XBee®/XBee-PRO SX

Radio Frequency (RF) Module

User Guide

Revision history—90001477

Revision	Date	Description
J	January 2019	Updated the Brazilian certifications.
K	January 2020	Added IFETEL certifications.
L	November 2020	Added the **C8** and **R?** commands. Added the **Sleep Immediate** and **Synchronous pulse** sections. Added information on randomizing a single I/O sample. Revised **BC**.
M	July 2021	Added safety instructions.
N	September 2021	Updated Mexican certifications.

Trademarks and copyright

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

© 2022 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 product warranty information, go to the following website:

www.digi.com/howtobuy/terms

Customer support

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

Product name and model

Product serial number (s)

Firmware version

Operating system/browser (if applicable)

Logs (from time of reported issue)

Trace (if possible)

Description of issue

Steps to reproduce

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

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (XBee®/XBee-PRO SX RF Module User Guide, 90001477 N) in the subject line of your email.

Contents

XBee®/XBee-PRO SX RF Module User Guide

Applicable firmware and hardware 14
Safety instructions 14
XBee modules 14

Technical specifications

Regulatory conformity summary 16
Power requirements 16
Networking specifications 16
Performance specifications 17
General specifications 18
GPIO specifications 19

Hardware

Mechanical drawings 21
Pin signals 22
Pin connection recommendations 24

Getting started with the XBee/XBee-PRO SX RF Module Development Kit

XBee SX Development Board 26
Connect XBee-PRO SX development boards to a PC 27
Connect the XBIB-U-SS development board to a PC 28
Configure the device using XCTU 28
Configure the devices for a range test 28
Perform a range test 29
Mesh network demonstration 30
Software libraries 32
XBee Network Assistant 32
XBee Multi Programmer 33

Modes

Transparent and API operating modes 35
Transparent operating mode 35

API operating mode 35
Comparing Transparent and API modes 35
Modes of operation 36
Receive mode 36
Transmit mode 37
Sleep mode 37
Command mode 37

Sleep modes

About sleep modes 41
Asynchronous modes 41
Synchronous modes 41
Normal mode 41
Asynchronous pin sleep mode 42
Asynchronous cyclic sleep mode 42
Asynchronous cyclic sleep with pin wake up mode 42
Synchronous sleep support mode 42
Synchronous cyclic sleep mode 43
The sleep timer 43
Indirect messaging and polling 43
Indirect messaging 43
Polling 44
Sleeping routers 44
Sleep coordinator sleep modes in the DigiMesh network 44
Synchronization messages 45
Become a sleep coordinator 47
Preferred sleep coordinator option 47
Resolution criteria and selection option 47
Commissioning Pushbutton option 48
Auto-early wake-up sleep option 50
Select sleep parameters 50
Sleep immediate 50
Synchronous pulse 51
Configuration 51
Accuracy 51
Start a sleeping synchronous network 51
Add a new node to an existing network 52
Change sleep parameters 53
Rejoin nodes that lose sync 54
Diagnostics 54
Query sleep cycle 54
Sleep status 55
Missed sync messages command 55
Sleep status API messages 55

Networking methods

The MAC and PHY layers 57
64-bit addresses 57
Make a unicast transmission 58
Make a broadcast transmission 58
Delivery methods 58
Point to Point / Point to Multipoint (P2MP) 58

Repeater/directed broadcast 59
DigiMesh networking 59

Serial communication

UART data flow 64
Serial data 64
SPI signals 64
Signal description 65
Slave mode characteristics 65
Full duplex operation 66
Low power operation 66
Configuration considerations 67
SPI and API mode 67
SPI parameters 67
Serial port selection 67
Serial receive buffer 68
Serial transmit buffer 68
UART flow control 68
CTS flow control 68
RTS flow control 68

AT commands

Special commands 71
AC (Apply Changes) 71
FR (Software Reset) 71
RE (Restore Defaults) 71
WR (Write) 71
MAC/PHY commands 72
AF (Available Frequencies) 72
CM (Channel Mask) 72
MF (Minimum Frequencies) 73
HP (Preamble ID) 74
ID (Network ID) 74
MT (Broadcast Multi-Transmits) 74
BR (RF Data Rate) 75
PL (TX Power Level) 75
RR (Unicast Mac Retries) 76
ED (Energy Detect) 77
Diagnostic commands - MAC statistics and timeouts 77
BC (Bytes Transmitted) 77
DB (Last Packet RSSI) 77
ER (Receive Count Error) 78
GD (Good Packets Received) 78
EA (MAC ACK Failure Count) 78
TR (Transmission Failure Count) 79
UA (Unicasts Attempted Count) 79
%H (MAC Unicast One Hop Time) 79
%8 (MAC Broadcast One Hop Time) 79
Network commands 79
CE (Routing / Messaging Mode) 80
C8 (Compatibility Options) 80
BH (Broadcast Hops) 81

NH (Network Hops) 81
MR (Mesh Unicast Retries) 81
NN (Network Delay Slots) 81
Addressing commands 82
SH (Serial Number High) 82
SL (Serial Number Low) 82
DH (Destination Address High) 82
DL (Destination Address Low) 83
TO (Transmit Options) 83
NI (Node Identifier) 83
NT (Network Discovery Back-off) 84
NO (Network Discovery Options) 84
CI (Cluster ID) 85
Diagnostic - addressing commands 85
N? (Network Discovery Timeout) 85
R? (Region) 85
Addressing discovery/configuration commands 86
AG (Aggregator Support) 86
DN (Discover Node) 86
ND (Network Discover) 87
FN (Find Neighbors) 88
Security commands 88
EE (Encryption Enable) 88
KY (AES Encryption Key) 89
KZ (Remote AT Command Password) 89
Serial interfacing commands 90
BD (Interface Data Rate) 90
NB (Parity) 91
SB (Stop Bits) 92
RO (Packetization Timeout) 92
FT (Flow Control Threshold) 92
AP (API Enable) 92
AO (API Options) 93
I/O settings commands 93
D0 (DIO0/AD0) 93
D1 (DIO1/AD1) 94
D2 (DIO2/AD2) 94
D3 (DIO3/AD3) 95
D4 (DIO4) 95
D5 (DIO5/ASSOCIATED_INDICATOR) 96
D6 (DIO6/RTS) 96
D7 (DIO7/CTS) 97
D8 (DIO8/SLEEP_REQUEST) 97
D9 (DIO9/ON_SLEEP) 98
P0 (DIO10/RSSI/PWM0 Configuration) 98
P1 (DIO11/PWM1 Configuration) 99
P2 (DIO12 Configuration) 99
P3 (DIO13/DOUT) 100
P4 (DIO14/DIN) 100
P5 (DIO15/SPI_MISO) 101
P6 (SPI_MOSI) 101
P7 (DIO17/SPI_SSEL) 102
P8 (DIO18/SPI_SCLK) 102
P9 (DIO19/SPI_ATTN) 103
PD (Pull Direction) 103

PR (Pull-up/Down Resistor Enable) 103
M0 (PWM0 Duty Cycle) 104
M1 (PWM1 Duty Cycle) 105
LT (Associated LED Blink Time) 105
RP (RSSI PWM Timer) 105
I/O sampling commands 106
AV (Analog Voltage Reference) 106
IC (DIO Change Detect) 106
IF (Sleep Sample Rate) 107
IR (Sample Rate) 107
IS (Immediate Sample) 108
TP (Board Temperature) 108
%V (Voltage Supply Monitoring) 108
I/O line passing commands 108
IU (I/O Output Enable) 109
IA (I/O Input Address) 109
T0 (D0 Timeout) 109
T1 (D1 Output Timeout) 110
T2 (D2 Output Timeout) 110
T3 (D3 Output Timeout) 110
T4 (D4 Output Timeout) 110
T5 (D5 Output Timeout) 111
T6 (D6 Output Timeout) 111
T7 (D7 Output Timeout) 111
T8 (D8 Timeout) 111
T9 (D9 Timeout) 112
Q0 (P0 Timeout) 112
Q1 (P1 Timeout) 112
Q2 (P2 Timeout) 112
Q3 (P3 Timeout) 113
Q4 (P4 Timeout) 113
PT (PWM Output Timeout) 113
Sleep commands 113
SM (Sleep Mode) 113
SO (Sleep Options) 114
SN (Number of Cycles Between ON_SLEEP) 115
SP (Sleep Time) 115
ST (Wake Time) 116
WH (Wake Host Delay) 116
Diagnostic - sleep status/timing commands 116
SS (Sleep Status) 116
OS (Operating Sleep Time) 117
OW (Operating Wake Time) 117
MS (Missed Sync Messages) 118
SQ (Missed Sleep Sync Count) 118
Command mode options 118
CC (Command Sequence Character) 118
CT (Command Mode Timeout) 119
CN (Exit Command Mode) 119
GT (Guard Times) 119
Firmware version/information commands 119
VR (Firmware Version) 119
HV (Hardware Version) 120
HS (Hardware Series) 120
DD (Device Type Identifier) 120

NP (Maximum Packet Payload Bytes) 120
CK (Configuration CRC) 120

Operate in API mode

API mode overview 123
Use the AP command to set the operation mode 123
API frame format 123
API operation (AP parameter = 1) 123
API operation with escaped characters (AP parameter = 2) 124
Length field 125
Frame data 125
Calculate and verify checksums 125

API frames

API frame exchanges 128
AT commands 128
Transmit and Receive RF data 128
Remote AT commands 129
Device Registration 129
Code to support future API frames 130
64-bit Transmit Request - 0x00 131
Description 131
Format 131
Examples 132
Local AT Command Request - 0x08 133
Description 133
Format 133
Examples 133
Queue Local AT Command Request - 0x09 135
Description 135
Examples 135
Transmit Request - 0x10 137
Description 137
Transmit options bit field 138
Examples 138
Explicit Addressing Command Request - 0x11 140
Description 140
64-bit addressing 140
Reserved endpoints 140
Reserved cluster IDs 140
Reserved profile IDs 140
Transmit options bit field 141
Examples 142
Remote AT Command Request - 0x17 144
Description 144
Format 144
Examples 145
Secure Remote AT Command Request frame - 0x18 147
Modem Status - 0x8A 150
Description 150
Examples 151
Extended Transmit Status - 0x8B 152

Description ... 152
Route Information - 0x8D ... 154
Description ... 154
Format ... 154
Examples ... 155
Aggregate Addressing Update - 0x8E ... 156
Description ... 156
Examples ... 156
64-bit Receive Packet - 0x80 ... 158
Description ... 158
Format ... 158
Examples ... 159
Local AT Command Response - 0x88 ... 160
Description ... 160
Examples ... 160
Transmit Status - 0x89 ... 162
Description ... 162
Delivery status codes ... 163
Examples ... 163
Receive Packet - 0x90 ... 164
Description ... 164
Examples ... 165
Explicit Receive Indicator - 0x91 ... 166
Description ... 166
Examples ... 167
Node Identification Indicator - 0x95 ... 168
Description ... 168
Examples ... 169
Remote AT Command Response- 0x97 ... 171
Description ... 171
Examples ... 172

Work with networked devices

Network commissioning and diagnostics ... 175
Local configuration ... 175
Remote configuration ... 175
Send a remote command ... 175
Apply changes on remote devices ... 175
Remote command response ... 175
Secure remote AT commands ... 176
How secure remote AT commands protect the network ... 176
Password protection ... 176
Password recovery ... 176
Secure remote command control ... 177
Security protections ... 177
Secure a network ... 177
Software downgrades ... 178
Establish and maintain network links ... 178
Build aggregate routes ... 178
DigiMesh routing examples ... 179
Replace nodes ... 179
Test links in a network - loopback cluster ... 180
Test links between adjacent devices ... 180
Example ... 181

RSSI indicators ... 182
Discover all the devices on a network ... 182
Trace route option ... 183
NACK messages ... 184
The Commissioning Pushbutton ... 184
Associate LED ... 186

Monitor I/O lines

Pin configurations ... 188
Queried sampling ... 189
Periodic I/O sampling ... 190
Detect digital I/O changes ... 190

I/O line passing

Configuration example ... 192

General Purpose Flash Memory

General Purpose Flash Memory ... 196
Access General Purpose Flash Memory ... 196
General Purpose Flash Memory commands ... 197
PLATFORM_INFO_REQUEST (0x00) ... 197
PLATFORM_INFO (0x80) ... 197
ERASE (0x01) ... 198
ERASE_RESPONSE (0x81) ... 198
WRITE (0x02) and ERASE_THEN_WRITE (0x03) ... 199
WRITE _RESPONSE (0x82) and ERASE_THEN_WRITE_RESPONSE (0x83) ... 200
READ (0x04) ... 200
READ_RESPONSE (0x84) ... 201
FIRMWARE_VERIFY (0x05) and FIRMWARE_VERIFY_AND_INSTALL(0x06) ... 201
FIRMWARE_VERIFY_RESPONSE (0x85) ... 202
FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86) ... 202

Update the firmware over-the-air

Over-the-air firmware updates ... 205
Distribute the new application ... 205
Example ... 205
Verify the new application ... 206
Install the application ... 206
Important considerations ... 206

Regulatory information

FCC (United States) ... 208
OEM labeling requirements ... 208
FCC notices ... 208
RF exposure statement ... 210
FCC antenna certifications ... 210
XBee-PRO SX antenna options (30 dBm maximum RF power) ... 211

XBee SX antenna options (13 dBm maximum RF power) 216
FCC publication 996369 related information 221
ISED (Innovation, Science and Economic Development Canada) 223
Labeling requirements 223
Transmitters for detachable antennas 223
Detachable antennas 223
ACMA (Australia) 224
Power requirements 224
RSM (New Zealand) 224
Power requirements 224
Brazil (Anatel) 225
ANATEL Brazil for XBee-PRO SX radio products (XBPSX) 225
SX ANATEL Brazil for XK9X-DMS-1 XBee SX RF Module Dev Kit (XK9X-DMS-1) 226
ANATEL Brazil for XBee SX radio products (XBSX) 227
IFETEL (Mexico) 228
OEM labeling requirements 228

PCB design and manufacturing

Recommended footprint and keepout 230
Design notes 232
Host board design 232
Improve antenna performance 233
RF pad version 233
Recommended solder reflow cycle 234
Flux and cleaning 235
Rework 235

XBee®/XBee-PRO SX RF Module User Guide

The XBee/XBee-PRO SX RF Module is an embedded radio frequency (RF) device that provides wireless connectivity to end-point devices in mesh networks.

The XBee/XBee-PRO SX RF Module delivers up to 1 Watt of RF power and has excellent receive sensitivity, low operating current, and exceptional performance in low power modes. The module's frequency hopping technology offers advanced interference immunity, affording long range data throughput even in challenging RF environments. The XBee/XBee-PRO SX RF Module uses a microprocessor that supports host communication through Serial Peripheral Interface (SPI) or universal asynchronous receiver/transmitter (UART), as well as digital, analog, and pulse width modulation (PWM) lines for interfacing with peripherals.

Applicable firmware and hardware 14
Safety instructions 14

Applicable firmware and hardware

This manual supports the following firmware:

- 0x900X USA and Canada, XBee/XBee-PRO SX
- 0x920x Australia, XBee/XBee-PRO SX
- 0x930X Brazil XBee/XBee-PRO SX
- 0x960X New Zealand, XBee SX only

Note The New Zealand firmware only works with non-PRO hardware.

It supports the following hardware:

- XBee/XBee-PRO SX RF Module

Safety instructions

XBee modules

- The XBee radio module cannot be guaranteed operation due to the radio link and so should not be used for interlocks in safety critical devices such as machines or automotive applications.
- The XBee radio module have not been approved for use in (this list is not exhaustive):
  - medical devices
  - nuclear applications
  - explosive or flammable atmospheres
- There are no user serviceable components inside the XBee radio module. Do not remove the shield or modify the XBee in any way. Modifications may exclude the module from any warranty and can cause the XBee radio to operate outside of regulatory compliance for a given country, leading to the possible illegal operation of the radio.
- Use industry standard ESD protection when handling the XBee module.
- Take care while handling to avoid electrical damage to the PCB and components.
- Do not expose XBee radio modules to water or moisture.
- Use this product with the antennas specified in the XBee module user guides.
- The end user must be told how to remove power from the XBee radio module or to locate the antennas 20 cm from humans or animals.

Technical specifications

Regulatory conformity summary 16
Power requirements 16
Networking specifications 16
Performance specifications 17
General specifications 18
GPIO specifications 19

Regulatory conformity summary

This table describes the agency approvals for the devices.

The Australia, New Zealand and Brazil certified devices are separate variants from the USA and Canada approved device.

Country	Approval	
	XBee-PRO SX	XBee SX
United States	FCC ID: MCQ-XBPSX	FCC ID: MCQ-XBSX
Canada	IC: 1846A-XBPSX	IC: 1846A-XBSX
Australia	RCM	RCM
New Zealand		R-NZ
Brazil	ANATEL: 05774-16-01209	ANATEL: 05776-16-01209
Mexico	RCPDIXB19-2288	RCPDIXB19-1819

Power requirements

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

Specification	Condition	XBee SX value	XBee-PRO SX value
Supply voltage range		2.4 to 3.6 VDC	2.6 to 3.6 VDC
Typical supply voltage		3.3 V	
Receive current	VCC = 3.3 V	40 mA	40 mA
Transmit current	VCC = 3.3 V	55 mA @ 13 dBm	900 mA @ 30 dBm
	VCC = 3.3 V	45 mA @ 10 dBm	640 mA @ 27 dBm
	VCC = 3.3 V	35 mA @ 0 dBm	350 mA @ 21.5 dBm
Sleep current	VCC 3.3 V, temperature = 25 °C	2.5 µA	2.5 µA

Networking specifications

The following table provides the networking specifications for the device.

Specification	Value
Modulation	Gaussian Frequency Shift Keying (GFSK)
Spreading technology	Frequency Hopping Spread Spectrum (FHSS)
Supported network	Peer-to-peer (master/slave relationship not required), point-to-point/point-to-multipoint, mesh

Specification	Value
topologies (software selectable)	
Encryption	Optional 256-bit Advanced Encryption Standard (AES) encryption which may be operated in either cipher block chaining (CBC) mode or counter (CTR) mode. Use the **EE** command to enable encryption. Use the **KY** command to set the encryption key.

Performance specifications

The following table describes the performance specifications for the devices.

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

Specification	Condition	XBee value	XBee-PRO value
Frequency range		ISM 902 to 928 MHz	
RF data rate (software selectable)	Low data rate	10 kb/s	
	Middle data rate	110 kb/s	
	High data rate	250 kb/s	
Transmit power (software selectable)		Up to 13 dBm	Up to 30 dBm[1]
	New Zealand specific	Up to 13 dBm, at the high data rate (**BR** 2) it is capped at 7.5 dBm	
Maximum data throughput	High data rate	120 kb/s	
Channels		10 hopping sequences share 50 frequencies	
Available channel frequencies	Low/middle data rate	101[2]	
	High data rate	50	

[1]30 dBm typical at 3.3 V and above. Maximum power will decrease at lower voltages. For more information on adjustable power levels, see PL (TX Power Level).
[2]The device hops on 50 channels selected using the **CM** command, from 101 available frequencies. For more details, see CM (Channel Mask).

Specification	Condition	XBee value	XBee-PRO value
Rural range line of sight	Low data rate	Up to 14.5 km (9 mi)	Up to 105 km (65 mi)[1]
Urban range line of sight	Low data rate	Up to 2.5 km (1.5 mi)	Up to 18 km (11 mi)[2]
Receiver sensitivity	Low data rate	-113 dBm	
	Middle data rate	-106 dBm	
	High data rate	-103 dBm	
Receiver IF selectivity	Low data rate ± 250 kHz	40 dB	
	Low data rate ± 500 kHz	50 dB	
	Middle data rate ± 250 kHz	30 dB	
	Middle data rate ± 500 kHz	40 dB	
	High data rate ± 500 kHz	30 dB	
	High data rate ± 1000 kHz	45 dB	
Receiver RF selectivity	Below 900 MHz and above 930 MHz	> 50 dB	
UART data rate (software selectable)		1200 - 921600 baud	
SPI clock rate		Up to 6 Mb/s	

General specifications

The following table describes the general specifications for the devices.

Specification	Value
Dimensions	3.38 x 2.21 x 0.32 cm (1.33 x 0.87 x 0.125 in)
Weight	3 g

[1]We estimate rural ranges based on a 14.5 km (9 mi) range test with dipole antennas.

[2]Range estimated assuming that the urban noise floor is approximately 15 dB higher than rural. The actual range depends on the setup and level of interference in your location.

Specification	Value
Restriction of Hazardous Substances (RoHS)	Compliant
Manufacturing	ISO 9001:2008 registered standards
Host interface connector	37 castellated SMT pads
Antenna connector options	U.FL or RF pad
Antenna impedance	50 Ω unbalanced
Maximum input RF level at antenna port	6 dBm
Operating temperature	-40 °C to 85 °C
Digital I/O	13 I/O lines, 5 output lines
Analog-to-digital converter (ADC)	4 10-bit analog inputs
Pulse width modulator (PWM)	2 outputs

GPIO specifications

The following table provides the electrical specifications for the GPIO pads.

