



# XBee DigiMesh 2.4

Radio Frequency (RF) Module

---

User Guide

## Revision history

90000991 S

Revision	Date	Description
R	November 2015	Revised the document as part of moving to online help.
S	January 2016	Updated several AT commands.

## 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.

© 2015 Digi International Inc. All rights reserved.

## 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 the product's warranty information, go to the following website:

<http://www.digi.com/howtobuy/terms>

## Customer support

Telephone (8:00 am — 5:00 pm U.S. Central Time):

Toll-free US and Canada: 866.912.3444

Worldwide: +1 952.912.3456

Online: [www.digi.com/support/eservice](http://www.digi.com/support/eservice)

Mail:

Digi International Inc.

11001 Bren Road East

Minnetonka, MN 55434  
USA

## Contents

---

Revision history .....	2
 <b>About the XBee DigiMesh 2.4 RF Module</b>	
XBee DigiMesh 2.4 overview .....	12
Antenna options .....	12
 <b>XBee DigiMesh 2.4 manufacturing standards</b>	
XBee DigiMesh 2.4 part numbers .....	14
 <b>XBee DigiMesh 2.4 technical specifications</b>	
XBee DigiMesh 2.4 performance specifications .....	15
XBee DigiMesh 2.4 power requirements .....	15
XBee DigiMesh 2.4 general specifications .....	17
XBee DigiMesh 2.4 network and security specifications .....	17
XBee DigiMesh 2.4 agency approvals .....	18
 <b>XBee DigiMesh 2.4 hardware</b>	
Mechanical drawings .....	19
Mounting considerations .....	20
Hardware diagram .....	21
XBee and XBee-PRO DigiMesh 2.4 pin signals .....	21
Recommended pin connections .....	23
Design notes .....	24
Board layout .....	24
Power supply design .....	24
Antenna performance .....	24

XBee DigiMesh 2.4 DC characteristics .....	27
XBee DigiMesh 2.4 ADC operating characteristics .....	27
XBee DigiMesh 2.4 ADC timing and performance characteristics .....	28

## XBee DigiMesh 2.4 modes

Transparent and API operating modes .....	30
Transparent operating mode .....	30
API operating mode .....	30
Comparing Transparent and API modes .....	31
XBee DigiMesh 2.4 additional modes .....	34
Command mode .....	34
Idle mode .....	34
Receive mode .....	34
Sleep modes .....	34
Transmit mode .....	35
Enter Command mode .....	35
Troubleshooting .....	35
Send AT commands .....	36
Respond to AT commands .....	36
Exit Command mode .....	36

## Configure the XBee/XBee-PRO DigiMesh 2.4

Software libraries .....	37
Configure the device using XCTU .....	37

## XBee DigiMesh 2.4 serial communication

Serial interface .....	38
UART data flow .....	38
Serial data .....	39
XBee DigiMesh 2.4 serial buffers .....	39
XBee DigiMesh 2.4 serial buffer issues .....	39
Serial flow control .....	40
CTS flow control .....	40
RTS flow control .....	41

## Work with networked devices

Network commissioning and diagnostics .....	42
---	----

Configure devices .....	42
Configure a device remotely .....	42
Send a remote command .....	42
Apply changes on remote devices .....	43
Remote command response .....	43
Establish and maintain network links .....	43
Build aggregate routes .....	43
Examples .....	44
Replace nodes .....	45
Test links in a network .....	45
Test links between adjacent devices .....	46
Example .....	47
RSSI indicators .....	48
Discover devices .....	49
Trace route option .....	49
NACK messages .....	51
The Commissioning Pushbutton .....	51
Monitor I/O lines .....	54
Queried sampling .....	54
Periodic I/O sampling .....	57
Detect digital I/O changes .....	57

## DigiMesh networking

DigiMesh networking .....	59
Routers and end devices .....	60
Network identifiers .....	60
Operating channels .....	60
Unicast addressing .....	60
Broadcast addressing .....	61
Routing .....	61
Route discovery .....	62
Throughput .....	62
Transmission timeouts .....	63
Unicast one hop time .....	63
Transmit a broadcast .....	64
Transmit a unicast with a known route .....	64
Transmit a unicast with an unknown route .....	64
Transmit a unicast with a broken route .....	65

## DigiMesh sleep modes

XBee DigiMesh 2.4 sleep modes .....	66
Asynchronous modes .....	66
Synchronous modes .....	66
Normal mode .....	67
Asynchronous Pin Sleep mode .....	67
Asynchronous Cyclic Sleep mode .....	67
Asynchronous Cyclic Sleep with Pin Wake Up mode .....	68
Synchronous Sleep Support mode .....	68
Synchronous Cyclic Sleep mode .....	68
The sleep timer .....	69
Coordinating sleep modes in the DigiMesh network .....	69
DigiMesh synchronization messages .....	70
Become a sleep coordinator .....	71
Preferred sleep coordinator option .....	72
Resolution criteria and selection option .....	72
Commissioning Pushbutton option .....	73
Auto-early wake-up sleep option .....	74
Select sleep parameters .....	74
Start a sleeping DigiMesh network .....	75
Adding a new node to an existing network .....	75
Change sleep parameters .....	76
Rejoin nodes that lose sync .....	77
XBee DigiMesh 2.4 diagnostics .....	78
Query sleep cycle .....	78
Sleep status .....	78
Missed sync messages command .....	78
Sleep status API messages .....	78

## XBee DigiMesh 2.4 AT commands

1S (XBee Sensor Sample) .....	79
AC (Apply Changes) .....	79
AG (Aggregator Support) .....	79
AO (API Options) .....	80
AP (API Enable) .....	80
BC (Bytes Transmitted) .....	81
BD (Baud Rate) .....	81
BH (Broadcast Hops) .....	82

CA (CCA Threshold) .....	83
CB (Commissioning Pushbutton) .....	83
CC (Command Sequence Character) .....	84
CE (Routing / Messaging Mode) .....	84
CH (Operating Channel) .....	84
CI (Cluster ID) .....	85
CK (Configuration CRC) .....	85
CN (Exit Command mode) .....	85
CT (Command mode Timeout) .....	86
D0 (DIO0/AD0) .....	86
D1 (DIO1/AD1) .....	86
D2 (DIO2/AD2) .....	87
D3 (DIO3/AD3) .....	88
D4 (DIO4/AD4) .....	88
D5 (DIO5/AD5/ASSOCIATED_INDICATOR) .....	89
D6 (DIO6/RTS) .....	90
D7 (DIO7/CTS) .....	90
D8 (DIO8/SLEEP_REQUEST) .....	91
D9 (DIO9/ON_SLEEP) .....	91
P0 (DIO10/RSSI/PWM0 Configuration) .....	92
P1 (DIO11/PWM1 Configuration) .....	93
P2 (DIO12 Configuration) .....	93
DB (Last Packet RSSI) .....	94
DD (Device Type Identifier) .....	94
DE (Destination Endpoint) .....	95
DH (Destination Address High) .....	95
DL (Destination Address Low) .....	95
DM (DigiMesh Options) .....	95
DN (Discover Node) .....	96
EA (MAC ACK Failure Count) .....	96
ED (Energy Detect) .....	97
EE (Encryption Enable) .....	97
Find Neighbors (FN) .....	98
FR (Software Reset) .....	98
FT (Flow Control Threshold) .....	98
GD (Good Packets Received) .....	99
GT (Guard Times) .....	99
HV (Hardware Version) .....	99
IC (DIO Change Detect) .....	99
ID (Network ID) .....	101



IF (Sleep Sample Rate) .....	101
IR (Sample Rate) .....	101
IS (Force Sample) .....	102
KY (AES Encryption Key) .....	102
LT (Associate LED Blink Time) .....	102
M0 (PWM0 Duty Cycle) .....	103
M1 (PWM1 Duty Cycle) .....	103
MR (Mesh Unicast Retries) .....	103
MS (Missed Sync Messages) .....	103
MT (Broadcast Multi-Transmits) .....	104
N? (Network Discovery Timeout) .....	104
NB (Parity) .....	104
ND (Network Discover) .....	105
NH (Network Hops) .....	105
NI (Node Identifier) .....	105
NN (Network Delay Slots) .....	106
NO (Network Discovery Options) .....	106
NP (Maximum Packet Payload Bytes) .....	107
NT (Network Discovery Back-off) .....	107
OS (Operating Sleep Time) .....	107
OW (Operating Wake Time) .....	108
PL (TX Power Level) .....	108
PR (Pull-up Resistor Enable) .....	109
RE (Restore Defaults) .....	110
RO (Packetization Timeout) .....	110
RP (RSSI PWM Timer) .....	111
RR (Unicast Retries) .....	111
SE (Source Endpoint) .....	111
SH (Serial Number High) .....	112
SL (Serial Number Low) .....	112
SM (Sleep Mode) .....	112
SN (Number of Cycles Between ON_SLEEP) .....	113
SO (Sleep Options) .....	114
SP (Sleep Time) .....	115
SQ (Missed Sleep Sync Count) .....	115
SS (Sleep Status) .....	115
ST (Wake Time) .....	116
TR (Transmission Failure Count) .....	116
UA (Unicasts Attempted Count) .....	117
VL (Version Long) .....	117

VR (Firmware Version) .....	117
WR (Write) .....	118
WH (Wake Host Delay) .....	118
%H (MAC Unicast One Hop Time) .....	118
%8 (MAC Broadcast One Hop Time) .....	119

## Operate in API mode

API mode overview .....	120
API frame specifications .....	120
Calculate and verify checksums .....	122
Escaped characters in API frames .....	123
API frames .....	123
API frame exchanges .....	124
Code to support future API frames .....	125
AT Command frame 0x08 .....	126
AT Command - Queue Parameter Value frame 0x09 .....	127
Transmit Request frame 0x10 .....	129
Explicit Addressing Command frame 0x11 .....	132
Remote AT Command Request frame 0x17 .....	136
AT Command Response frame 0x88 .....	139
Modem Status frame 0x8A .....	140
Transmit Status frame 0x8B .....	141
Route Information Packet 0x8D .....	144
Aggregate Addressing Update frame 0x8E .....	148
Receive Packet frame 0x90 .....	151
Explicit Rx Indicator frame 0x91 .....	153
Data Sample Rx Indicator frame 0x92 .....	156
Node Identification Indicator frame 0x95 .....	159
Remote Command Response frame 0x97 .....	164

## XBee DigiMesh 2.4 certifications

XBee DigiMesh 2.4 agency certifications - United States .....	167
United States (FCC) .....	167
OEM labeling requirements .....	167
FCC notices .....	168
RF exposure statement .....	168
XBee DigiMesh 2.4 FCC-approved antennas (2.4 GHz) .....	169
XBee DigiMesh 2.4 agency certifications - Australia (C-Tick) .....	178

Labeling requirements .....	178
XBee DigiMesh 2.4 agency certifications - Brazil Agência Nacional de Telecomunicações (Anatel) .....	178
XBee DigiMesh 2.4 agency certifications - Industry Canada (IC) .....	179
Labeling requirements .....	179
XBee DigiMesh 2.4 agency certifications - European Telecommunications Standards Institute (ETSI) .....	180
OEM labeling requirements .....	180
Restrictions .....	181
Declarations of conformity .....	181
Approved antennas .....	181
XBee DigiMesh 2.4 agency certifications - Japan (Telec) .....	182
Labeling requirements .....	182

## About the XBee DigiMesh 2.4 RF Module

---

### XBee DigiMesh 2.4 overview

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

You can build networks of up to 32 nodes using these devices. For larger networks of up to 1,000 or more nodes, we offer RF optimization services to assist with proper network configuration.

Contact Digi Technical Support for more details.

### Antenna options

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

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

The U.FL connection allows for connectivity to an external antenna. U.FL is a small antenna connection for use with a pigtail connector. A pigtail is a short (typically 4 - 6 in) cable that either terminates into an external antenna port such as an RPSMA, N or TNC connection or an antenna. You would attach the RPSMA connector to a bulkhead. These options allow you to mount the device away from the edge of the enclosure in your product and centrally locate the radio. U.FL is fragile and is not designed for multiple

insertions without a specialized tool to separate the pigtail without damaging the connector; for more information, see [http://www.digikey.com/product-detail/en/U.FL-LP\(V\)-N-2/HR5017-ND/513034](http://www.digikey.com/product-detail/en/U.FL-LP(V)-N-2/HR5017-ND/513034).

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

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

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

## XBee DigiMesh 2.4 manufacturing standards

---

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

The Federal Communications Commission (FCC) approves the devices for use in the United States. For details, see [XBee DigiMesh 2.4 agency certifications - United States on page 167](#).

If a system contains XBee-PRO DigiMesh 2.4 RF Modules, the system inherits Digi's certifications.

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

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

We optimize the devices for use in the United States and Canada. For a complete list of agency approvals, see [XBee DigiMesh 2.4 certifications on page 167](#).

## XBee DigiMesh 2.4 part numbers

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

[www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4#partnumbers](http://www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4#partnumbers)

## XBee DigiMesh 2.4 technical specifications

The following tables provide the device's technical specifications.

### XBee DigiMesh 2.4 performance specifications

This table describes the performance specifications for the XBee/XBee-PRO 2.4 DigiMesh devices.

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

### XBee DigiMesh 2.4 power requirements

The following table describes the power requirements for the XBee/XBee-PRO 2.4 DigiMesh devices.

Specification	XBee	XBee-PRO
Supply voltage	2.8 - 3.4 VDC	2.8 - 3.4 VDC

Specification	XBee	XBee-PRO
Transmit current	45 mA (@ 3.3 V)	250 mA (@ 3.3 V) (150 mA for the international variant) RPSMA device only: 340 mA (@ 3.3 V) (180 mA for the international variant)
Idle / receive current	50 mA (@ 3.3 V)	55 mA (@ 3.3 V)
Power down current (pin sleep)	<10 $\mu$ A	<10 $\mu$ A
Power down current (cyclic sleep)	<50 $\mu$ A	<50 $\mu$ A



## XBee DigiMesh 2.4 general specifications

The following table describes the general specifications for the devices.

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

## XBee DigiMesh 2.4 network and security specifications

The following describes the network and security specifications for the XBee/XBee-PRO 2.4 DigiMesh devices.

Specification	XBee	XBee-PRO
Supported network topologies	Mesh, point-to-point, point-to-multipoint, peer-to-peer	Mesh, point-to-point, point-to-multipoint, peer-to-peer
Number of channels (software selectable)	16 direct sequence channels	12 direct sequence channels

Specification	XBee	XBee-PRO
Addressing options	PAN ID, channel and 64-bit addresses	PAN ID, channel and 64-bit addresses
Encryption	128 bit Advanced Encryption Standard (AES)	128 bit AES

## XBee DigiMesh 2.4 agency approvals

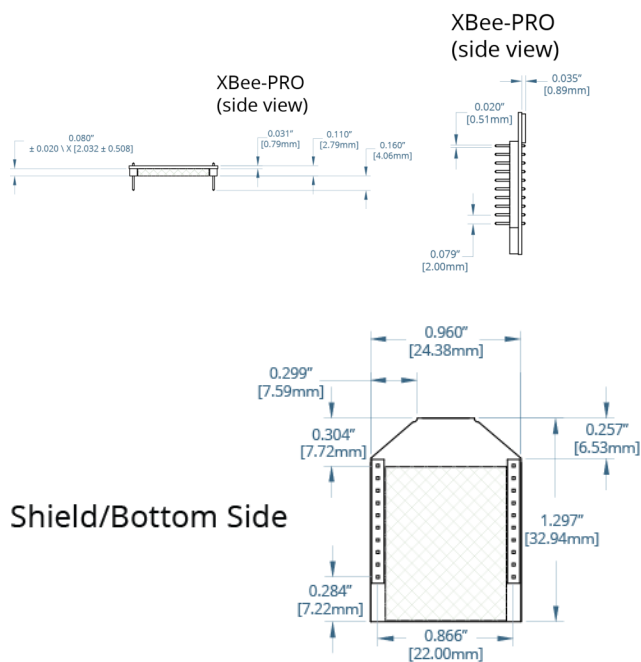
This table describes the agency approvals for the XBee 2.4 DigiMesh devices.

