum gain of antenna
Integral PCB	0.0 dBi	0.0 dB
Dipole	2.1 dBi	0.63 dB
Omni-directional	15.0 dBi	1.25 dB
Yagi	15.0 dBi	1.25 dB
Flat panel	19.0 dBi	1.25 dB

RF exposure

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

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

FCC publication 996369 related information

In publication 996369 section D03, the FCC requires information concerning a module to be presented by OEM manufacturers. This section assists in answering or fulfilling these requirements.

2.1 General

No requirements are associated with this section.

2.2 List of applicable FCC rules

This module conforms to FCC Part 15.247.

2.3 Summarize the specific operational use conditions

Certain approved antennas require attenuation for operation. For the XBee/XBee-PRO S2C Zigbee modules, see FCC-approved antennas (2.4 GHz).

Host product user guides should include the antenna table if end customers are permitted to select antennas.

2.4 Limited module procedures

Not applicable.

2.5 Trace antenna designs

While it is possible to build a trace antenna into the host PCB, this requires at least a Class II permissive change to the FCC grant which includes significant extra testing and cost. If an embedded trace antenna is desired, select the XBee module variant with the preferred antenna.

2.6 RF exposure considerations

For RF exposure considerations see RF exposure and FCC-approved antennas (2.4 GHz).

Host product manufacturers need to provide end-users a copy of the "RF Exposure" section of the manual: RF exposure.

2.7 Antennas

A list of approved antennas is provided for the XBee/XBee-PRO S2C Zigbee modules. See FCC-approved antennas (2.4 GHz).

2.8 Label and compliance information

Host product manufacturers need to follow the sticker guidelines outlined in OEM labeling requirements.

2.9 Information on test modes and additional testing requirements

Contact a Digi sales represntative for information on how to configure test modes for XBee/XBee-PRO S2C Zigbee modules.

2.10 Additional testing, Part 15 Subpart B disclaimer

All final host products must be tested to be compliant to FCC Part 15 Subpart B standards. While the XBee/XBee-PRO S2C Zigbee modules were tested to be complaint to FCC unintentional radiator standards, FCC Part 15 Subpart B compliance testing is still required for the final host product. This testing is required for all end products, and XBee/XBee-PRO S2C Zigbee module Part 15 Subpart B compliance does not affirm the end product's compliance.

See FCC notices for more details.

Europe (CE)

The XBee ZigBee RF Modules (non-PRO variants) have been tested for use in several European countries. For a complete list, refer to www.digi.com/resources/certifications.

If XBee ZigBee RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product with articles 3.1a and 3.1b of the Radio Equipment Directive. A Declaration of Conformity must be issued for each of these standards and kept on file as described in the Radio Equipment Directive.

Furthermore, the manufacturer must maintain a copy of the XBee ZigBee RF Modules user guide documentation and ensure the final product does not exceed the specified power ratings, antenna specifications, and/or installation requirements as specified in the user guide.

Maximum power and frequency specifications

For the S2C through-hole device:
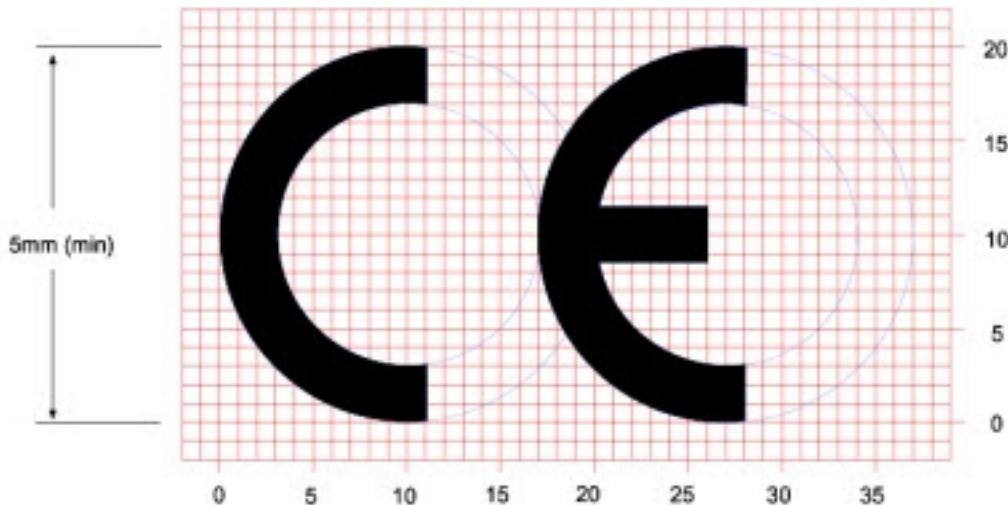
- Maximum power: 9.82 mW (9.92 dBm) Equivalent Isotropically Radiated Power (EIRP) at normal condition.
- Frequencies: 5 MHz channel spacing, beginning at 2405 MHz and ending at 2480 MHz.

For the S2C surface-mount device:

- Maximum power: 12.65 mW (11.02 dBm) EIRP.
- Frequencies: 5 MHz channel spacing, beginning at 2405 MHz and ending at 2480 MHz.

OEM labeling requirements

The "CE" marking must be affixed to a visible location on the OEM product. The following figure shows CE labeling requirements.