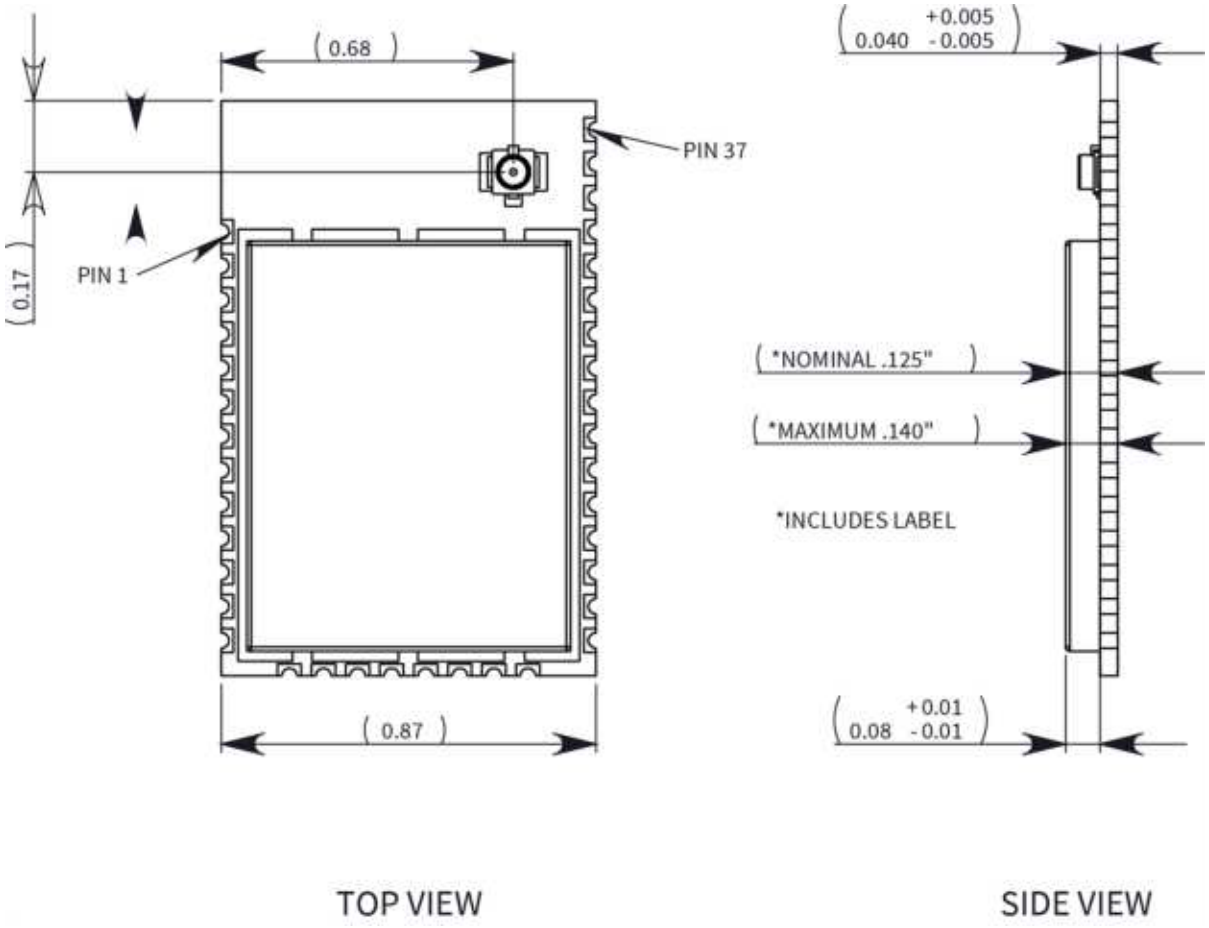
GPIO electrical specification	Value
Voltage - supply	2.4 - 3.6 V
Low Schmitt switching threshold	0.3 * VCC
High Schmitt switching threshold	0.7 * VCC
Input current for logic 0	-0.1 µA
Input current for logic 1	0.1 µA
Input pull-up resistor value	40 kΩ
Input pull-down resistor value	40 kΩ
Output voltage for logic 0	0.05 * VCC
Output voltage for logic 1	0.95 * VCC
Output source/sink current	1 mA
Total output current (for GPIO pads)	20 mA

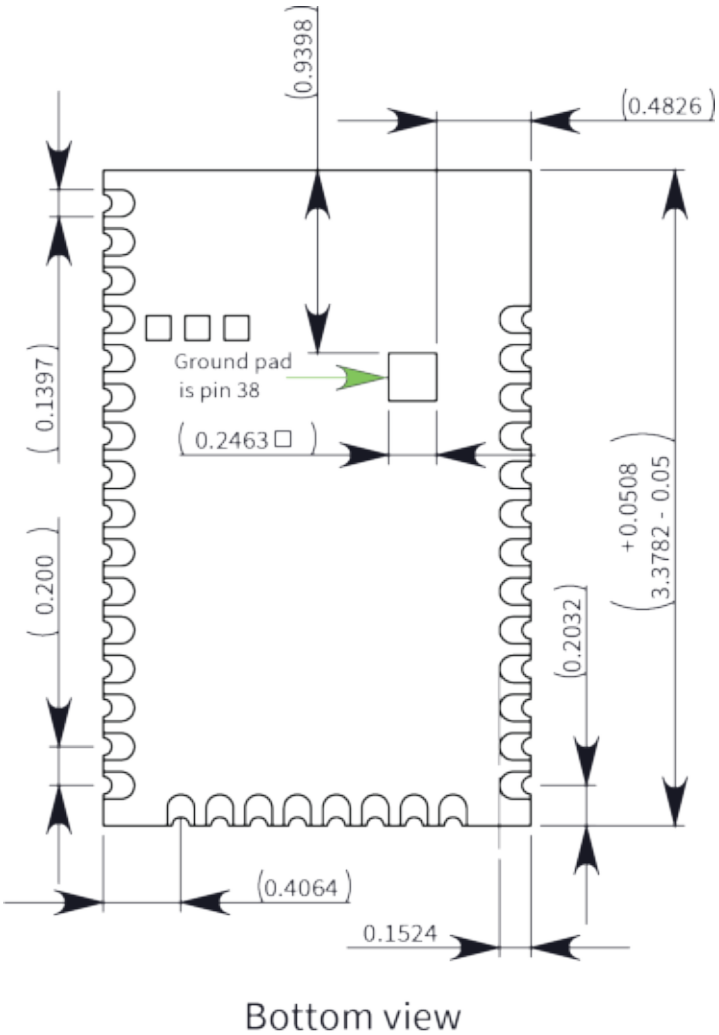
Hardware

Mechanical drawings 21
Pin signals 22

Mechanical drawings

The following figures show the XBee/XBee-PRO SX RF Module mechanical drawings. All dimensions are in centimeters. The XBee/XBee-PRO SX RF Module differs from other surface-mount XBee modules. It has an additional ground pad on the underside of the module used for heat dissipation. For more details, see PCB design and manufacturing.





Pin signals

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

Pin	Name	I/O	Default state	Function
1	GND	-	-	Ground
2	VCC	I	-	Power supply
3	DIO13/DOUT	I/O	Output	GPIO / UART data out
4	DIN/DIO14/$\overline{\text{CONFIG}}$	I/O	Input	GPIO / UART data in
5	DIO12	I/O	Disabled	GPIO

Pin	Name	I/O	Default state	Function
6	$\overline{\text{RESET}}$	I	-	Drive low to reset device. Do not drive pin high; pin may only be driven open drain or low. Pin has an internal 20k pullup resistor. The minimum reset pulse time is 100 ns.
7	DIO10/RSSI/PWM0	I/O	Output	GPIO / RX Signal Strength Indicator
8	DIO11/PWM1	I/O	Disabled	GPIO / Pulse Width Modulator
9	[Reserved]	-	-	Do not connect
10	DIO8/$\overline{\text{DTR}}$ /SLEEP_RQ	I/O	Input	GPIO / Pin Sleep Control line ($\overline{\text{DTR}}$ on the development board)
11	GND	-	-	Ground
12	DO19/SPI_$\overline{\text{ATTN}}$	O	Output	GPO / Serial Peripheral Interface (SPI) Attention or UART Data Present indicator
13	GND	-	-	Ground
14	DO18/SPI_CLK	I/O[1]	Input	GPO / SPI clock
15	DO17/SPI_$\overline{\text{SSEL}}$	I/O[2]	Input	GPO / SPI not select
16	DO16/SPI_MOSI	I/O[3]	Input	GPO / SPI Data In
17	DO15/SPI_MISO	O	Output	GPO/SPI Data Out Tri-stated when SPI_$\overline{\text{SSEL}}$ is high
18	[Reserved]	-	-	Do not connect
19	[Reserved]	-	-	Do not connect
20	[Reserved]	-	-	Do not connect
21	[Reserved]	-	-	Do not connect
22	GND	-	-	Ground
23	[Reserved]	-	-	Do not connect
24	DIO4	I/O	Disabled	GPIO
25	DIO7/$\overline{\text{CTS}}$	I/O	Output	GPIO / UART Clear to Send Flow Control
26	DIO9/ON/$\overline{\text{SLEEP}}$	I/O	Output	GPIO / Module Sleep Status Indicator
27	V_{REF}	-	-	Feature not supported on this device. Used on

[1]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.
[2]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.
[3]Pins 14-16 are inputs in SPI mode only. In general purpose I/O pin mode you can only use them as digital outputs.

Pin	Name	I/O	Default state	Function
				other XBee devices for analog voltage reference.
28	DIO5/ASSOC	I/O	Output	GPIO / Associate Indicator
29	DIO6/$\overline{\text{RTS}}$	I/O	Disabled	GPIO / UART Request to Send Flow Control
30	DIO3/AD3	I/O	Disabled	GPIO / Analog Input
31	DIO2/AD2	I/O	Disabled	GPIO / Analog Input
32	DIO1/AD1	I/O	Disabled	GPIO / Analog Input
33	DIO0/AD0	I/O	Input	GPIO / Analog Input / Commissioning Pushbutton
34	[Reserved]	-	-	Do not connect
35	GND	-	-	Ground
36	RF_PAD	I/O	-	RF connection for RF pad variant
37	[Reserved]	-	-	Do not connect
38	GND	-	-	Ground pad for heat transfer to host PCB. Located on the underside of the XBee module.

Pin connection recommendations

The only required pin connections are VCC, GND, DOUT and DIN. To support serial firmware updates, you should connect VCC, GND, DOUT, DIN, RTS, and SLEEP (DTR).

Getting started with the XBee/XBee-PRO SX RF Module Development Kit

This section provides getting started instructions if you have an XBee/XBee-PRO SX RF Module development kit:

- USA XK9X-DMS-0
- Australia XK9X-DMS-2
- Brazil XK9X-DMS-1

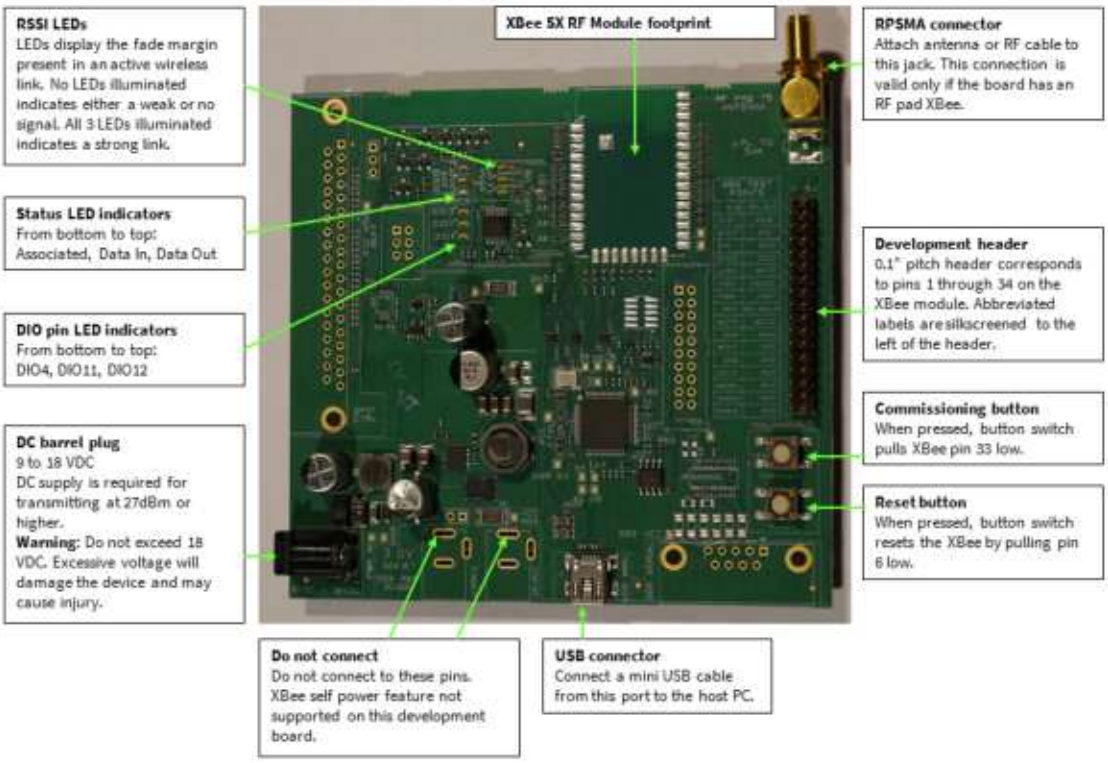
This section describes how to set up an XBee network and adjust the XBee/XBee-PRO SX RF Module settings.

This section provides information on the Development Board for the XBee/XBee-PRO SX RF Module and getting started instructions if you have an XBee/XBee-PRO SX RF Module Development Kit.

XBee SX Development Board 26
Connect XBee-PRO SX development boards to a PC 27
Connect the XBIB-U-SS development board to a PC 28
Configure the device using XCTU 28
Configure the devices for a range test 28
Perform a range test 29
Mesh network demonstration 30
Software libraries 32
XBee Network Assistant 32
XBee Multi Programmer 33

XBee SX Development Board

The following figure shows the XBee SX development board with onboard XBee-PRO SX RF pad module and the table that follows explains the callouts in the picture.



Number	Item	Description
1	RSSI LEDs	LEDs display the fade margin present in an active wireless link. No LEDs illuminated indicates either a weak or no signal. All three LEDs illuminated indicates a strong link.
2	Status LED indicators	From bottom to top: Associated, Data In, Data Out
3	DIO pin LED indicators	From bottom to top: DIO4, DIO11, DIO12
4	DC barrel plug	9 to 18 VDC DC supply is required for transmitting at 27 dBm or higher. **WARNING!** Do not exceed 18 VDC. Excessive voltage will damage the device and may cause injury.
5	Do not connect	Do not connect to these pins. This development board does not support the device's self power feature.
6	USB connector	Connect a mini USB cable from this port to the host PC.

Number	Item	Description
7	Reset button	When pressed, the button switch resets the device by pulling pin 6 low.
8	Commissioning button	When pressed, the button switch pulls the device's pin 33 low.
9	Development header	0.1” pitch header corresponds to pins 1 through 34 on the XBee/XBee-PRO SX RF Module. Abbreviated labels are silkscreened to the left of the header.
10	RPSMA connector	Attach an antenna or RF cable to this jack. This connection is valid only if the board has an RF pad XBee.
11	XBee/XBee-PRO SX RF Modulefootprint	

Connect XBee-PRO SX development boards to a PC

1. Connect both SX development boards to power supplies and plug the power supplies into an outlet. The power supply is required when using the XBee-PRO SX because the current draw when transmitting will exceed what a USB port can supply.
2. Connect the SX development boards to the USB port on a PC via the mini-USB cables. Separate the SX development boards by at least 2 m (6 ft).
3. Connect the antennas to the RPSMA connector on the SX development boards.

The following table shows the XBee SX development kit contents.

Description	Quantity	Part number
XBee surface-mount (SMT) socket development board	1	XBIB-U-SS
SX development board with onboard XBee-PRO SX RF pad module	2	USA - XBIB-XBP9XR-0 Australia - XBIB-XBP9XR-2 Brazil - XBIB-XBP9XR-1
Standard USB cable	1	N/A
Mini USB cable	2	N/A
12 VDC power supply	2	Australia and Brazil include packages of AC adapters
RPSMA antenna	3	A09-HASM-675
U.FL to RPSMA adapter cable	1	JF1R6-CR3-4I
XBee SX U.FL module	1	USA - XB9X-DMUS-001 Australia - XB9X-DMUS-021 Brazil - XB9X-DMUS-011

Connect the XBIB-U-SS development board to a PC

This step is optional. It shows how to set up the standalone XBee SX module on the XBIB-U-SS development board, which you can substitute as one of the range test devices or use with the other two devices to create a DigiMesh network.

You can find reference information on the XBIB-U-SS at this link: http://ftp1.digi.com/support/documentation/xbibuss_referenceguide.pdf.

To connect the XBee devices to the XBIB-U-SS development boards included in the kit:

CAUTION! Make sure the board is not powered by either the USB or a battery when you plug in the XBee module.

1. Plug one XBee SX RF Module into the XBIB-U-SS development board.
2. Once the XBee module is plugged into the board (and not before), connect the board to your computer using the USB cable provided.
3. Connect the U.FL to RPSMA adapter cable to the XBee device and an antenna.

Configure the device using XCTU

XBee Configuration and Test Utility (XCTU) is a multi-platform program that enables users to interact with Digi radio frequency (RF) devices through a graphical interface. The application includes built-in tools that make it easy to set up, configure, and test Digi RF devices.

For instructions on downloading and using XCTU, see the XCTU User Guide.

Once you install XCTU, click the XCTU icon to open the program.

Click **Discover devices** and follow the instructions. XCTU should discover the connected XBee/XBee-PRO SX RF Modules using the provided settings.

Click **Add selected devices**. The devices appear in the **Radio Modules** list. You can click a module to view and configure its individual settings. For more information on these items, see AT commands.

Configure the devices for a range test

For devices to communicate with each other, you must configure them so they are in the same network. To obtain all possible data from the remote device, you must also set the local device to API mode. For more information on API mode, see Operate in API mode.

For devices to communicate with each other, you configure them so they are in the same network. You also set the local device to API mode to obtain all possible data of the remote device.

When you connect the development board to a PC for the first time, the PC automatically installs drivers, which may take a few minutes to complete.

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

   ID: 2015
   NI: LOCAL_DEVICE
   AP: API Mode Without Escapes [1]
4. Click the **Write radio settings** button.

5. Select the other module and click the **Default firmware settings** button.
6. Configure the following parameters:

   ID: 2015
   NI: REMOTE_DEVICE
   AP: Transparent Mode [0]
7. Click the **Write radio settings** button.

   After you write the radio settings for each device, their names appear in the **Radio Modules** area. The Port indicates that the LOCAL_DEVICE is in API mode.
8. Disconnect REMOTE_DEVICE from your computer and remove it from XCTU; leave its power supply connected and plugged in.
9. Place REMOTE_DEVICE at the testing location and connect its power supply.
10. If you have not already done so, connect LOCAL_DEVICE to a battery. Its USB cable should still be connected to the laptop.

Perform a range test

A range test is a simple point to point wireless demonstration that tests the devices' ability to transmit to and receive from each other. Wireless environments vary dramatically depending on many factors and the range test allows you to experiment with the devices in your own environment.

The range test instructions assume that you are using a laptop to configure the devices. You will need a battery to power the device connected to the laptop (the "local device") so that you can move around. Alternatively you can set the local device to **PL**: 21.5 dBm [0], and the device will run off of the USB port exclusively. However, the achievable range will significantly decrease if you use the lower power setting.

WARNING! Keep the boards 2 m (6 ft) apart when transmitting to protect the device's front end.

Follow these steps to perform the range test:

1. Open the **Tools** menu within XCTU and select the **Range Test** option.
2. Your local devices are listed on the left side of the Devices Selection section. Select the local device and click **the Discover remote devices** button .
3. When the discovery process finishes, the remote device is displayed in the **Discovering remote devices...** dialog. Click **Add selected devices**.
4. Select the remote device from the **Discovered device** list located on the right panel inside.
5. Click **Start Range Test**.
6. Range test data is represented in the chart. By default, 100 packets are sent for the test. XCTU displays the instant local and remote RSSI in two separate controls, as well as the number of packets sent and received. Remote devices only report their RSSI value when the local device is operating in API mode (by setting **AP** to 1).
7. Test the wireless link by moving your laptop and local device setup away from the remote device. Watch the range test status indicators: RSSI will decrease as you get farther away, and eventually you will see the percentage of successful packets drop below 100%, indicating you are approaching the limits of the range.

Watch the range test status indicators: RSSI will decrease the further away you are, and if you continue to move the setup further away, eventually you will see the percentage of successful packets drop below 100%, indicating you are approaching the limits of the range.

8. Click **Stop Range Test** to stop the process at any time.

Mesh network demonstration

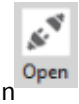
1. Connect the two XBee PRO SX development boards and the XBIB-U-SS (with the XBee SX installed) to your computer. Open XCTU and find the three XBee devices.
2. Configure all three devices with the following parameters and write the settings. Note that XBee A (SENDER) should be the XBee SX device running on the XBIB-U-SS development board.

Parameter	XBee A (XBee SX)	XBee B (XBee PRO SX)	XBee C (XBee PRO SX)	Effect
NI	SENDER	RECEIVER	BRIDGE	Defines the node identifier, a human-friendly name for the module. **CAUTION!** The default NI value is a blank space. Make sure to delete the space when you change the value.
ID	2015	2015	2015	Defines the network that a device will attach to. This must be the same for all devices in your network.
PL	Lowest [0]	Lowest [0]	Lowest [0]	Defines the transmitter output power level. Set it to the lowest level to facilitate the execution of the example.
DH	0	0	0	Defines the destination address (high part) to transmit the data to.
DL	FFFF	FFFF	FFFF	Defines the destination address (low part) to transmit the data to. You can use the 000000000000FFFF address to send a broadcast message.
RP	5	5	5	Defines the time that the RSSI LED will be on when the device receives a packet. 5 (hexadecimal) = 5 (decimal) x 100 ms = 500 ms.

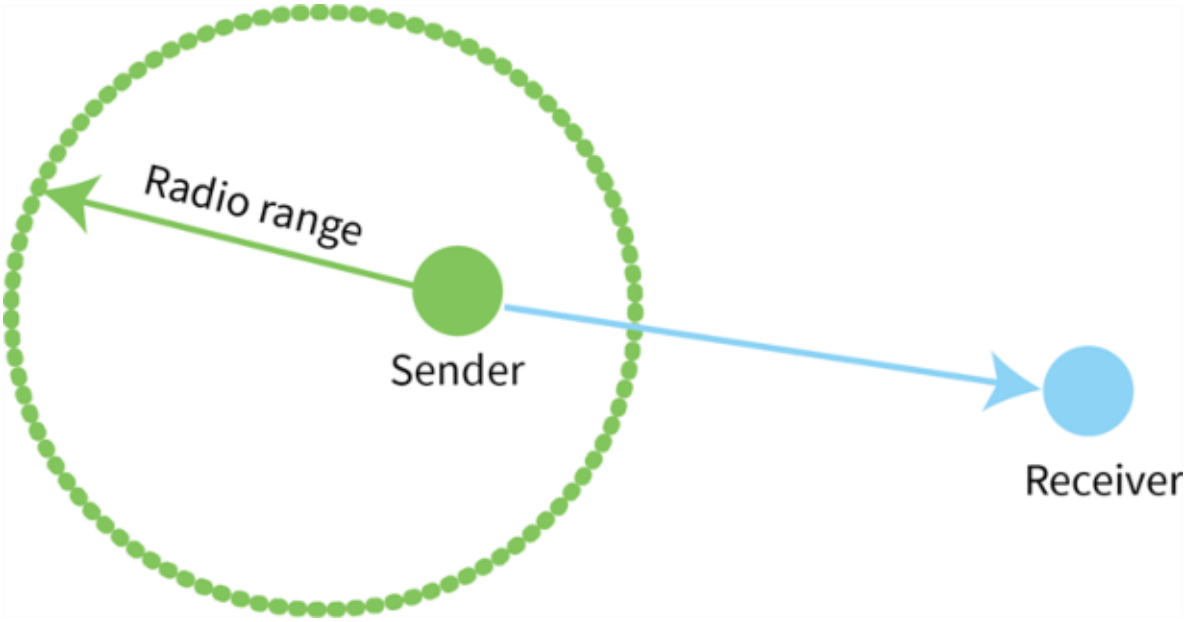
To set up a DigiMesh network, you must first connect some XBee devices to a portable battery so you can move around with them.

1. Keep SENDER connected to the computer.
2. Disconnect RECEIVER from the computer and connect to a portable battery.
3. Disconnect BRIDGE from the computer and from power.