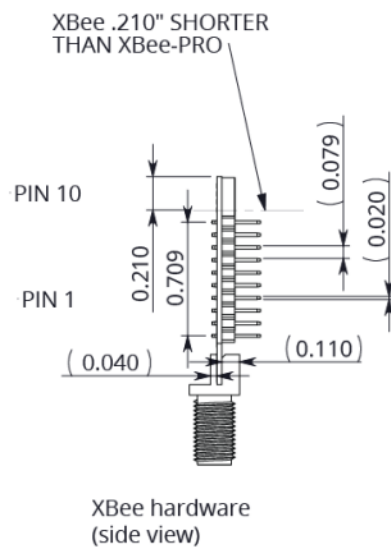
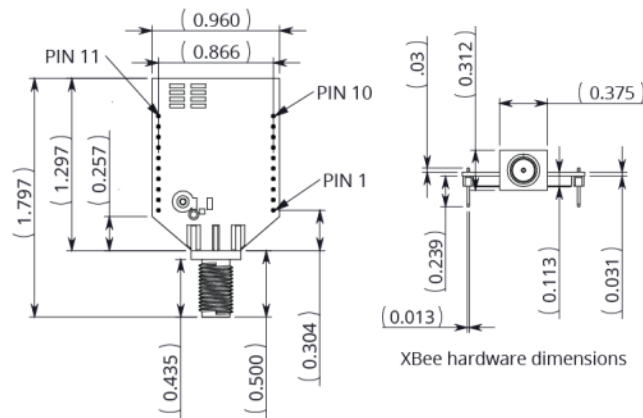
Specification	XBee	XBee-PRO
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A-XBEE	4214A-XBEEPRO
Europe (CE)	ETSI	ETSI (maximum 10 dBm transmit power output)
RoHS	Lead-free and RoHS compliant	Lead-free and RoHS compliant
Japan	R201WW07215214	R201WW08215111 (maximum 10 dBm transmit power output)
Australia	C -Tick	C -Tick
Brazil	ANATEL 0369-15-1209	ANATEL 0378-15-1209
See <a href="#">XBee DigiMesh 2.4 certifications on page 167</a> for region-specific certification requirements.		

## XBee DigiMesh 2.4 hardware

### Mechanical drawings

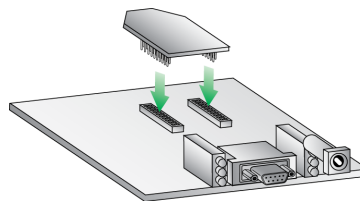
The following figures show the mechanical drawings for the devices. The second drawing shows the RPSMA device. The drawings do not show antenna options. All dimensions are in inches.





## Mounting considerations

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



Century Interconnect and Samtec manufacture the 2 x 10 pin 2 mm spacing receptacles on Digi development boards. Several other manufacturers provide comparable mounting solutions; we currently use the following receptacles:

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

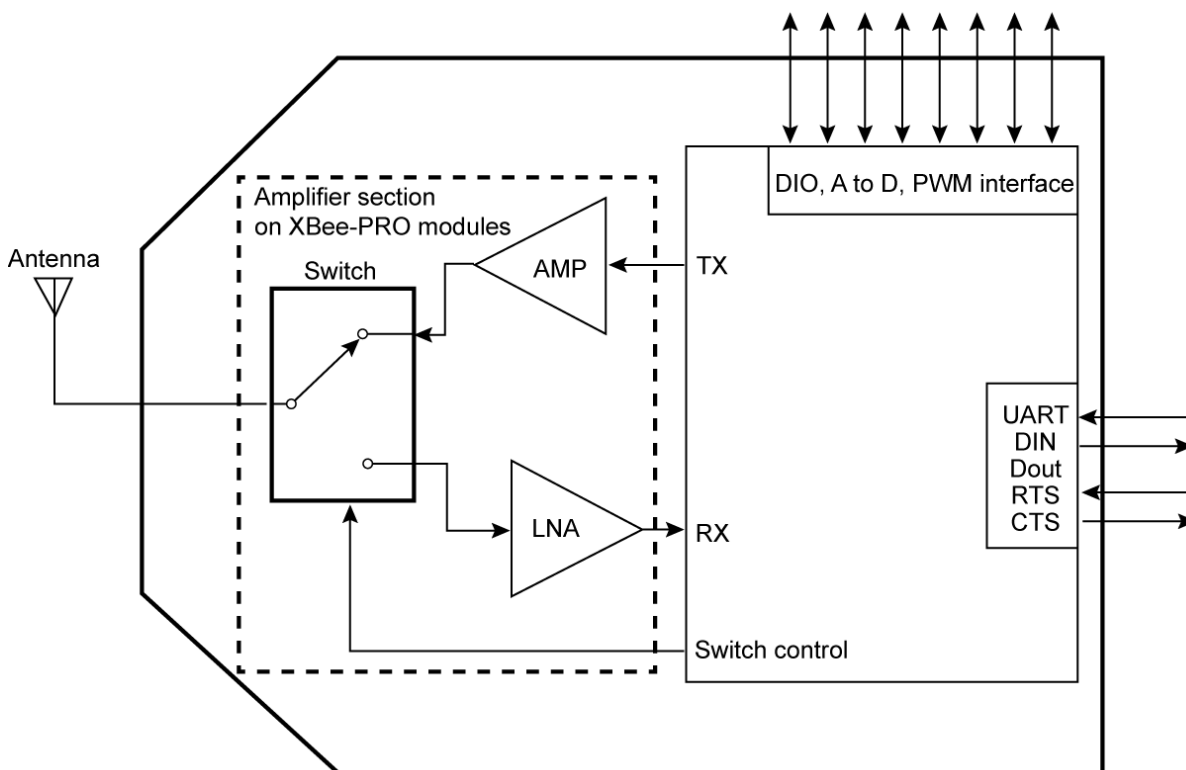
---

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

---

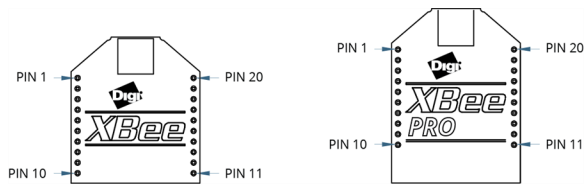
## Hardware diagram

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



## XBee and XBee-PRO DigiMesh 2.4 pin signals

The following table shows the pin signals and their descriptions.



Pin #	Pin Name	Direction	Description
1	Vcc	-	Power supply
2	DOUT	Output	UART data out
3	DIN/CONFIG	Input	UART data in
4	DIO12	Either	Digital I/O 12
5	RESET	Input/open drain output	Device reset. The reset pulse must be at least 100 us. Drive this line as an open drain/collector. The device drives this line low when a reset occurs. Never drive this line high.
6	PWM0/RSSI/DIO10	Either	PWM output 0 / RX signal strength indicator / Digital I/O
7	PWM/DIO11	Either	PWM output 1 / Digital I/O 11
8	Reserved	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ/ DIO8	Either	Pin sleep control line or Digital I/O 8
10	GND	-	Ground
11	AD4/ DIO4	Either	Analog input 4 or Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear-to-send flow control or Digital I/O 7
13	ON/SLEEP	Output	Device Status Indicator or Digital I/O 9

Pin #	Pin Name	Direction	Description
14	VREF	-	You must connect this line if you want to use analog I/O sampling. Must be between 2.6 V and Vcc.
15	Associate / DIO5/AD5	Either	Associated indicator, Digital I/O 5
16	RTS/ DIO6	Either	Request-to-send flow control, Digital I/O 6
17	AD3 / DIO3	Either	Analog input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Either	Analog input 0, Digital I/O 0, or Commissioning Button

**Notes:**

The table specifies signal direction with respect to the device.

The device includes a 50 k $\Omega$  pull-up resistor attached to RESET.

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

Leave any unused pins disconnected.

## Recommended pin connections

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

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

You can connect other pins to external circuitry for convenience of operation including the Associate LED pin (pin 15) and the Commissioning Button 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 analog sampling, attach the VREF pin (pin 14) to a voltage reference.

## Design notes

The XBee devices 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.

## Board layout

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

## 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  $\mu$ F and 8.2 pF capacitor as near as possible to pin 1 on the PCB. If you are using a switching regulator for the power supply, switch the frequencies above 500 kHz. Limit the power supply ripple to a maximum 100 mV peak to peak.

## Antenna performance

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

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

- metal poles
- metal studs
- structure beams
- concrete, which is usually reinforced with metal rods

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

- vehicles
- elevators
- ventilation ducts

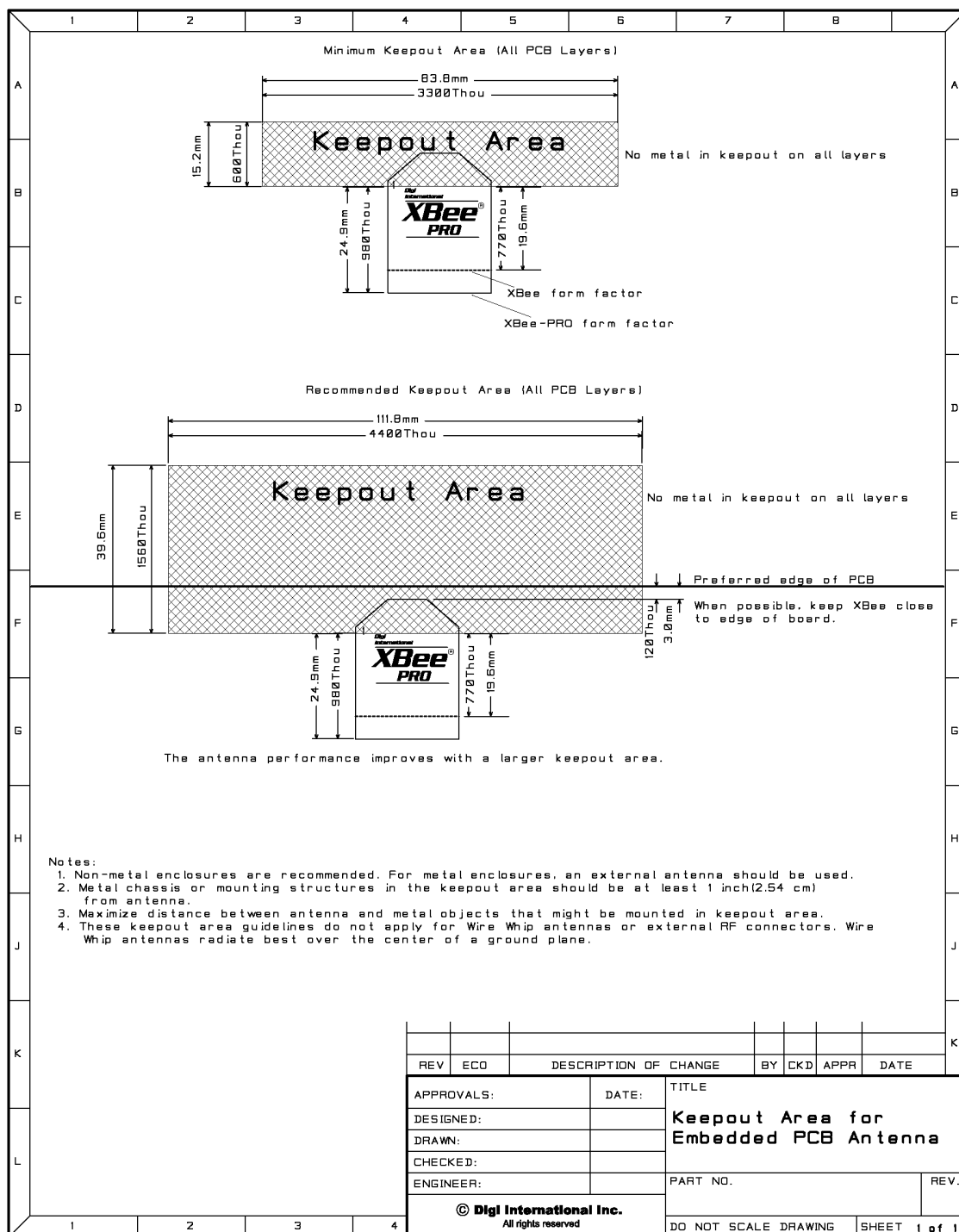


- refrigerators
- microwave ovens
- batteries
- tall electrolytic capacitors

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

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

For the best results, mount the device at the edge of the host PCB. Ensure that the ground, power, and signal planes are vacant immediately below the antenna section. We recommend that you allow a “keepout” area, which the following drawing shows.



## XBee DigiMesh 2.4 DC characteristics

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

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

## XBee DigiMesh 2.4 ADC operating characteristics

The following table displays the ADC timing and performance characteristics.

Symbols	Parameter	Condition	Min	Typical	Max	Units
$V_{REFH}$	VREF-analog-to-digital converter reference range		2.08	-	$V_{DDAD}$	V

<sup>1</sup>Maximum electrical operating range, not valid conversion range.

<sup>2</sup>Maximum electrical operating range, not valid conversion range.

<sup>3</sup>Maximum electrical operating range, not valid conversion range.

<sup>4</sup>Maximum electrical operating range, not valid conversion range.

Symbols	Parameter	Condition	Min	Typical	Max	Units
$I_{REF}$	VREF-reference supply current	Enabled	-	200	-	$\mu A$
		Disabled or sleep mode	-	< 0.01	0.02	$\mu A$
$V_{INDC}^1$	Analog input voltage		$V_{SSAD} - 0.3$	-	$V_{SSAD} + 0.3$	V

## XBee DigiMesh 2.4 ADC timing and performance characteristics

The following table displays the ADC timing and performance characteristics.<sup>2</sup>

Symbols	Parameter	Condition	Min	Typical	Max	Units
$R_{AS}$	Source impedance at input <sup>3</sup>		-	-	10	k $\Omega$
RES	Ideal resolution (1 LSB) <sup>4</sup>	$2.08V > V_{DDAD} > 3.6V$	2.031		3.516	mV
DNL	Differential non-linearity <sup>5</sup>		-	$\pm 0.5$	$\pm 1.0$	LSB
INL	Integral non-linearity <sup>6</sup>		-	$\pm 0.5$	$\pm 1.0$	LSB

<sup>1</sup>Analog input must be between  $V_{REFL}$  and  $V_{REFH}$  for valid conversion. Values greater than  $V_{REFH}$  will convert to \$3FF.

<sup>2</sup>All Accuracy numbers are based on processor and system being in WAIT state (very little activity and no IO switching) and that adequate low-pass filtering is present on analog input pins (filter with 0.01  $\mu F$  to 0.1  $\mu F$  capacitor between analog input and  $V_{REFL}$ ). Failure to observe these guidelines may result in system or microcontroller noise causing accuracy errors which will vary based on board layout and the type and magnitude of the activity. Data transmission and reception during data conversion may cause some degradation of these specifications, depending on the number and timing of packets. It is advisable to test the ADCs in your installation if best accuracy is required.

<sup>3</sup>RAS is the real portion of the impedance of the network driving the analog input pin. Values greater than this amount may not fully charge the input circuitry of the ATD resulting in accuracy error.

<sup>4</sup>The resolution is the ideal step size or  $1LSB = (V_{REFH} - V_{REFL})/1024$ .

<sup>5</sup>Differential non-linearity is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to and from the current code.

<sup>6</sup>Integral non-linearity is the difference between the transition voltage to the current code and the adjusted ideal transition voltage for the current code. The adjusted ideal transition voltage is  $(Current\ Code.1/2) * (1 / ((V_{REFH} + E_{FS}) / (V_{REFL} + E_{ZS})))$ .

Symbols	Parameter	Condition	Min	Typical	Max	Units
$E_{ZS}$	Zero-scale error <sup>1</sup>		-	±0.4	±1.0	LSB
$F_{FS}$	Full-scale error <sup>2</sup>		-	±0.4	±1.0	LSB
$E_{IL}$	Input leakage error <sup>3</sup>		-	±0.05	±5.0	LSB
$E_{TU}$	Total unadjusted error <sup>4</sup>		-	±1.1	±2.5	LSB

<sup>1</sup>Zero-scale error is the difference between the transition to the first valid code and the ideal transition to that code. The ideal transition voltage to a given code is  $(Code.1/2) \cdot (1/(V_{REFH} - V_{REFL}))$ .

<sup>2</sup>Full-scale error is the difference between the transition to the last valid code and the ideal transition to that code. The ideal transition voltage to a given code is  $(Code.1/2) \cdot (1/(V_{REFH} - V_{REFL}))$ .

<sup>3</sup>Input leakage error is error due to input leakage across the real portion of the impedance of the network driving the analog pin. Reducing the impedance of the network reduces this error.

<sup>4</sup>Total unadjusted error is the difference between the transition voltage to the current code and the ideal straight-line transfer function. This measure of error includes inherent quantization error (1/2 LSB) and circuit error (differential, integral, zero-scale, and full-scale) error. The specified value of  $E_{TU}$  assumes zero  $E_{IL}$  (no leakage or zero real source impedance).

## XBee DigiMesh 2.4 modes

---

### Transparent and API operating modes

The firmware operates in several different modes. Two top-level modes establish how the device communicates with other devices through its serial interface: Transparent operating mode and API operating mode.

### Transparent operating mode

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

### API operating mode

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

There are two types of API operating modes: one with escaped characters and another without escaped characters.

- Without escaped characters (**AP** = 1). This mode eliminates escaping character sequences. This makes it simpler to create code and libraries, but runs a minor risk of lost frames or errors due to the possibility that payload data can be confused with frame structure. We do not recommend this mode for noisy radio environments and where payload data may include special characters (specifically 0x7E, 0x7D, 0x11, and 0x13).

- With escaped characters (**AP** = 2). API escaped operating mode works similarly to API mode. The only difference is that when working in API escaped mode, the software must escape any payload bytes that match API frame specific data, such as the start-of-frame byte (0x7E).

## Comparing Transparent and API modes

XBee devices can use their serial connections in two ways: Transparent operating mode or API operating mode. You can use a mixture of devices running API mode and transparent mode in a network. The following table provides a comparison of the two modes.