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

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

Important note

Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. Refer to the radio regulatory agency in the desired countries of operation for more information.

Declarations of conformity

Digi has issued Declarations of Conformity for the XBee RF Modules concerning emissions, EMC, and safety. For more information, see www.digi.com/resources/certifications.

Antennas

The following antennas have been tested and approved for use with the XBee/XBee-PRO Zigbee RF Module:

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

- Dipole (2.1 dBi, Omni-directional, Articulated RPSMA, Digi part number A24-HABSM)
- PCB Antenna (0.0 dBi)
- Monopole Whip (1.5 dBi)

ISED (Innovation, Science and Economic Development Canada)

Labeling requirements

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product enclosure must display the following text.

For XBee ZB surface-mount:

Contains Model XBee S2C Radio, IC: 1846A-XBS2C

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

For XBee-PRO ZB surface-mount:

Contains Model PS2CSM Radio, IC: 1846A-PS2CSM

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Note Legacy XBee-PRO SMT (Model: PRO S2C; hardware version 21xx) has IC: 1846A-XBPS2C.

For XBee ZB through-hole:

Contains Model S2CTH Radio, IC: 1846A-S2CTH

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

For XBee-PRO ZB through-hole:

Contains Model PS2CTH Radio, IC: 1846A-PS2CTH

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Transmitters for detachable antennas

This device has been designed to operate with the antennas listed in the previous table and having a maximum of 19 dB. Antennas not included in this list or having a gain greater than 19 dB are strictly prohibited for use with this device. The required antenna impedance is 50 ohms.

Detachable antenna

To reduce potential radio interference to other users, the antenna type and gain should be so chosen that the equivalent, isotropically radiated power (EIRP) is not more than permitted for successful communication.

For XBee S2D SMT:

Contains Model S2D SMT, IC: 1846A-S2DSM

RF Exposure

CAUTION! This equipment is approved for mobile and base station transmitting devices only. Antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

ATTENTION! Cet équipement est approuvé pour la mobile et la station base dispositifs d'émission seulement. Antenne(s) utilisé pour cet émetteur doit être installé pour fournir une distance de séparation d'au moins 20 cm à partir de toutes les personnes et ne doit pas être situé ou fonctionner en conjonction avec tout autre antenne ou émetteur.

Transmitters with Detachable Antennas

This radio transmitter (IC: 1846A-S2DSM) has been approved by Industry Canada to operate with the antenna types listed in FCC-approved antennas (2.4 GHz) with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Le présent émetteur radio (IC: 1846A-S2DSM) a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Detachable Antenna

Under Industry Canada regulations, this radio transmitter may operate using only an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peutfonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvépour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillageradioélectrique à l'intention des autres

utilisateurs, il faut choisir le type d'antenne etson gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépassepas l'intensité nécessaire àl'établissement d'une communication satisfaisante.

Australia (RCM)

Only XBee S2C and XBee-PRO S2C modules comply with requirements to be used in end products in Australia and New Zealand. All products with EMC and radio communications must have registered RCM and R-NZ marks. Registration to use the compliance mark will only be accepted from Australia or New Zealand manufacturers or importers, or their agents.

In order to have a RCM or R-NZ mark on an end product, a company must comply with a or b below.

    a. have a company presence in Australia or New Zealand.

    b. have a company/distributor/agent in Australia or New Zealand that will sponsor the importing of the end product.

Contact Digi for questions related to locating a contact in Australia and New Zealand.

ANATEL (Brazil)

The XBee ZB RF devices comply with Brazil ANATEL standards. The following information is required in the user manual for the product containing the radio and on the product containing the radio (in Portuguese):

Digi Model: XB24CZ7PIS-004, XB24CZ7PISB003, XB24CZ7RIS-004, XB24CZ7RISB003, XB24CZ7UIS-004 e XB24CZ7UISB003

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

Model:	S2CTH
Part Number	XB24CZ7PIT-004, XB24CZ7PITB003, XB24CZ7WIT-004, XB24CZ7WITB003, XB24CZ7SIT-004, XB24CZ7SITB003, XB24CZ7UIT-004, XB24CZ7UITB003

4556-15-1209

ANATEL (01)07899029306018

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

Model:	PS2CTH
Part Number	XBP24CZ7PIT-004, XBP24CZ7PITB003, XBP24CZ7WIT-004, XBP24CZ7WITB003, XBP24CZ7SIT-004, XBP24CZ7SITB003, XBP24CZ7UIT-004, XBP24CZ7UITB003

4077-15-1209

(01)07899029305967

ANATEL

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

The XBee-Pro ZB RF devices comply with Brazil ANATEL standards. The following information is required in the user manual for the product containing the radio and on the product containing the radio (in Portuguese):

Digi Model: XBP24CZ7PIS-004, XBP24CZ7RIS-004, XBP24CZ7UIS-004

Modelo: PS2CSM

ANATEL

Agência Nacional de Telecomunicações

07440-19-01209

"Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados"