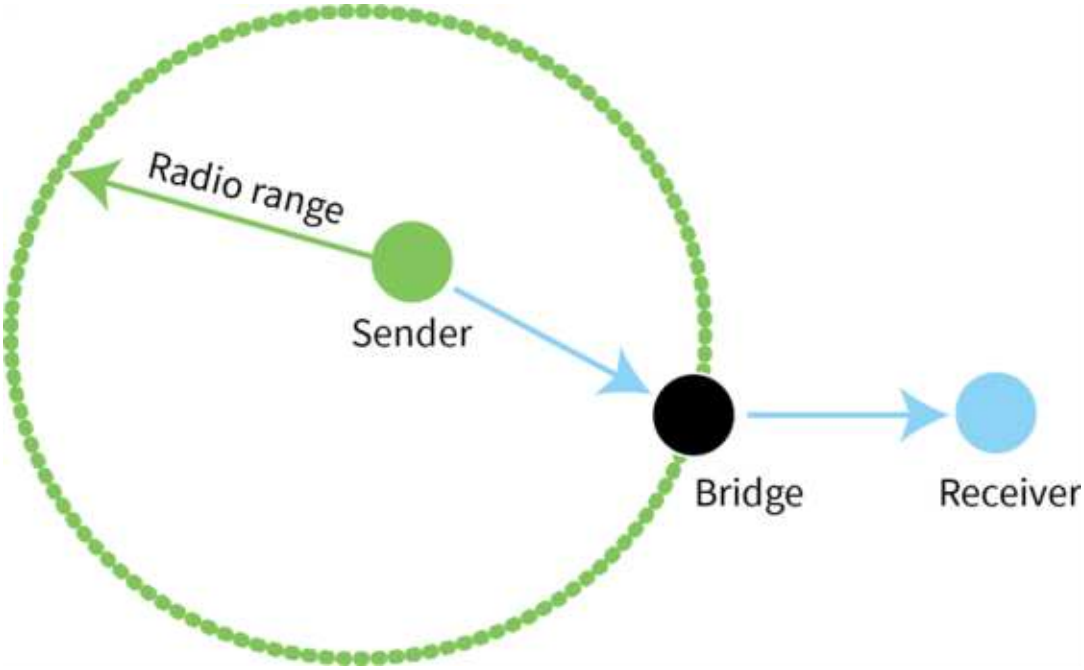
You can now start sending messages. Use the following steps to simulate a simple DigiMesh network by configuring SENDER and RECEIVER to communicate, moving the two devices out of range of each other, and adding the BRIDGE node to relay the messages between SENDER and RECEIVER. You will start by configuring SENDER to send a broadcast message every second. You can use the XCTU console or any serial port terminal application. This tutorial uses the XCTU console.

1. Switch to the **Consoles** working mode using the button in the top menu.
2. Open the serial connection of the device: select the SENDER device in the **Radio Modules** section, and click the **Open serial connection** button 

.
The background changes to green to indicate that the connection is open.
3. Click the plus sign in the **Send Frames** panel to add a new packet. Type the message "Hello!".
4. To start sending this message every second, in the **Send sequence** box:
   a. Set 1000 ms as transmit interval.
   b. Select **Loop infinitely**.
   c. Click **Start sequence**.
5. Notice that the yellow Data Out LED of RECEIVER illuminates briefly every second. This means that the device is successfully receiving the messages. The white RSSI LEDs will also light up to indicate the strength of the signal received.
6. Move RECEIVER away from SENDER until the Data Out LED does not blink anymore, meaning it has moved out of range.



7. Place BRIDGE about halfway between SENDER and RECEIVER and plug it in. BRIDGE joins the network, creates a bridge between the other two nodes, and relays messages from SENDER to RECEIVER.



Software libraries

One way to communicate with the XBee/XBee-PRO SX RF Module is by using a software library. The libraries available for use with the XBee/XBee-PRO SX RF Module include:

- XBee Java library
- XBee Python library

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

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

XBee Network Assistant

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

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

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

To install the XBee Network Assistant:

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

XBee Multi Programmer

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

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

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

Download the XBee Multi Programmer application from: Digi XBee Multi Programmer

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

Modes

Transparent and API operating modes 35
Modes of operation 36

Transparent and API operating modes

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

Transparent operating mode

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

While operating in Transparent operating mode, the device uses the **DH** and **DL** parameters to determine the destination address used for RF transmissions. Transparent operating mode is not available when using the SPI interface.

API operating mode

API operating mode is an alternative to Transparent operating mode. API mode is a frame-based protocol that allows you to direct data on a packet basis. The device communicates UART or SPI data in packets, also known as API frames. This mode allows for structured communications with computers and microcontrollers.

The advantages of API operating mode include:

- It is easier to send information to multiple destinations
- The host receives the source address for each received data frame
- You can change parameters without entering Command mode
- You can query or set a configuration parameter while a pending command—for example **ND**—is in progress. This cannot be done in Command mode.

For more information, see API frame specifications.

Comparing Transparent and API modes

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

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

Feature	Description
Transparent mode features	
Simple interface	All received serial data is transmitted unless the device is in Command mode
Easy to support	It is easier for an application to support Transparent operation and Command mode
API mode features	
Easy to manage data transmissions to	Transmitting RF data to multiple remote devices only requires the

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

Modes of operation

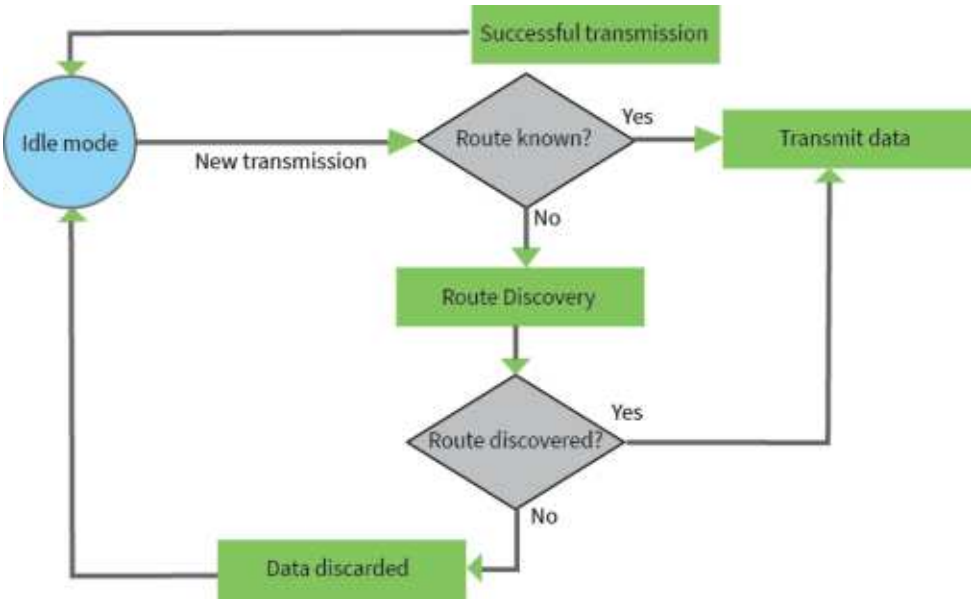
Receive mode

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

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 received data on the RF network, that data must meet the following criteria:

- Network ID match [ID (Network ID)]
- Preamble ID match [HP (Preamble ID)]
- Address match [SH (Serial Number High) and SL (Serial Number Low)]

Transmit mode



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

Sleep mode

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

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

For more information, see Sleep modes.

Command mode

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

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

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

Enter Command mode

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

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

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

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

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

Troubleshooting

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

There are two alternative ways to enter Command mode:

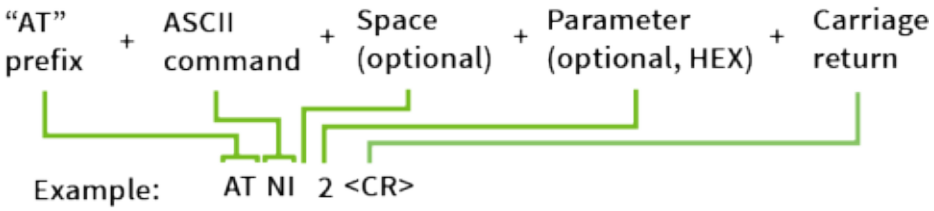
- A serial break for six seconds enters Command mode. You can issue the "break" command from a serial console, it is often a button or menu item.
- Asserting DIN (serial break) upon power up or reset enters Command mode. XCTU guides you through a reset and automatically issues the break when needed.

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

Send AT commands

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

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



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

Multiple AT commands

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

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

Parameter format

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

Response to AT commands

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

Apply command changes

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

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

Make command changes permanent

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

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

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

Exit Command mode

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

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

Sleep modes

About sleep modes 41
Normal mode 41
Asynchronous pin sleep mode 42
Asynchronous cyclic sleep mode 42
Asynchronous cyclic sleep with pin wake up mode 42
Synchronous sleep support mode 42
Synchronous cyclic sleep mode 43
The sleep timer 43
Indirect messaging and polling 43
Sleeping routers 44
Sleep coordinator sleep modes in the DigiMesh network 44
Synchronization messages 45
Become a sleep coordinator 47
Select sleep parameters 50
Sleep immediate 50
Synchronous pulse 51
Start a sleeping synchronous network 51
Add a new node to an existing network 52
Change sleep parameters 53
Rejoin nodes that lose sync 54
Diagnostics 54

About sleep modes

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

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

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

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

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

Asynchronous modes

- Do not use asynchronous sleep modes in a synchronous sleeping network, and vice versa.
- Use the asynchronous sleep modes to control the sleep state on a device by device basis.
- Do not use devices operating in asynchronous sleep mode to route data.
- We strongly encourage you to set asynchronous sleeping devices as end-devices using the **CE** command. This prevents the node from attempting to route data.
- Transmissions sent to an asynchronously sleeping device are not buffered and will be lost if the receiving device is not actively awake when the transmission occurred. If you require two-way communication, you can use Indirect Messaging for P2MP unicast messages. For more information, see Indirect messaging and polling.

Synchronous modes

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

This forms a cyclic sleeping network.

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

Normal mode

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

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

Use mains-power for devices in Normal mode.

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

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

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

Asynchronous pin sleep mode

Set **SM** to 1 to enter asynchronous pin sleep mode.

Pin sleep allows the device to sleep and wake according to the state of the $\overline{\text{SLEEP}}$_RQ pin (pin 10).

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

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

Asynchronous cyclic sleep mode

Set **SM** to 4 to enter asynchronous cyclic sleep mode.

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

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

The ON_$\overline{\text{SLEEP}}$ line (pin pin 26) is asserted (high) when the device wakes, and is de-asserted (low) when the device sleeps.

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

Asynchronous cyclic sleep with pin wake up mode

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

This mode is a slight variation on asynchronous cyclic sleep mode (**SM** = 4) that allows you to wake a device prematurely by asserting the SLEEP_RQ pin (pin 10).

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

Synchronous sleep support mode

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

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

Sleep support nodes are especially useful:

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

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

Synchronous cyclic sleep mode

Set **SM** to 8 to enter synchronous cyclic sleep mode.

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

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

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

If **D9** = 1 (ON/$\overline{\text{SLEEP}}$ enabled) on a cyclic sleep node, the ON/$\overline{\text{SLEEP}}$ line asserts when the device is awake and de-asserts when the device is asleep.

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

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

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

The sleep timer

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

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

Indirect messaging and polling

To enable reliable communication with sleeping devices, you can use the **CE** (Routing/Messaging Mode) command to enable indirect messaging and polling. Indirect messaging only affects Point-to-Multipoint (P2MP) unicast transmissions.

Indirect messaging

Indirect messaging is a communication mode designed for communicating with asynchronous sleeping devices. A device can enable indirect messaging by making itself an indirect messaging coordinator with the **CE** command. An indirect messaging coordinator does not immediately transmit a P2MP unicast when it is received over the serial port. Instead the device holds onto the data until it is requested via a poll. On receiving a poll, the indirect messaging coordinator sends a queued data packet (if available) to the requestor.

Because it is possible for a polling device to be eliminated, a mechanism is in place to purge unrequested data packets. If the coordinator holds an indirect data packet for an indirect messaging poller for more than 2.5 times its **SP** value, then the packet is purged. We suggest setting the **SP** of the coordinator to the same value as the highest **SP** time that exists among the pollers in the network. If

the coordinator is in API mode, a TxStatus message is generated for a purged data packet with a status of 0x75 (**INDIRECT_MESSAGE_UNREQUESTED**).

An indirect messaging coordinator queues up as many data packets as it has buffers available. After the coordinator uses all of its available buffers, it holds transmission requests unprocessed on the serial input queue. After the serial input queue is full, the device de-asserts CTS (if hardware flow control is enabled). After receiving a poll or purging data from the indirect messaging queue the buffers become available again.

Indirect messaging only functions with P2MP unicast messages. Indirect messaging has no effect on P2MP broadcasts, directed broadcasts, repeater packets, or DigiMesh packets. These messages are sent immediately when received over the serial port and are not put on the indirect messaging queue.

Polling

Polling is the automatic process by which a node can request data from an indirect messaging coordinator. To enable polling on a device, configure it as an indirect messaging poller with the **CE** command and set its **DH**:**DL** registers to match the **SH**:**SL** registers of the device that will function as the Indirect Messaging Coordinator. When you enable polling, the device sends a P2MP poll request regularly to the address specified by the **DH**:**DL** registers. When the device sends a P2MP unicast to the destination specified by the **DH**:**DL** of a polling device, the data also functions as a poll.

When a polling device is also an asynchronous sleeping device, that device sends a poll shortly after waking from sleep. After that first poll is sent, the device sends polls in the normal manner described previously until it returns to sleep.

Sleeping routers

The Sleeping Router feature of DigiMesh makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time. This forms a cyclic sleeping network. For more information, see Become a sleep coordinator.

Sleep coordinator sleep modes in the DigiMesh network

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

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

If you change the **SP** or **ST** parameters, the network does not apply the new settings until the beginning of the next wake time. For more information, see Change sleep parameters.

A sleeping router network is robust enough that an individual node can go several cycles without receiving a sync message, due to RF interference, for example. As a node misses sync messages, the time available for transmitting messages during the wake time reduces to maintain synchronization accuracy. By default, a device reduces its active sleep time progressively as it misses sync messages.

Synchronization messages

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

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

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

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

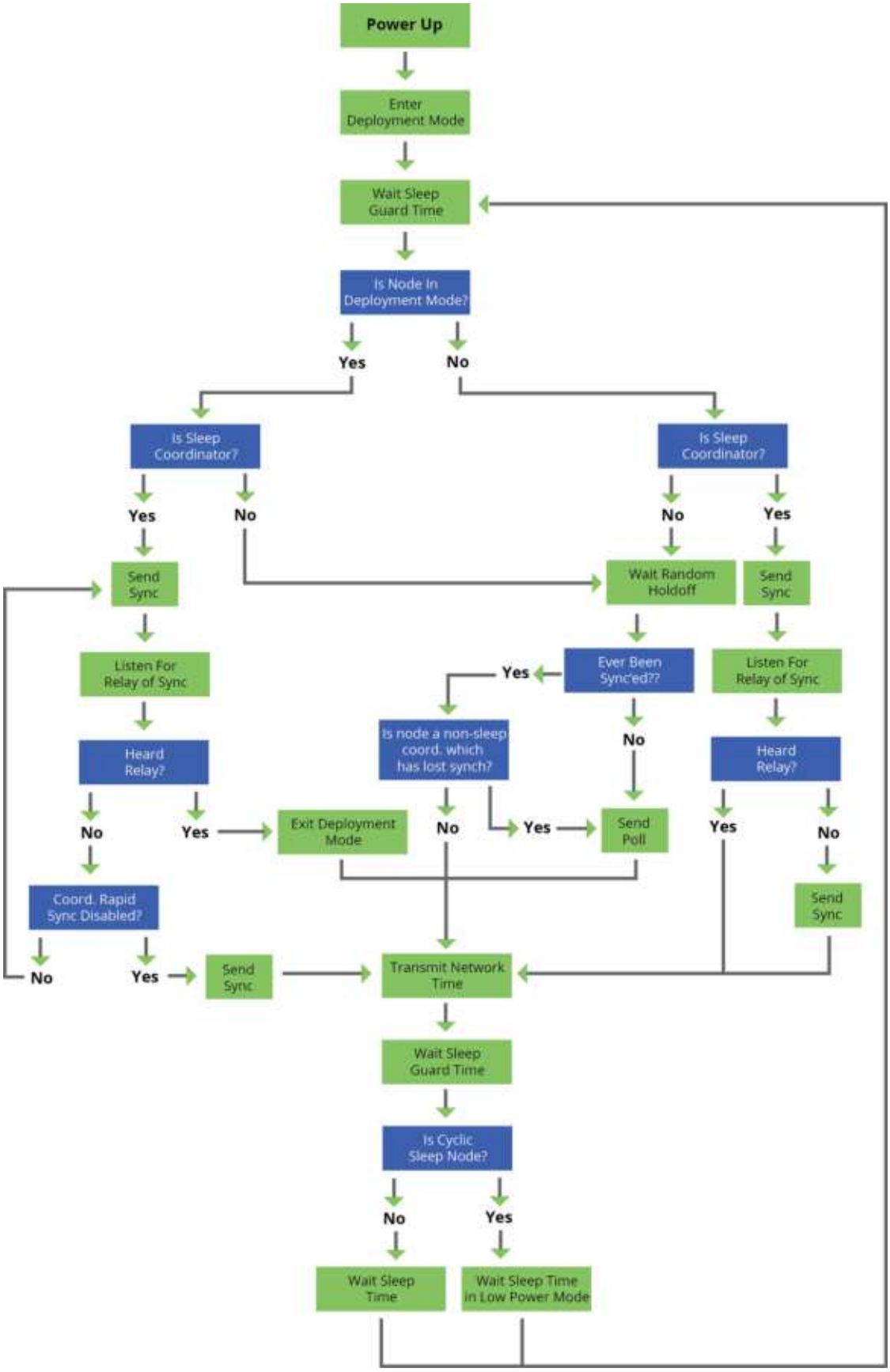
Use the **SO** (sleep options) command to disable deployment mode. This option is enabled by default.

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

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

If you use the **SO** command to configure nodes as non-coordinators, and if the non-coordinators go six or more sleep cycles without hearing a sync, they send a message requesting sync at the beginning of their wake period.

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


Power Up
Enter Deployment Mode
Wait Sleep Guard Time
Is Node In Deployment Mode?
Yes
No
Is Sleep Coordinator?
Yes
No
Send Sync
Listen For Relay of Sync
Heard Relay?
No
Yes
Exit Deployment Mode
Coord. Rapid Sync Disabled?
No
Yes
Send Sync
Is Sleep Coordinator?
No
Yes
Wait Random Holdoff
Send Sync
Ever Been Sync'ed??
Yes
No
Is node a non-sleep coord. which has lost synch?
No
Yes
Send Poll
Listen For Relay of Sync
Heard Relay?
Yes
No
Send Sync
Transmit Network Time
Wait Sleep Guard Time
Is Cyclic Sleep Node?
No
Yes
Wait Sleep Time
Wait Sleep Time in Low Power Mode

Become a sleep coordinator

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

- Define a preferred sleep coordinator
- A potential sleep coordinator misses three or more sync messages
- Press the Commissioning Pushbutton twice on a potential sleep coordinator
- Change the sleep timing values on a potential sleep coordinator

Preferred sleep coordinator option

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

A node with the sleep coordinator bit set always sends a sync message at the beginning of a wake cycle. To avoid network congestion and synchronization conflicts, do not set this bit on more than one node in the network.

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

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

CAUTION! Use the preferred sleep coordinator bit with caution. The advantages of using the option become weaknesses if you use it on a node that is not in the proper position or configuration. Also, it is not valid to have the sleep coordinator option bit set on more than one node at a time.

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

Resolution criteria and selection option

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

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

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

- Misses three or more sync messages.
- It is not configured as a non-coordinator by setting bit 1 of **SO**.

If such a node wins out in the selection process, it becomes the new network sleep coordinator.

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

1. Newer sleep parameters always take priority over older sleep parameters. The age of the sleep parameters is determined by a sequence number that increments when an overriding sync is sent.
2. Otherwise, the node with the preferred sleep coordinator bit set takes precedence.
3. Otherwise, a sleep support node—**SM 7**—takes priority over a node that is not a sleep support node—**SM 8**.
4. Otherwise, the node with highest serial number becomes the sleep coordinator.

Commissioning Pushbutton option

Use the Commissioning Pushbutton to select a device to act as the sleep coordinator. The Commissioning Pushbutton is mapped to DIO0 (pin 33) and enabled by default.

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

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

Overriding syncs

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

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

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

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

Sleep guard times

To compensate for variations in the timekeeping hardware of the various devices in a sleeping router network, the network allocates sleep guard times at the beginning and end of the wake period. The size of the sleep guard time varies based on the sleep and wake times you select and the number of sleep cycles that elapse since receiving the last sync message. The sleep guard time guarantees that a destination module will be awake when the source device sends a transmission. As a node misses

more and more consecutive sync messages, the sleep guard time increases in duration and decreases the available transmission time.

Auto-early wake-up sleep option

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

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

Use the **SO** command to disable auto-early wake-up sleep. This option is enabled by default.

Select sleep parameters

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

1. Choose **NN** and **NH**.

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

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

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

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

3. Select the duty cycle you want.
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.

Sleep immediate

In order to ensure that the needed messages have time to traverse the network, **ST** must be sufficiently large. Additionally, your application is a factor in determining what **ST** should be. When **ST**/**SP** increases, the batteries burn out faster. Yet, **ST** must be large enough for a functional network.

To mitigate this problem, the Sleep Immediate command is available (in version **0x9009** and newer). The Sleep Immediate (**SI**) command can be sent by your application after it determines that all needed transmissions are completed. This not only puts the node that issues the command asleep, but it also sends a broadcast to put the whole network to sleep. The network will then remain asleep for the remainder of the wake time and the subsequent sleep time. Then the entire network will awaken again, resuming the same sleep cycle as before.

In the event that one or more nodes fail to receive the sleep immediate broadcast, they will not get the power savings, but they will still remain synchronized to the network because the sleep cycle would not have changed.

Synchronous pulse

The Synchronous pulse feature supports user applications that desire synchronizing operations across multiple devices in a network. This is done by pulsing the **DIO9**—ON_SLEEP—pin simultaneously on all devices in the network. Typically, this feature is used with a mains-powered application and transmissions are not inhibited except during the small window of time allocated for sending synchronization messages throughout the network.