Transparent operating mode	API operating mode
<p>When to use:</p> <ul style="list-style-type: none"> <li>• The conditions for using API mode do not apply.</li> </ul>	<p>When to use:</p> <ul style="list-style-type: none"> <li>• The device sends wireless data to multiple destinations.</li> <li>• The device configures remote XBee devices in the network.</li> <li>• The device receives wireless data packets from multiple XBee devices, and the application needs to identify which devices send each packet.</li> <li>• The device receives I/O samples from remote XBee devices.</li> </ul>

Transparent operating mode	API operating mode
<p>Advantages:</p> <ul style="list-style-type: none"><li>• Provides a simple interface that makes it easy to get started with XBee devices.</li><li>• It is easy for an application to support; what you send is exactly what other modules get, and vice versa.</li><li>• Works very well for two-way communication between XBee devices.</li></ul>	<p>Advantages:</p> <ul style="list-style-type: none"><li>• You can set or read the configuration of remote XBee devices in the network.</li><li>• You can transmit data to one or multiple destinations; this is much faster than transparent mode where the configuration must be updated to establish a new destination.</li><li>• Received data includes the sender's address.</li><li>• Received data includes transmission details and reasons for success or failure.</li><li>• This mode has several advanced features, such as advanced networking diagnostics, and firmware upgrades.</li></ul>



Transparent operating mode	API operating mode
<p>Disadvantages:</p> <ul style="list-style-type: none"> <li>• You cannot set or read the configuration of remote XBee devices in the network.</li> <li>• You must first update the configuration to establish a new destination and transmit data.</li> <li>• You cannot identify the source of received data, as it does not include the sender's address.</li> <li>• Received data does not include transmission details or the reasons for success or failure.</li> <li>• This mode does not offer the advanced features of API mode, including advanced networking diagnostics, and firmware upgrades.</li> </ul>	<p>Disadvantages:</p> <ul style="list-style-type: none"> <li>• The interface is more complex; data is structured in packets with a specific format.</li> <li>• This mode is more difficult to support; transmissions are structured in packets that need to be parsed (to get data) or created (to transmit data).</li> <li>• Sent data and received data are not identical; received packets include some control data and eXTend vBa information.</li> </ul>

## XBee DigiMesh 2.4 additional modes

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

### Command mode

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

### 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.

### Receive mode

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

- ID match
- Channel match
- Address match

### Sleep modes

Sleep modes allows the device to enter states of low power consumption when not in use. The device is almost completely off during sleep, and is incapable of sending or receiving data until it wakes up. XBee devices support both pin sleep, where the module enters sleep mode upon pin transition, and cyclic sleep, where the module sleeps for a fixed time. While asleep, nodes cannot receive RF messages or read commands from the UART port.

The sleep modes are:

- Normal mode. Normal mode is the default for a newly powered-on node. In this mode, a node does not sleep. Normal mode nodes should be mains-powered.
- Asynchronous Pin Sleep mode. This mode allows the device to sleep and wake according to the state of the Sleep\_RQ pin (pin 9).

- Asynchronous Cyclic Sleep Mode. This mode allows the device to sleep for a specified time and wake for a short time to poll.
- Asynchronous Cyclic Sleep with Pin Wake Up mode. In this mode you can wake the device up prematurely using the Sleep\_RQ pin.
- Synchronous Sleep Support mode. A node in this mode synchronizes itself with a sleeping network, but does not sleep itself. At any time, the node responds to new nodes that attempt to join the sleeping network using a sync message.
- Synchronous Cyclic Sleep mode. A node in synchronous cyclic sleep mode sleeps for a programmed time, wakes in unison with other nodes, exchanges data and sync messages, and then returns to sleep.

## Transmit mode

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

## Enter Command mode

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

---

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

---

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

Once the device is in Command mode, it listens for user input and is able to receive AT commands on the UART. If **CT** time (default is 10 seconds) passes without any user input, the device drops out of Command mode and returns to Transparent operating mode.

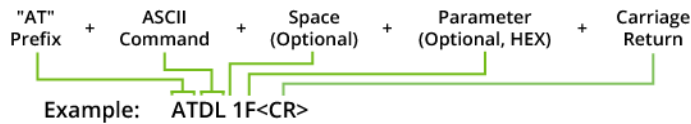
You can customize the guard times and timeout in the device's configuration settings. For information on how to do this, see [CC \(Command Sequence Character\) on page 84](#), [CT \(Command mode Timeout\) on page 86](#) and [GT \(Guard Times\) on page 99](#).

## Troubleshooting

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

## Send AT commands

Once the device enters Command mode, use the syntax in the following figure to send AT commands.



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

The preceding example would change the device's destination address (Low) to 0x1F.

To store the new value to non-volatile (long term) memory, send the **WR** (Write) command. This allows parameter values that you modify to persist in the device's registry after a reset. Otherwise, the device restores parameters to the previous values after a reset.

## Respond to AT commands

When you send a command to the device, the device parses and runs the command. If the command runs successfully, the device returns an **OK** message. If the command errors, the device returns an **ERROR** message.

## Exit Command mode

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

## Configure the XBee/XBee-PRO DigiMesh 2.4

---

### Software libraries

One way to communicate with the XBee device is by using a software library. There are two primary libraries available for use with the devices:

- [XBee Java library](#)
- [XBee ANSI C library](#)

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

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

### Configure the device using XCTU

XBee Configuration and Test Utility (XCTU) is a multi-platform program that enables developers 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 full support of the XBee/XBee-PRO DigiMesh 2.4, you must use XCTU version 6.3.0 or higher.

For instructions on downloading and using XCTU, go to:

<http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>

## XBee DigiMesh 2.4 serial communication

---

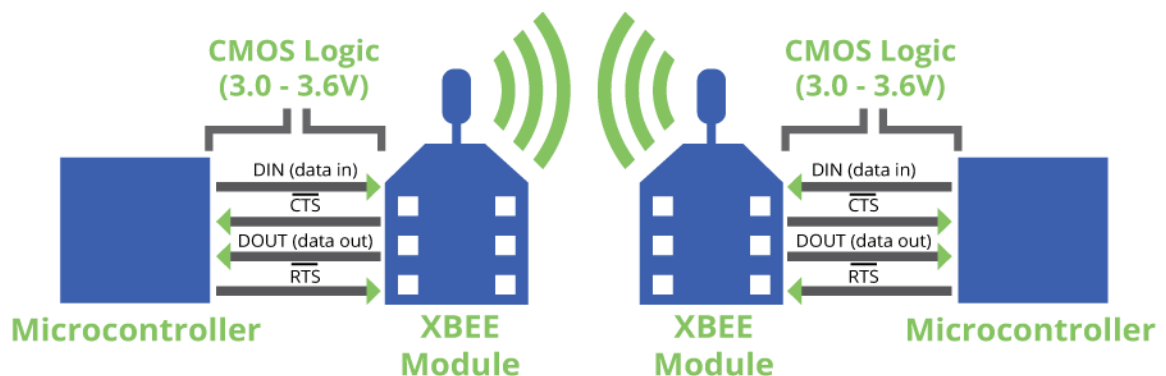
### Serial interface

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

### UART data flow

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

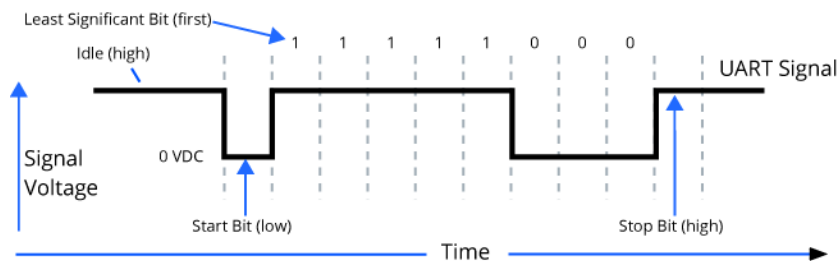
Devices that have a UART interface can connect directly to the pins of the XBee/XBee-PRO DigiMesh 2.4 as the following figure shows. The figure shows system data flow in a UART-interfaced environment. Low-asserted signals have a horizontal line over the signal name.



## Serial data

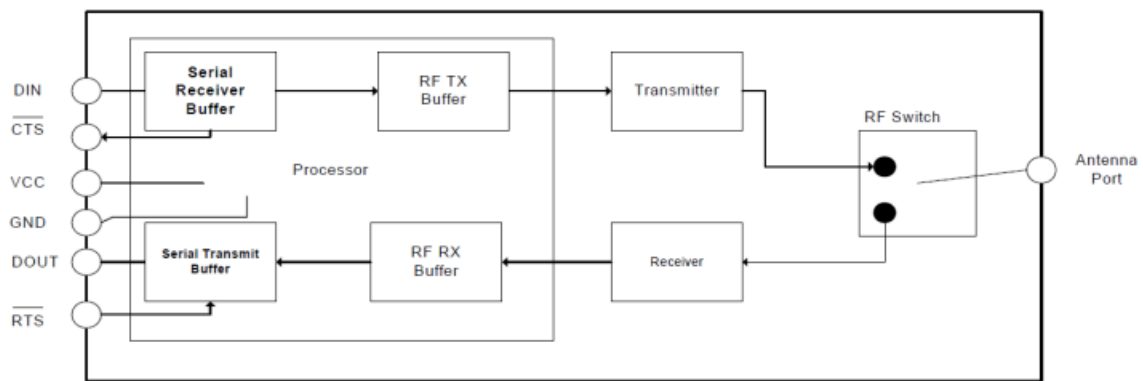
A device sends data to the device's UART through pin 3 (DIN) as an asynchronous serial signal. When the device is not transmitting data, the signal idles high.

For serial communication to occur, you must configure the UART of both devices 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. It shows UART data packet 0x1F (decimal number 31) as transmitted through the device.



## XBee DigiMesh 2.4 serial buffers

The XBee devices maintain buffers to collect serial and RF data that they receive. 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.



## XBee DigiMesh 2.4 serial buffer issues

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

## Serial receive buffer

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

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

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

## Serial transmit buffer

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

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

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

## Serial flow control

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

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

$\overline{\text{CTS}}$  flow control is enabled by default; you can disable it with the **D7** command. When the serial receive buffer fills with the number of bytes specified by the **FT** parameter, the device de-asserts  $\overline{\text{CTS}}$  (sets it high) to signal the host device to stop sending serial data. The device re-asserts  $\overline{\text{CTS}}$  when less than FT-16 bytes are in the UART receive buffer; for more information, see [FT \(Flow Control Threshold\) on page 98](#).



## **RTS flow control**

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

## Work with networked devices

---

### Network commissioning and diagnostics

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

### Configure devices

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

### Configure a device remotely

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

### Send a remote command

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

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

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

XCTU has a Frames Generator tool that can assist you with building and sending a remote AT frame.

## Apply changes on remote devices

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

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

## Remote command response

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

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

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

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

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

## Establish and maintain network links

### Build aggregate routes

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

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

Upon deploying a DigiMesh network, you can issue the **AG** command on the desired aggregator node to cause all nodes in the network to build routes to the aggregator node. The command can optionally be used to automatically update the **DH/DL** registers to match the MAC address of the aggregator node.

The **AG** command requires a 64-bit parameter. The parameter indicates the current value of the **DH/DL** registers on a device which should be replaced by the 64-bit address of the node sending the AG broadcast. If it is not desirable to update the **DH/DL** of the module receiving the AG broadcast then the invalid address of 0xFFFFE can be used. API enabled modules output an Aggregator Update API frame (0x8E) if they update their **DH/DL** address; for a description of the frame, see [Aggregate Addressing Update frame 0x8E on page 148](#).

All devices which receive an **AG** broadcast update their routing table information to build a route to the sending device, regardless of whether or not their **DH/DL** address is updated. The devices use this routing information for future DigiMesh unicast transmissions.

## Examples

### Example one:

In a scenario where you deploy a network, and then you want to update the **DH** and **DL** registers of all the devices in the network so that they use the MAC address of the aggregator node, which has the MAC address 0x0013A2004052C507, you could use the following technique.

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

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

### Example two:

If you want all of the nodes in the network to build routes to an aggregator node with a MAC address of 0x0013A2004052C507 without affecting the **DH** and **DL** registers of any nodes in the network:

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

## Replace nodes

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

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

### Example three:

You were using a device with serial number 0x0013A2004052C507 as a network aggregator. You replace the aggregator with a device with serial number 0x0013A200F5E4D3B2.

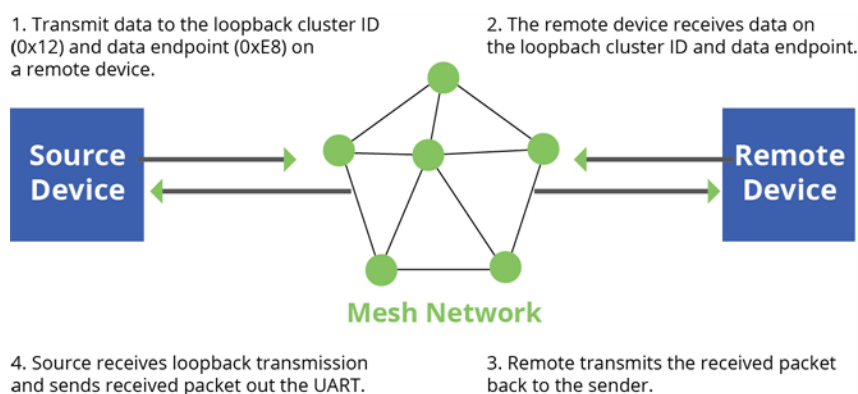
Send the ATAG0013A2004052C507 command to the new device. This causes all devices which had a **DH** and **DL** register setting of 0x0013A2004052C507 to update their **DH** and **DL** register setting to the MAC address of the sending device (0x0013A200F5E4D3B2).

## Test links in a network

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

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

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



The configuration steps for sending data to the loopback cluster ID depend on what mode the device is in. For details on setting the mode, see [AP \(API Enable\) on page 80](#). The following sections list the steps based on the device's mode.

## Transparent operating mode configuration (AP=0)

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

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

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

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

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

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

## Test links between adjacent devices

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

To perform a link test:

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

Number of Bytes	Field Name	Description
8	Destination address	The address the device uses to test its link.
2	Payload size	The size of the test packet. Use the NP command to query the maximum payload size for the device.
2	Iterations	The number of packets to send. This must be a number between 1 and 4000.

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

Number of Bytes	Field Name	Description
8	Destination address	The address the device used to test its link.
2	Payload size	The size of the test packet device A sent to test the link.
2	Iterations	The number of packets that device A sent.
2	Success	The number of packets that were successfully acknowledged.
2	Retries	The number of MAC retries used to transfer all the packets.
1	Result	0x00 - the command was successful. 0x03 - invalid parameter used.
1	RR	The maximum number of MAC retries allowed.
1	maxRSSI	The strongest RSSI reading observed during the test.
1	minRSSI	The weakest RSSI reading observed during the test.
1	avgRSSI	The average RSSI reading observed during the test.

## Example

Suppose that you want to test the link between device A (**SH/SL** = 0x0013a20040521234) and device B (**SH/SL**=0x0013a2004052abcd) by transmitting 1000 40-byte packets:

- Send the following API packet to the serial interface of the device that will output the results, device C.
- Device C can be the same as device A or B.

In the following example packet, whitespace marks fields, bold text is the payload portion of the packet:

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

Device C may return the following packet:

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

This means:

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

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

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

## RSSI indicators

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

You can use the **DB** command to measure the RSSI on a device. **DB** returns the RSSI value measured in -dBm of the last packet the device received. This number can be misleading in multi-hop DigiMesh networks. The **DB** value only indicates the received signal strength of the last hop. If a transmission spans multiple hops, the **DB** value provides no indication of the overall transmission path, or the quality of the worst link, it only indicates the quality of the last link.

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

1. Enable the RSSI pulse-width modulation (PWM) functionality using the **P0** command.
2. Use the RSSI/PWM module pin (pin 6). When the device receives data, it sets the RSSI PWM to a value based on the RSSI of the packet it receives.

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.



## Discover devices

### Discover all the devices on a network

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

1. The device sends a broadcast **ND** command through the network.
2. All devices that receive the command send a response that includes their addressing information, node identifier string and other relevant information. For more information on the node identifier string, see [NI \(Node Identifier\) on page 105](#).

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

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

- The device that sends the **ND** includes its **NT** setting in the transmission to provide a delay window for all devices in the network.
- On large networks, you may need to increase **NT** to improve the reliability of network discovery.
- The default **NT** value is 0x82 (13 seconds).

### Discover devices within RF range

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

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

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

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

## Trace route option

In many networks, it is useful to determine the route that a DigiMesh unicast takes to its destination; particularly, when you set up a network or want to diagnose problems within a network. Because of the large number of Route Information Packet frames that a unicast with trace route enabled can generate,

we suggest you only use the trace route option for occasional diagnostic purposes and not for normal operations.