South Korea
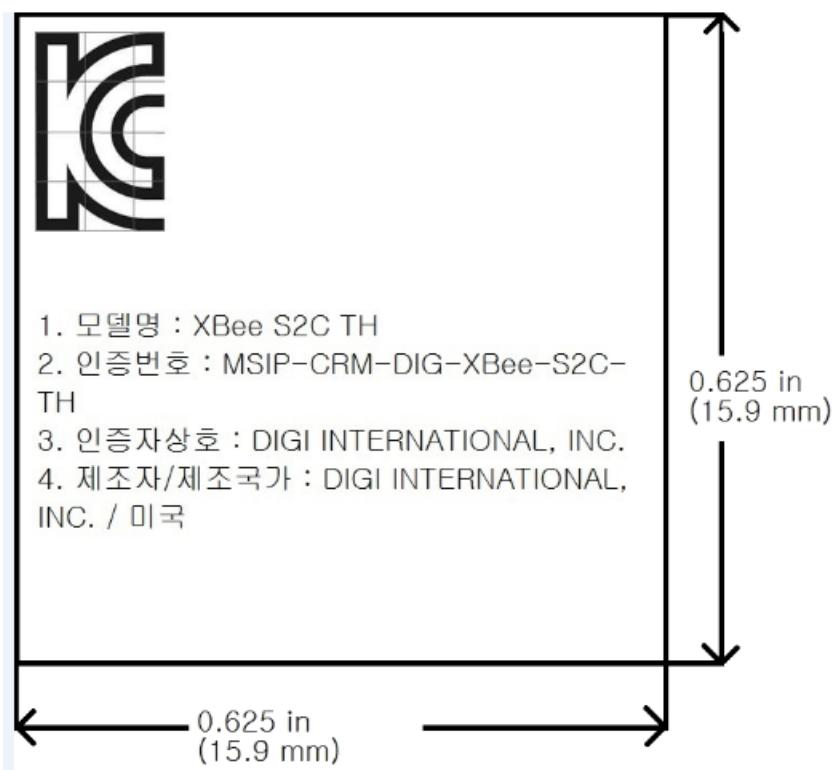
The low-power XBee S2C TH and XBee S2C devices have received South Korean approvals. To show conformity to the certificate, you must add a label with the South Korean product information to the XBee S2C Zigbee RF Module.

For the through-hole device, you can place the label on the reverse side.

Recommended label material: Abraham Technical (700342) MFG P/N TAAE-014250.

The label size is: 15.9 mm x 15.9 mm (0.625 in x 0.625 in)

The complete label information is as follows:
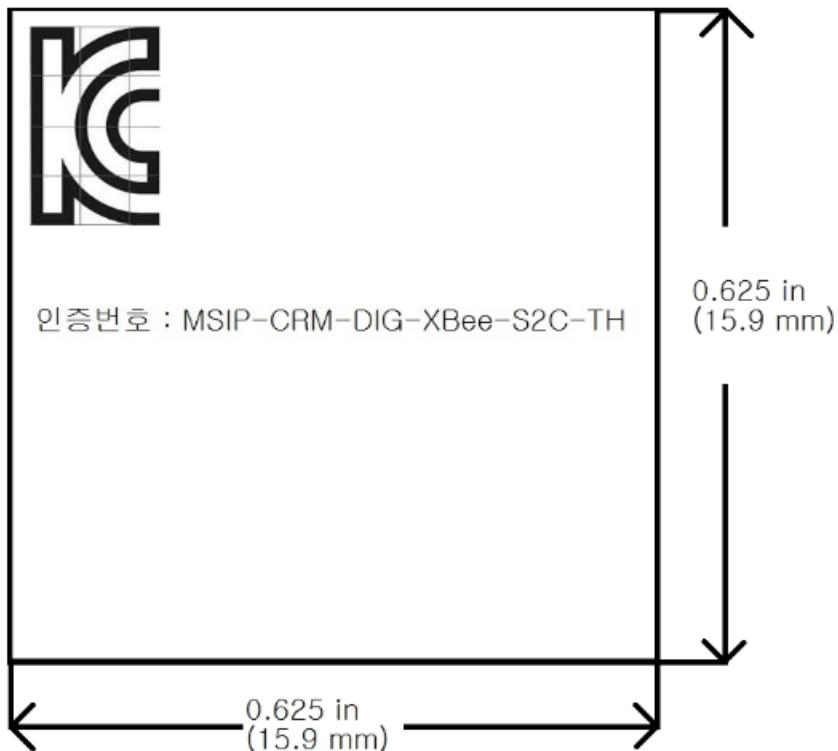
The KCC logo must be at least 5 mm tall.

The text shown in the label is:

1. 모델명 : XBee S2C TH
2. 인증번호 : MSIP-CRM-DIG-XBee-S2C-TH
3. 인증자상호 : DIGI INTERNATIONAL, INC.
4. 제조자/제조국가 : DIGI INTERNATIONAL, INC. / 미국

If the label size does not accommodate the required content, you can use abbreviated information, as follows:

The KCC logo must be at least 5 mm tall.