Configuration

Generally, synchronous pulse is configured the same way as synchronous sleep. See the table below:

AT Command	Value	Description
D9	6	This causes a synchronous pulse line to be toggled low and then high on the **DIO9** pin throughout the network. Note that **D9** must be set to **1** for the ON_SLEEP indicator. This means that it is not possible to have both a synchronous pulse and an ON_SLEEP indicator. However, if a synchronously sleeping network is desired, **D9** may still be used as a synchronous pulse with the obvious limitation that the network will not be able to transmit while it is asleep.
SM	7 or 8	Setting **SM** to **7** or **8** allows the synchronization messages to be sent. They will not be sent and this feature will be disabled if **SM** is set to any other value.
SP	0	Even with **SM** set to **8**, the network still stays awake by setting **SP** to **0** on the sleep coordinator.
ST	??	Set **ST** to the frequency desired for the synchronous pulse. If it is set too small, it is rounded up to a value that allows a synchronous pulse to be output every time **DIO9** pulses. Some parameters—such as **NH**—can be made smaller in order to reduce the minimum value for **ST**. If it still is not possible to get synchronous pulses at small enough intervals to satisfy the application, then the microcontroller connected to **DIO9** will be responsible for subdividing the time between each synchronous pulse.

Accuracy

Due to variations is traffic patterns and hopping patterns, perfect accuracy cannot be guaranteed. However, this feature is usually accurate within + or - 100 micro-seconds. If greater accuracy than that is required, this solution will not work for your application.

Start a sleeping synchronous network

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

1. Set **SO** to 1 to enable the preferred sleep coordinator option on one of the nodes.
2. Set its **SM** to a synchronous sleep compatible mode (7 or 8) with its **SP** and **ST** set to a quick cycle time. The purpose of a quick cycle time is to allow the network to send commands quickly through the network during commissioning.
3. Power on the new nodes within range of the sleep coordinator. The nodes quickly receive a sync message and synchronize themselves to the short cycle **SP** and **ST** set on the sleep coordinator.
4. Configure the new nodes to the sleep mode you want, either cyclic sleeping modes or sleep support modes.
5. Set the **SP** and **ST** values on the sleep coordinator to the values you want for the network.
6. In order to reduce the possibility of an unintended overriding sync, set **SP** and **ST** to the intended sleep/wake cycle on all nodes in the network. Be sure that **ST** is large enough to prevent it from being inadvertently increased by changing **NN**, **NH**, or **MT**.
7. Wait a sleep cycle for the sleeping nodes to sync themselves to the new **SP** and **ST** values.
8. Disable the preferred sleep coordinator option bit on the sleep coordinator unless you want a preferred sleep coordinator.
9. Deploy the nodes to their positions.

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

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

Add a new node to an existing network

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

1. Power the new node on within range of a sleep support node. Sleep support nodes are always awake and able to respond to sync requests promptly.

2. You can wake a sleeping cyclic sleep node in the network using the Commissioning Pushbutton. Place the new node in range of the existing cyclic sleep node. Wake the existing node by holding down the Commissioning Pushbutton for two seconds, or until the node wakes. The existing node stays awake for 30 seconds and responds to sync requests while it is awake.

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

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

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

Change sleep parameters

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

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

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

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

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

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

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

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

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

Rejoin nodes that lose sync

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

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

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

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

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

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

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

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

Diagnostics

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

Query sleep cycle

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

Sleep status

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

Missed sync messages command

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

Sleep status API messages

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

Networking methods

The MAC and PHY layers 57
64-bit addresses 57
Make a unicast transmission 58
Make a broadcast transmission 58
Delivery methods 58

The MAC and PHY layers

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

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

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

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

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

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

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

64-bit addresses

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

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

Make a unicast transmission

To transmit to a specific device in Transparent operating mode:

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

To transmit to a specific device in API operating mode:

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

Make a broadcast transmission

To transmit to all devices in Transparent operating mode:

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

To transmit to all devices in API operating mode:

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

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

Delivery methods

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

The XBee/XBee-PRO SX RF Module supports three delivery methods:

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

Point to Point / Point to Multipoint (P2MP)

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

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

In P2MP mode, the originator sends a unicast packet directly to the destination device—which must be in range. All other devices in range ignore the unicast.

In P2MP mode the originator sends a broadcast **MT**+1 times, but the receiving nodes do not rebroadcast the packet. So, like the unicast transmission, the receiving device must be in range.

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

Repeater/directed broadcast

This mode of operation allows messages to traverse a network and arrive at the destination node without requiring the overhead of route discovery. This is done in the network layer which does one of two things when it receives a message:

1. If this node is the destination node, then the message is output on the serial port.
2. If this node is not the destination node, then the message is rebroadcast.

Since the message is a broadcast at the lowest layer, all nodes will receive the broadcasted or re-broadcasted message, but only the node whose address is the final destination will send the message to the serial port.

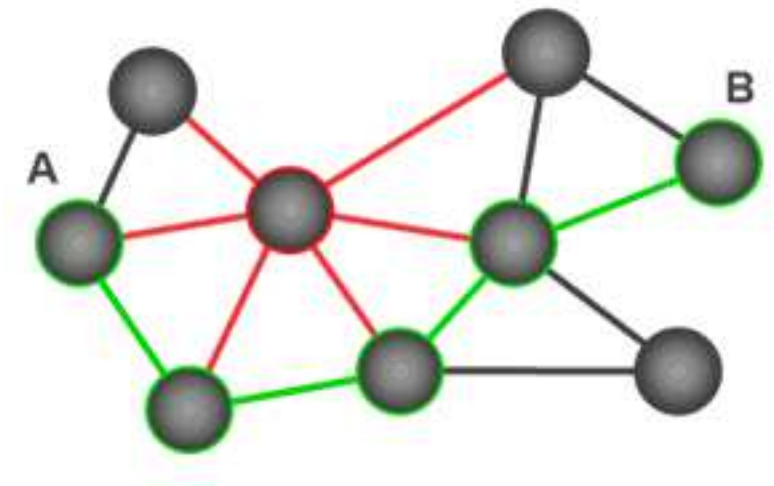
The nodes that are not the final destination follow all the rules that otherwise apply to broadcasts:

1. Hops are limited by **NH** and **BH**.
2. Non-routing nodes—with **CE** = **2**—will not repeat directed broadcasts.
3. Broadcasts are sent **MT**+**1** times.

DigiMesh networking

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

- **Routing**. With this technique, the message is propagated along a path by hopping from node to node until it reaches its final destination.
- **Ad-hoc network creation**. This is an automated process that creates an entire network of nodes on the fly, without any human intervention.
- **Self-healing**. This process automatically figures out if one or more nodes on the network is missing and reconfigures the network to repair any broken routes.
- **Peer-to-peer architecture**. No hierarchy and no parent-child relationships are needed.
- **Quiet protocol**. Routing overhead will be reduced by using a reactive protocol similar to AODV.
- **Route discovery**. Rather than maintaining a network map, routes will be discovered and created only when needed.
- **Selective acknowledgments**. Only the destination node will reply to route requests.
- **Reliable delivery**. Reliable delivery of data is accomplished by means of acknowledgments.
- **Sleep modes.** Low power sleep modes with synchronized wake are supported with variable sleep and wake times.



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

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

Note Mesh networks use more bandwidth for administration and therefore have less available for payloads.

Unicast addressing

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

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

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

Routing

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

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

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

Route discovery

Route discovery is a process that occurs when:

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

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

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

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

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

Transmission timeouts

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

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

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

- Broadcast transmission
- Unicast transmission on a known route
- Unicast transmission on an unknown route
- Unicast transmission on a broken route

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

Unicast one hop time

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

Transmit a broadcast

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

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

The **NH** and **%H** parameters define the maximum broadcast delay as follows:

BroadcastTxTime = NH * NN * %8

Unless **BH** < **NH**, in which case the formula is:

BroadcastTxTime = BH * NN * %8

Transmit a unicast with a known route

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

You can estimate the timeout in the following manner:

knownRouteUnicastTime=2*NH*MR*unicastOneHopTime

Transmit a unicast with an unknown route

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

unknownRouteUnicastTime=BroadcastTxTime+
(**NH***unicastOneHopTime)+knownRouteUnicastTime

Transmit a unicast with a broken route

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

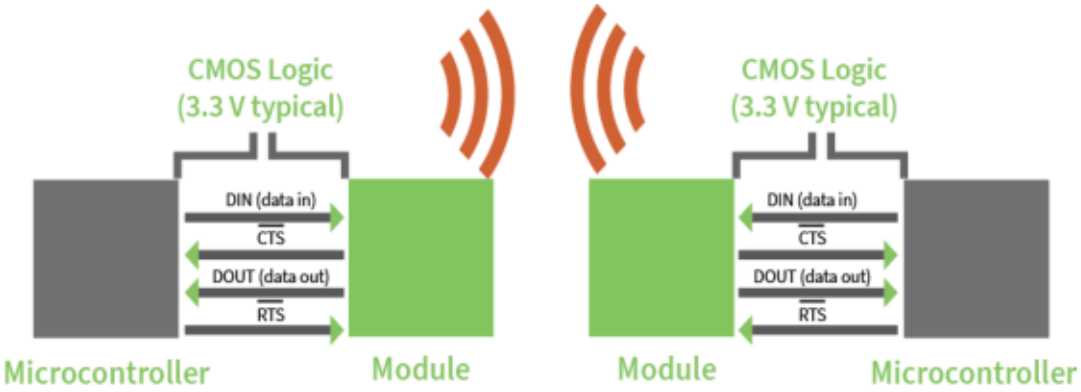
brokenRouteUnicastTime=BroadcastTxTime+(NH*unicastOneHopTime)+
(2*knownRouteUnicastTime)

Serial communication

UART data flow 64
SPI signals 64
Slave mode characteristics 65
Full duplex operation 66
Low power operation 66
Configuration considerations 67
SPI and API mode 67
SPI parameters 67
Serial port selection 67
UART flow control 68

UART data flow

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

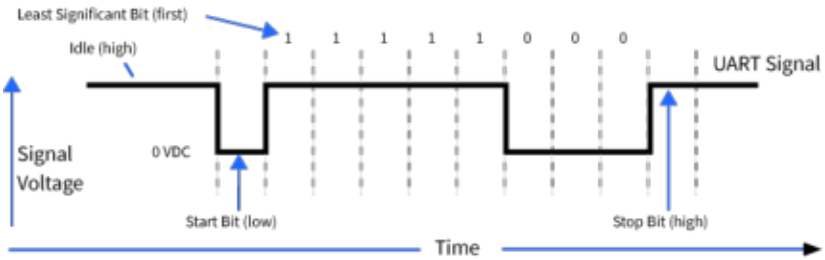


Serial data

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

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

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



SPI signals

The XBee/XBee-PRO SX RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The SPI port uses the following signals on the device:

Signal	Pin number	Applicable AT command
SPI_$\overline{\text{ATTN}}$ (Attention)	12	**P9**
SPI_SCLK (Serial clock)	14	**P8**

Signal	Pin number	Applicable AT command
SPI_$\overline{\text{SSEL}}$ (Slave select)	15	**P7**
SPI_MOSI (Master out, Slave in)	16	**P6**
SPI_MISO (Master in, Slave out)	17	**P5**

By default, the inputs have pull-up resistors enabled. Use the **PR** command to disable the pull-up resistors. When the SPI pins are not connected but the pins are configured for SPI operation, then the device requires the pull-ups for proper UART operation.

Signal description

SPI_MISO: When SPI_$\overline{\text{SSEL}}$ is asserted (low) and SPI_CLK is active, the device outputs the data on this line at the SPI_CLK rate. When SPI_$\overline{\text{SSEL}}$ is de-asserted (high), you should tri-state this output such that another slave device can drive the line.

SPI_MOSI: The SPI master outputs data on this line at the SPI_CLK rate after it selects the desired slave. When you configure the device for SPI operations, this pin is an input.

SPI_SCLK: The SPI master outputs a clock on this pin, and the rate must not exceed the maximum allowed, 6 Mb/s. When you configure the device for SPI operations, this pin is an input. This signal enables serial communication with the slave.

SPI_$\overline{\text{SSEL}}$: The SPI master outputs a low signal on this line to select the desired slave. When you configure the device for SPI operations, this pin is an input. This signal clocks data transfers on MOSI and MISO.

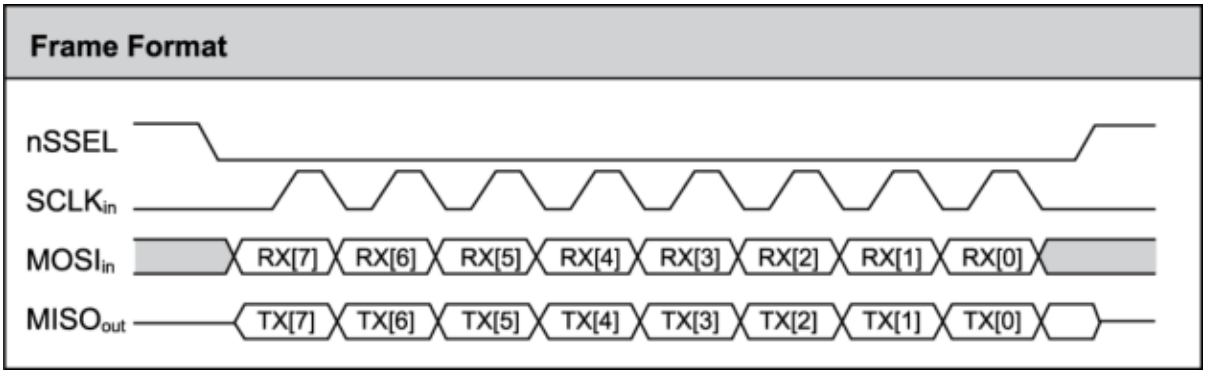
SPI_$\overline{\text{ATTN}}$: The device asserts this pin low when it has data to send to the SPI master. When you configure this pin for SPI operations, it is an output (not tri-stated). This signal alerts the master that the slave has data queued to send. The device asserts this pin as soon as data is available to send to the SPI master and it remains asserted until the SPI master has clocked out all available data.

Slave mode characteristics

In slave mode, the following apply:

- SPI Clock rates up to 6 MHz (6 Mb/s) are possible.
- Data is MSB first.
- It uses Frame Format Mode 0. This means CPOL= 0 (idle clock is low) and CPHA = 0 (data is sampled on the clock's leading edge). The picture below diagrams Mode 0.
- The SPI port is setup for API mode and is equivalent to **AP** = 1.

The following picture shows the frame format for SPI communications.

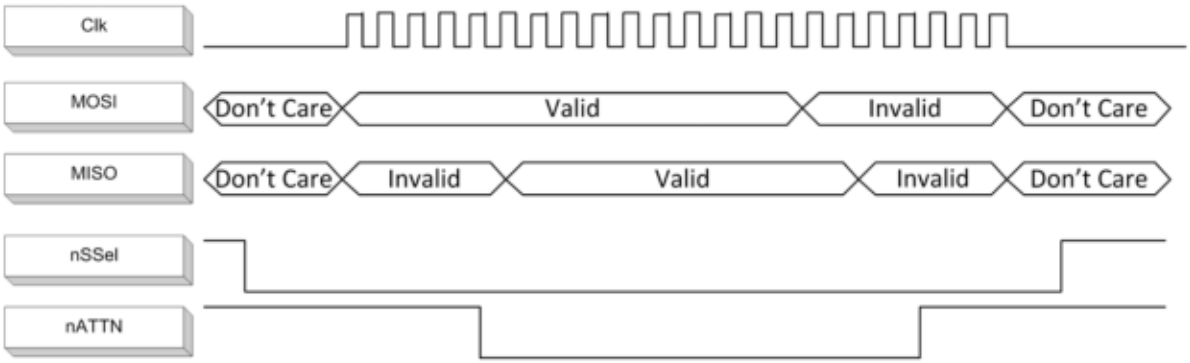


Full duplex operation

When using SPI on the XBee/XBee-PRO SX RF Module the device uses API operation (**AP** = 1) without escaped characters to packetize data. SPI is a full duplex protocol, even when data is only available in one direction. This means that whenever a device receives data, it also transmits, and that data is normally invalid. Likewise, whenever a device transmits data, invalid data is probably received. To determine whether or not received data is invalid, the firmware places the data in API packets.

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

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



Low power operation

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

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

If SLEEP_REQUEST is not configured as a peripheral and SPI_SSEL is configured as a peripheral, then pin sleep is controlled by SPI_SSEL rather than by SLEEP_REQUEST. Asserting SPI_SSEL (pin 15) by driving it low either wakes the device or keeps it awake. Negating SPI_SSEL by driving it high puts the device to sleep.

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

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

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

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

Configuration considerations

The configuration considerations are:

- How do you select the serial port? For example, should you use the UART or the SPI port?
- If you use the SPI port, what data format should you use in order to avoid processing invalid characters while transmitting?
- What SPI options do you need to configure?

SPI and API mode

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

SPI parameters

Most host processors with SPI hardware allow you to set the bit order, clock phase and polarity. For communication with all XBee/XBee-PRO SX RF Modules, the host processor must set these options as follows:

- Bit order: send MSB first
- Clock phase (CPHA): sample data on first (leading) edge
- Clock polarity (CPOL): first (leading) edge rises

All XBee/XBee-PRO SX RF Modules use SPI mode 0 and MSB first. Mode 0 means that data is sampled on the leading edge and that the leading edge rises. MSB first means that bit 7 is the first bit of a byte sent over the interface.

Serial port selection

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

When both the UART and SPI ports are enabled on power-up, all serial data goes out the UART. As soon as input occurs on either port, that port is selected as the active port and no input or output is allowed on the other port until the next device reset.

If you change the configuration so that only one port is configured, then that port is the only one enabled or used. If the parameters are written with only one port enabled, then the port that is not enabled is not used even temporarily after the next reset.

If both ports are disabled on reset, the device uses the UART in spite of the wrong configuration so that at least one serial port is operational.

Serial receive buffer

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

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

Serial transmit buffer

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

UART flow control

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

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

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

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

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

$\overline{\text{RTS}}$ 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.

The UART Data Present Indicator is a useful feature when using $\overline{\text{RTS}}$ flow control. When enabled, the DIO19 line asserts (low asserted) when UART data is queued to be transmitted from the device. For more information, see P9 (DIO19/SPI_ATTN).

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

AT commands

Special commands 71
MAC/PHY commands 72
Diagnostic commands - MAC statistics and timeouts 77
Network commands 79
Addressing commands 82
Diagnostic - addressing commands 85
Addressing discovery/configuration commands 86
Security commands 88
Serial interfacing commands 90
I/O settings commands 93
I/O sampling commands 106
I/O line passing commands 108
Sleep commands 113
Diagnostic - sleep status/timing commands 116
Command mode options 118
Firmware version/information commands 119

Special commands

The following commands are special commands.

AC (Apply Changes)

This command applies to the XBee/XBee-PRO SX RF Module.

Immediately applies new settings without exiting Command mode.

Parameter range

N/A

Default

N/A

FR (Software Reset)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

N/A

Default

N/A

RE (Restore Defaults)

This command applies to the XBee/XBee-PRO SX RF Module.

Restore device parameters to factory defaults.

Parameter range

N/A

Default

N/A

WR (Write)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

N/A

Default

N/A

MAC/PHY commands

The following AT commands are MAC/PHY commands.

AF (Available Frequencies)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Channels for these data rates are spaced 250 kHz apart.

Parameter range

0x7FF - 0x1F FFFF FFFF FFFF FFFF FFFF FFFF

Default

Region	BR	Bitfield	Operating frequency range	Channel spacing
United States / Canada	0/1	0x1F ffff FFFF ffff FFFF ffff FFFF	902,500 to 927,500 kHz	250 kHz
United States / Canada	2	0x03 FFFF ffff FFFF	902,750 to 927,250 kHz	500 kHz
Australia	0/1	0x01 FFFF ffff FFFF	915,500 to 927,500 kHz	250 kHz
Australia	2	0Xff FFFF	915,750 to 927,250 kHz	500 kHz
New Zealand	0/1	0x1FF ffff FFFF	917,500 to 927,500 kHz	250 kHz
New Zealand	2	0x7FF	922,250 to 927,250 kHz	500 kHz
Brazil	0/1	0x1F ffff FFFF ffff FFFF	902,625 to 907,125 kHz and 915,375 to 927,625 kHz	250 kHz
Brazil	2	0x07 ffff FFFF	902,750 to 906,350 kHz and 915,800 to 927,050 kHz	450 kHz
France	0/1	0x7f FFFF	915,250 to 920,750 kHz	250 kHz
France	2	0x0FFF	915,250 to 920,750 kHz	500 kHz

CM (Channel Mask)

This command applies to the XBee/XBee-PRO SX RF Module.

CM allows you to selectively enable or disable channels used for RF communication. This is useful to avoid using frequencies that experience unacceptable levels of RF interference, or to operate two networks of radios on separate frequencies.

When **CM** is queried, it returns the operating channel mask based on what value **BR** is set to. When **BR** is set to **2**, a fixed channel mask is used (see the defaults below). A user-defined **CM** value is only used when **BR** is set to **0** or **1**.

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

Note For Australia and New Zealand, **CM** is read-only.

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

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

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

The default **CM** mask spaces the channels across the entire 900 MHz band.

Parameter range

United States/Canada: 0x1 FFFF FFFF FFFF - 0x1F FFFF FFFF FFFF FFFF FFFF FFFF

Australia/New Zealand: Read-only

Brazil: 0x03 FFFF FFFF FFFF - 0x1F FFFF FFFF FFFF FFFF

Default

Country	Default CM when BR is 0 or 1	Default CM when BR is 2
United States/Canada	0x05 5555 5555 5555 5555 5555 5555	0x00 0000 0000 0003 FFFF FFFF FFFF
Australia	0x00 0000 0000 0000 01FF FFFF FFFF	0x00 0000 0000 0000 0000 00FF FFFF
New Zealand	0x00 0000 0000 0000 01FF FFFF FFFF	0x00 0000 0000 0000 0000 0000 07FF
Brazil	0x00 0000 001F FFFF FFFF FFF8 0000	0x00 0000 0000 0000 0007 FFFF FFFF

MF (Minimum Frequencies)

This command applies to the XBee/XBee-PRO SX RF Module.

This read-only command returns the number of hopping channels that the device uses to comply with its region of operation. You can use this information to determine which available frequencies you want to enable with the **CM** command. **MF** may vary depending on the **BR** setting.

Parameter range

read-only

Default

United States/Canada: 0x32 (50 channels)

The Default value of **MF** for the Australia/New Zealand firmware varies depending on the value you set **BR** to:

BR value	Australia default	New Zealand default	Brazil default
0	0x31 (49 channels)	0x29 (41 channels)	0 - 0x32 (50 channels)
1	0x31 (49 channels)	0x29 (41 channels)	1 - 0x32 (50 channels)
2	0x18 (24 channels)	0x0B (11 channels)	2 - 0x23 (35 channels)

HP (Preamble ID)

This command applies to the XBee/XBee-PRO SX RF Module.

The preamble ID for which the device communicates. Only devices with matching preamble IDs can communicate with each other. Different preamble IDs minimize interference between multiple sets of devices operating in the same vicinity. When receiving a packet, the device checks this before the network ID, as it is encoded in the preamble, and the network ID is encoded in the MAC header.