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

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

The Route Information Packet frame contains:

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

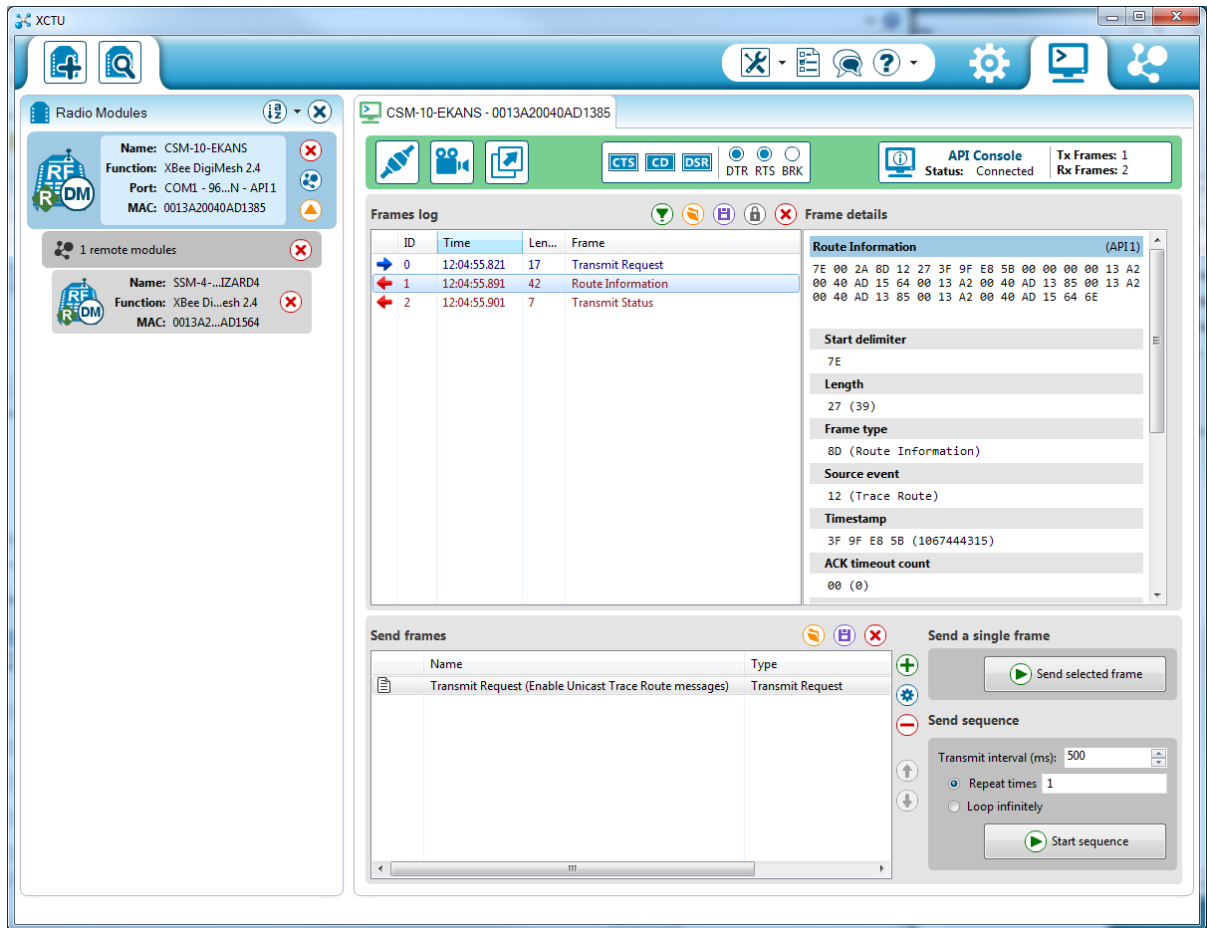
For a full description of the Route Information Packet frame, see [Route Information Packet 0x8D on page 144](#).

### Trace route example

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

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

The following figure shows a trace route example:



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

## NACK messages

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

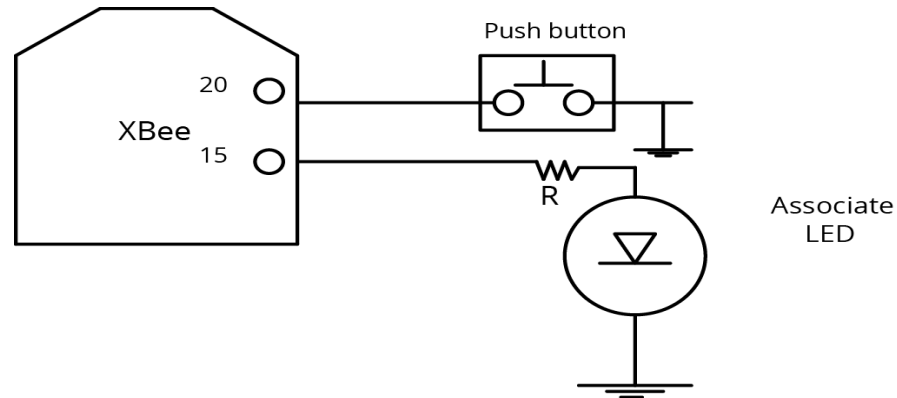
If you use this option, then when a MAC acknowledgment failure occurs on one of the hops to the destination device, the device generates a Route Information Packet (0x8D) frame and sends it to the originator of the unicast.

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

## The Commissioning Pushbutton

Devices support a set of commissioning and LED functions to help you deploy and commission modules. These functions include the Commissioning Pushbutton definitions and the associated LED functions. The

following diagram shows how the hardware can support these features.



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

## Definitions

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

You must perform multiple button presses within two seconds.

The following table provides the pushbutton definitions.

Button presses	Sleep configuration and sync status	Action
1	Not configured for sleep	Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission blink their Associate LED rapidly for one second. All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART.
1	Configured for asynchronous sleep	Wakes the device for 30 seconds. Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission blink their Associate LED rapidly for one second. All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART.

Button presses	Sleep configuration and sync status	Action
1	Configured for synchronous sleep	Wakes the module for 30 seconds or until the synchronized network goes to sleep. Queues a Node Identification broadcast transmission that it sends at the beginning of the next network wake cycle. All devices that receive this transmission blink their Associate LED rapidly for one second. All devices in API operating mode that receive this transmission send a Node Identification Indicator frame (0x95) out their UART.
2	Not configured for synchronous sleep	No effect.
2	Configured for synchronous sleep	Causes a node configured with sleeping router nomination enabled to immediately nominate itself as the network sleep coordinator. For more information, see <a href="#">SO (Sleep Options) on page 114</a> .
4	Any	Sends an ATRE command to restore device parameters to default values.

### Use the Commissioning Pushbutton

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

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

If you enable the Commissioning Pushbutton during sleep, it increases the sleeping current draw, especially in Asynchronous pin sleep (SM1) mode. When asleep, hold down the Commissioning Pushbutton for up to two seconds to wake the device from sleep, then issue the two or four button presses.

## Monitor I/O lines

Devices support both analog input and digital IO line modes on several configurable pins.

### Queried sampling

The following table provides typical parameters for the pin configuration commands.

Pin command parameter	Description
0	Unmonitored digital input.
1	Reserved for pin-specific alternate functionality.
2	Analog input (A/D pins) or PWM output (PWM pins).
3	Digital input, monitored.
4	Digital output, low.
5	Digital output, high.
6-9	Alternate functionality, where applicable.

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

XBee pin names	XBee pin number	Configuration command
CD / DIO12	4	P2
PWM0 / RSSI / DIO10	6	P0
PWM1 / DIO11	7	P1
DTR / SLEEP_RQ / DIO8	9	D8
AD4 / DIO4	11	D4

XBee pin names	XBee pin number	Configuration command
CTS / DIO7	12	D7
ON_SLEEP / DIO9	13	D9
ASSOC / AD5 / DIO5	15	D5
RTS / DIO6	16	D6
AD3 / DIO3	17	D3
AD2 / DIO2	18	D2
AD1 / DIO1	19	D1
AD0 / DIO0 / Commissioning Pushbutton	20	D0

Use the PR command to enable the pullup resistors for each digital input.

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

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

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



If you issue the IS command using an API frame, then the device returns an AT Command Response (0x88) frame with the IO data included in the command data portion of the packet.

Example	Sample AT response
0x01\r	[1 sample set]
0x0C0C\r	[Digital Inputs: DIO 2, 3, 10, 11 enabled]
0x03\r	[Analog Inputs: A/D 0, 1 enabled]
0x0408\r	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0\r	[Analog input: ADIO 0= 0x3D0]
0x0124\r	[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 module at a periodic rate. You set the periodic sample rate with the IR command.

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

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

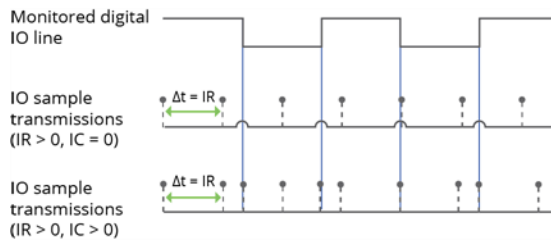
Only devices with API operating mode enabled will send I/O data samples out their UART. Devices that are not in API operating mode discard the I/O data samples they receive.

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

## Detect digital I/O changes

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

The figure below shows how edge detection can work with periodic sampling. In the figure, the grey dashed lines with a dot on top represent samples taken from the monitored DIO line. The top graph shows only IR samples, the bottom graph shows a combination of IR samples and IC (change detect). In the top graph, the humps indicate that the sample was not taken at that exact moment and needed to wait for the next IR sample period.



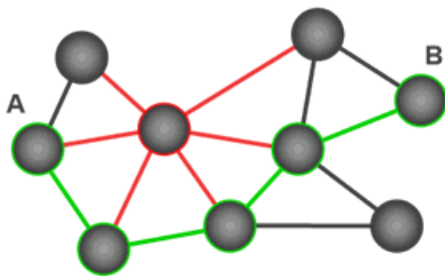
## DigiMesh networking

---

### DigiMesh networking

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

- **Routing.** With this technique, the message is propagated along a path by hopping from node to node until it reaches its final destination.
- **Ad-hoc network creation.** This is an automated process that creates an entire network of nodes on the fly, without any human intervention.
- **Self-healing.** This process automatically figures out if one or more nodes on the network is missing and reconfigures the network to repair any broken routes.



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

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

---

**Note** Mesh networks use more bandwidth for administration and therefore have less available for payloads. They can also be more complex to configure and debug in some cases.

---

## Routers and end devices

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

## Network identifiers

You define DigiMesh networks with a unique network identifier. Use the ID command to set this identifier. For devices to communicate, you must configure them with the same network identifier.

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

## Operating channels

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

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

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

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

A device will:

- Only receive data from other devices within the same network (with the same ID value) and using the same channel (with the same CH value).
- Only be able to transmit data to other devices within the same network (with the same ID value) and using the same channel (with the same CH value).

## Unicast addressing

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

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

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

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

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

## Broadcast addressing

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

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

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

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

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

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

## Routing

A device within a mesh network determines reliable routes using a routing algorithm and table. The routing algorithm uses a reactive method derived from Ad-hoc On-demand Distance Vector (AODV). The firmware uses an associative routing table to map a destination node address with its next hop. A device sends a message to the next hop address, and the message either reaches its destination or forwards to an intermediate router that routes the message on to its destination.

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

Packet tracking prevents a node from resending a broadcast message more than **MT+1** times.

## Route discovery

Route discovery is a process that occurs when:

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

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

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

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

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

## Throughput

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

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

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

Configuration	Data throughput
1 hop, encryption disabled	27.0 kb/s
3 hop, encryption disabled	10.9 kb/s

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

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

## Transmission timeouts

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

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

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

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

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

## Unicast one hop time

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

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

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

## Transmit a broadcast

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

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

The **NH**, **NN**, and **MT** parameters define the maximum broadcast delay as follows:

$$\text{BroadcastTxTime} = \text{NN} * \text{NH} * (\text{MT} + 1) * 18 \text{ms}$$

## Transmit a unicast with a known route

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

You can estimate the timeout in the following manner:

$$\text{knownRouteUnicast} = 2 * \text{NH} * \text{MR} * \text{unicastOneHopTime}$$

## Transmit a unicast with an unknown route

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

$$\text{unknownRouteUnicast} = \text{BroadcastTxTime} + \text{NH} * \text{unicastOneHopTime} + \text{knownRouteUnicast}$$



## Transmit a unicast with a broken route

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

$$\text{brokenRouteUnicast} = \text{BroadcastTxTime} + \text{NH} * \text{unicastOneHopTime} + 2 * \text{knownRouteUnicast}$$

## DigiMesh sleep modes

---

### XBee DigiMesh 2.4 sleep modes

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

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

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

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

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

### Asynchronous modes

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

### Synchronous modes

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

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

## Normal mode

Set SM to 0 to enter Normal mode.

When you power-on a device, it defaults to Normal mode. If a device is in this mode, it does not sleep.

Use mains-power for devices in Normal mode.

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

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

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

## Asynchronous Pin Sleep mode

Set SM to 1 to enter Asynchronous Pin Sleep mode.

Pin sleep allows the device to sleep and wake according to the state of the Sleep\_RQ pin (pin 9).

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

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

## Asynchronous Cyclic Sleep mode

Set SM to 4 to enter Asynchronous Cyclic Sleep mode.

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

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

Assert the ON/SLEEP pin (high) to wake the device.

De-assert ON/SLEEP (low) to put the device to sleep.

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

## **Asynchronous Cyclic Sleep with Pin Wake Up mode**

Set SM to 5 to enter Asynchronous Cyclic Sleep with Pin Wake Up mode.

This mode is a slight variation on Asynchronous Cyclic Sleep mode (SM=4) that allows you to wake a device prematurely by asserting the Sleep\_RQ pin (pin 9).

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

## **Synchronous Sleep Support mode**

Set SM to 7 to enter Synchronous Sleep Support mode.

A device in Synchronous Sleep Support mode synchronizes itself with a sleeping network, but does not sleep itself.

At any time, the node responds to new nodes that attempt to join the sleeping network with a sync message.

A sleep support node only transmits normal data when the other nodes in the sleeping network are awake.

Sleep support nodes are especially useful when:

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

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

## **Synchronous Cyclic Sleep mode**

Set SM to 8 to enter Synchronous Cyclic Sleep mode.

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

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

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

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

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

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

Configure all nodes in a synchronous sleep network to operate in either Synchronous Sleep Support mode or Synchronous Cyclic Sleep mode.

Asynchronous sleeping nodes are not compatible with synchronous sleep nodes.

## The sleep timer

In Asynchronous Cyclic Sleep Mode and Asynchronous Cyclic Sleep with Pin Wake Up modes (SM = 4 or SM = 5), if the device receives serial or RF data, it starts a sleep timer (time until sleep).

If the device receives any data serially or by RF link, the timer resets.

Set the duration of the timer using the ST command.

When the sleep timer expires the device returns to sleep.

## Coordinating sleep modes in the DigiMesh network

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

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

If you change the SP or ST parameters, the network does not apply the new settings until the beginning of the next wake time.

A sleeping router network is robust enough that an individual node can go several cycles without receiving a sync message, due to RF interference, for example. As a node misses sync messages, the time available

for transmitting messages during the wake time reduces to maintain synchronization accuracy. By default, a device reduces its active sleep time progressively as it misses sync messages.

## DigiMesh synchronization messages

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

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

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

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

Use the **SO** (sleep options) command to disable deployment mode.