The text shown on the label is:

> 인증번호 : MSIP-CRM-DIG-XBee-S2C-TH

For the surface-mount version, the label will overlay the existing product label.
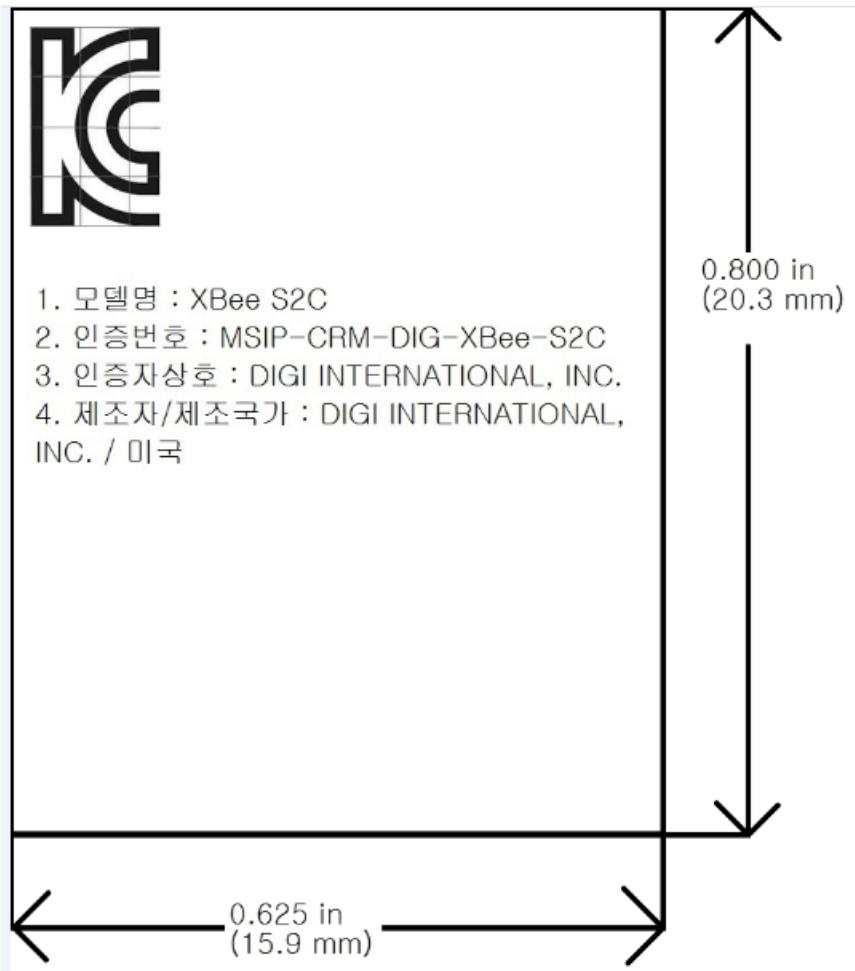

⚠️ **CAUTION!** By placing a label over the existing label, the certifications for Europe (CE), Australia, New Zealand (RCM), and Japan will no longer apply.

Recommended label material: Abraham Technical TELT-000465.

The label size is: 15.9 mm x 20.3 mm (0.625 in x 0.8 in)

The complete label information is as follows:
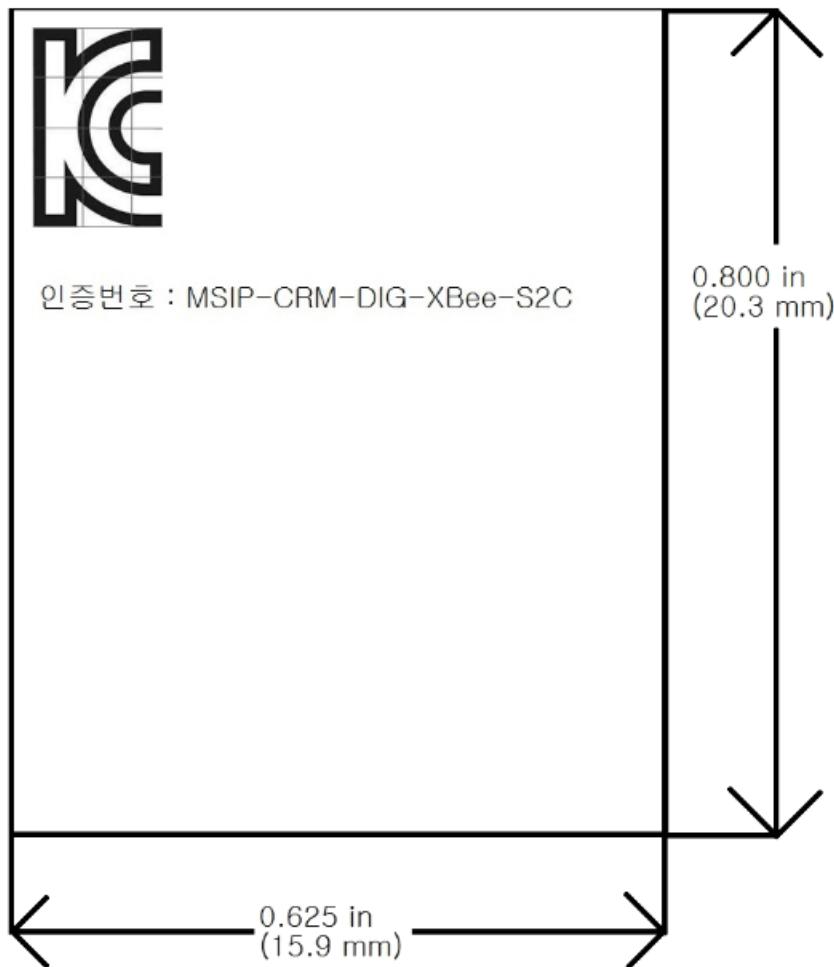
The KCC logo must be at least 5 mm tall.