Parameter range

0 - 9 (usually)

0 - 7 (For some regions and values of **BR**. See BR (RF Data Rate) for details.)

Default

0

ID (Network ID)

This command applies to the XBee/XBee-PRO SX RF Module.

Set or read the user network identifier.

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

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

When receiving a packet, the device check this after the preamble ID. If you are using Original equipment manufacturer (OEM) network IDs, **0xFFFF** uses the factory value.

Parameter range

0 - 0x7FFF

Default

0x7FFF

MT (Broadcast Multi-Transmits)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 5

Default

3

BR (RF Data Rate)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets and reads the device's RF data rate (the rate at which the device transmits and receives RF data over-the-air).

Synchronous sleep is not supported when **BR** = **0**. All devices on the network must have the same **BR** value set in order to communicate. **BR** directly affects the range of the device. The higher the RF data rate, the lower the receive sensitivity.

BR also affects the number of hopping patterns (HP) and the number of channels based on region as shown in the following table:

BR	Region	# of hopping patterns (HP)	# of channels
0 or 1	Australia	10	49
0 or 1	Brazil	10	50
0 or 1	New Zealand	10	41
0 or 1	France	8	12
2	Australia	8	24
2	Brazil	10	35
2	New Zealand	8	11
2	France	8	12

Parameter range

0 - 2

Parameter	RF data rate	Receiver sensitivity
0	10 kb/s	-113 dBm
1	110 kb/s	-106 dBm
2	250 kb/s	-103 dBm

Default

2

PL (TX Power Level)

This command applies to the XBee/XBee-PRO SX RF Module.

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

The XBee-PRO SX requires the power supply to be above 3.3 V to ensure 30 dBm output power. The following table shows the typical values over supply voltage. If using a USB interface board with an XBee-PRO SX module, you must supply external DC power.

Power supply	Output power @ PL = 2
3.3 to 3.6 V	30 dBm typical
3.0 V	29 dBm typical
2.6 V	27 dBm typical

Note The device generates heat during RF transmission. There is an additional ground pad on the underside of the module that is used for heat dissipation. For more details, see PCB design and manufacturing.

Parameter range

0 - 2

Setting	XBee Tx power level	XBee-PRO Tx Power level
0	0 dBm	21.5 dBm
1	10 dBm	27 dBm
2	13 dBm	30 dBm

New Zealand specific:

Setting	Low/middle data rate	High data rate
0	0 dBm	0 dBm
1	10 dBm	7.5 dBm
2	13 dBm	7.5 dBm

Default

2

RR (Unicast Mac Retries)

This command applies to the XBee/XBee-PRO SX RF Module.

Set or read the maximum number of MAC level packet delivery attempts for unicasts. If **RR** is non-zero, the sent unicast packets request an acknowledgment from the recipient. Unicast packets can be retransmitted up to **RR** times if the transmitting device does not receive a successful acknowledgment.

Parameter range

0 - 0xF

Default

0xA (10 retries)

ED (Energy Detect)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x3A98 (15 seconds)

Default

N/A

Diagnostic commands - MAC statistics and timeouts

The following AT commands are MAC diagnostic commands and timeouts. Diagnostic commands are typically volatile and will not persist across a power cycle.

BC (Bytes Transmitted)

This command applies to the XBee/XBee-PRO SX RF Module.

The number of RF bytes transmitted. The firmware counts bytes in every retry and retransmission. A packet includes not only the payload, but also the preamble, the MAC header, the network header, the application header, encryption overhead, and the CRC. The purpose of this count is to estimate battery life by tracking time spent performing transmissions.

BC stops counting when it reaches a max value of **0xffffFFFF**. But, you can reset the counter to any 32-bit value—for example **0**—by appending a hexadecimal parameter to the command.

Parameter range

0 - 0xFFFFFFFF

Default

N/A (0 after reset)

DB (Last Packet RSSI)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

The RSSI measurement is accurate within ±2 dB from approximately -50 dBm down to sensitivity.

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

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

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

Parameter range

0x28 - 0x6E (-40 dBm to -110 dBm) [read-only]

Default

0

ER (Receive Count Error)

This command applies to the XBee/XBee-PRO SX RF Module.

This count increments when a device receives a packet that contains integrity errors of some sort. When the number reaches 0xFFFF, the firmware does not count further events.

To reset the counter to any 16-bit 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).

Occasionally random noise can cause this value to increment.

The **ER** parameter is not reset by pin, serial port or cyclic sleep modes.

Default

N/A

GD (Good Packets Received)

This command applies to the XBee/XBee-PRO SX RF Module.

This count increments when a device receives a good frame with a valid MAC header on the RF interface. Received MAC ACK packets do not increment this counter. Once the number reaches 0xFFFF, it does not count further events.

To reset the counter to any 16-bit unsigned value, append a hexadecimal parameter to the command.

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

Parameter range

0 - 0xFFFF

Default

N/A

EA (MAC ACK Failure Count)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0 - 0xFFFF

Default

N/A

TR (Transmission Failure Count)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0xFFFF

Default

N/A

UA (Unicasts Attempted Count)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0 - 0xFFFF

Default

N/A

%H (MAC Unicast One Hop Time)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

[read-only]

Default

N/A

%8 (MAC Broadcast One Hop Time)

This command applies to the XBee/XBee-PRO SX RF Module.

Parameter range

[read-only]

Default

N/A

Network commands

The following commands are network commands.

CE (Routing / Messaging Mode)

This command applies to the XBee/XBee-PRO SX RF Module.

The routing and messaging mode of the device.

A routing device repeats broadcasts. Indirect Messaging Coordinators do not transmit point-to-multipoint unicasts until an end device requests them. Setting a device as a poller causes it to regularly send polls to its Indirect Messaging Coordinator. Nodes can also be configured to route, or not route, multi-hop packets.

Sets or displays the behavior (End Device versus Coordinator) of the device.

Sets or displays whether the device is a coordinator.

Parameter range

0 - 6

Parameter	Description	Routes packets
0	Standard router	Yes
1	Indirect message coordinator	Yes
2	Non-routing device	No
3	Non-routing coordinator	No
4	Indirect message poller	Yes
5	N/A	N/A
6	Non-routing poller	No

Default

0

C8 (Compatibility Options)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the operational compatibility with older versions of the firmware.

Parameter range

4

Bit field:

Bit	Meaning	Setting	Description
2	Encryption compatibility	0	When encryption is enabled, AES Counter mode is used with a 256-bit key.
		1	When encryption is enabled AES CBC mode is used with a 256-bit key. This bit should be set when using encryption with versions of the firmware prior to 9009. It may be used in all circumstances if set consistently throughout the network, but it disables the more secure counter mode encryption.

All other bits should be 0 for future compatibility.

Default

0x04

BH (Broadcast Hops)

This command applies to the XBee/XBee-PRO SX RF Module.

The maximum transmission hops for broadcast data transmissions. This will not affect Point-to-Multipoint transmissions (**TO** = 40).

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

Parameter range

0 - 0x20

Default

0

NH (Network Hops)

This command applies to the XBee/XBee-PRO SX RF Module.

The maximum number of hops expected to be seen in a network route. This value limits the number of hops used in a route discovery; thereby, it limits the number of hops allowed in a unicast.

Additionally, **NH** is used to calculate the timeouts for a network acknowledgment.

Parameter range

1 - 0x20 (1 - 32 hops)

Default

7

MR (Mesh Unicast Retries)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

We recommend that you set this value to **1**.

Parameter range

0 - 7 mesh unicast retries

Default

1

NN (Network Delay Slots)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

1 - 0x5 network delay slots

Default

3

Addressing commands

The following AT commands are addressing commands.

SH (Serial Number High)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

SL (Serial Number Low)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

DH (Destination Address High)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

0x000000000000FFFF is the broadcast address.

Parameter range

0 - 0xFFFFFFFF

Default

0

DL (Destination Address Low)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

0x000000000000FFFF is the broadcast address.

Parameter range

0 - 0xFFFFFFFF

Default

0xFFFF

TO (Transmit Options)

This command applies to the XBee/XBee-PRO SX RF Module.

The bitfield that configures the transmit options for Transparent mode.

The device's transmit options. The device uses these options for all transparent transmissions. API transmissions can override this using the TxOptions field in the API frame.

DigiMesh is not supported if **BR** is set to **0**, transmitted packets will be sent as point-to-point/multipoint in this case.

Parameter range

0 - 0xFF

Bit	Meaning	Description
6,7	Delivery method	b’00 = <invalid option> b’01 = Point-multipoint b'10 = Repeater mode—directed broadcast of packets b’11 = DigiMesh—not available when **BR** = **0**

When you set **BR** to **0** the **TO** option has the DigiMesh and Repeater mode disabled automatically.

Default

0x40 When **BR** = **0**

0xC0 When **BR** = **1**

NI (Node Identifier)

This command applies to the XBee/XBee-PRO SX RF Module.

Stores the node identifier string for a device, which is a user-defined name or description of the device. This can be up to 20 ASCII characters.

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

Parameter range

A string of case-sensitive ASCII printable characters from 0 to 20 bytes in length. A carriage return or a comma automatically ends the command.

Default

0x20 (an ASCII space character)

NT (Network Discovery Back-off)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays 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.

Parameter range

0x20 - 0x2EE0 (x 100 ms)

Default

0x82 (13 seconds)

NO (Network Discovery Options)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0x0 - 0x7 (bit field)

Bit field

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

Default

0x0

CI (Cluster ID)

This command applies to the XBee/XBee-PRO SX RF Module.

The application layer cluster ID value. The device uses this value as the cluster ID for all data transmissions.

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

Supported Cluster IDs

0x11: Transparent data (default)

0x12: Loopback

0x14: Test Link Request

0x94: Test Link Result

0x23: Memory Access (GPM)

Parameter range

0 - 0xFFFF

Default

0x11 (Transparent data cluster ID)

Diagnostic - addressing commands

The following AT command is a Diagnostic - addressing command.

N? (Network Discovery Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

[read-only]

Default

N/A

R? (Region)

This command applies to the XBee/XBee-PRO SX RF Module.

Indicates the region of the module.

- USA/Canada = 1
- Australia = 2

- Brazil = 3
- New Zealand = 6

Parameter range

1, 2, 3, 6 [read-only]

Default

N/A

Addressing discovery/configuration commands

AG (Aggregator Support)

This command applies to the XBee/XBee-PRO SX RF Module.

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

- The receiving node establishes a DigiMesh route back to the originating node, if there is space in the routing table.
- The **DH** and **DL** of the receiving node update to the address of the originating node if the **AG** parameter matches the current **DH**/**DL** of the receiving node.
- API-enabled devices with updated **DH** and **DL** send an Aggregate Addressing Update frame (0x8E) out the serial port.

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

Parameter range

Any 64-bit address

Default

N/A

DN (Discover Node)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

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

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

1. The receiving device returns 0xFFFE followed by its 64-bit extended addresses in a Remote AT Command Response- 0x97.
2. The device returns an **ERROR** message if it is given without a destination node (that is without a parameter) or if the given destination node does not respond within **N?** milliseconds.

Parameter range

20-byte ASCII string

Default

N/A

ND (Network Discover)

This command applies to the XBee/XBee-PRO SX RF Module.

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 Local AT Command Response - 0x88 or Remote AT Command Response- 0x97 frame, respectively.

The command reports the following information after a jittered time delay.

SH<CR> (4 bytes)

SL<CR> (4 bytes)

NI <CR> (variable, 0-20 bytes plus 0x00 character)

DEVICE_TYPE<CR> (1 byte)

STATUS<CR> (1 byte: reserved)

PROFILE_ID<CR> (2 bytes)

MANUFACTURER_ID<CR> (2 bytes)

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

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

After (**NT** * 100) milliseconds, the command ends by returning a <CR>.

If you send **ND** through a local AT Command (0x08) API frame, each network node returns a separate AT Command Response (0x88) or Remote Command Response (0x97) frame, respectively. The data consists of the bytes listed above without the carriage return delimiters. The **NI** string ends in a 0x00 null character.

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

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

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

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

Parameter range

N/A

Default

N/A

FN (Find Neighbors)

This command applies to the XBee/XBee-PRO SX RF Module.

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

MY<CR> (always 0xFFFE)

SH<CR>

SL<CR>

NI<CR> (Variable length)

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

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

STATUS<CR> (1 byte: reserved)

PROFILE_ID<CR> (2 bytes)

MANUFACTURER_ID<CR> (2 bytes)

DIGI DEVICE TYPE<CR> (4 bytes. Optionally included based on NO (Network Discovery Options) settings.)

RSSI OF LAST HOP<CR> (1 byte. Optionally included based on NO (Network Discovery Options) settings.)

<CR>

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

If you send the **FN** command through a local AT Command (0x08) or remote AT command (0x17) API frame, each response returns as a separate AT Command Response (0x88) or Remote Command Response (0x97) frame, respectively. The data consists of the bytes in the previous list without the carriage return delimiters. The **NI** string ends in a 0x00 null character.

Parameter range

N/A

Default

N/A

Security commands

The following AT commands are security commands.

EE (Encryption Enable)

This command applies to the XBee/XBee-PRO SX RF Module.

Enable or disable 256-bit Advanced Encryption Standard (AES) encryption.

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

See C8 (Compatibility Options) that affects whether to use CBC mode or CTR mode encryption. A 256-bit AES key is used in both instances.

Parameter range

0 - 1

Parameter	Description
0	Encryption Disabled
1	Encryption Enabled

Default

0

KY (AES Encryption Key)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets the 256-bit 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

256-bit value (64 Hexadecimal digits)

Default

N/A

0

KZ (Remote AT Command Password)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets a password to inhibit transmission and reception of remote AT commands without including the password in the request. The password is write-only and cannot be changed without knowing the current password. When you set **KZ**, it disables Remote AT Command Request - 0x17 frames and you must use Secure Remote AT Command Request frame - 0x18 frames.

To change the password, enter two ASCII strings of 16 characters or less, separated by a space character. The first string is the old password and the second string is the new password. All ASCII characters other than white space characters and comma (**,**) are allowed in a password. Although we do not recommend it, you can include non-printable characters in the password if you enter them in API mode. The software allows all characters that are not password delimiters (SP, comma, CR, LF, TAB, vertical tab, and form feed), as part of the password.

Each device has its own password, which may or may not match passwords on another device in the network.

You must enable encryption when using a password so that even if the password is reset to the default value of **0** by using of RE (Restore Defaults), access to an active network is still inhibited due to the network key (**KY**) also being lost.

Parameter range

3 to 33 ASCII characters (a minimum of three characters are required to enter two passwords separated by a space)

Default

0 in ASCII.

Note When **KZ** is set to any value other than the default, it cannot be read. However, when **KZ** is set to the default value it reads **0**.

Serial interfacing commands

The following AT commands are serial interfacing commands.

BD (Interface Data Rate)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets and reads the serial interface data rate (baud rate) between the device and the host. The baud rate is the rate that the host sends serial data to the device.

When you make an update to the interface data rate, the change does not take effect until the host issues the **CN** command and the device returns the **OK** response.

The **BD** parameter does not affect the RF data rate. If you set the interface data rate higher than the RF data rate, you may need to implement a flow control configuration.

The range between standard and non-standard baud rates (0x9B - 0x4B0) is invalid. The range between 0x2580 and 0x4AFF is also invalid.

Non-standard interface data rates

The firmware interprets any value within 0x4B0 - 0x2580 and 0x4B00 - 0x1C9468 as an actual baud rate. When the host sends a value above 0x4B0, the firmware stores the closest interface data rate represented by the number in the **BD** register. For example, to set a rate of 19200 b/s, send the following command line: **ATBD4B00**.

Note When using XCTU, you can only set and read non-standard interface data rates using the XCTU Serial Console tool. You cannot access non-standard rates through the configuration section of XCTU.

When you send the **BD** command with a non-standard interface data rate, the UART adjusts to accommodate the interface rate you request. In most cases, the clock resolution causes the stored **BD** parameter to vary from the sent parameter. Sending **ATBD** without an associated parameter value returns the value actually stored in the device's **BD** register.

The following table provides the parameters sent versus the parameters stored.

BD parameter sent (HEX)	Interface data rate (b/s)	BD parameter stored (HEX)
0	1200	0
4	19,200	4
7	115,200	7
1C200	115,200	1B207

Parameter ranges

0 - xA (standard rates)

0x4B0 - 0x1C9468 (non-standard rates; 0x2581 to 0x4AFF not supported)

Parameter	Configuration (b/s)
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200
8	230400
9	460800
10	921600

Default

3

NB (Parity)

This command applies to the XBee/XBee-PRO SX RF Module.

Set or read the serial parity settings for UART communications.

Parameter range

0x00 - 0x02

Parameter	Description
0x00	No parity
0x01	Even parity
0x02	Odd parity

Parameter	Description
0	No parity
1	Even parity
2	Odd parity

Default

0x00

SB (Stop Bits)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the number of stop bits in the data packet.

Parameter range

0 - 1

Parameter	Configuration
0	One stop bit
1	Two stop bits

Default

0

RO (Packetization Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

Set or read the number of character times of inter-character silence required before transmission begins when operating in Transparent mode.

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

Parameter range

0 - 0xFF (x character times)

Default

3

FT (Flow Control Threshold)

This command applies to the XBee/XBee-PRO SX RF Module.

Set or display the flow control threshold.

The device de-asserts $\overline{\text{CTS}}$ when **FT** bytes are in the UART receive buffer. It re-asserts $\overline{\text{CTS}}$ when less than **FT**-16 bytes are in the UART receive buffer.

Parameter range

0x11 - 0x16F bytes

Default

0x13F

AP (API Enable)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

Parameter range

0 - 2

Parameter	Description
0	Transparent mode, API mode is off. All UART input and output is raw data and the device uses the **RO** and **RB** parameters to delineate packets.
1	API Mode Without Escapes. The device packetizes all UART input and output data in API format, without escape sequences.
2	API Mode With Escapes. The device is in API mode and inserts escaped sequences to allow for control characters. The device passes XON (0x11), XOFF (0x13), Escape (0x7D), and start delimiter 0x7E as data.

Default

0

AO (API Options)

This command applies to the XBee/XBee-PRO SX RF Module.

The API data frame output format for RF packets received.

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

Parameter range

0 - 2

Parameter	Description
0	API Rx Indicator - 0x90, this is for standard data frames.
1	API Explicit Rx Indicator - 0x91, this is for Explicit Addressing data frames.
2	XTend DigiMesh API Rx Indicator - 0x80

Default

0

I/O settings commands

The following AT commands are I/O settings commands.

D0 (DIO0/AD0)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the DIO0/AD0 configuration (pin 33).

Parameter range

0 - 5

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

Default

1

D1 (DIO1/AD1)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO1/AD1 configuration (pin 32).

Parameter range

0, 2 - 5

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

Default

0

D2 (DIO2/AD2)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO2/AD2 configuration (pin 31).

Parameter range

0, 2 - 5

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

Default

0

D3 (DIO3/AD3)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO3/AD3 configuration (pin 30).

Parameter range

0, 2 - 5

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

Default

0

D4 (DIO4)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO4 configuration (pin 24).

Parameter range

0, 2 - 5

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

Default

0

D5 (DIO5/ASSOCIATED_INDICATOR)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO5/ASSOCIATED_INDICATOR configuration (pin 28).

Parameter range

0, 1, 3 - 5

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

Default

1

D6 (DIO6/RTS)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO6/$\overline{\text{RTS}}$ configuration (pin 29).

Parameter range

0, 1, 3 - 5

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

Default

0

D7 (DIO7/CTS)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration (pin 25).

Parameter range

0, 1, 3 - 7

Parameter	Description
0	Disabled
1	$\overline{\text{CTS}}$ flow control
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high
6	RS-485 Tx enable, low Tx (0 V on transmit, high when idle)
7	RS-485 Tx enable high, high Tx (high on transmit, 0 V when idle)

Default

0x1

D8 (DIO8/SLEEP_REQUEST)

This command applies to the XBee/XBee-PRO SX RF Module.

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

This line is also used with Pin Sleep, but pin sleep ignores the **D8** configuration. It is always used to control pin sleep, regardless of configuration of **D8**.

Parameter range

0, 1, 3 - 5

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

Default

1

D9 (DIO9/ON_SLEEP)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO9/ON_$\overline{\text{SLEEP}}$ configuration (pin 26).

Parameter range

0, 1, 3 - 6

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

Default

1

P0 (DIO10/RSSI/PWM0 Configuration)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the PWM0/RSSI/DIO10 configuration (pin 7).
When configured as a PWM output, you can use **M0** to set the PWM duty cycle.

Parameter range

0 - 5

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

Default

1

P1 (DIO11/PWM1 Configuration)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO11/PWM1 configuration (pin 8).
When configured as a PWM1 output, you can use **M1** to set the PWM duty cycle.

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	32.768 kHz clock output
2	PWM1 output
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P2 (DIO12 Configuration)

This command applies to the XBee/XBee-PRO SX RF Module.
Sets or displays the DIO12 configuration (pin 5).

Parameter range

0, 3 - 6

Parameter	Description
0	Disabled
1	N/A
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high
6	RX LED

Default

0

P3 (DIO13/DOUT)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the DIO13/DOUT configuration (pin 3).

Parameter range

0, 1

Parameter	Description
0	Disabled
1	UART DOUT enabled

Default

1

P4 (DIO14/DIN)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the DIO14/DIN/$\overline{\text{CONFIG}}$ configuration (pin 4).

Sets or displays the DIO14/DIN configuration (pin 4).

The device enters Command mode at 9600 baud if you enable DIN and either of the following conditions is met:

- A six second serial break is received during normal operation.
- DIN is driven low upon power up or reset.

The device sends an **OK** response out of the UART when it enters Command mode in this way.

Parameter range

0 - 1

Parameter	Description
0	Disabled
1	UART DIN/$\overline{\text{CONFIG}}$ enabled

Default

1

P5 (DIO15/SPI_MISO)

This command applies to the XBee/XBee-PRO SX RF Module. Sets or displays the DIO15/SPI_MISO configuration (pin 17).

Parameter range

0, 1, 4, 5

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

Default

1

P6 (SPI_MOSI)

This command applies to the XBee/XBee-PRO SX RF Module. Sets or displays the DIO16/SPI_MOSI configuration (pin 16).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_MOSI
2	N/A
3	N/A

Parameter	Description
4	Digital output low
5	Digital output, high

Default

1

P7 (DIO17/SPI_SSEL)

This command applies to the XBee/XBee-PRO SX RF Module. Sets or displays the DIO17/SPI_$\overline{\text{SSEL}}$ configuration (pin 15).

Parameter range

0, 1, 4, 5

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

Default

1

P8 (DIO18/SPI_SCLK)

This command applies to the XBee/XBee-PRO SX RF Module. Sets or displays the DIO18/SPI_SCLK configuration (pin 14).

Parameter range

0, 1, 4, 5

Parameter	Description
0	Disabled
1	SPI_SCLK
2	N/A
3	N/A

Parameter	Description
4	Digital output low
5	Digital output high

Default

1

P9 (DIO19/SPI_ATTN)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the DIO19/SPI_$\overline{\text{ATTN}}$ configuration (pin 12).