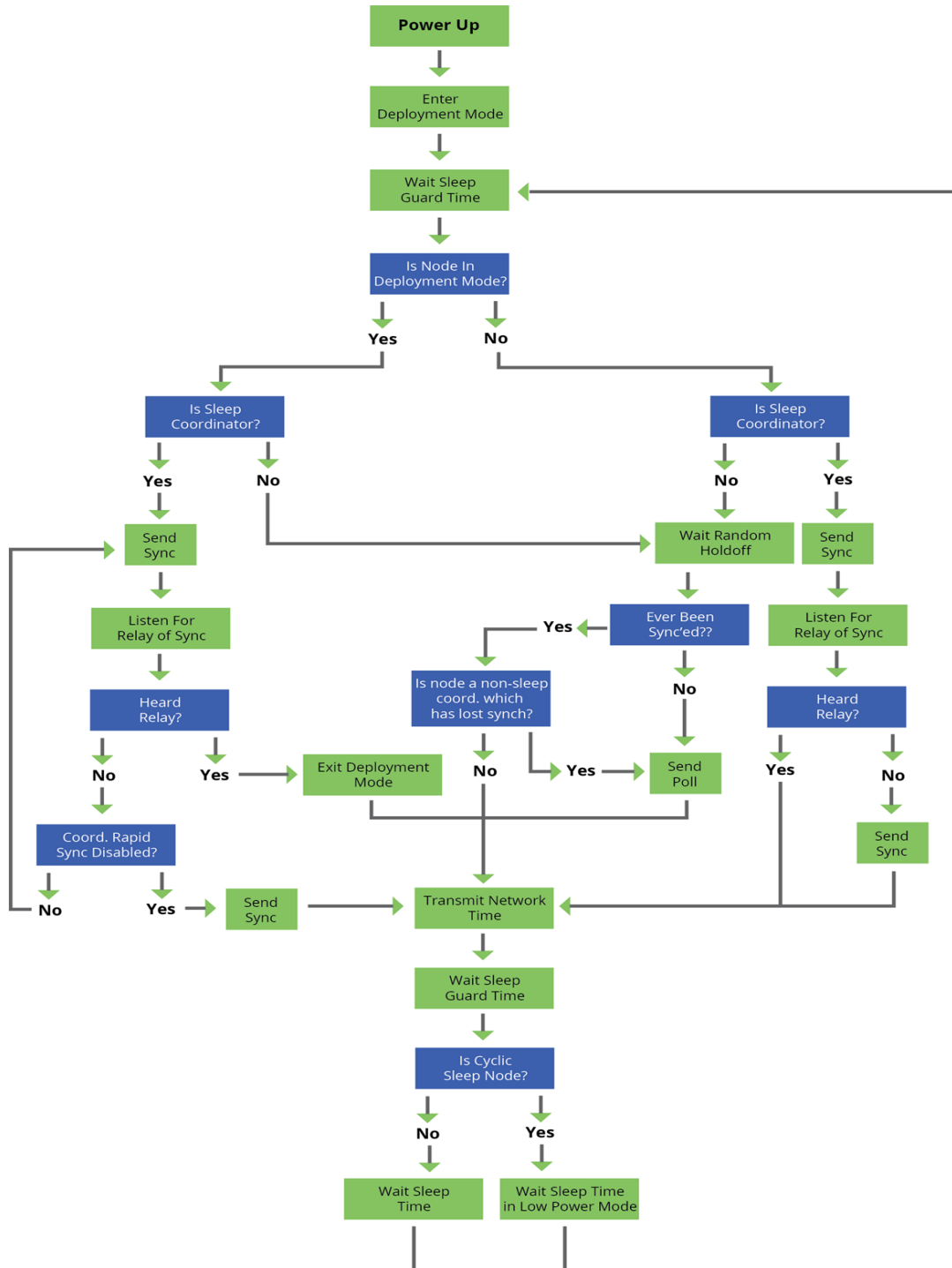
A sleep coordinator that is not in deployment mode sends a sync message at the beginning of the wake cycle. The sleep coordinator listens for a neighboring node to relay the sync. If it does not hear the relay, the sleep coordinator sends the sync one additional time.

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

If you use the SO command to configure nodes as non-coordinators:

- If the non-coordinators go six or more sleep cycles without hearing a sync, they send a message requesting sync at the beginning of their wake period.

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



## Become a sleep coordinator

In DigiMesh 2.4 networks, a node can become a sleep coordinator in one of four ways:

## Preferred sleep coordinator option

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

A node with the sleep coordinator bit set always sends a sync message at the beginning of a wake cycle. For this reason, do not set this bit on more than one node in the network.

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

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

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

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

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

## Resolution criteria and selection option

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

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

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

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

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

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

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



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

## **Commissioning Pushbutton option**

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

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

Only sleep coordinator nodes honor Commissioning Pushbutton nomination requests.

## **Changing sleep parameters**

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

For normal operations, a device uses the sleep and wake parameters it gets from the sleep sync message, not the ones specified in its SP and ST parameters. It does not update the SP and ST parameters with the values of the sync message.

Use the OS and OW commands to query the current network sleep and wake times used by the node.

Changing network parameters can cause a node to become a sleep coordinator and change the sleep settings of the network. The following commands can cause this to occur: NH, NN, and MR.

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

## **Sleep guard times**

To compensate for variations in the timekeeping hardware of the various devices in a sleeping router network, the network allocates sleep guard times at the beginning and end of the wake time. The size of the sleep guard time varies based on the sleep and wake times you select and the number of sleep cycles that elapse since receiving the last sync message. The sleep guard time guarantees that a destination

module will be awake when the source device sends a transmission. As a node misses more and more consecutive sync messages, the sleep guard time increases in duration and decreases the available transmission time.

## Auto-early wake-up sleep option

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

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

Use the SO command to disable auto-early wake-up sleep.

## Select sleep parameters

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

1. Choose NN and NH.

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

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

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

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

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

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

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

## Start a sleeping DigiMesh network

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

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

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

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

## Adding a new node to an existing network

To add a new node to the network, the node must receive a sync message from a node already in the network. On power-up, an unsynchronized, sleep compatible node periodically sends a broadcast

requesting a sync message and then sleeps for its SP period. Any node in the network that receives this message responds with a sync. Because the network can be asleep for extended periods of time, and cannot respond to requests for sync messages, there are methods that can be used to sync a new node while the network is asleep.

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

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

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

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

## Change sleep parameters

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

If you use a preferred sleep coordinator or if you know which node acts as the sleep coordinator, we suggest that you use this node to make changes to network settings.

If you do not know the network sleep coordinator, you can use any node that does not have the non-sleep coordinator sleep option bit set. For details on the bit, see [SO \(Sleep Options\) on page 114](#).

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

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

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

Using this sleep option increases reliability but may decrease battery life.

Nodes using this sleep option that miss sync messages increase their wake time and decrease their sleep time during cycles where they miss the sync message. This increases power consumption.

When changing between two sets of sleep settings, choose settings so that the wake periods of the two sleep settings occur at the same time. In other words, try to satisfy the following equation:  $(SP1 + ST1) = N * (SP2 + ST2)$ , where  $SP1/ST1$  and  $SP2/ST2$  are the desired sleep settings and  $N$  is an integer.

## Rejoin nodes that lose sync

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

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

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

- If you disable the non-sleep coordinator bit in the SO command on multiple devices in the network, they are eligible for the network to nominate them as a sleep coordinator. For more details, see [SO \(Sleep Options\) on page 114](#).
- If the devices in the network do not use the auto early wake-up sleep option.

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

1. Place a sleep support node in range of both subnets.
2. Select a node in the subnet that you want the other subnet to sync with.
3. Use this node to slightly change the sleep cycle settings of the network, for example, increment ST.

4. Wait for the subnet's next wake cycle. During this cycle, the node you select to change the sleep cycle parameters sends the new settings to the entire subnet it is in range of, including the sleep support node that is in range of the other subnet.
5. Wait for the out of sync subnet to wake up and send a sync. When the sleep support node receives this sync, it rejects it and sends a sync to the subnet with the new sleep settings.
6. The subnets will now be in sync. You can remove the sleep support node.
7. If you want, you can change the sleep cycle settings back to what they were.

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

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

## XBee DigiMesh 2.4 diagnostics

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

### Query sleep cycle

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

### Sleep status

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

### Missed sync messages command

Use the MS command to query the number of cycles that elapse since the device receives a sync message.

### Sleep status API messages

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

## **XBee DigiMesh 2.4 AT commands**

---

### **1S (XBee Sensor Sample)**

Forces a device sensor to take a sample. You can only send 1S to a device sensor using a Remote Command Request (0x17) frame.

#### **Parameter range**

N/A

#### **Default**

N/A

### **AC (Apply Changes)**

Immediately applies new settings without exiting Command mode.

#### **Parameter range**

N/A

#### **Default**

N/A

### **AG (Aggregator Support)**

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

- The receiving node establishes a DigiMesh route back to the originating node, if there is space in the routing table.
- The DH and DL of the receiving node update to the address of the originating node if the AG parameter matches the current DH/DL of the receiving node.

- API-enabled XBees with updated DH and DL send an Aggregate Addressing Update frame (0x8E) out the UART.

---

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

---

### Parameter range

Any 64-bit number

### Default

N/A

## AO (API Options)

The API data frame output format for RF packets received.

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

### Parameter range

0-2

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

### Default

0

## AP (API Enable)

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

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

### Parameter range

0-2



Parameter	Description
0	Transparent Mode. All UART input and output is raw data and the device uses the RO parameter to delineate packets.
1	API Mode Without Escapes. The device packetizes all UART input and output data in API format, without escape sequences.
2	API Mode With Escapes. The device is in API mode and inserts escaped sequences to allow for control characters. The device passes 0x11, 0x13, 0x7d, and 0x7e as data.

### Default

0

## BC (Bytes Transmitted)

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

This number rolls over to zero from 0xFFFF.

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

### Parameter range

0 - 0xFFFF

### Default

0

## BD (Baud Rate)

Set or read the serial interface baud rate for communication between the serial port of the device and the host.

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

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

**Parameter range**

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

Parameter	Description
0	1200 b/s
1	2400 b/s
2	4800 b/s
3	9600 b/s
4	19200 b/s
5	38400 b/s
6	57600 b/s
7	115200 b/s
0x39 to 0xF4240 if the host supports it.	

**Default**

3 (9600 b/s)

**BH (Broadcast Hops)**

The maximum transmission hops for broadcast data transmissions.

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

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

**Parameter range**

0 - 0x20

**Default**

0

## CA (CCA Threshold)

Set or read the Clear Channel Assessment (CCA) threshold. If the device detects energy above the CCA threshold, it will not transmit.

The CCA parameter is measured in units of -dBm.

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

### Parameter range

0x0 - 0x50 -dBm

### Default

0x0 (CCA disabled)

### Example

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

### ETSI compliance (Europe)

Use the following settings for ETSI compliance.

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

## 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 ATCB1 command to perform the action of pressing the commissioning pushbutton once.

### Parameter range

0-4

### Default

N/A

## CC (Command Sequence Character)

The character value you use to enter Command mode. Use **CC** to set or read the character.

The default value (0x2B) is the ASCII code for the plus (+) character. You must enter it three times within the guard time to enter Command mode from Idle mode. To enter Command mode, there is also a required period of silence before and after the command sequence characters of the Command mode sequence (**GT** + **CC** + **GT**). The period of silence prevents inadvertently entering Command mode.

### Parameter range

0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

### Default

0x2B

## CE (Routing / Messaging Mode)

The routing and messaging network type of the device. End devices do not propagate broadcasts and will not become intermediate nodes on a route.

### Parameter range

0-2

Parameter	Description
0	Standard router
1	N/A
2	End Device

### Default

0

## CH (Operating Channel)

Set or read the operating channel number devices use to transmit and receive data to each other. The devices use 802.15.4 channel numbers.

### Parameter range

0xb - 0x1a

**Default**

0xc

**CI (Cluster ID)**

Sets or reads the default cluster identifier value that the device uses for serial data transmissions.

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

**Parameter range**

0 - 0xFFFF

**Default**

0x11 (Transparent data cluster ID)

**CK (Configuration CRC)**

Reads the cyclic redundancy check (CRC) of the current AT command configuration settings.

This command allows you to detect an unexpected configuration change on a device. Use the code that the device returns to determine if a node has the configuration you want.

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

**Parameter range**

0 - 0xFFFFFFFF

**Default**

N/A

**CN (Exit Command mode)**

Makes the device exit Command mode.

**Parameter range**

N/A

**Default**

N/A

## CT (Command mode Timeout)

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

### Parameter range

2 - 0x1770 (x 100 ms)

### Default

0x64 (10 seconds)

## D0 (DIO0/AD0)

The DIO0/AD0 pin configuration (pin 20).

### Parameter range

Parameter	Description
0x0	Disabled
0x1	Commissioning Pushbutton
0x2	ADC
0x3	Digital Input
0x4	Digital output, low
0x5	Digital output, high

### Default

0x1

## D1 (DIO1/AD1)

The DIO1/AD1 pin configuration (pin 19).

**Parameter range**

Parameter	Description
Parameter	Description
0x0	Disabled
0x1	N/A
0x2	ADC
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x0

**D2 (DIO2/AD2)**

The DIO2/AD2 pin configuration (pin 18).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	N/A
0x2	ADC
0x3	Digital input
0x4	Digital output, low

Parameter	Description
0x5	Digital output, high

**Default**

0x0

**D3 (DIO3/AD3)**

The DIO3/AD3 pin configuration (pin 17).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	N/A
0x2	ADC
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x0

**D4 (DIO4/AD4)**

The DIO4/AD4 pin configuration (pin 11).

**Parameter range**

Parameter	Description
0x0	Disabled



Parameter	Description
0x1	N/A
0x2	ADC
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x0

**D5 (DIO5/AD5/ASSOCIATED\_INDICATOR)**

The DIO5/AD5/ASSOCIATED\_INDICATOR pin configuration (pin 15).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	Associate LED indicator - blinks when associated.
0x2	ADC
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x1

## D6 (DIO6/RTS)

The DIO6/RTS pin configuration (pin 16).

### Parameter range

Parameter	Description
0x0	Disabled
0x1	RTS flow control
0x2	N/A
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

### Default

0x0

## D7 (DIO7/CTS)

The DIO7/CTS pin configuration (pin 12).

### Parameter range

Parameter	Description
0x0	Disabled
0x1	CTS flow control
0x2	N/A
0x3	Digital input

Parameter	Description
0x4	Digital output, low
0x5	Digital output, high
0x6	RS-485 Tx enable, low Tx (0 V on transmit, high when idle)
0x7	RS-485 Tx enable high, high Tx (high on transmit, 0 V when idle)

### Default

0x1

## D8 (DIO8/SLEEP\_REQUEST)

The DIO8/SLEEP\_REQUEST pin configuration (pin 9).

### Parameter range

Parameter	Description
0x0	Disabled
0x1	Sleep request
0x2	N/A
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

### Default

0x1

## D9 (DIO9/ON\_SLEEP)

The DIO9/ON\_SLEEP pin configuration (pin 13).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	ON/SLEEP output
0x2	N/A
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x1

**P0 (DIO10/RSSI/PWM0 Configuration)**

The DIO10/RSSI/PWM0 pin configuration (pin 6).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	RSSI PWM0 output
0x2	PWM0 output
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x1

**P1 (DIO11/PWM1 Configuration)**

The DIO11/PWM1 pin configuration (pin 7).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	N/A
0x2	PWM1 output
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x0

**P2 (DIO12 Configuration)**

The DIO12 pin configuration (pin 4).

**Parameter range**

Parameter	Description
0x0	Disabled
0x1	N/A
0x2	N/A

Parameter	Description
0x3	Digital input
0x4	Digital output, low
0x5	Digital output, high

**Default**

0x0

**DB (Last Packet RSSI)**

This command reports the received signal strength of the last RF data packet that a device receives. The DB command only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link.

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

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

**Parameter range**

N/A

**Default**

0

**DD (Device Type Identifier)**

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

**Parameter range**

0 - 0xFFFFFFFF

**Default**

0x50000

## DE (Destination Endpoint)

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

### Parameter range

0-0xFF

### Default

0xE8

## DH (Destination Address High)

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

0x000000000000FFFF is the broadcast address.

### Parameter range

0 - 0xFFFFFFFF

### Default

0

## DL (Destination Address Low)

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

0x000000000000FFFF is the broadcast address.

### Parameter range

0 - 0xFFFFFFFF

### Default

0xFFFF

## DM (DigiMesh Options)

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

Bit:

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

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

### Parameter range

0 - 0x03 (bit field)

### Default

0

## DN (Discover Node)

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

The following events occur after DN discovers the destination node; if you have AT firmware:

1. The device sets DL and DH to the extended (64-bit) address of the device with the matching NI string.
2. The receiving device returns OK (or ERROR).
3. The device exits Command mode to allow for immediate communication.

If you have API firmware:

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

### Parameter range

20-byte ASCII string

### Default

N/A

## EA (MAC ACK Failure Count)

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

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



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

### Parameter range

N/A

### Default

N/A

## ED (Energy Detect)

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

### Parameter range

0-0x3A98

### Default

N/A

## EE (Encryption Enable)

Enable or disable 128-bit Advanced Encryption Standard (AES) encryption.

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

### Parameter range

0 - 1

Parameter	Description
0	Disabled
1	Enabled

### Default

0

## Find Neighbors (FN)

Discovers and reports all devices found within immediate RF range. FN reports the following information for each device it discovers.

```
MY<CR> (always 0xFFFE)
SH<CR>
SL<CR>
NI<CR> (Variable length)
PARENT_NETWORK_ADDRESS<CR> (2 Bytes) (always 0xFFFE)
DEVICE_TYPE<CR> (1 Byte: 0=Coord, 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<DR> (1 Byte. Optionally included based on NO settings.)
<CR>
```

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

If you send the FN command through a local API frame, each response returns as a separate AT Command Response (0x88) or Remote Command Response (0x97) frame, respectively. The data consists of the bytes in the previous list without the carriage return delimiters. The NI string ends in a 0x00 null character.

### Parameter range

N/A

### Default

N/A

## FR (Software Reset)

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

### Parameter range

N/A

### Default

N/A

## FT (Flow Control Threshold)

Set or read the flow control threshold.

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

### Parameter range

0x11 - 0xee

Bytes

### Default

0xBE

## GD (Good Packets Received)

This count increments when a device receives a good frame with a valid MAC header on the RF interface. Once the number reaches 0xFFFF, it does not count further events. To reset the counter to any 16-bit value, append a hexadecimal parameter to the command.

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

### Parameter range

N/A

### Default

N/A

## GT (Guard Times)

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

### Parameter range

2 - 0xCE4 (x 1 ms)

### Default

0x3E8 (one second)

## HV (Hardware Version)

Read the device's hardware version number.