The text shown in the label is:

1. 모델명 : XBee S2C
2. 인증번호 : MSIP-CRM-DIG-XBee-S2C
3. 인증자상호 : DIGI INTERNATIONAL, INC.
4. 제조자/제조국가 : DIGI INTERNATIONAL, INC. / 미국

If the label size does not accommodate the required content, you can use the abbreviated information, as follows:

인증번호 : MSIP–CRM–DIG–XBee–S2C

0.800 in
(20.3 mm)

0.625 in
(15.9 mm)

The KCC logo must be at least 5 mm tall.

The text shown in the label is:

인증번호 : MSIP-CRM-DIG-XBee-S2C

IFETEL (Mexico)

Manufacturer: Digi International

Country: USA

Brand: Digi

The Tariff Code (HS) for all XBee/XBee-PRO S2C radio modules is 8517-62-15.

Model: PS2CSM (XBee-Pro S2C surface-mount)

IFETEL (IFT) number RCPDIPS19-1817-A1 applies to these XBee-Pro S2C radios:

XBP24CZ7UIS-004	XBP24CZ7PIS-004	XBP24CZ7RIS-004	XBP24CZ7UISB003
XBP24CZ7PISB003	XBP24CZ7RISB003	XBP24CAUIS-001	XBP24CAPIS-001
XBP24CARIS-001	XBP24CDMUIS-001	XBP24CDMPIS-001	XBP24CDMRIS-001

Model: PS2CTH (XBee-Pro S2C through-hole)

IFETEL (IFT) number RCPDIPS19-1817 applies to these XBee-Pro S2C radios:

XBP24CZ7UITB003A	XBP24CZ7UITB003B	XBP24CZ7UITB003C	XBP24CZ7WITB003A
XBP24CZ7WIT-004	XBP24CZ7UIT-004	XBP24CZ7SIT-004	XBP24CZ7PIT-004
XBP24CZ7WITB003	XBP24CZ7UITB003	XBP24CZ7SITB003	XBP24CZ7PITB003
XBP24CAWIT-001	XBP24CAUIT-001	XBP24CASIT-001	XBP24CAPIT-001
XBP24CDMWIT-001	XBP24CDMUIT-001	XBP24CDMSIT-001	XBP24CDMPIT-001

Model: XBEE S2C (XBee S2C surface-mount)

IFETEL (IFT) number RCPDIXB19-1821 applies to these S2C radios:

XB24CZ7UIS-004	XB24CZ7PIS-004	XB24CZ7RIS-004	XB24CZ7UISB003
XB24CZ7PISB003	XB24CZ7RISB003	XB24CAUIS-001	XB24CAPIS-001
XB24CARIS-001	XB24CDMUIS-001	XB24CDMPIS-001	XB24CDMRIS-001

Model: S2CTH (XBee S2C through-hole)

IFETEL (IFT) number RCPDIS219-1821-A1 applies to these S2C radios:

XB24CZ7WIT-004,	XB24CZ7UIT-004,	XB24CZ7SIT-004,	XB24CZ7PIT-004,
XB24CZ7WITB003,	XB24CZ7UITB003,	XB24CZ7SITB003,	XB24CZ7PITB003,
XB24CAWIT-001,	XB24CAUIT-001,	XB24CASIT-001,	XB24CAPIT-001,
XB24CDMWIT-001,	XB24CDMUIT-001,	XB24CDMSIT-001,	XB24CDMPIT-001

OEM labeling requirements

WARNING! The Original Equipment Manufacturer (OEM) must ensure that Mexico IFT labeling requirements are met.

The IFETEL number for the XBee3 product must be listed either on the end product, on the packaging, in the manual, or in the software with the following phrase, substituting the IFT number for the XBee S2C product you are using for the XXXXXXXXX-XXXX:

"Este equipo contiene el módulo XBee3 con Número IFETEL: XXXXXXXXX-XXXX"

or

"Este equipo contiene el módulo XBee3 con IFT #: XXXXXXXXX-XXXX"

The following paragraph must also be present in the User Manual for the end product:

"La operación de este equipo está sujeta a las siguientes dos condiciones: (1) es posible que este equipo o dispositivo no cause interferencia perjudicial y (2) este equipo o dispositivo debe aceptar cualquier interferencia, incluyendo la que pueda causar su operación no deseada."

Migrating from XBee through-hole to XBee surface-mount devices

The XBee surface-mount and through-hole devices are designed to be compatible and offer the same basic feature set. As mentioned previously, the surface-mount form factor has more I/O pins.

This section provides information to help users migrate from the surface-mount to the Through-hole form factor.