Parameter range

0, 1, 4 - 6

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

Default

1

PD (Pull Direction)

This command applies to the XBee/XBee-PRO SX RF Module.

The resistor pull direction bit field (1 = pull-up, 0 = pull-down) for corresponding I/O lines that are set by the **PR** command.

Parameter range

0x0 - 0xFFFFF

Default

0xFFFFF

PR (Pull-up/Down Resistor Enable)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

Bit	I/O line	Module pin
0	DIO4/AD4	24
1	DIO3/AD3	30
2	DIO2/AD2	31
3	DIO1/AD1	32
4	DIO0/AD0	33
5	DIO6/$\overline{\text{RTS}}$	29
6	DIO8/SLEEP_REQUEST	9
7	DIO14/DIN/$\overline{\text{CONFIG}}$	4
8	DIO5/ASSOCIATE	28
9	DIO9/On/$\overline{\text{SLEEP}}$	26
10	DIO12	5
11	DIO10/RSSI/PWM0	7
12	DIO11/PWM1	8
13	DIO7/$\overline{\text{CTS}}$	25
14	DIO13/DOUT	3
15	DIO15/SPI_MISO	17
16	DIO16/SPI_MOSI	16
17	DIO17/SPI_$\overline{\text{SSEL}}$	15
18	DIO18/SPI_SCLK	14
19	DIO19/SPI_$\overline{\text{ATTN}}$	12

Parameter range

0 - 0xFFFFF (bit field)

Default

0xFFFFF

M0 (PWM0 Duty Cycle)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Use the **P0** command to configure the line as a PWM output.

Parameter range

0 - 0x3FF

Default

0

M1 (PWM1 Duty Cycle)

This command applies to the XBee/XBee-PRO SX RF Module.

The duty cycle of the PWM1 line (pin 8).

Use the **P1** command to configure the line as a PWM output.

Parameter range

0 - 0x3FF

Default

0

LT (Associated LED Blink Time)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

Parameter range

0x14 - 0xFF (x 10 ms)

Default

0

RP (RSSI PWM Timer)

This command applies to the XBee/XBee-PRO SX RF Module.

The PWM timer expiration in 0.1 seconds. **RP** sets the duration of pulse width modulation (PWM) signal output on the RSSI pin. The signal duty cycle updates with each received packet and shuts off when the timer expires.

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

Parameter range

0 - 0xFF (x 100 ms)

Default

0x28 (four seconds)

I/O sampling commands

The following AT commands configure I/O sampling parameters.

AV (Analog Voltage Reference)

This command applies to the XBee/XBee-PRO SX RF Module.

The analog voltage reference used for A/D sampling.

Parameter range

0, 1

Parameter	Description
0	1.25 V reference
1	2.5 V reference

Default

1

IC (DIO Change Detect)

This command applies to the XBee/XBee-PRO SX RF Module.

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

IC works with the individual pin configuration commands (**D0** - **D9**, **P0** - **P2**). If you enable a pin as a digital I/O, use the **IC** command to force an immediate I/O sample transmission when the DIO state changes. If sleep is enabled, the edge transition must occur during a wake period to trigger a change detect.

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

Set unused bits to 0.

Bit	I/O line	Module pin
0	DIO0	33
1	DIO1	32
2	DIO2	31
3	DIO3	30
4	DIO4	24
5	DIO5	28
6	DIO6	29
7	DIO7	25
8	DIO8	10

Bit	I/O line	Module pin
9	DIO9	26
10	DIO10	7
11	DIO11	8
12	DIO12	5

Parameter range

0 - 0xFFFF (bit field)

Default

0

IF (Sleep Sample Rate)

This command applies to the XBee/XBee-PRO SX RF Module.

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

In addition, setting **IF** to zero allows I/O samples to occur before the device goes to sleep and to occur thereafter every wake cycle specified by **IR**.

To enable periodic sampling, set **IR** to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin. The sample rate is measured in milliseconds. For more information, see the following commands:

Parameter range

0 - 0xFF

Default

1

IR (Sample Rate)

This command applies to the XBee/XBee-PRO SX RF Module.

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

To enable periodic sampling, set **IR** to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin (see D0 (DIO0/AD0) -D9 (DIO9/ON_SLEEP), P0 (DIO10/RSSI/PWM0 Configuration)- P2 (DIO12 Configuration).

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

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

IS (Immediate Sample)

This command applies to the XBee/XBee-PRO SX RF Module.

Immediately forces an I/O sample to be generated. If you issue the command to the local device, the sample data is sent out the local UART. If sent remotely, the sample data is returned as a AT Command Response - 0x88.

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

Parameter range

N/A

Default

N/A

TP (Board Temperature)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

This is a read-only parameter

Default

N/A

%V (Voltage Supply Monitoring)

This command applies to the XBee/XBee-PRO SX RF Module.

Displays the supply voltage of the device in mV units.

Parameter range

This is a read-only parameter

Default

N/A

I/O line passing commands

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

I/O Line Passing allows the digital and analog inputs of a remote device to affect the corresponding outputs of the local device.

You can perform Digital Line Passing on any of the Digital I/O lines. Digital Inputs directly map to Digital Outputs of each digital pin.

Analog Line Passing can be performed only on the first two ADC lines:

- ADC0 corresponds with PWM0
- ADC1 corresponds with PWM1

IU (I/O Output Enable)

This command applies to the XBee/XBee-PRO SX RF Module.

Enable or disable I/O data received to be sent out UART/SPI using an API frame when **AP** = 1 or 2 and when I/O line passing is enabled.

Parameter range

0 - 1

Parameter	Description
0	Disabled
1	Enabled

Default

1

IA (I/O Input Address)

This command applies to the XBee/XBee-PRO SX RF Module.

The source address of the device to which outputs are bound. Setting all bytes to 0xFF disables I/O line passing. Setting **IA** to 0xFFFF allows any I/O packet addressed to this device (including broadcasts) to change the outputs.

The source address of the device to which outputs are bound. If an I/O sample is received from the address specified, any pin that is configured as a digital output or PWM changes its state to match that of the I/O sample.

Set **IA** to 0xFFFFFFFFFFFFFFFF to disable I/O line passing.

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

Parameter range

0 - 0xFFFF FFFF FFFF FFFF

Default

0xFFFFFFFFFFFFFFFF (I/O line passing disabled)

T0 (D0 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T1 (D1 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T2 (D2 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T3 (D3 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T4 (D4 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T5 (D5 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T6 (D6 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T7 (D7 Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T8 (D8 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T9 (D9 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q0 (P0 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q1 (P1 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q2 (P2 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q3 (P3 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

Specifies how long pin P3 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q4 (P4 Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

Specifies how long pin P4 holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

PT (PWM Output Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0x1770 (x 100 ms)

Default

0xFF

Sleep commands

The following AT commands are sleep commands.

SM (Sleep Mode)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the sleep mode of the device.

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

Synchronous modes are unavailable if **BR** = **0** and are incompatible with asynchronous modes.

Parameter range

0, 1, 4, 5, 7, 8

Parameter	Description
0	Normal.
1	Asynchronous Pin Sleep. In this mode, the SLEEP_RQ line controls the sleep/wake state of the device.
2	N/A
3	N/A
4	Asynchronous Cyclic Sleep. In this mode, the device periodically sleeps and wakes based on the **SP** and **ST** commands.
5	Asynchronous Cyclic Sleep Pin Wake. When you assert the SLEEP_RQ pin, the device enters a cyclic sleep mode similar to Asynchronous Cyclic Sleep. When you de-assert the SLEEP_RQ pin, the device immediately wakes up. The device does not sleep when you de-assert the SLEEP_RQ pin.
6	N/A
7	Sleep Support—this may also be used for synchronous pulse.
8	Synchronized Cyclic Sleep—this may also be used for synchronous pulse.

Default

0

SO (Sleep Options)

This command applies to the XBee/XBee-PRO SX RF Module.

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

0 - 0x13E

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

Bit	Option
0	Preferred sleep coordinator; setting this bit causes a sleep compatible device to always act as sleep coordinator
1	Non-sleep coordinator; setting this bit causes a device to never act as a sleep coordinator
2	Enable API sleep status messages
3	Disable early wake-up for missed syncs
4	Enable node type equality (disables seniority based on device type)

Bit	Option
5	Disable coordinator rapid sync deployment mode
8	Always wake for **ST** time.
10	Enable sync sleep randomize I/O sampling

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

Bit	Option
8	Always wake for **ST** time
10	Enable sync sleep randomize I/O sampling

Default

0x2 (non-sleep coordinator)

SN (Number of Cycles Between ON_SLEEP)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

This command does not work with synchronous sleep devices.

Parameter range

1 - 0xFFFF

Default

1

Example

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

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

SP (Sleep Time)

This command applies to the XBee/XBee-PRO SX RF Module.

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

A time of **0** is used to indicate no sleep at all and is used for synchronous pulse on **DIO9**.

Parameter range

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

Default

0x12C (3 seconds)

ST (Wake Time)

This command applies to the XBee/XBee-PRO SX RF Module.

Sets or displays the wake time of the device.

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

For devices in synchronous sleep or synchronous pulse, **ST** defines the amount of time that a device stays awake when operating in cyclic sleep mode. The command adjusts the value upwards automatically if it is too small to function properly based on other settings.

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

Parameter range

0x1 - 0x36EE80 (x 1 ms) (one hour)

Default

0xD08 (3.3 seconds)

WH (Wake Host Delay)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

Diagnostic - sleep status/timing commands

The following AT commands are Diagnostic sleep status/timing commands.

SS (Sleep Status)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Bit	Description
0	This bit is true when the network is in its wake state.
1	This bit is true if the node currently acts as a network sleep coordinator.
2	This bit is true if the node ever receives a valid sync message after it powers on.
3	This bit is true if the node receives a sync message in the current wake cycle.
4	This bit is true if you alter the sleep settings on the device so that the node nominates itself and sends a sync message with the new settings at the beginning of the next wake cycle.
5	This bit is true if you request that the node nominate itself as the sleep coordinator using the Commissioning Pushbutton or the **CB**2 command.
6	This bit is true if the node is currently in deployment mode.
All other bits	Reserved. Ignore all non-documented bits.

Parameter range

N/A

[read-only]

Default

N/A

OS (Operating Sleep Time)

This command applies to the XBee/XBee-PRO SX RF Module.

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

If the device synchronizes with a sleeping router network, **OS** may differ from **SP**.

Parameter range

N/A

Default

N/A

OW (Operating Wake Time)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

If the device synchronizes with a sleeping router network, **OW** may differ from **ST**.

Parameter range

N/A

Default

N/A

MS (Missed Sync Messages)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

N/A

Default

N/A

SQ (Missed Sleep Sync Count)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Set the value to 0 to reset this value.

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

Parameter range

0 - 0xFFFF

Default

N/A

Command mode options

The following commands are Command mode option commands.

CC (Command Sequence Character)

This command applies to the XBee/XBee-PRO SX RF Module.

The character value the device uses to enter Command mode.

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

Parameter range

0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

Default

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

CT (Command Mode Timeout)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

2 - 0x1770 (x 100 ms)

Default

0x64 (10 seconds)

CN (Exit Command Mode)

This command applies to the XBee/XBee-PRO SX RF Module.

Immediately exits Command Mode and applies pending changes.

Parameter range

N/A

Default

N/A

GT (Guard Times)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0x2 - 0x3E8 (x 1 ms)

Default

0x3E8 (one second)

Firmware version/information commands

The following AT commands are firmware version/information commands.

VR (Firmware Version)

This command applies to the XBee/XBee-PRO SX RF Module.

Reads the firmware version on a device.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in the factory

HV (Hardware Version)

This command applies to the XBee/XBee-PRO SX RF Module.

Display the hardware version number of the device.

Parameter range

0 - 0xFFFF [read-only]

Default

Set in firmware

HS (Hardware Series)

This command applies to the XBee/XBee-PRO SX RF Module.

Read the device's hardware series number.

Parameter range

0 - 0xFFFF [read-only]

Default

0x0A00 - set in the firmware

DD (Device Type Identifier)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

0 - 0xFFFFFFFF

Default

0xF0000

NP (Maximum Packet Payload Bytes)

This command applies to the XBee/XBee-PRO SX RF Module.

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

Parameter range

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

Default

0x100

CK (Configuration CRC)

This command applies to the XBee/XBee-PRO SX RF Module.

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

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

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

Parameter range

N/A

Default

N/A

Operate in API mode

API mode overview 123
Use the AP command to set the operation mode 123
API frame format 123

API mode overview

By default, the XBee/XBee-PRO SX RF Module acts as a serial line replacement (Transparent operation), it queues all UART data that it receive through the DI pin for RF transmission. When the device receives an RF packet, it sends the data out the DO pin with no additional information.

The following behaviors are inherent to Transparent operation:

- If device parameter registers are to be set or queried, a special operation is required for transitioning the device into Command 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 destination of individual data packets or when you need to know which node a data packet was sent 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.

Use the AP command to set the operation mode

Use AP (API Enable) to specify the operation mode:

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

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

API frame format

An API frame consists of the following:

- Start delimeter
- Length
- Frame data
- Checksum

API operation (AP parameter = 1)

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

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

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

API operation with escaped characters (AP parameter = 2)

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

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

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

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

Escaped characters in API frames

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

The following data bytes need to be escaped:

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

To escape a character:

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

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

Example: escape an API frame

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

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

You must escape the 0x13 byte:

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

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

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

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

Length field

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

Frame data

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

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

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

Multi-byte values are sent big-endian.

Calculate and verify checksums

To calculate the checksum of an API frame:

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

To verify the checksum of an API frame:

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

Example

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

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

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

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

Add these hex bytes:

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

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

If an API data packet is composed with an incorrect checksum, the XBee/XBee-PRO SX RF Module will consider the packet invalid and will ignore the data.

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

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

API frames

The following sections document API frame types.

API frame exchanges 128
Code to support future API frames 130
64-bit Transmit Request - 0x00 131
Local AT Command Request - 0x08 133
Queue Local AT Command Request - 0x09 135
Transmit Request - 0x10 137
Explicit Addressing Command Request - 0x11 140
Remote AT Command Request - 0x17 144
Secure Remote AT Command Request frame - 0x18 147
Modem Status - 0x8A 150
Extended Transmit Status - 0x8B 152
Route Information - 0x8D 154
Aggregate Addressing Update - 0x8E 156
64-bit Receive Packet - 0x80 158
Local AT Command Response - 0x88 160
Transmit Status - 0x89 162
Receive Packet - 0x90 164
Explicit Receive Indicator - 0x91 166
Node Identification Indicator - 0x95 168
Remote AT Command Response- 0x97 171

API frame exchanges

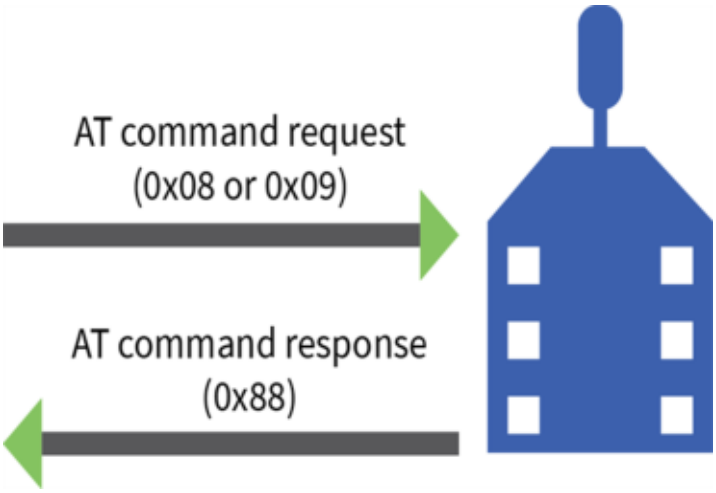
Every outgoing API frame has a corresponding response (or ACK) frame that indicates the success or failure of the outgoing API frame. This section details some of the common API exchanges that occur.

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

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

AT commands

The following image shows the API frame exchange that takes place on the serial interface (UART or SPI) when you send a 0x08 AT Command Request or 0x09 AT Command-Queue Request to read or set a device parameter. To disable the 0x88 AT Command Response, set the frame ID to 0 in the request.



Transmit and Receive RF data

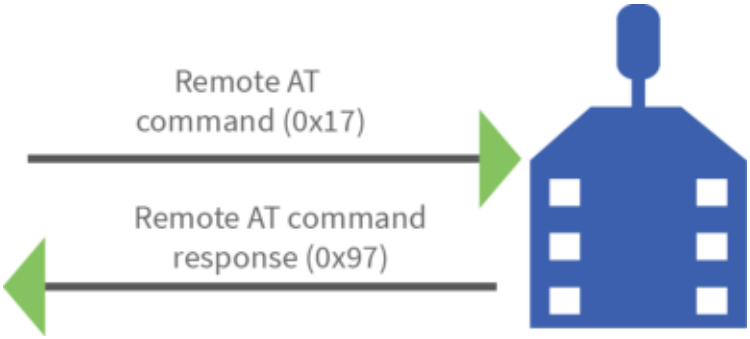
The following image shows the API exchanges that take place on the serial interface when a device sends a 0x10, or 0x11 Transmit Request to another device.

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

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

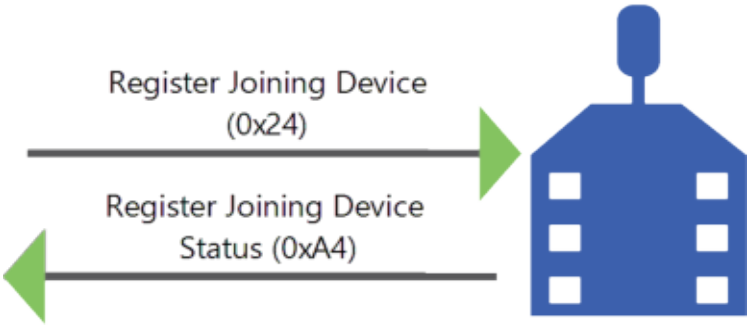
Remote AT commands

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



Device Registration

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



Code to support future API frames

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

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

               case RX_IO_SAMPLE_FRAME:
                       //process IO sample frame
                       break;

               case NODE_IDENTIFICATION_FRAME:
                       //process node identification frame
                       break;


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

64-bit Transmit Request - 0x00

Response frame: Transmit Status - 0x89

Description

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

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

Format

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

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

Examples

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

64-bit unicast

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

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

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

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

64-bit broadcast

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

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

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

Local AT Command Request - 0x08

Response frame: Local AT Command Response - 0x88

Description

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

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

Format

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

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

Examples

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

Set the local command parameter

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

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

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

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

Query local command parameter

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

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

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

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

Queue Local AT Command Request - 0x09

Response frame: Local AT Command Response - 0x88

Description

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

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

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

Format

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

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

Examples

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

Queue setting local command parameter

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

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

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

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

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

Query local command parameter

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

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

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

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

Transmit Request - 0x10

Response frame: Extended Transmit Status - 0x8B

Description

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

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

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

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node

Format

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

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

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

Transmit options bit field

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

0	Disable ACK [**0x01**]	Disable acknowledgments on all unicasts.
1	Disable route discoveries [**0x02**]	Disable Route Discovery on all DigiMesh unicasts.
2	Unicast NACK [**0x04**]	Enable unicast NACK messages on DigiMesh transmissions When set, a failed transmission will generate a Route Information - 0x8D frame for diagnosis.
3	Unicast trace route [**0x08**]	Enable a unicast Trace Route on DigiMesh transmissions When set, the transmission will generate a Route Information - 0x8D frame.
5	Reserved	<set this bit to 0>
6,7	Delivery method	b'00 = If **BR** = **0** then packet type will be Point-multipoint otherwise it will be DigiMesh b'01 = Point-multipoint [**0x40**] b'10 = Repeater mode, Directed Broadcast [**0x80**] b'11 = DigiMesh—not available when **BR** = **0** [**0xC0**]

Examples

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

64-bit unicast

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

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

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

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

64-bit broadcast

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

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

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

Explicit Addressing Command Request - 0x11

Response frame: Extended Transmit Status - 0x8B

Description

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

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

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

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node

Reserved endpoints

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

The active Digi endpoints are:

- **0xE8** - Digi Data endpoint
- **0xE6** - Digi Device Object (DDO) endpoint

Reserved cluster IDs

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

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

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

Reserved profile IDs

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

Format

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

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

Transmit options bit field

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

0	Disable ACK [**0x01**]	Disable acknowledgments on all unicasts.

1	Disable route discoveries [**0x02**]	Disable Route Discovery on all DigiMesh unicasts.
2	Unicast NACK [**0x04**]	Enable unicast NACK messages on DigiMesh transmissions When set, a failed transmission will generate a Route Information - 0x8D frame for diagnosis.
3	Unicast trace route [**0x08**]	Enable a unicast Trace Route on DigiMesh transmissions When set, the transmission will generate a Route Information - 0x8D frame.
5	Reserved	<set this bit to 0>
6,7	Delivery method	b'00 = If **BR** = **0** then packet type will be Point-multipoint otherwise it will be DigiMesh b'01 = Point-multipoint [**0x40**] b'10 = Repeater mode, Directed Broadcast [**0x80**] b'11 = DigiMesh—not available when **BR** = **0** [**0xC0**]

Examples

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

64-bit unicast

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

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

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

Frame type	Frame ID	64-bit dest	Reserved	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0x87	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0011	0xC105	0x00	0x00	0x547844617461
Explicit request	*Matches response*	*Destination*	*Unused*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Use **TO***	*"TxData"*

Loopback Packet

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

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

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

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

Frame type	Frame ID	64-bit dest	Reserved	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0xF8	0x0013A200 12345678	0xFFFE	0xE8	0xE8	0x0012	0xC105	0x00	0x01	0x547844617461
Explicit request	*Matches response*	*Destination*	*Unused*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Disable retries*	*"TxData"*

Remote AT Command Request - 0x17

Response frame: Remote AT Command Response- 0x97

Description

This frame type is used to query or set AT command parameters on a remote device.

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

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Request - **0x17**.
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device.
13	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
15	8-bit	**Remote command options**	Bit field of options that apply to the remote AT command request: ■ **Bit 0**: Disable ACK [**0x01**] ■ **Bit 1**: Apply changes on remote [**0x02**] • If not set, changes will not applied until the device receives an **AC** command or a subsequent

Offset	Size	Frame Field	Description
			command change is received with this bit set ■ Bit 2: Reserved (set to 0) ■ Bit 3: Reserved (set to 0) ■ **Bit 4**: Send the remote command securely [**0x10**] **Note** Option values may be combined. Set all unused bits to 0.
16	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
18-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

Set remote command parameter

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

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

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

Frame type	Frame ID	64-bit dest	Reserved	Command options	AT command	Parameter value
0x17	0x27	0x0013A200 12345678	0xFFFE	0x02	0x4E49	0x52656D6F7465
Request	*Matches response*		*Unused*	*Apply Change*	*"NI"*	*"Remote"*

Queue remote command parameter change

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

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

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

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

Query remote command parameter

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

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

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

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

Secure Remote AT Command Request frame - 0x18

Description

The 0x18 frame is like the 0x17 frame except that it requires a password to function; See KZ (Remote AT Command Password). Use this frame to securely send remote AT commands. The frame requires a correct password to be included as part of the 0x18 command.