### Parameter range

0 - 0xFFFF (read-only)

### Default

Set in the factory

## IC (DIO Change Detect)

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

IC works with the individual pin configuration commands (D0 - D9, P0 - P2). If a pin is enabled as a digital I/O, you can use the IC command to force an immediate I/O sample transmission when the DIO state changes. The data transmission contains only DIO data.

IC is a bitmask you can use to enable or disable edge detection on individual channels.

Set unused bits to 0.

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

### Parameter range

0x0 - 0xffff

**Default**

0x0

**ID (Network ID)**

Set or read the user network identifier.

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

When receiving a packet this is checked after the preamble ID. If using OEM network IDs, 0xFFFF uses the factory value.

You can use the **WR** command to write **ID** changes to non-volatile memory.

**Parameter range**

0x0 - 0xFFFF

**Default**

0x7FFF

**IF (Sleep Sample Rate)**

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

**Parameter range**

0x1 - 0xff

**Default**

0x1

**IR (Sample Rate)**

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

To enable periodic sampling, you must set IR to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin. Refer to [D0 \(DIO0/AD0\) on page 86](#) - [D9 \(DIO9/ON\\_SLEEP\) on page 91](#) and [P0 \(DIO10/RSSI/PWM0 Configuration\) on page 92](#) - [P2 \(DIO12 Configuration\) on page 93](#).

**Parameter range**

0x0 - 0xffff (ms)

**Default**

0x0

**IS (Force Sample)**

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

**Parameter range**

N/A

**Default**

N/A

**KY (AES Encryption Key)**

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

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

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

**Parameter range**

128-bit value

**Default**

N/A

**LT (Associate LED Blink Time)**

The Associate LED blink time. If you use the D5 command to enable the Associate LED functionality (DIO5/Associate pin), this value determines the on and off blink times for the LED.

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

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

**Parameter range**

0x14 - 0xff (x 10 ms)

**Default**

0x0

## M0 (PWM0 Duty Cycle)

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

### Parameter range

0x0 - 0x3ff

### Default

0x0

## M1 (PWM1 Duty Cycle)

The duty cycle of the PWM1 line. Use the P1 command to configure the line as a PWM.

### Parameter range

0x0 - 0x3ff

### Default

0x0

## MR (Mesh Unicast Retries)

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

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

We recommend that you set this value to 1.

### Parameter range

0 - 7 mesh unicast retries

### Default

1

## MS (Missed Sync Messages)

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

### Parameter range

N/A

**Default**

N/A

**MT (Broadcast Multi-Transmits)**

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

**Parameter range**

0 - 0xF

**Default**

3

**N? (Network Discovery Timeout)**

The maximum response time, in milliseconds, for network discovery responses (ND) and discover node (DN) responses. The timeout is based on the network discovery back-off time (NT) and the network propagation time.

**Parameter range**

Read-only

**Default**

0x3D6A

**NB (Parity)**

Set or read the parity settings for UART communications.

**Parameter range**

0 - 4

Parameter	Description
0	No parity
1	Even parity
2	Odd parity



Parameter	Description
3	Mark parity (forced high)
4	Space parity (forced low)

**Default**

0

**ND (Network Discover)**

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

**Parameter range**

N/A

**Default**

N/A

**NH (Network Hops)**

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

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

**Parameter range**

1 - 20 hops

**Default**

7

**NI (Node Identifier)**

Stores the node identifier string for a device. This can be up to 20 ASCII characters.

The string only accepts printable ASCII data in Command mode.

The string cannot start with a space.

Use a carriage return or comma to end the command.

The command automatically ends when you enter the maximum bytes for the string.

Use the **ND** command to return this string to easily identify devices.

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

### Parameter range

Up to 20-byte ASCII string

### Default

0x20 (an ASCII space character)

## NN (Network Delay Slots)

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

One network delay slot is approximately 13 ms.

### Parameter range

1 - A network delay slots

### Default

3

## NO (Network Discovery Options)

Set or read the network discovery options value for the **ND** command on a particular device. The options bit field value changes the behavior of the **ND** command and/or changes:

- What optional values the device returns when it receives an **ND** command or API Node Identification Indicator (0x95) frame.

### Parameter range

0x0 - 0x7 (bit field)

### Bit field

Option	Description
0x01	Append the <b>DD</b> value to <b>ND</b> responses or API node identification frames.

Option	Description
0x02	Local device sends <b>ND</b> response frame when <b>ND</b> is issued.
0x04	Append the RSSI of the last hop for DigiMesh networks to <b>ND</b> or <b>FN</b> responses or API node identification frames.

### Default

0x0

## NP (Maximum Packet Payload Bytes)

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

### Parameter range

0 - 0xFFFF (bytes)

### Default

N/A

## NT (Network Discovery Back-off)

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

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

### Parameter range

0x20 - 0x2EE0 (x 100 ms)

### Default

0x82 (13 seconds)

## OS (Operating Sleep Time)

Reads the current network sleep time that the device is synchronized to, in units of 10 milliseconds. If the device has not been synchronized, then OS returns the value of SP.

If the device synchronizes with a sleeping router network, OS is different from SP.

**Parameter range**

N/A

**Default**

N/A

**OW (Operating Wake Time)**

Reads the current network wake time that a device is synchronized to, in 1 ms units.

If the device has not been synchronized, then OW returns the value of ST.

If the device synchronizes with a sleeping router network, OS is different from ST.

**Parameter range**

N/A

**Default**

N/A

**PL (TX Power Level)**

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

**Parameter range**

0 - 4

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

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

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

Setting	Power level
0	+10 dBm
1	+12 dBm
2	+14 dBm
3	+16 dBm
4	+18 dBm

### Default

4

### PR (Pull-up Resistor Enable)

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

Bit	I/O line
0	DIO4/AD4
1	DIO3/AD3
2	DIO2/AD2
3	DIO1/AD1
4	DIO0/AD0
5	DIO6/RTS
6	DIO8/Sleep_Request

Bit	I/O line
7	DIO14/DIN/Config
8	DIO5/AD6/Associate
9	DIO9/On/nSleep
10	DIO12
11	DIO10/RSSI/PWM0
12	DIO11/PWM1
13	DIO7/CTS
14	DOUT (pin 2)

**Parameter range**

0x0 - 0x7fff (bit field)

**Default**

0x1fff

**RE (Restore Defaults)**

Restore device parameters to factory defaults. Does not exit out of Command mode.

**Parameter range**

N/A

**Default**

N/A

**RO (Packetization Timeout)**

Set or read the number of character times of inter-character silence required before transmission begins.

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

**Parameter range**

0x0 - 0xff (x character times)

**Default**

0x3

**RP (RSSI PWM Timer)**

The PWM timer expiration in 0.1 seconds. RP sets the duration of 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**

0x0 - 0xff (x 100 ms)

**Default**

0x28 (four seconds)

**RR (Unicast Retries)**

Set or read the maximum number of MAC level packet delivery attempts for unicasts. If **RR** is non-zero, the packets a device sends request an acknowledgment, and can be present up to **RR** times if the device does not receive any acknowledgments.

**Parameter range**

0 - 0xF

**Default**

0xA

**SE (Source Endpoint)**

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

**Parameter range**

0-0xFF

**Default**

0xE8

## SH (Serial Number High)

Reads the upper 32 bits of the device's unique IEEE 64-bit extended address. The 64-bit source address is always enabled. This value is read-only and it never changes.

### Parameter range

0 - 0xFFFFFFFF

### Default

Set in the factory.

## SL (Serial Number Low)

Reads the lower 32 bits of the device's unique IEEE 64-bit extended address. The 64-bit source address is always enabled. **SL** is read-only and it is also the serial number of the node.

### Parameter range

0 - 0xFFFFFFFF

### Default

Set in the factory

## SM (Sleep Mode)

Set or read the sleep mode of the device.

Normal mode is always awake. Pin sleep modes allow you to wake the device with the SLEEP\_REQUEST line. Asynchronous cyclic mode sleeps for SP time and briefly wakes, checking for activity. Sleep Support mode is always awake but can effectively communicate with SM8 nodes. Synchronized Cyclic Sleep nodes sleep for SP and wake for ST time. Synchronous modes are not compatible with asynchronous modes.

### Parameter range

Parameter	Description
0x0	Normal. The user did not enable a sleep mode.
0x1	Asynchronous Pin Sleep. In this mode, the SLEEP_RQ line controls the sleep/wake state of the device.
0x2	N/A



Parameter	Description
0x3	N/A
0x4	Asynchronous Cyclic Sleep. In this mode, the device periodically sleeps and wakes based on the SP and ST commands.
0x5	Asynchronous Cyclic Sleep Pin Wake. When you assert the SLEEP_REQUEST pin, the device enters a cyclic sleep mode similar to Asynchronous Cyclic Sleep. When you de-assert the SLEEP_REQUEST pin, the device immediately wakes up. The device does not sleep when you de-assert the SLEEP_REQUEST pin.
0x6	N/A
0x7	Sleep Support
0x8	Synchronized Cyclic Sleep

**Default**

0x0

**SN (Number of Cycles Between ON\_SLEEP)**

Set or read the number of sleep periods value. This command controls the number of sleep periods that must elapse between assertions of the ON\_SLEEP line during the wake time of Asynchronous Cyclic Sleep. This allows external circuitry to sleep longer than the SP time.

During cycles when ON\_SLEEP is de-asserted, the device wakes up and checks for any serial or RF data. If it receives any such data, then it asserts the ON\_SLEEP line and the device wakes up fully. Otherwise, the device returns to sleep after checking.

This command does not work with synchronous sleep devices.

**Parameter range**

0x1 - 0xffff

**Default**

0x1

## Example

Set to 1 to set ON\_SLEEP high after each SP time (default).

If SN=3, the ON\_SLEEP line asserts only every third wakeup; SN=9, every 9th wakeup; and so forth.

## SO (Sleep Options)

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

You can set or clear any of the available sleep option bits.

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

## Parameter range

0x0 - 0xffff

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

Bit	Option
0	Preferred sleep coordinator
1	Non-sleep coordinator
2	Enable API sleep status messages
3	Disable early wake-up
4	Enable node type equality
5	Disable lone coordinator sync repeat

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

Bit	Option
8	Always wake for ST time

## Default

0x2

## SP (Sleep Time)

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

### Parameter range

0x1 - 0x15f900 (x 10 ms)

### Default

0xC8

## SQ (Missed Sleep Sync Count)

Counts the number of sleep cycles in which the device does not receive a sleep sync.

Set the value to 0 to reset this value.

When the value reaches 0xFFFF it does not increment anymore.

### Parameter range

N/A

### Default

N/A

## SS (Sleep Status)

Queries a number of Boolean values that describe the device's status.

Bit	Description
0	This bit is true when the network is in its wake state.
1	This bit is true if the node currently acts as a network sleep coordinator.
2	This bit is true if the node ever receives a valid sync message after it powers on.
3	This bit is true if the node receives a sync message in the current wake cycle.
4	This bit is true if you alter the sleep settings on the module so that the node nominates itself and sends a sync message with the new settings at the beginning of the next wake cycle.

Bit	Description
5	This bit is true if you request that the node nominate itself as the sleep coordinator using the commissioning pushbutton or the CB2 command.
6	This bit is true if the node is currently in deployment mode.
All other bits	Reserved. All non-documented bits can be any value you can ignore them.

**Parameter range**

N/A

**Default**

N/A

**ST (Wake Time)**

Sets or reads the wake time of the device.

For devices in asynchronous sleep, ST defines the amount of time that a device stays awake after it receives RF or serial data.

For devices in synchronous sleep, the minimum wake time is a function of MT, SP, NH, NN, and platform dependent values. If you increase SP, NH, NN, or MT, the ST value raises automatically. The maximum value is an hour (0x36ee80 ms).

**Parameter range**

0x1 - 0x36ee80 (x 1 ms)

**Default**

0x7D0 (2 seconds)

**TR (Transmission Failure Count)**

This count increments whenever a MAC transmission attempt exhausts all MAC retries without ever receiving a MAC acknowledgment message from the destination node. Once the number reaches 0xFFFF, it does not count further events. To reset the counter to any 16-bit value, append a hexadecimal parameter to the command.

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

### Parameter range

N/A

### Default

N/A

## UA (Unicasts Attempted Count)

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

This count increments whenever a MAC unicast transmission occurs that requests an ACK. Once the number reaches 0xFFFF, it does not count further events. You can reset the counter to any 16-bit value by appending a hexadecimal parameter to the command.

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

### Parameter range

0 - 0xFFFF

### Default

0

## VL (Version Long)

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

### Parameter range

N/A

### Default

N/A

## VR (Firmware Version)

Reads the firmware version on a device.

### Parameter range

0 - 0xFFFFFFFF (read-only)

### Default

Set in the factory

## WR (Write)

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

---

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

---

### Parameter range

N/A

### Default

N/A

## WH (Wake Host Delay)

Sets or reads the wake host timer value. You can use WH to give a sleeping host processor sufficient time to power up after the device asserts the ON\_SLEEP line.

If you set WH to a non-zero value, this timer specifies a time in milliseconds that the device allows after waking from sleep before sending data out the UART or transmitting an I/O sample. If the device receives serial characters, the WH timer stops immediately.

When in synchronous sleep, the device shortens its sleep period by the WH value to ensure that it is prepared to communicate when the network wakes up. When in this sleep mode, the device always stay awake for the WH time plus the amount of time it takes to transmit a one-hop unicast to another node.

### Parameter range

0x0 - 0xffff (x 1 ms)

### Default

0x0

## %H (MAC Unicast One Hop Time)

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

### Parameter range

This parameter is read-only.

### Default

N/A

## %8 (MAC Broadcast One Hop Time)

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

### Parameter range

This is a read-only parameter

### Default

N/A

## Operate in API mode

---

### API mode overview

As an alternative to Transparent operating mode, you can use API operating mode. API mode requires that your application communicates data with the device using software frames in a particular order. The API specifies how to send commands, receive command responses and device status messages from the device using UART data frames.

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

### API frame specifications

The firmware supports two API operating modes: with escaped characters and without 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:

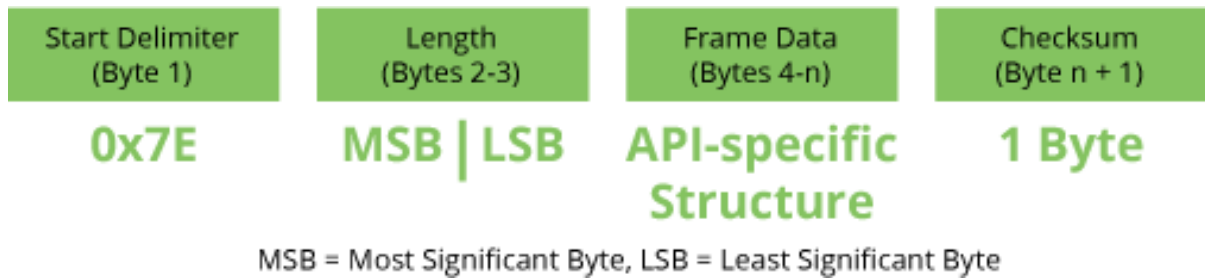
1. AP = 0: Transparent operating mode, UART serial line replacement with API modes disabled.
2. AP = 1: API operation.
3. AP = 2: API operation with escaped characters.

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

### API operation (AP parameter = 1)

When you enable this mode, the following diagram illustrates the data frame structure:





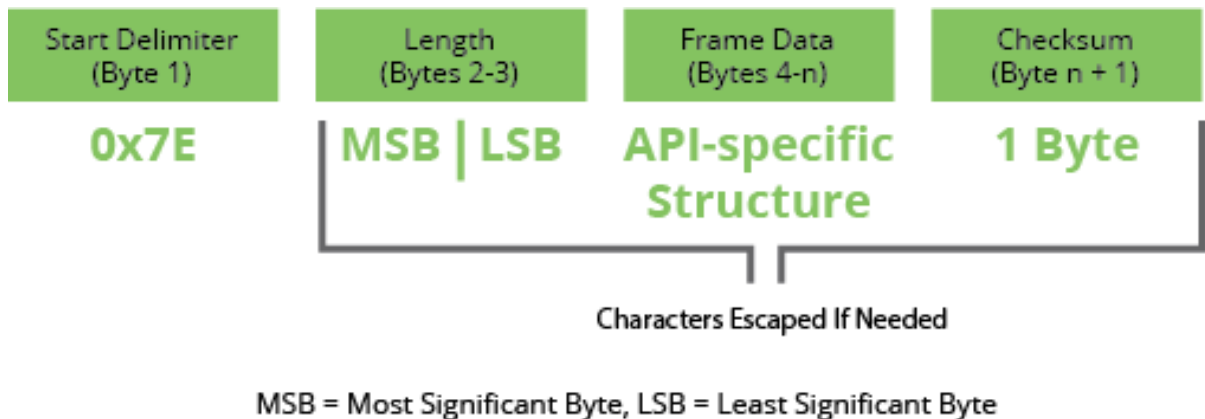
The firmware silently discards any data it receives prior to the start delimiter. If the device does not receive the frame correctly or if the checksum fails, the device discards the frame.