Pin mapping

Mapping of the surface-mount (SMT) pads to the through-hole (TH) pins is shown in the table below. The pin names are from the S2C SMT device.
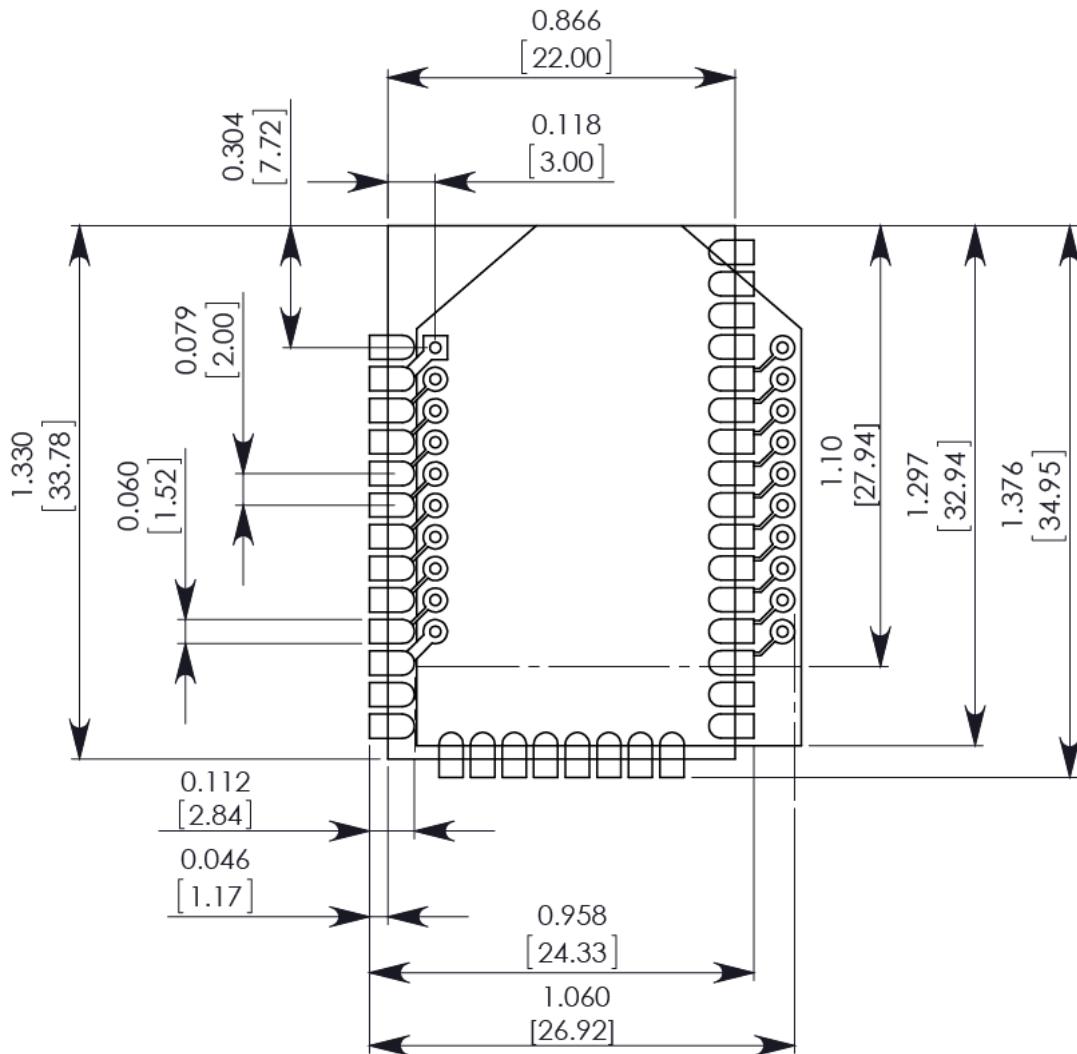
SMT Pin #	Name	TH Pin #
1	GND	
2	V_{CC}	1
3	DOUT / DIO13	2
4	DIN / $\overline{CONFIG}$ / DIO14	3
5	DIO12	4
6	$\overline{RESET}$	5
7	RSSI PWM / DIO10	6
8	PWM1 / DIO11	7
9	[reserved]	8
10	$\overline{DTR}$ / SLEEP_RQ / DIO8	9
11	GND	10
12	SPI_$\overline{ATTN}$ / $\overline{BOOTMODE}$ / DIO19	
13	GND	
14	SPI_CLK / DIO18	
15	SPI_$\overline{SSEL}$ / DIO17	
16	SPI_MOSI / DIO16	
17	SPI_MISO / DIO15	
18	[reserved]	
19	[reserved]	
20	[reserved]	
21	[reserved]	
22	GND	
23	[reserved]	
24	DIO4	11
25	$\overline{CTS}$ / DIO7	12
26	ON / $\overline{SLEEP}$ / DIO9	13
27	V_{REF}	14

SMT Pin #	Name	TH Pin #
28	ASSOCIATE / DIO5	15
29	$\overline{RTS}$ / DIO6	16
30	AD3 / DIO3	17
31	AD2 / DIO2	18
32	AD1 / DIO1	19
33	AD0 / DIO0	20
34	[reserved]	
35	GND	
36	RF	
37	[reserved]	

Mounting

One important difference between the surface-mount and the through-hole devices is how they mount to the PCB. Different mounting techniques are required.

We designed a footprint that allows either device to be attached to a PCB as shown in the following diagram. The dimensions without brackets are in inches, and those in brackets are in millimeters.

The round holes in the diagram are for the through-hole design, and the semi-oval pads are for the surface-mount design. Pin 1 of the through-hole design lines up with pad 1 of the surface-mount design, but the pins are actually offset by one pad (see Pin mapping). By using diagonal traces to connect the appropriate pins, the layout works for both modules.

For information on attaching the SMT device, see Manufacturing information.