If **KZ** is set:

- The 0x17 frame fails.
- The 0x18 frame only works if it includes the correct password and encryption is enabled.
- Remote commands from other nodes are only accepted if they were sent from a node which also had **KZ** set.

If **KZ** is not set:

- You may send 0x17 frames.
- You may send the 0x18 frame with the password field set with ASCII **0**.

Format

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

Frame data fields	Offset	Description
Frame type	3	0x18
Frame ID	4	Identifies the frame to correlate with a subsequent remote command response (0x97).
Password Length	5	Specifies how many bytes are in the Password field to follow. Does not include the Length byte itself.
Password	6	The password itself.
64-bit destination address	7+password length	The 64-bit address of the destination node which is receiving this command, MSB first, LSB last.
Reserved	15+password length	This field is unused for DigiMesh, but it can be used in protocols that support 16-bit addresses, for example Zigbee. For DigiMesh, the value of these two bytes does not matter.
Command Options	17+password length	If bit 1 is set (0x02), the remote node immediately applies changes in the AT command. If bit 1 is clear, you must send an **AC** command for the change to take effect.
AT Command	18+password length	Command name: two ASCII characters that identify the command.

Frame data fields	Offset	Description
Command Parameter	20+password length	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 table shows an example frame to send a secure remote AT command to a node at address **0x0013a200 12345678** to set the baud rate to **07**, which translates to 115,200 b/s. If the password configured with the **KZ** command is **Nugget**, encryption is enabled with **EE**, and the receiving node is also password protected for remote commands, the baud rate change will apply immediately.

Frame data fields	Offset	Example
Start delimiter	0	0x7E
Length	1	0x00 (MSB)
	2	0x17 (LSB)
Frame type	3	0x18
Frame ID	4	0x01
Password Length	5	0x06
Password	6	0x4E (N)
	7	0x75 (u)
	8	0x67 (g)
	9	0x67 (g)
	10	0x65 (e)
	11	0x74 (t)
64-bit destination address	12	0x00 (MSB)
	13	0x13
	14	0xA2
	15	0x00
	16	0x12
	17	0x34
	18	0x56
	19	0X78 (LSB)

Frame data fields	Offset	Example
Reserved	20	0xFF
	21	0xFE
Command Options	22	0x02 (apply changes)
AT Command	23	0x42 (B)
	24	0x44 (D)
Command Parameter	25	0x07
Checksum	26	0x21

Modem Status - 0x8A

Description

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

Format

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Modem Status - **0x8A**
4	8-bit	**Modem status**	Complete list of modem statuses: **0x00** = Hardware reset or power up **0x01** = Watchdog timer reset **0x02** = Joined network **0x03** = Left network **0x06** = Coordinator started **0x07** = Network security key was updated **0x0B** = Network woke up **0x0C** = Network went to sleep **0x0D** = Voltage supply limit exceeded **0x0E** = Digi Remote Manager connected **0x0F** = Digi Remote Manager disconnected **0x11** = Modem configuration changed while join in progress **0x12** = Access fault **0x13** = Fatal error **0x3B** = Secure session successfully established **0x3C** = Secure session ended **0x3D** = Secure session authentication failed **0x3E** = Coordinator detected a PAN ID conflict but took no action **0x3F** = Coordinator changed PAN ID due to a conflict **0x32** = BLE Connect **0x33** = BLE Disconnect **0x34** = Bandmask configuration failed **0x35** = Cellular component update started **0x36** = Cellular component update failed **0x37** = Cellular component update completed **0x38** = XBee firmware update started **0x39** = XBee firmware update failed **0x3A** = XBee firmware update applying **0x40** = Router PAN ID was changed by coordinator due to a conflict **0x42** = Network Watchdog timeout expired **0x80** through **0xFF** = Stack error

Offset	Size	Frame Field	Description
			Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Statuses for specific modem types are listed here.

Examples

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

Boot status

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

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

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

Extended Transmit Status - 0x8B

Request frames:

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

Description

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

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x8B**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
7	8-bit	**Transmit retry count**	The number of application transmission retries that occur.
8	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = MAC ACK failure **0x02** = CCA/LBT failure **0x03** = Indirect message unrequested / no spectrum available **0x21** = Network ACK failure **0x25** = Route not found **0x31** = Internal resource error **0x32** = Resource error lack of free buffers, timers, etc. **0x74** = Data payload too large **0x75** = Indirect message unrequested
9	8-bit	**Discovery**	Complete list of delivery statuses:

Offset	Size	Frame Field	Description
		status	**0x00** = No discovery overhead **0x02** = Route discovery
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Route Information - 0x8D

Request frames:

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

Description

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Route Information - **0x8D**
4	8-bit	**Source event**	Event that caused the route information to be generated: **0x11** = NACK **0x12** = Trace route
5	8-bit	**Data length**	The number of bytes that follow, excluding the checksum. If the length increases, new items have been added to the end of the list for future revisions.
6	32-bit	**Timestamp**	System timer value on the node generating the Route Information Packet.The timestamp is in microseconds. Only use this value for relative time measurements because the time stamp count restarts approximately every hour.
10	8-bit	**ACK timeout count**	The number of MAC ACK timeouts that occur.
11	8-bit	**TX blocked count**	The number of times the transmission was blocked due to reception in progress.
12	8-bit	**Reserved**	Not used.
14	64-bit	**Destination address**	The 64-bit IEEE address of the final destination node of this network-level transmission.
21	64-bit	**Source**	The 64-bit IEEE address of the source node of this network-level

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

Examples

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

Routing information

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

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

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

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

Aggregate Addressing Update - 0x8E

Description

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

Format

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

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

Examples

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

Aggregate address update

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

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

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

64-bit Receive Packet - 0x80

Request frames:

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

Description

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

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

Format

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

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

Offset	Size	Frame Field	Description
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

64-bit unicast

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

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

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

Local AT Command Response - 0x88

Request frames:

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

Description

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

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

Format

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

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

Examples

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

Set local command parameter

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

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

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

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

Query local command parameter

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

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

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

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

Transmit Status - 0x89

Request frames:

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

Description

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

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x89**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = No ACK received **0x02** = CCA failure **0x03** = Indirect message unrequested **0x04** = Transceiver was unable to complete the transmission **0x21** = Network ACK failure **0x22** = Not joined to network **0x2C** = Invalid frame values (check the phone number) **0x31** = Internal error **0x32** = Resource error - lack of free buffers, timers, etc. **0x34** = No Secure Session Connection **0x35** = Encryption Failure **0x74** = Message too long **0x76** = Socket closed unexpectedly **0x78** = Invalid UDP port **0x79** = Invalid TCP port

Offset	Size	Frame Field	Description
			0x7A = Invalid host address **0x7B** = Invalid data mode **0x7C** = Invalid interface. **0x7D** = Interface not accepting frames. **0x7E** = A modem update is in progress. Try again after the update is complete. **0x80** = Connection refused **0x81** = Socket connection lost **0x82** = No server **0x83** = Socket closed **0x84** = Unknown server **0x85** = Unknown error **0x86** = Invalid TLS configuration (missing file, and so forth) **0x87** = Socket not connected **0x88** = Socket not bound Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Delivery status codes

Protocol-specific status codes follow

Examples

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

Successful transmission

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

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

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

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

Receive Packet - 0x90

Request frames:

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

Description

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Receive Packet - **0x90**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
14	8-bit	**Receive options**	Bit field of options that apply to the received message: ▪ **Bit 0**: Packet was Acknowledged [**0x01**] ▪ **Bit 1**: Packet was sent as a broadcast [**0x02**] ▪ **Bit 2**: Reserved ▪ **Bit 3**: Reserved ▪ **Bit 4**: Reserved ▪ **Bit 5**: Reserved ▪ **Bit 6**: Reserved ▪ **Bit 6, 7**: DigiMesh delivery method • b’00 = <invalid option> • b’01 = Point-multipoint [**0x40**] • b’10 = Directed Broadcast [**0x80**] • b’11 = DigiMesh [**0xC0**]

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

Examples

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

64-bit unicast

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

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

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

Explicit Receive Indicator - 0x91

Request frames:

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

Description

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Receive Indicator - **0x91**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
14	8-bit	**Source endpoint**	Endpoint of the source that initiated transmission.
15	8-bit	**Destination endpoint**	Endpoint of the destination that the message is addressed to.
16	16-bit	**Cluster ID**	The Cluster ID that the frame is addressed to.
18	16-bit	**Profile ID**	The Profile ID that the fame is addressed to.
20	8-bit	**Receive options**	Bit field of options that apply to the received message for packets sent using Digi endpoints (0xDC-0xEE): - **Bit 0**: Packet was Acknowledged [**0x01**] - **Bit 1**: Packet was sent as a broadcast [**0x02**] - **Bit 2**: Reserved - **Bit 3**: Reserved - **Bit 4**: Reserved

Offset	Size	Frame Field	Description
			■ **Bit 5**: Reserved ■ **Bit 6**: Reserved ■ **Bit 6, 7**: DigiMesh delivery method • b'00 = <invalid option> • b'01 = Point-multipoint [**0x40**] • b'10 = Directed Broadcast [**0x80**] • b'11 = DigiMesh [**0xC0**] **Note** Option values may be combined.
21-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

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

64-bit unicast

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

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

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

Node Identification Indicator - 0x95

Description

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

A node identifies itself to the network under these conditions:

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

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

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

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Node Identification Indicator - **0x95**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
12	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
14	8-bit	**Options**	Bit field of options that apply to the received message: ▪ Bit 0: Packet was Acknowledged [**0x01**] ▪ **Bit 1**: Packet was sent as a broadcast [**0x02**] ▪ **Bit 2**: Reserved ▪ Bit 4: Reserved ▪ Bit 5: Reserved ▪ **Bit 6, 7**: DigiMesh delivery method • b’00 = <invalid option> • b’01 = Point-multipoint [**0x40**] • b’10 = Directed Broadcast [**0x80**] • b’11 = DigiMesh [**0xC0**]

Offset	Size	Frame Field	Description
			Note Option values may be combined.
15	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
17	64-bit	**64-bit remote address**	The 64-bit address of the device that sent the Node Identification.
25	variable (2-byte minimum)	**Node identification string**	Node identification string on the remote device set by NI (Node Identifier). The identification string is terminated with a NULL byte (0x00).
27+NI	16-bit	**Reserved**	Unused, but this field is typically set to **0xFFFE**.
29+NI	8-bit	**Network device type**	What type of network device the remote identifies as: 0 = Coordinator 1 = Router 2 = End Device
30+NI	8-bit	**Source event**	The event that caused the node identification broadcast to be sent. 0 = Reserved 1 = Frame sent by node identification pushbutton event—see D0 (DIO0/AD0).
31+NI	16-bit	**Digi Profile ID**	The Digi application Profile ID—**0xC105**.
33+NI	16-bit	**Digi Manufacturer ID**	The Digi Manufacturer ID—**0x101E**.
35+NI	32-bit	**Device type identifier (optional)**	The user-defined device type on the remote device set by DD (Device Type Identifier). Only included if the receiving device has the appropriate NO (Network Discovery Options) bit set.
EOF-1	8-bit	**RSSI (optional)**	The RSSI of the last hop that relayed the message. Only included if the receiving device has the appropriate NO (Network Discovery Options) bit set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte—between length and checksum.

Examples

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

Identify remote device

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

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

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

Frame type	64-bit source	Reserved	Options	64-bit remote	NI String	Reserved	Device type	Event	Profile ID	MFG ID
0x95	0x0013A2001234567 8	0xFFFE	0xC2	0x0013A2001234567 8	0x4C483735 00	0xFFFE	0x01	0x01	0xC105	0x101E
Identification		*Unused*	*DigiMesh broadcast*		*"LH75" + null*	*Unused*	*Router*	*Button press*	*Digi*	*Digi*

Remote AT Command Response- 0x97

Request frame: Remote AT Command Request - 0x17

Description

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

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

Format

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

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

Examples

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

Set remote command parameter

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

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

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

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

Transmission failure

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

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

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

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

Query remote command parameter

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

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

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

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

Work with networked devices

Network commissioning and diagnostics 175
Local configuration 175
Remote configuration 175
Secure remote AT commands 176
Establish and maintain network links 178
Test links in a network - loopback cluster 180
Test links between adjacent devices 180

Network commissioning and diagnostics

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

Local configuration

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

Remote configuration

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

Send a remote command

To send a remote command, populate the Remote AT Command Request - 0x17 with:

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

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

XCTU has a Frames Generator tool that can assist you with building and sending a remote AT frame; see Frames generator tool in the *XCTU User Guide*.

Apply changes on remote devices

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

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

Remote command response

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

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

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

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

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

Secure remote AT commands

Configuring a network through remote AT commands is powerful, but it also leaves the door open for configuration mistakes, or worse, malicious actions by an intruder. This is because it is easy to bring a network down through remote AT commands.

We introduced secure remote AT commands to protect against such network attacks by requiring the user to enter a correct password with each remote command.

How secure remote AT commands protect the network

If a network exists without encryption enabled, anyone can introduce a new node into the network, configure the correct network parameters, and operate as a member of the network. Using remote commands, the network could be disabled.

If a network exists with encryption enabled, then the above intrusion would not be possible without knowing the **KY** value. However, this would not prevent the intruder from hijacking a node operating in an encrypted network and using that node (that is already a member of the network with a correct encryption key) to negatively affect the network through remote AT commands.

Secure remote AT commands protect against a hijacked device by requiring the correct password to be included with each remote command. Just like the network key, the password cannot be read from the device. Unlike the network key, it must be entered from the serial port for each remote command and the password may or may not be the same for each node in the network. In fact, for extra security (and more difficulty with network management), each node may have a different password.

Password protection

Enter the password with the **KZ** command. It is only used locally, which implies that it is not sent over the air. Secure remote AT commands do not offer authentication in the sense of a password being sent to a server which verifies the password. The password is only checked locally.

The default password is ASCII **0**. In order to set the password on a device, the old password must be followed by the new password. See KZ (Remote AT Command Password).

Password recovery

Inevitably, someone will be locked out of a node on their own network due to forgetting the password. If this occurs, you can enter RE (Restore Defaults), which restores the password to the default value of **0**.

Unfortunately, this is not without a side effect: **RE** also clears the network key. Therefore, you must re-enter all network operating parameters, including the network key after sending **RE**. If both the **KY** and **KZ** parameters are forgotten, the only remaining alternative is to enter a new network key for each node in the network.

Secure remote command control

Assuming the password is known, you can use the Secure Remote AT Command Request frame - 0x18 to send secure remote commands and include the known password in that frame. Such frames are free to be effective on any node in the network whether or not that node has a password set to a non-default value.

If the password is not set (**KZ** is **0**), then you can send 0x17 frames the same as they could before the 0x18 frame type was introduced. However such frames will be rejected when sent to any node with the password set (**KZ** is not **0**). Also, if the password is not set, you can send a 0x18 command with the password field set to **0**.

An undisclosed indicator in the OTA frame tells if the remote AT command is secure or not. This causes the receiving node to reject the incoming remote command if the receiving node has its password set.

Security protections

In order to harden the network against unauthorized access, several security protections are in place. In this context, a secure node is one which has the password set to a value other than **0**, which is the default.

Operational rule	0x97 error code
0x17 frames are not sent from a secure node.	0x09
0x18 frames are not sent if encryption is not enabled.	0x0A
0x18 frames are not sent unless the password matches the value given by the **KZ** command.	0x08
Secure nodes that receive remote AT commands do not execute those commands unless they are sent from a secure node.	0x09

The error code tells which operational rule was broken.

Note Encryption must be enabled in order to send secure remote AT commands.

Secure a network

You can use remote commands to secure an existing network from a single gateway node. To apply KZ (Remote AT Command Password) across a network the entire network must already have encryption enabled. Use the following procedure:

1. Ensure encryption is already enabled on every node in the network.
2. Use Local AT Command Request - 0x08 to configure **KZ** on the origin node.
3. Use Secure Remote AT Command Request frame - 0x18 frames to set **KZ** remotely on every node in the network.
4. Keep track of the passwords for all protected devices in a secure manner.

Note All gateway devices (any device that will receive 0x18 frames over its serial interface) must be kept physically secure. If an attacker gains physical access to a gateway device, he or she could

observe the password as it is communicated from the controller to the XBee/XBee-PRO SX RF Module. For the same reason, we recommend using at least two different passwords on a network—one for gateway devices, and one for standalone devices.

Software downgrades

When software version 9007 is installed, it prohibits a downgrade to the previous version. This rule is in place to protect against the following potential exploit:

1. A device with encryption enabled and a password set is hijacked from an operating network.
2. The intruder knows neither the network key nor the password, which are both stored on the node.
3. The intruder downgrades to the previous version of the firmware, which does not have a **KZ** command implemented.
4. In the downgrade process, the only parameters saved in flash will be those known by the old firmware loaded. An old firmware knows about **KY** and **EE**, but not **KZ**. Therefore, after the downgrade **KY** is saved but not **KZ**.
5. The intruder upgrades back to version 9007. Since **KZ** is no longer in flash, it reverts to its default value of **0**. Since **KY** is still in flash it is still correct just as it was before the downgrade.
6. The intruder has a node in the network that is encrypted, has the right key, and is not secure.
7. Next, the intruder enters a password that he or she knows.
8. Finally, the intruder can send secure remote AT commands throughout the network, using his or her own password, and defeating the original attempt to secure the network.

Reverting to a previous version of the software from version 9007 is prohibited in order to defeat this scenario.

Establish and maintain network links

Build aggregate routes

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

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

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

The **AG** command requires a 64-bit parameter. The parameter indicates the current value of the **DH**/**DL** registers on a device; typically you should replace this value with the 64-bit address of the node sending the **AG** broadcast. However, if you do not want to update the **DH**/**DL** of the device

receiving the **AG** broadcast you can use the invalid address of 0xFFFE. The receiving nodes that are configured in API mode output an Aggregator Update API frame (0x8E) if they update their **DH**/**DL** address; for a description of the frame, see Aggregate Addressing Update - 0x8E.

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

DigiMesh routing examples

Example one:

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

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

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

Example two:

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

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

Replace nodes

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

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

Example three:

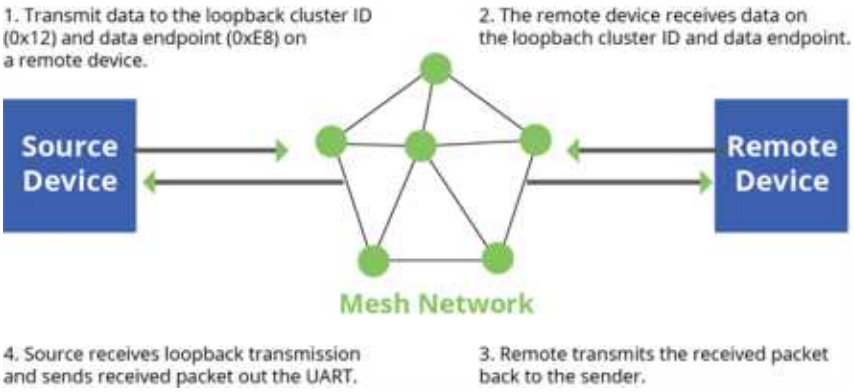
This example shows how to cause all devices to update their **DH** and **DL** registers to the MAC address of the sending device. In this case, assume you are using a device with a serial number of 0x0013A200 4052C507 as a network aggregator, and the sending device has a MAC address of 0x0013A200 F5E4D3B2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

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

Test links in a network - loopback cluster

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

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



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

Transparent operating mode configuration (AP = 0)

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

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

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

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

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

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

Test links between adjacent devices

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

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

1. Use device A in API mode (**AP** = **1**) to send an Explicit Addressing Command (0x11) frame to device B.
2. Address the frame to the Test Link Request Cluster ID (0x0014) and destination endpoint: 0xE6.

3. Include a 12-byte payload in the Explicit Addressing Command frame with the following format:

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

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

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

Example

Suppose that you want to test the link between device A (**SH**/**SL** = 0x0013A200 40521234) and device B (**SH**/**SL**=0x0013A 200 4052ABCD) by transmitting 1000 40-byte packets:

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

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

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

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

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

This means:

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

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

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

The device that sends the request for initiating the Test link and outputs the result does not need to be the sender or receiver of the test. It is possible for a third node, "device C", to request device A to perform a test link against device B and send the results back to device C to be output. It is also possible for device B to request device A to perform the previously mentioned test. In other words, the frames can be sent by either device A, device B or device C and in all cases the test is the same: device A sends data to device B and reports the results.

RSSI indicators

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

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

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

1. Use the RSSI module pin (pin 7). When the device receives data, it sets the RSSI PWM duty cycle to a value based on the RSSI of the packet it receives.

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

Discover all the devices on a network

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

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

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

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

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

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 Local AT Command Request - 0x08.

To use **FN** remotely, send the target node a Remote AT Command Request - 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 Local AT Command Response - 0x88.

Trace route option

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

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

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

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

The Route Information Packet frame contains:

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

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

Trace route example

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

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

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

NACK messages

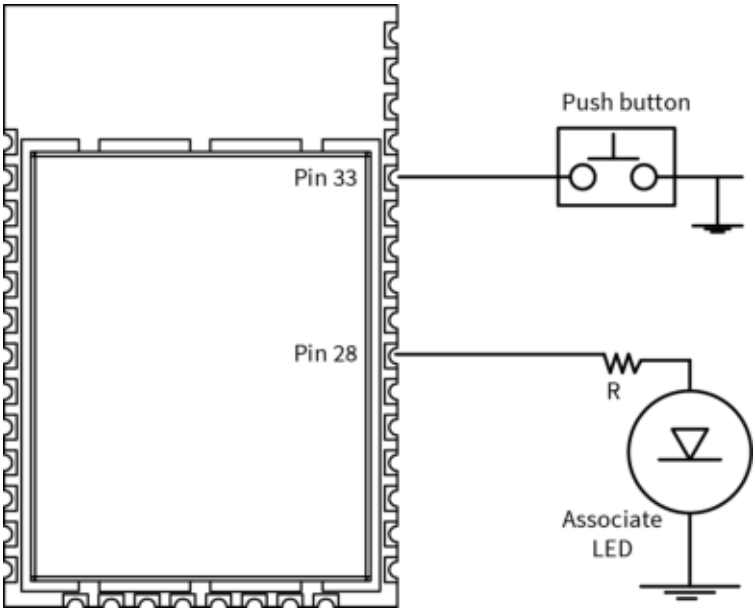
Transmit Request (0x10 and 0x11) frames contain a negative-acknowledge character (NACK) API option (Bit 2 of the Transmit Options field).