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

API mode 2 is rarely necessary to use. API mode 1 is fully capable of transmitting all characters including those that can be escaped. If you think you need to use API mode 2, contact Digi Support so we can discuss your application. For details, read the following article:

[http://knowledge.digi.com/articles/Knowledge\\_Base\\_Article/Escaped-Characters-and-API-Mode-2](http://knowledge.digi.com/articles/Knowledge_Base_Article/Escaped-Characters-and-API-Mode-2)

When you enable this mode, the following diagram illustrates the data frame structure:



The following paragraphs explain what the fields within the API frame mean.

### Start delimiter

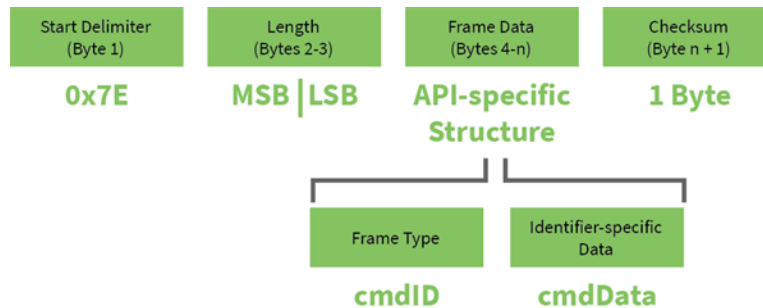
The first byte of a frame consists of a special sequence of bits that indicate the beginning of a data frame. Its value is always 0x7E. This allows an application to easily detect a new incoming frame.

### Length

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

## Frame data

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



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

The Frame Type byte indicates which API messages the data bytes (Identifier-specific data) will contain.

## Checksum

The value 0xFF minus the 8-bit sum of bytes from offset 3 to this byte. Checksum is the last byte of the frame and helps test data integrity.

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

## Calculate and verify checksums

To calculate the checksum of an API frame:

1. Add all bytes of the packet, excluding 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 will equal 0xFF.

## Escaped characters in API frames

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

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

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

To escape a character:

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

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

## API frames

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

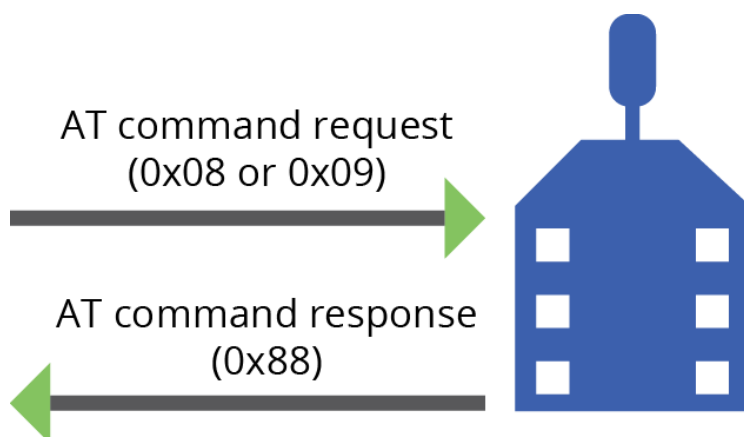
API frame name	API ID
AT Command	0x08
AT Command-Queue Parameter Value	0x09
Transmit Request	0x10
Explicit Addressing Command Frame	0x11
Remote Command Request	0x17
AT Command Response	0x88

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

## API frame exchanges

### AT commands

The following image shows the API frame exchange that takes place at the UART when you send an AT command request to read or set a device parameter. To disable the response, set the frame ID to 0 in the request.



## Transmit and Receive RF data

The following image shows the API exchanges that take place at the UART when a device sends RF data to another device. The device sends the Transmit Status frame at the end of a data transmission unless you set the frame ID to 0 in the transmit request.

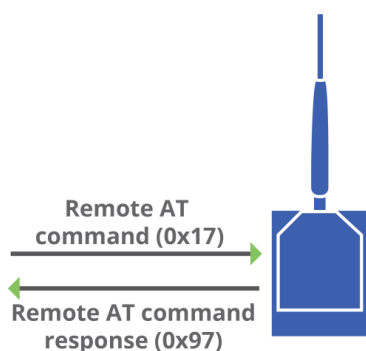
If the packet cannot be delivered to the destination, the Transmit Status frame indicates the cause of failure.

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



## Remote AT commands

The following image shows the API frame exchanges that take place at the UART when you send a remote AT command. If the remote device does not receive the remote command, it does not send a Remote Command Response frame out the UART.



## Code to support future API frames

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

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

```

        case RX_IO_SAMPLE_FRAME:
            //process IO sample frame
            break;

        case NODE_IDENTIFICATION_FRAME:
            //process node identification frame
            break;

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

```

## AT Command frame 0x08

### Description

Use this frame to query or set device parameters on the local device. This API command applies changes after running the command.

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x08
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.
AT command	6-7	Command name: two ASCII characters that identify the AT command.
Parameter value	8-n	If present, indicates the requested parameter value to set the given register. If no characters are present, queries the register.

### Example

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05
Frame type	3	0x08
Frame ID	4	0x52
AT command	5	0x4E (N)
	6	0x48 (H)
Parameter value ( <b>NH2</b> = two network hops)	7	0x02
Checksum	8	0x0D

## AT Command - Queue Parameter Value frame 0x09

### Description

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

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

The device immediately returns the register queries (parameter value readings).

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x09
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.
AT command	6-7	Command name: two ASCII characters that identify the AT command.
Parameter value	8-n	If present, indicates the requested parameter value to set the given register. If no characters are present, queries the register.

### Example

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

---

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

---

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05
Frame type	3	0x09
Frame ID	4	0x01
AT command	5	0x42 (B)
	6	0x44 (D)



Frame fields	Offset	Example
Parameter value ( <b>BD7</b> = 115200 baud)	7	0x07
Checksum	8	0x68

## Transmit Request frame 0x10

### Description

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

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x10
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.
64-bit Destination Address	6-13	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF
Reserved	14-15	Set to 0xFFFE.
Broadcast Radius	16	Sets the maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius is set to the maximum hops value.

Frame fields	Byte	Description
Transmit Options	17	0x01 = Disable ACK 0x02 = Disable network address discovery Set all other bits to 0.
RF Data	18-n	Up to <b>NP</b> bytes per packet. Sent to the destination device.

### Example

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

Frame Fields	Offset	Example
Start Delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x16
Frame Type	3	0x10
Frame ID	4	0x01

Frame Fields	Offset	Example
64-bit Destination Address	MSB 5	0x00
	6	0x13
	7	0xA2
	8	0x00
	9	0x40
	10	0x0A
	11	0x01
	LSB 12	0x27
Reserved	13	0xFF
	14	0xFE
Broadcast Radius	15	0x00
Transmit Options	16	0x00

Frame Fields	Offset	Example
RF Data	17	0x54
	18	0x78
	19	0x44
	20	0x61
	21	0x74
	22	0x61
	23	0x30
	24	0x41
Checksum	25	0x13

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

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

## Explicit Addressing Command frame 0x11

### Description

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

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

Set the 64-bit destination as follows:

- For broadcast transmission: 0x000000000000FFFF
- For unicast transmission: the address of the destination node you want for unicast transmissions

Set the reserved field to 0xFFFE.

Use the **NP** command to read the maximum number of payload bytes.

## Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x11
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.
64-bit Destination Address	6-13	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF
Reserved	14-15	Set to 0xFFFE.
Source Endpoint	16	Source Endpoint for the transmission.
Destination Endpoint	17	Destination Endpoint for the transmission.
Cluster ID	18-19	The Cluster ID that the host uses in the transmission.
Profile ID	20-21	The Profile ID that the host uses in the transmission.
Broadcast Radius	21-22	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius set to the network maximum hops value. If the broadcast radius exceeds the value of <b>NH</b> then the devices use the value of <b>NH</b> as the radius. Only broadcast transmissions use this parameter.

Frame fields	Byte	Description
Transmission Options	23	0x01 = Disable ACK 0x02 = Disable network address discovery Set all other bits to 0.
Data Payload	24-n	Up to <b>NP</b> bytes per packet. Sent to the destination device.

### Example

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

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x1A
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	0x01
	10	0x23
	11	0x84
	LSB12	0x00
Reserved	13	0xFF
	14	0xFE
Source endpoint	15	0xE8
Destination endpoint	16	0xE8
Cluster ID	17	0x00
	18	0x11
Profile ID	19	0xC1
	20	0x05
Broadcast radius	21	0x00
Transmit options	22	0x00

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

## Remote AT Command Request frame 0x17

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x17
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.



Frame fields	Byte	Description
64-bit Destination Address	6-13	MSB first, LSB last. Set to the 64-bit address of the destination device. Broadcast = 0x000000000000FFFF.
Reserved	14-15	Set to 0xFFFFE.
Remote Command Options	16	0x02 = Apply changes on remote. If you do not set this, you must send the <b>AC</b> command for changes to take effect. Set all other bits to 0.
AT Command	17-18	Command name: two ASCII characters that identify the command.
Command Parameter	19-n	If present, indicates the parameter value you request for a given register. If no characters are present, it queries the register. Numeric parameter values are given in binary format.

### Example

The following example sends a remote command to:

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

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x10

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

## AT Command Response frame 0x88

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x88
Frame ID	5	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> , the device does not send a response.
AT Command	6-7	Command name: two ASCII characters that identify the command.
Command Status	8	0 = OK 1 = ERROR 2 = Invalid command 3 = Invalid parameter
Command data	9-n	The register data in binary format. If the host sets the register, the device does not return this field.

### Example

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x05
Frame type	3	0x88
Frame ID	4	0x01
AT command	5	0x42 (B)
	6	0x44 (D)
Command status	7	0x00
Command data		In this example, if <b>BD</b> was queried instead of being set, its current value would be returned here
Checksum	8	0xF0

## Modem Status frame 0x8A

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x8A

Frame fields	Byte	Description
Status	5	0x00 = Hardware reset 0x01 = Watchdog timer reset 0x0B = Network woke up 0x0C = Network went to sleep

### Example

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

Frame fields	Offset	Example
Start Delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x02
Frame Type	3	0x8A
Status	4	0x00
Checksum	5	0x75

## Transmit Status frame 0x8B

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	3	0x8B
Frame ID	4	Identifies the serial interface data frame being reported. If Frame ID = 0 in the associated request frame then no response frame is delivered.
Reserved	5-6	Reserved.
Transmit Retry Count	7	The number of application transmission retries that occur.
Delivery Status	8	0x00 = Success 0x01 = MAC ACK failure 0x02 = Collision avoidance failure 0x21 = Network ACK failure 0x25 = Route not found 0x31 = Internal resource error 0x32 = Internal error 0x74 = Payload too large
Discovery Status	9	0x00 = No discovery overhead 0x02 = Route discovery

### Example

In the following example, the destination device reports that a unicast data transmission was successful and a route discovery occurred using a frame ID of 0x47.

Frame Fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x07

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

## Route Information Packet 0x8D

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x8D
Source event	5	0x11 = NACK 0x12 = Trace route
Length	6	The number of bytes that follow, excluding the checksum. If the length increases, new items have been added to the end of the list for future revisions.
Timestamp	7-10	System timer value on the node generating the Route Information Packet. The timestamp is in microseconds. Only use this value for relative time measurements because the time stamp count restarts about every hour.
ACK timeout count	11	The number MAC ACK timeouts that occur.
Reserved	12-13	Reserved.
Destination address	14-21	The address of the final destination node of this network-based transmission.
Source address	22-29	Address of the source node of this network-based transmission.



Frame fields	Byte	Description
Responder address	30-37	Address of the node that generates this Route Information packet after it sends (or attempts to send) the packet to the next hop (the Receiver node).
Receiver address	38-45	Address of the node that the device sends (or attempts to send) the data packet.

### Example

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

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x2A
Frame type	3	0x8D
Source event	4	0x12
Length	5	0x2B
Timestamp	MSB 6	0x9C
	7	0x93
	8	0x81
	LSB 9	0x7F

Frame fields	Offset	Example
ACK timeout count	10	0x00
Reserved	11	0x00
Reserved	12	0x00
Destination address	MSB 13	0x00
	14	0x13
	15	0xA2
	16	0x00
	17	0x40
	18	0x52
	19	0xAA
	LSB 20	0xAA

Frame fields	Offset	Example
Source address	MSB 21	0x00
	22	0x13
	23	0xA2
	24	0x00
	25	0x40
	26	0x52
	27	0xDD
	LSB 28	0xDD
Responder address	MSB 29	0x00
	30	0x13
	31	0xA2
	32	0x00
	33	0x40
	34	0x52
	35	0xBB
	LSB 36	0xBB

Frame fields	Offset	Example
Receiver address	MSB 37	0x00
	38	0x13
	39	0xA2
	40	0x00
	41	0x40
	42	0x52
	43	0xCC
	LSB 44	0xCC
Checksum	45	0xCE

## Aggregate Addressing Update frame 0x8E

### Description

The following steps generate this frame:

1. Send an **AG** command to a node in API operating mode.
2. The node outputs the Aggregate Addressing Update (0x8E) frame.
3. The node updates its **DH** and **DL** registers.

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x8E

Frame fields	Byte	Description
Format ID	5	Byte reserved to indicate format of additional packet information which may be added in future firmware revisions. In the current firmware revision, this field returns 0x00.
New address	6-13	Address to which <b>DH</b> and <b>DL</b> are being set.
Old address	14-21	Address to which <b>DH</b> and <b>DL</b> were previously set.

### Example

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

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

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

## Receive Packet frame 0x90

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x90
Frame ID	4	Identifies the data frame for the host to correlate with a subsequent ACK. If set to <b>0</b> in Command mode, the device does not send a response.
64-bit source address	5-12	The sender's 64-bit address. MSB first, LSB last.
Reserved	13-14	Reserved.
Receive options	15	Bit field: 0x01 = Packet acknowledged 0x02 = Packet was a broadcast packet Ignore all other bits.
Received data	16-n	The RF data that the device receives.

### Example

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

following frame out its UART.

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x12
Frame type	3	0x90
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	12	0xFF
	13	0xFE
Receive options	14	0x01



Frame fields	Offset	Example
Received data	15	0x52
	16	0x78
	17	0x44
	18	0x61
	19	0x74
	20	0x61
Checksum	21	0x11

## Explicit Rx Indicator frame 0x91

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x91
64-bit source address	5-12	The sender's 64-bit address.
Reserved	13-14	Reserved.
Source endpoint	15	Endpoint of the source that initiates transmission.
Destination endpoint	16	Endpoint of the destination that the message is addressed to.

Frame fields	Byte	Description
Cluster ID	17-18	The Cluster ID that the frame is addressed to.
Profile ID	19-20	The Profile ID that the frame is addressed to.
Receive options	21	Bit field: 0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet Ignore all other bits.
Received data	22-n	Received RF data.

### Example

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

If a device sends the transmission:

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

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x18
Frame type	3	0x91

Frame fields	Offset	Example
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	12	0xFF
	13	0xFE
Source endpoint	14	0xE8
Destination endpoint	15	0xE8
Cluster ID	16	0x00
	17	0x11
Profile ID	18	0xC1
	19	0x05
Receive options	20	0x02

Frame fields	Offset	Example
Received data	21	0x52
	22	0x78
	23	0x44
	24	0x61
	25	0x74
	26	0x61
Checksum	27	0x68

## Data Sample Rx Indicator frame 0x92

### Description

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte (s)	Description
Frame type	4	0x92
64-bit source address	5-12	The sender's 64-bit address.

Frame fields	Byte (s)	Description
Reserved	13-14	Reserved.
Receive options	15	Bit field: 0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet Ignore all other bits
Number of samples	16	The number of sample sets included in the payload. Always set to 1.
Digital channel mask	17-18	Bitmask field that indicates which digital IO lines on the 17 0x1C remote have sampling enabled, if any.
Analog channel mask	19	Bitmask field that indicates which analog IO lines on the remote have sampling enabled, if any.
Digital samples (if included)	20-21	If the sample set includes any digital IO lines (Digital channel mask > 0), these two bytes contain samples for all enabled digital IO lines. DIO lines that do not have sampling enabled return 0. Bits in these two bytes map the same as they do in the Digital channel mask field.
Analog sample	22-23	If the sample set includes any analog IO lines (Analog channel mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from ADO/DIO0 to AD3/DIO3.

### Example

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

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

The complete example frame is:

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x14
Frame-specific data		
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x52
	10	0x2B
	LSB 11	0xAA
Reserved	MSB 12	0xfffe
	LSB 13	0x84
Receive options	14	0x01
Number of samples	15	0x01

Frame fields	Offset	Example
Digital channel mask	16	0x00
	17	0x1C
Analog channel mask	18	0x02
Digital samples (if included)	19	0x00
	20	0x14
Analog sample	21	0x02
	22	0x25
Checksum	23	0xF5

## Node Identification Indicator frame 0x95

### Description

A device receives this frame when:

- a device's **AO** = 0
- it transmits a node identification message to identify itself

The data portion of this frame is similar to a network discovery response; see [ND \(Network Discover\) on page 105](#).