Manufacturing information

The XBee/XBee-PRO Zigbee RF Module is designed for surface-mounting on the OEM PCB. It has castellated pads to allow for easy solder attaching and inspection. The pads are all located on the edge of the device so there are no hidden solder joints on these devices.

Recommended solder reflow cycle

The following table lists the recommended solder reflow cycle. The chart shows the temperature setting and the time to reach the temperature.

Time (seconds)	Temperature (°C)
30	65
60	100
90	135
120	160
150	195
180	240
210	260

The maximum temperature should not exceed 260 ℃.

The device reflows during this cycle, and must not be reflowed upside down. Be careful not to jar the device while the solder is molten, as parts inside the device can be removed from their required locations.

Hand soldering is possible and should be done in accordance with approved standards.
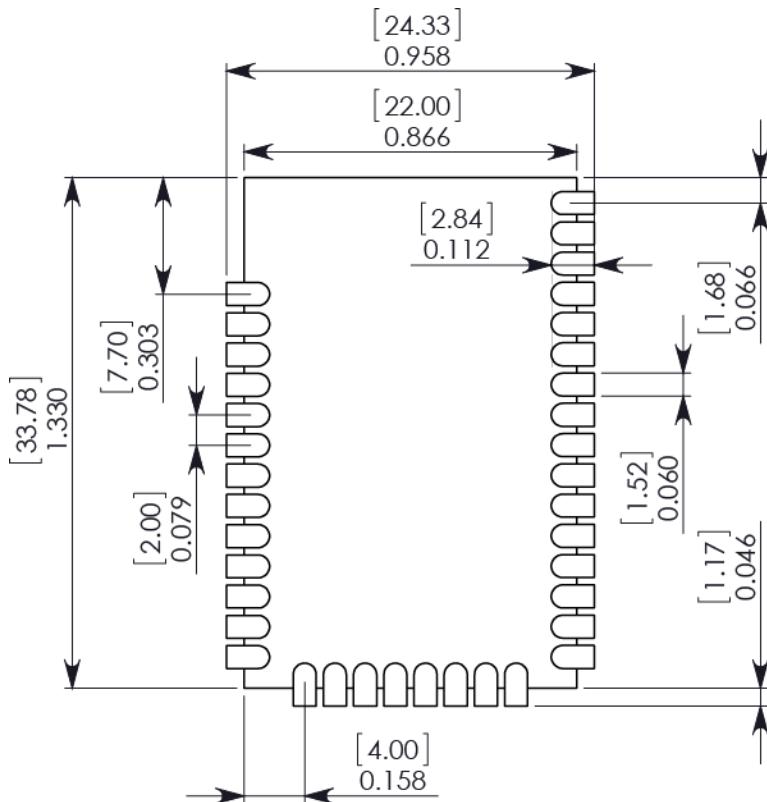
The XBee/XBee-PRO Zigbee RF Modules are level 3 Moisture Sensitive Devices. When using this kind of device, consider the relative requirements in accordance with standard IPC/JEDEC J-STD-020.

In addition, note the following conditions:

a. Calculated shelf life in sealed bag: 12 months at <40 ℃ and <90% relative humidity (RH).

b. Environmental condition during the production: 30 ℃ /60% RH according to IPC/JEDEC J-STD -033C, paragraphs 5 through 7.

c. The time between the opening of the sealed bag and the start of the reflow process cannot exceed 168 hours if condition b) is met.

d. Baking is required if conditions b) or c) are not met.

e. Baking is required if the humidity indicator inside the bag indicates a RH of 10% more.

f. If baking is required, bake modules in trays stacked no more than 10 high for 4-6 hours at 125 ℃.

Recommended footprint and keepout
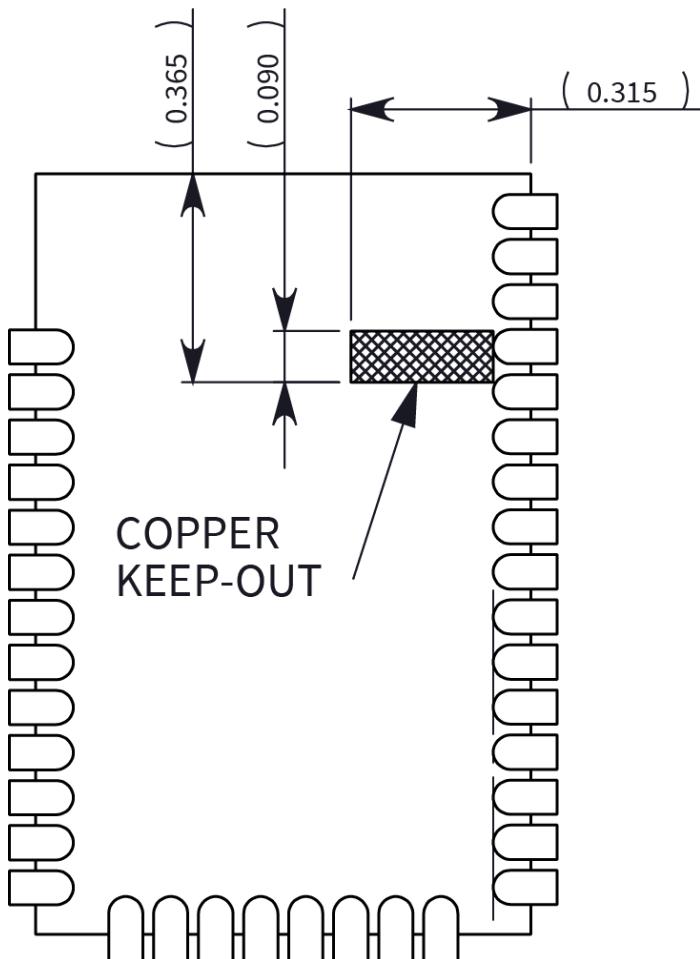
We recommend that you use the following PCB footprints for surface-mounting. The dimensions without brackets are in inches, and those in brackets are in millimeters.