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x95

Frame fields	Byte	Description
64-bit source address	5-12	The sender's 64-bit address.
Reserved	13-14	Reserved.
Receive options	15	Bit field: 0x01 = Packet acknowledged 0x02 = Packet was a broadcast packet 0x40 = Point-multipoint packet 0x80 = Directed broadcast packet 0xC0 = DigiMesh packet Ignore all other bits
Reserved	16-17	Reserved.
64-bit remote address	18-25	Indicates the 64-bit address of the remote device that transmitted the Node Identification Indicator frame.
NI string	26-27	Node identifier string on the remote device. The NI string is terminated with a NULL byte (0x00).
Reserved	28-29	Reserved.
Device type	30	0=Coordinator 1=Normal Mode 2=End Device - For more options, see <a href="#">NO (Network Discovery Options) on page 106</a> .
Source event	31	1=Frame sent by node identification pushbutton event - See <a href="#">D0 (DIO0/AD0) on page 86</a> .



Frame fields	Byte	Description
Digi Profile ID	32-33	Set to Digi's application profile ID.
Digi Manufacturer ID	34-35	Set to Digi's Manufacturer ID.
Digi DD value (optional)	36-39	Reports the <b>DD</b> value of the responding device. To enable this field, use the <b>NO</b> command.
RSSI (optional)	40	Received signal strength indicator. To enable this field, use the <b>NO</b> command.

### Example

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

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

Frame fields	Offset	Example
Start delimiter	0	0x7E
Length	MSB 1	0x00
	LSB 2	0x25
Frame type	3	0x95

Frame fields	Offset	Example
64-bit source address	MSB 4	0x00
	5	0x13
	6	0xA2
	7	0x00
	8	0x40
	9	0x74
	10	0x02
	LSB 11	0xAC
Reserved	12	0xFF
	13	0xFE
Receive options	14	0xC2
Reserved	15	0xFF
	16	0xFE

Frame fields	Offset	Example
64-bit remote address	MSB 17	0x00
	18	0x13
	19	0xA2
	20	0x00
	21	0x40
	22	0x74
	23	0x02
	LSB 24	0xAC
NI string	25	0x20
	26	0x00
Reserved	27	0xFF
	28	0xFE
Device type	29	0x01
Source event	30	0x01
Digi Profile ID	31	0xC1
	32	0x05
Digi Manufacturer ID	33	0x10
	34	0x1E

Frame fields	Offset	Example
Digi DD value (optional)	35	0x00
	36	0x0C
	37	0x00
	38	0x00
RSSI (optional)	39	0x2E
Checksum	40	0x33

## Remote Command Response frame 0x97

### Description

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

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

### Format

The following table provides the contents of the frame. For details on frame structure, see [API frame specifications on page 120](#).

Frame fields	Byte	Description
Frame type	4	0x97
Frame ID	5	This is the same value that is passed in to the request.
64-bit source (remote) address	6-13	The address of the remote device returning this response.
Reserved	14-15	Reserved.
AT commands	16-17	The name of the command.

Frame fields	Byte	Description
Command status	18	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
Command data	19-n	The value of the required register.

### Example

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

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

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

## XBee DigiMesh 2.4 certifications

---

### XBee DigiMesh 2.4 agency certifications - United States

#### United States (FCC)

XBee and XBee-PRO DigiMesh 2.4 RF Modules comply with Part 15 of the FCC rules and regulations. If you are integrating the XBee into another product, you must comply with the labeling requirements, FCC notices and antenna usage guidelines.

To fulfill FCC certification requirements, the OEM must comply with the following regulations:

- The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.
- XBee-PRO RF modules may only be used with antennas that have been tested and approved for use with this module; refer to [XBee DigiMesh 2.4 FCC-approved antennas \(2.4 GHz\) on page 169](#).

#### OEM labeling requirements



**Warning:** As an Original Equipment Manufacturer (OEM), you must include a clearly visible label on the outside of the final product enclosure that displays the following content:

---

Contains FCC ID: OUR-XBEE/OUR-XBEEPRO<sup>1</sup>

The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (i.) this device may not cause harmful interference and (ii.) this device must accept any interference received, including interference that may cause undesired operation.

---

<sup>1</sup>The FCC ID for the XBee is “OUR-XBEE.” The FCC ID for the XBee-PRO is “OUR-XBEEPRO.”

## FCC notices

**IMPORTANT:** The XBee-PRO OEM RF Module has been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

**IMPORTANT:** OEMs must test final product to comply with unintentional radiators (FCC section 15.107 and 15.109) before declaring compliance of their final product to Part 15 of the FCC rules.

**IMPORTANT:** The RF module has been certified for remote and base radio applications. If the module is used for portable applications, the following applies:

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

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect equipment and receiver to outlets on different circuits.
- Consult the dealer or an experienced radio/TV technician for help.

## RF exposure statement

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

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



## XBee DigiMesh 2.4 FCC-approved antennas (2.4 GHz)

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

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

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

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

The antennas in the following tables are approved for use with the XBee/XBee-PRO DigiMesh 2.4 RF module. We do not carry all of these antenna variants. Contact Digi Sales for the available antennas.

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

Part Number	Type (Description)	Gain	Application*	Minimum Separation
A24-HASM-450	Dipole (half-wave articulated RPSMA - 4.5")	2.1 dBi	Fixed/mobile	20 cm
29000095	Dipole (half-wave articulated RPSMA - 4.5")	2.1 dBi	Fixed/mobile	20 cm
A24-HABSM	Dipole (articulated RPSMA)	2.1 dBi	Fixed	20 cm

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

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

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
Yagi class antennas					
A24-Y4NF	Yagi (4-element)	6.0 dBi	Fixed	2 m	

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-Y6NF	Yagi (6-element)	8.8 dBi	Fixed	2 m	1.7 dB
A24-Y7NF	Yagi (7-element)	9.0 dBi	Fixed	2 m	1.9 dB
A24-Y9NF	Yagi (9-element)	10.0 dBi	Fixed	2 m	2.9 dB
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed	2 m	3.9 dB
A24-Y12NF	Yagi (12-element)	12.0 dBi	Fixed	2 m	4.9 dB
A24-Y13NF	Yagi (13-element)	12.0 dBi	Fixed	2 m	4.9 dB
A24-Y15NF	Yagi (15-element)	12.5 dBi	Fixed	2 m	5.4 dB
A24-Y16NF	Yagi (16-element)	13.5 dBi	Fixed	2 m	6.4 dB
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5 dBi	Fixed	2 m	6.4 dB
A24-Y18NF	Yagi (18-element)	15.0 dBi	Fixed	2 m	7.9 dB
Omni-directional class antennas					

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-F2NF	Omni-directional (fiberglass base station)	2.1 dBi	Fixed/mobile	20 cm	
A24-F3NF	Omni-directional (fiberglass base station)	3.0 dBi	Fixed/mobile	20 cm	
A24-F5NF	Omni-directional (fiberglass base station)	5.0 dBi	Fixed/mobile	20 cm	
A24-F8NF	Omni-directional (fiberglass base station)	8.0 dBi	Fixed	2 m	
A24-F9NF	Omni-directional (fiberglass base station)	9.5 dBi	Fixed	2 m	0.2 dB
A24-F10NF	Omni-directional (fiberglass base station)	10.0 dBi	Fixed	2 m	0.7 dB
A24-F12NF	Omni-directional (fiberglass base station)	12.0 dBi	Fixed	2 m	2.7 dB

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-F15NF	Omni-directional (fiberglass base station)	15.0 dBi	Fixed	2 m	5.7 dB
A24-W7NF	Omni-directional (base station)	7.2 dBi	Fixed	2 m	
A24-M7NF	Omni-directional (mag-mount base station)	7.2 dBi	Fixed	2 m	
Panel class antennas					
A24-P8SF	Flat panel	8.5 dBi	Fixed	2 m	1.5 dB
A24-P8NF	Flat panel	8.5 dBi	Fixed	2 m	1.5 dB
A24-P13NF	Flat panel	13.0 dBi	Fixed	2 m	6 dB
A24-P14NF	Flat panel	14.0 dBi	Fixed	2 m	7 dB
A24-P15NF	Flat panel	15.0 dBi	Fixed	2 m	8 dB
A24-P16NF	Flat panel	16.0 dBi	Fixed	2 m	9 dB

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

### High gain antenna summary

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

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

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

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

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

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
Yagi class antennas					
A24-Y4NF	Yagi (4-element)	6.0 dBi	Fixed	2 m	8.1 dB
A24-Y6NF	Yagi (6-element)	8.8 dBi	Fixed	2 m	10.9 dB
A24-Y7NF	Yagi (7-element)	9.0 dBi	Fixed	2 m	11.1 dB

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-Y9NF	Yagi (9-element)	10.0 dBi	Fixed	2 m	12.1 dB
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed	2 m	13.1 dB
A24-Y12NF	Yagi (12-element)	12.0 dBi	Fixed	2 m	14.1 dB
A24-Y13NF	Yagi (13-element)	12.0 dBi	Fixed	2 m	14.1 dB
A24-Y15NF	Yagi (15-element)	12.5 dBi	Fixed	2 m	14.6 dB
A24-Y16NF	Yagi (16-element)	13.5 dBi	Fixed	2 m	15.6 dB
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5 dBi	Fixed	2 m	15.6 dB
A24-Y18NF	Yagi (18-element)	15.0 dBi	Fixed	2 m	17.1 dB
Omni-directional class antennas					
A24-F2NF	Omni-directional (fiberglass base station)	2.1 dBi	Fixed/Mobile	20 cm	4.2 dB

Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-F3NF	Omni-directional (fiberglass base station)	3.0 dBi	Fixed/Mobile	20 cm	5.1 dB
A24-F5NF	Omni-directional (fiberglass base station)	5.0 dBi	Fixed/Mobile	20 cm	7.1 dB
A24-F8NF	Omni-directional (fiberglass base station)	8.0 dBi	Fixed	2 m	10.1 dB
A24-F9NF	Omni-directional (fiberglass base station)	9.5 dBi	Fixed	2 m	11.6 dB
A24-F10NF	Omni-directional (fiberglass base station)	10.0 dBi	Fixed	2 m	12.1 dB
A24-F12NF	Omni-directional (fiberglass base station)	12.0 dBi	Fixed	2 m	14.1 dB
A24-F15NF	Omni-directional (fiberglass base station)	15.0 dBi	Fixed	2 m	17.1 dB



Part Number	Type (Description)	Gain	Application*	Minimum Separation	Required Cable Loss
A24-W7NF	Omni-directional (base station)	7.2 dBi	Fixed	2 m	9.3 dB
A24-M7NF	Omni-directional (mag-mount base station)	7.2 dBi	Fixed	2 m	9.3 dB
Panel class antennas					
A24-P8SF	Flat panel	8.5 dBi	Fixed	2 m	8.6 dB
A24-P8NF	Flat panel	8.5 dBi	Fixed	2 m	8.6 dB
A24-P13NF	Flat panel	13.0 dBi	Fixed	2 m	13.1 dB
A24-P14NF	Flat panel	14.0 dBi	Fixed	2 m	14.1 dB
A24-P15NF	Flat panel	15.0 dBi	Fixed	2 m	15.1 dB
A24-P16NF	Flat panel	16.0 dBi	Fixed	2 m	16.1 dB
A24-P19NF	Flat panel	19.0 dBi	Fixed	2 m	19.1 dB
<p>* If you are using the RF module in a portable application or if the module is used in a handheld device and the antenna is less than 20 cm from the human body when the device is in operation: The integrator may be responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1091 and FCC Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields, OET Bulletin and Supplement C. See the note under FCC notices for more information. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person.</p>					

## XBee DigiMesh 2.4 agency certifications - Australia (C-Tick)

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

### Labeling requirements

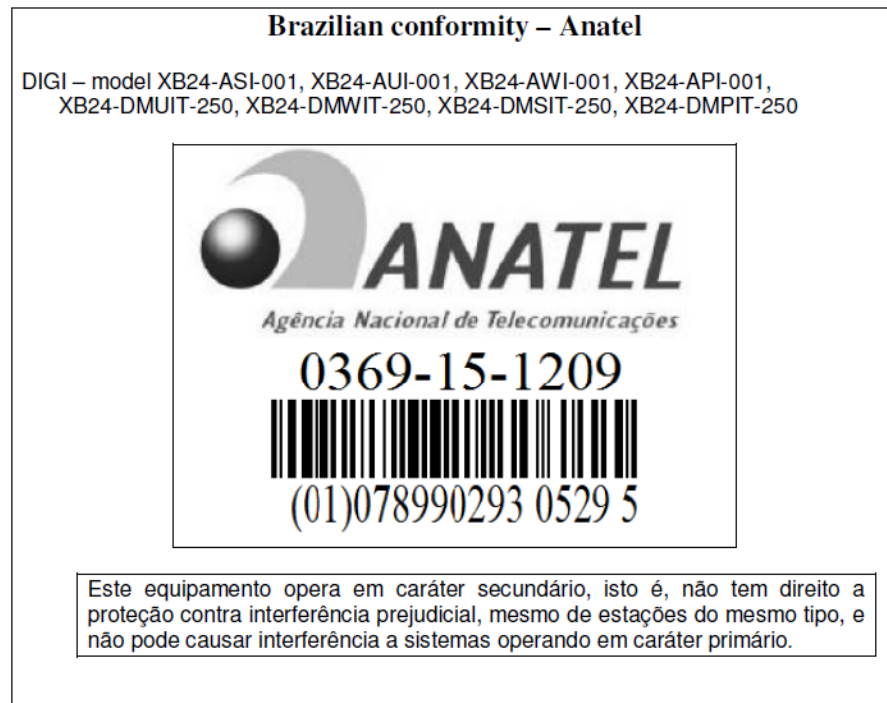
In order to place a C-Tick mark on an end product, a company must comply with 1 or 2 below:

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

Contact Digi for questions about locating a contact in Australia.

## XBee DigiMesh 2.4 agency certifications - Brazil Agência Nacional de Telecomunicações (Anatel)

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



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



## XBee DigiMesh 2.4 agency certifications - Industry Canada (IC)

### Labeling requirements

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

Contains Model XBee Radio, IC: 4214A-XBEE

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

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

## XBee DigiMesh 2.4 agency certifications - European Telecommunications Standards Institute (ETSI)

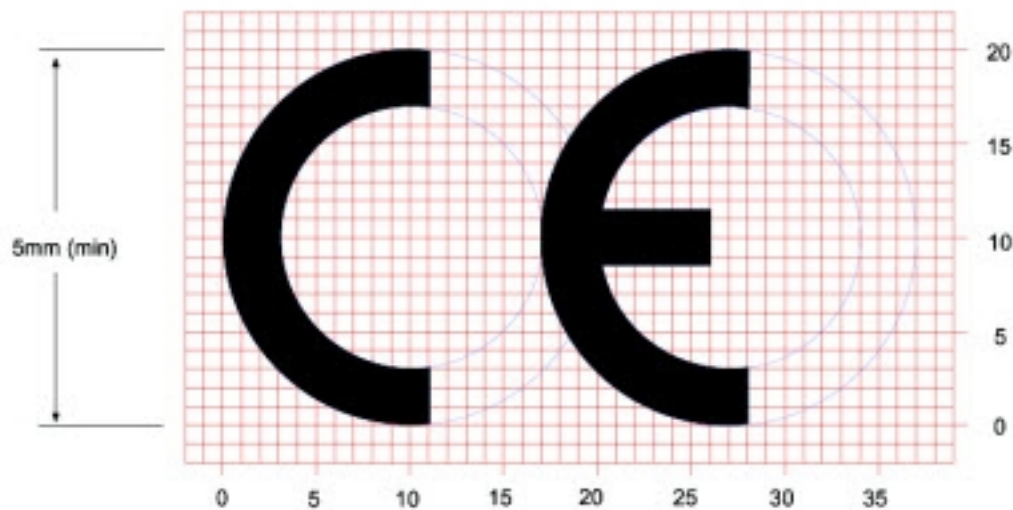
The XBee-PRO RF Module is certified for use in several European countries; for a complete list, go to [www.digi.com](http://www.digi.com).

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

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

### OEM labeling requirements

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



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

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

## Restrictions

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

France imposes restrictions on the 2.4 GHz band. Go to [www.art-telcom.fr](http://www.art-telcom.fr) or contact Digi for more information.

## Declarations of conformity

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

---

**Note** Digi does not list the entire set of standards that must be met for each country. Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. For more information relating to European compliance of an OEM product incorporating the XBee-PRO RF Module, contact Digi, or refer to the following web sites:

- CEPT ERC 70-03E, technical requirements, European restrictions and general requirements: available at [www.ero.dk/](http://www.ero.dk/).
  - R&TTE Directive, equipment requirements and market placement: available at [www.ero.dk/](http://www.ero.dk/).
- 

## Approved antennas

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

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

### Yagi

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

### Omni-directional

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

### Flat panel

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

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

---

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

---

### **Dipole**

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

### **Chip antenna**

-1.5 dBi.

### **Attached monopole whip**

1.5 dBi.

### **Integrated PCB antenna**

-0.5 dBi

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

## **XBee DigiMesh 2.4 agency certifications - Japan (Telec)**

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

### **Labeling requirements**

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

R201WW07215214 (XBee)

R201WW08215111 (XBee-PRO)