Match the solder footprint to the copper pads, but you may need to adjust it depending on the specific needs of assembly and product standards. We recommend a stencil thickness of 0.15 mm (0.005 in). Place the component last and set the placement speed to the slowest setting.

While the underside of the module is mostly coated with solder resist, we recommend that the copper layer directly below the module be left open to avoid unintended contacts. Copper or vias must not interfere with the three exposed RF test points on the bottom of the module (see below). Furthermore, these modules have a ground plane in the middle on the back side for shielding purposes, which can be affected by copper traces directly below the module.

Flux and cleaning

Digi recommends that a "no clean" solder paste be used in assembling these devices. This eliminates the clean step and ensures unwanted residual flux is not left under the device where it is difficult to remove.

In addition the following issues can occur:

- Cleaning with liquids can result in liquid remaining under the shield or in the gap between the device and the OEM PCB. This can lead to unintended connections between pads on the device.

- The residual moisture and flux residue under the device are not easily seen during an inspection process.

Factory recommended best practice is to use a "no clean" solder paste to avoid these issues and ensure proper device operation.

Reworking

Never perform rework on the device itself. The device has been optimized to give the best possible performance, and reworking the device itself will void warranty coverage and certifications. We recognize that some customers choose to rework and void the warranty. The following information

serves as a guideline in such cases to increase the chances of success during rework, though the warranty is still voided.

The device may be removed from the OEM PCB by the use of a hot air rework station, or hot plate. Be careful not to overheat the device. During rework, the device temperature may rise above its internal solder melting point and care should be taken not to dislodge internal components from their intended positions.

Load Zigbee firmware on 802.15.4 devices

Background

Our XBee/XBee-PRO 802.15.4 RF modules are built on the same hardware as the XBee/XBee-PRO S2C ZB RF Modules. It is possible to load Zigbee firmware on existing 802.15.4 modules. The "XBee/XBee-PRO S2C 802.15.4 part numbers" table shows which part numbers are compatible with Zigbee firmware.

Zigbee modules are approved for operation in some regions that 802.15.4 modules are not (see the table below). If your region is not listed as certified for the 802.15.4 module you purchased, it may be illegal for you to operate in your region, even if you load Zigbee firmware onto the module, because it is not labeled with the appropriate certification markings. Please check with local laws or your regional Digi International representative for more information.

Module name	Approved regions
XBee S2C Zigbee	USA, Canada, Europe, Australia/New Zealand, Brazil, and Japan
XBee-PRO S2C Zigbee	USA, Canada, Australia/New Zealand, and Brazil (through-hole devices only)
XBee S2C 802.15.4	USA, Canada, Europe
XBee-PRO S2C 802.15.4	USA, Canada

CAUTION! The antenna cable loss requirements for the 802.15.4 firmware are different than the Zigbee firmware for gain antennas exceeding 2.1 dBi. If you migrate an 802.15.4 device to Zigbee firmware, and are using gain antennas, you must adhere to the cable loss requirements found in Regulatory information.

XBee/XBee-PRO S2C 802.15.4 part numbers	Form factor	Hardware version (HV)
XB24CAPIS-004 XB24CARIS-004 XB24CAUIS-004	XBee SMT	0x22
XB24CAPIT-004 XB24CASIT-004 XB24CAUIT-004 XB24CAWIT-004	XBee TH	0x2E
XBP24CAPIS-004 XBP24CARIS-004 XBP24CAUIS-004	XBee SMT	0x30
XBP24CAPIT-004 XBP24CASIT-004 XBP24CAUIT-004 XBP24CAWIT-004	XBee TH	0x2D

In addition to the differences between the 802.15.4 and Zigbee protocols, some of the operational features are different between the two firmware versions. For example, the XBee-PRO 802.15.4 supports fewer channels than the Zigbee firmware. It is important that you read and understand this user guide before developing with the Zigbee firmware.

Load Zigbee firmware

To load Zigbee firmware on an existing 802.15.4 module, use the following instructions. You must use the serial interface to perform this update. The device does not support OTA updates for changing 802.15.4 to ZB firmware or vice versa.

1. Verify that your device's part number (listed on the label) is included in the list shown in the table above.

2. Install the device in a Digi development board and connect it to your PC.

3. The next steps involve loading firmware using XCTU. To download XCTU and read detailed instructions about it, go to:

   https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

4. When you get to the **Update firmware** dialog box, in the **Function set** area, click the **Zigbee** option, and the newest firmware version.

5. Click **Update** and follow the instructions.

6. When the updating process successfully completes, your device runs Zigbee firmware. You can change back to 802.15.4 firmware at any time by following the same process and selecting the 802.15.4 firmware option instead.