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

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

The Commissioning Pushbutton

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



To support the Commissioning Pushbutton and its associated LED functions, connect a pushbutton and an LED to device pins 33 and 28 respectively.

Definitions

To enable the Commissioning Pushbutton functionality on pin 33, 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.
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.

Button presses	Sleep configuration and sync status	Action
		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 SO (Sleep Options).
4	Any	Sends an **RE** command to restore device parameters to default values.

Use the Commissioning Pushbutton

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

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

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

Associate LED

The Associate pin (pin 28) provides an indication of the device's sleep status and diagnostic information. To take advantage of these indications, connect an LED to the Associate pin.

To enable the Associate LED functionality, set the **D5** command to 1; it is enabled by default. If enabled, the Associate pin is configured as an output. This section describes the behavior of the pin.

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

Use the **LT** command to override the blink rate of the Associate pin. If you set **LT** to 0, the device uses the default blink time: 500 ms for a sleep coordinator, 250 ms otherwise.

The following table describes the Associate LED functionality.

Sleep mode	LED status	Meaning
0	On, blinking	The device has power and is operating properly

Sleep mode	LED status	Meaning
1, 4, 5	Off	The device is in a low power mode
1, 4, 5	On, blinking	The device has power, is awake and is operating properly
7	On, solid	The network is asleep, or the device has not synchronized with the network, or has lost synchronization with the network
7, 8	On, slow blinking (500 ms blink time)	The device is acting as the network sleep coordinator and is operating properly
7, 8	On, fast blinking (250 ms blink time)	The device is properly synchronized with the network
8	Off	The device is in a low power mode
8	On, solid	The device has not synchronized or has lost synchronization with the network

Diagnostics support

The Associate pin works with the Commissioning Pushbutton to provide additional diagnostic behaviors to aid in deploying and testing a network. If you press the Commissioning Pushbutton once, the device transmits a broadcast Node Identification Indicator (0x95) frame at the beginning of the next wake cycle if the device is sleep compatible, or immediately if the device is not sleep compatible. If you enable the Associate LED functionality using the **D5** command, a device that receives this transmission blinks its Associate pin rapidly for one second.

Monitor I/O lines

Pin configurations

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

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

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

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

Device pin name	Device pin number	Configuration command
DIO12	5	**P2**
PWM0 / RSSI / DIO10	7	**P0**
PWM1 / DIO11	8	**P1**
$\overline{\text{DTR}}$ / SLEEP_RQ / DIO8	10	**D8**
DIO4	24	**D4**
$\overline{\text{CTS}}$ / DIO7	25	**D7**
ON/$\overline{\text{SLEEP}}$ / DIO9	26	**D9**
ASSOC / AD5 / DIO5	15	**D5**

Device pin name	Device pin number	Configuration command
$\overline{\text{RTS}}$ / DIO6	29	**D6**
AD3 / DIO3	30	**D3**
AD2 / DIO2	31	**D2**
AD1 / DIO1	32	**D1**
AD0 / DIO0 / Commissioning Pushbutton	33	**D0**

Use the **PR** command to enable internal pull up/down resistors for each digital input. Use the **PD** command to determine the direction of the internal pull up/down resistor.

Queried sampling

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

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

Field	Name	Description
1	Sample sets	Number of sample sets in the packet. Always set to 1.
2	Digital channel mask	Indicates which digital I/O lines have sampling enabled. Each bit corresponds to one digital I/O line on the device. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 4 = DIO4 bit 5 = ASSOC/DIO5 bit 6 = RTS/DIO6 bit 7 = CTS/GPIO7 bit 8 = DTR / SLEEP_RQ / DIO8 bit 9 = ON_SLEEP / DIO9 bit 10 = RSSI/DIO10 bit 11 = PWM/DIO11 bit 12 = CD/DIO12 For example, a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital I/O.
1	Analog channel mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit 4 = AD4/DIO4

Field	Name	Description
		bit 5 = ASSOC/AD5/DIO5
Variable	Sampled data set	If you enable any digital I/O lines, the first two bytes of the data set indicate the state of all enabled digital I/O. Only digital channels that you enable in the Digital channel mask bytes have any meaning in the sample set. If do not enable any digital I/O on the device, it omits these two bytes. Following the digital I/O data (if there is any), each enabled analog channel returns two bytes. The data starts with AIN0 and continues sequentially for each enabled analog input channel up to AIN5.

If you issue the **IS** command using a a local or remote AT Command API frame, then the device returns an AT Command Response (0x88) frame with the I/O data included in the command data portion of the packet.

Example	Sample AT response
0x01	[1 sample set]
0x0C0C	[Digital Inputs: DIO 2, 3, 10, 11 enabled]
0x03	[Analog Inputs: A/D 0, 1 enabled]
0x0408	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0	[Analog input: ADIO 0 = 0x3D0]
0x0124	[Analog input: ADIO 1 =0x120]

Periodic I/O sampling

Periodic sampling allows a device to take an I/O sample and transmit it to a remote device at a periodic rate. Use the **IR** command to set the periodic sample rate.

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

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

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

A device with sleep enabled transmits periodic I/O samples at the **IR** rate until the **ST** time expires and the device can resume sleeping. For more information about setting sleep modes, see Sleep modes.

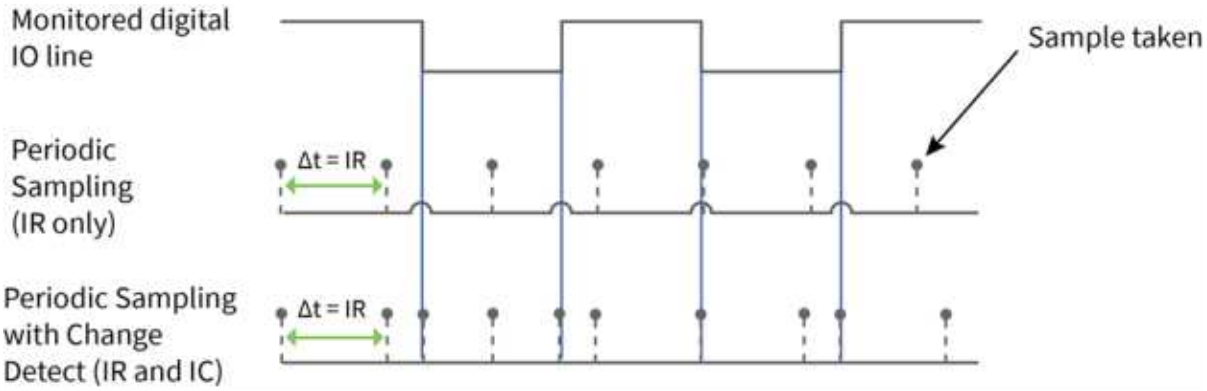
Detect digital I/O changes

When the network is setup for synchronous sleep there is an option to randomize a single I/O sample by setting **SO** bit 10—enables random I/O samples—and setting **IR** to zero where the sample will be randomized over the **ST** time period. In addition you can use **IR** in conjunction with **SO** bit 10 which will randomize the first I/O sample over the **IR** time period and thereafter I/O samples will occur on

the **IR** interval sending x samples or x samples - 1. If the **IR** interval is greater than the **ST** time period then **ST** time will be used to randomize the I/O sample.

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

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



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

I/O line passing

You can configure XBee/XBee-PRO SX RF Modules to perform analog and digital line passing. When a device receives an RF I/O sample data packet, you can set up the receiving device to update any enabled outputs (PWM and DIO) based on the data it receives.

Digital I/O lines are mapped in pairs; pins configured as digital input on the transmitting device affect the corresponding digital output pin on the receiving device. For example: DI5 (pin 25) can only update DO5 (pin 25).

For Analog Line Passing, the XBee/XBee-PRO SX RF Module has two PWM output pins that simulate the voltage measured by the ADC lines AD0 and AD1. For example, when configured as an ADC, AD0 (pin 33) updates PWM0 (pin 7); AD1 (pin 32) updates PWM1 (pin 8).

The default setup is for outputs to not be updated. Instead, a device sends I/O sample data out the serial interface if the device is configured for API mode (**AP** = 1 or 2). You can use the **IU** command to disable sample data output.

To enable updating the outputs, set the **IA** (I/O Input Address) parameter with the address of the device that has the appropriate inputs enabled. This effectively binds the outputs to a particular device's input. This does not affect the ability of the device to receive I/O line data from other devices - only its ability to update enabled outputs. Set the **IA** parameter to 0xFFFF (broadcast address) to set up the device to accept I/O data for output changes from any device on the network.

For line passing to function, the device configured with inputs must generate sample data. Refer to Pin configurations for information on how to configure digital and analog sampling.

When outputs are changed from their non-active state, the device can be setup to return the output level to its non-active state. The timers are set using the **T*n*** (**D*n*** Output Timer) and **PT** (PWM Output Timeout) commands. The timers are reset every time the device receives a valid I/O sample packet with a matching **IA** address. You can adjust the **IC** (Change Detect) and **IR** (Sample Rate) parameters on the transmitting device to keep the outputs set to their active output if the system needs more time than the timers can handle.

Configuration example

As an example for a simple digital and analog link, you could set a pair of RF devices as follows:

Command	Description	Device A	Device B
SH	Serial Number High	0x0013A200	0x0013A200
SL	Serial Number Low	0x12345678	0xABCDABCD
DH	Destination High	0x0013A200	0x00000000

Command	Description	Device A	Device B
DL	Destination Low	0xABCDABCD	0x0000FFFF (broadcast)
IA	I/O Input Address	0x0013A200ABCDABCD	0x0013A20012345678
IR	Sample Rate	0x7D0 (2 seconds)	0 (disabled)
IC	DIO Change Detect	0 (disabled)	0x1000 (DIO3 only)
D1	DIO1/AD1	2 : ADC input	N/A
P1	DIO11/PWM1	N/A	2: PWM1 output
PT	PWM Output Timeout	N/A	0x1E (3 seconds)
D2	DIO2/AD2	3: Digital input	5: Digital output, HIGH
D3	DIO3/AD3	5: Digital output, HIGH	3: Digital input
T3	DIO3 Timeout	0x64 (10 seconds)	N/A

Command	Description	Device A	Device B
SH	Serial Number High	0x0013A200	0x0013A200
SL	Serial Number Low	0x12345678	0xABCDABCD
DH	Destination High	0x0013A200	0x00000000
DL	Destination Low	0xABCDABCD	0x0000FFFF (broadcast)
IA	I/O Input Address	0x0013A200ABCDABCD	0x0013A20012345678
IR	Sample Rate	0x7D0 (2 seconds)	0 (disabled)
IC	DIO Change Detect	0 (disabled)	0x8 (DIO3 only)
D1	DIO1/AD1	2 : ADC input	N/A
P1	DIO11/PWM1	N/A	2: PWM1 output
PT	PWM Output Timeout	N/A	0x1E (3 seconds)
D2	DIO2/AD2	3: Digital input	5: Digital output, HIGH
D3	DIO3/AD3	5: Digital output, HIGH	3: Digital input
T3	DIO3 Timeout	0x64 (10 seconds)	N/A

In the example, both devices have I/O Line Passing enabled with appropriate inputs and outputs configured. The **IA** parameter determines which device on the network is allowed to affect the device's outputs.

Device A takes a periodic sample of all I/O lines every two seconds and transmits it as a unicast transmission to the address defined by **DH** and **DL** (in this case, Device B). Device B does not periodically sample, instead it monitors DIO3 for a binary change. When it detects a change on that pin, it generates a sample and transmits it as a broadcast to all devices on the network.

When Device B receives a sample packet from Device A:

- DIO2 on Device B outputs the state of DIO2 from Device A.
- PWM1 outputs a duty cycle equivalent to the analog voltage read on AD1 of Device A.
- A PWM timeout has been set to three seconds; if no sample is received, PWM1 returns to 0 V after this period.

When Device A receives a sample packet from Device B:

- DIO3 on Device A outputs the state of DIO3 from Device B.
- A DIO3 timeout has been set to 10 seconds; if no sample is received, DIO3 reverts to a HIGH state after this period.

Note By default, all Digital I/O lines have internal pull-up resistors enabled with the **PR** command. This causes inputs to float high. You can use the **PD** command to change the direction of the internal pull-up/down resistors. The XBee/XBee-PRO SX RF Module uses an internal reference voltage of 2.5 V for ADC lines, but you can use the **AV** command to set it to 1.25 VDC.

General Purpose Flash Memory

General Purpose Flash Memory 196
Access General Purpose Flash Memory 196
General Purpose Flash Memory commands 197

General Purpose Flash Memory

XBee/XBee-PRO SX RF Modules provides 119 512-byte blocks of flash memory that an application can read and write to. This memory provides a non-volatile data storage area that an application uses for many purposes. Some common uses of this data storage include:

- Storing logged sensor data
- Buffering firmware update data for a host microcontroller
- Storing and retrieving data tables needed for calculations performed by a host microcontroller

The General Purpose Memory (GPM) is also used to store a firmware update file for over-the-air firmware updates of the device itself.

Access General Purpose Flash Memory

To access the GPM of a target node locally or over-the-air, send commands to the MEMORY_ACCESS cluster ID (0x23) on the DIGI_DEVICE endpoint (0xE6) of the target node using explicit API frames. For a description of Explicit API frames, see Operate in API mode.

To issue a GPM command, format the payload of an explicit API frame as follows:

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	Specific GPM commands are described in detail in the topics that follow.
1	1	GPM_OPTIONS	Command-specific options.
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM.
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block.
6	2*	GPM_NUM_BYTES	The number of bytes in the GPM_DATA field, or in the case of a READ, the number of bytes requested.
8	varies	GPM_DATA	
* Specify multi-byte parameters with big-endian byte ordering.			

When a device sends a GPM command to another device via a unicast, the receiving device sends a unicast response back to the requesting device's source endpoint specified in the request packet. It does not send a response for broadcast requests. If the source endpoint is set to the DIGI_DEVICE endpoint (0xE6) or Explicit API mode is enabled on the requesting device, then the requesting node outputs a GPM response as an explicit API RX indicator frame (assuming it has API mode enabled).

The format of the response is similar to the request packet:

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	This field is the same as the request field.
1	1	GPM_STATUS	Status indicating whether the command was successful.
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM.
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block.
6	2*	GPM_NUM_BYTES	The number of bytes in the GPM_DATA field.
8	varies	GPM_DATA	
* Specify multi-byte parameters with big-endian byte ordering.			

General Purpose Flash Memory commands

This section provides information about commands that interact with GPM:

PLATFORM_INFO_REQUEST (0x00)

A PLATFORM_INFO_REQUEST frame can be sent to query details of the GPM structure.

Field name	Command-specific description
GPM_CMD_ID	Should be set to PLATFORM_INFO_REQUEST (0x00).
GPM_OPTIONS	This field is unused for this command. Set to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	No data bytes should be specified for this command.

PLATFORM_INFO (0x80)

When a PLATFORM_INFO_REQUEST command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to PLATFORM_INFO (0x80).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.

Field name	Command-specific description
GPM_BLOCK_NUM	Indicates the number of GPM blocks available.
GPM_START_INDEX	Indicates the size, in bytes, of a GPM block.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

A PLATFORM_INFO_REQUEST sent to a device with a serial number of 0x0013a200407402AC should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 00 00 0000 0000 0000 24

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 80 00 0077 0200 0000 EB

ERASE (0x01)

The ERASE command erases (writes all bits to binary 1) one or all of the GPM flash blocks. You can also use the ERASE command to erase all blocks of the GPM by setting the GPM_NUM_BYTES field to 0.

Field name	Command-specific description
GPM_CMD_ID	Should be set to ERASE (0x01).
GPM_OPTIONS	There are currently no options defined for the ERASE command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be erased. When erasing all GPM blocks, this field is ignored (set to 0).
GPM_START_INDEX	The ERASE command only works on complete GPM blocks. The command cannot be used to erase part of a GPM block. For this reason GPM_START_INDEX is unused (set to 0).
GPM_NUM_BYTES	Setting GPM_NUM_BYTES to 0 has a special meaning. It indicates that every flash block in the GPM should be erased (not just the one specified with GPM_BLOCK_NUM). In all other cases, the GPM_NUM_BYTES field should be set to the GPM flash block size.
GPM_DATA	No data bytes are specified for this command.

ERASE_RESPONSE (0x81)

When an ERASE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to ERASE_RESPONSE (0x81).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

To erase flash block 42 of a target radio with serial number of 0x0013a200407402ac format an ERASE packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 01 00 002A 0000 0200 37

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 81 00 002A 0000 0000 39

WRITE (0x02) and ERASE_THEN_WRITE (0x03)

The WRITE command writes the specified bytes to the GPM location specified. Before writing bytes to a GPM block it is important that the bytes have been erased previously. The ERASE_THEN_WRITE command performs an ERASE of the entire GPM block specified with the GPM_BLOCK_NUM field prior to doing a WRITE.

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE (0x02) or ERASE_THEN_WRITE (0x03).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be written.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be written.
GPM_NUM_BYTES	Set to the number of bytes specified in the GPM_DATA field. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. The maximum payload size can be queried with the **NP** command.
GPM_DATA	The data to be written.

WRITE _RESPONSE (0x82) and ERASE_THEN_WRITE_RESPONSE (0x83)

When a WRITE or ERASE_THEN_WRITE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE_RESPONSE (0x82) or ERASE_THEN_WRITE_RESPONSE (0x83)
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time
GPM_BLOCK_NUM	Matches the parameter passed in the request frame
GPM_START_INDEX	Matches the parameter passed in the request frame
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0
GPM_DATA	No data bytes are specified for these commands

Example

To write 15 bytes of incrementing data to flash block 22 of a target radio with serial number of 0x0013a200407402ac a WRITE packet should be formatted as follows (spaces added to delineate fields):

7E 002B 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 02 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C5

Assuming all transmissions were successful and that flash block 22 was previously erased, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 82 00 0016 0000 0000 4C

READ (0x04)

You can use the READ command to read the specified number of bytes from the GPM location specified. Data can be queried from only one GPM block per command.

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ (0x04).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be read.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be read.
GPM_NUM_BYTES	Set to the number of data bytes to be read. Only one GPM block can

Field name	Command-specific description
	be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. You can query the maximum payload size with the **NP** AT command.
GPM_DATA	No data bytes should be specified for this command.

READ_RESPONSE (0x84)

When a READ command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ_RESPONSE (0x84).
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field.
GPM_DATA	The bytes read from the GPM block specified.

Example

To read 15 bytes of previously written data from flash block 22 of a target radio with serial number of 0x0013a200407402ac a READ packet should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 04 00 0016 0000 000F 3B

Assuming all transmissions were successful and that flash block 22 was previously written with incrementing data, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 0029 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 84 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C3

FIRMWARE_VERIFY (0x05) and FIRMWARE_VERIFY_AND_INSTALL (0x06)

Use the FIRMWARE_VERIFY and FIRMWARE_VERIFY_AND_INSTALL commands when remotely updating firmware on a device. For more information about firmware updates, see Update the firmware over-the-air. These commands check if the GPM contains a valid over-the-air update file. For the FIRMWARE_VERIFY_AND_INSTALL command, if the GPM contains a valid firmware image then the device resets and begins using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY (0x05) or FIRMWARE_VERIFY_AND_INSTALL (0x06)
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

FIRMWARE_VERIFY_RESPONSE (0x85)

When a FIRMWARE_VERIFY command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_RESPONSE (0x85)
GPM_STATUS	A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. A 0 in the least significant bit indicates the GPM does contain a valid firmware image. All other bits are reserved at this time.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86)

When a FIRMWARE_VERIFY_AND_INSTALL command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame only if the GPM memory does not contain a valid image. If the image is valid, the device resets and begins using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_AND_INSTALL_RESPONSE (0x86).
GPM_STATUS	A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. All other bits are reserved at this time.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.

Field name	Command-specific description
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command.

Example

To verify a firmware image previously loaded into the GPM on a target device with serial number 0x0013a200407402ac, format a FIRMWARE_VERIFY packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 05 00 0000 0000 0000 1F

Assuming all transmissions were successful and that the firmware image previously loaded into the GPM is valid, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 85 00 0000 0000 0000 5F

Update the firmware over-the-air

This section provides instruction on how to update your firmware using wired updates and over-the-air updates.

Over-the-air firmware updates 205
Distribute the new application 205
Verify the new application 206
Install the application 206

Over-the-air firmware updates

There are two methods of updating the firmware on the device. You can update the firmware locally with XCTU using the device's serial port interface. You can also update firmware using the device's RF interface (over-the-air updating.)

The over-the-air firmware update method provided is a robust and versatile technique that you can tailor to many different networks and applications. OTA updates are reliable and minimize disruption of normal network operations.

In the following sections, we refer to the node that will be updated as the target node. We refer to the node providing the update information as the source node. In most applications the source node is locally attached to a computer running update software.

There are three phases of the over-the-air update process:

1. Distribute the new application
2. Verify the new application
3. Install the application

Distribute the new application

The first phase of performing an over-the-air update on a device is transferring the new firmware file to the target node. Load the new firmware image in the target node's GPM prior to installation. XBee/XBee-PRO SX RF Modules use an encrypted binary (.ebin) file for both serial and over-the-air firmware updates. These firmware files are available on the Digi Support website and via XCTU.

Send the contents of the .ebin file to the target device using general purpose memory WRITE commands. Erase the entire GPM prior to beginning an upload of an .ebin file. The contents of the .ebin file should be stored in order in the appropriate GPM memory blocks. The number of bytes that are sent in an individual GPM WRITE frame is flexible and can be catered to the user application.

Example

The example firmware version has an .ebin file of 55,141 bytes in length. Based on network traffic, we determine that sending a 128 byte packet every 30 seconds minimizes network disruption. For this reason, you would divide and address the .ebin as follows:

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
0	0	128	0 to 127
0	128	128	128 to 255
0	256	128	256 to 383
0	384	128	384 to 511
1	0	128	512 to 639
1	128	128	640 to 767
-	-	-	-
-	-	-	-

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
-	-	-	-
107	0		54784 to 54911
107	128		54912 to 55039
107	256	101	55040 to 55140

Verify the new application

For an uploaded application to function correctly, every single byte from the .ebin file must be properly transferred to the GPM. To guarantee that this is the case, GPM VERIFY functions exist to ensure that all bytes are properly in place. The FIRMWARE_VERIFY function reports whether or not the uploaded data is valid. The FIRMWARE_VERIFY_AND_INSTALL command reports if the uploaded data is invalid. If the data is valid, it begins installing the application. No installation takes place on invalid data.

Install the application

When the entire .ebin file is uploaded to the GPM of the target node, you can issue a FIRMWARE_VERIFY_AND_INSTALL command. Once the target receives the command it verifies the .ebin file loaded in the GPM. If it is valid, then the device installs the new firmware. This installation process can take up to eight seconds. During the installation the device is unresponsive to both serial and RF communication. To complete the installation, the target module resets. AT parameter settings which have not been written to flash using the **WR** command will be lost.

Important considerations

Write all parameters with the **WR** command before performing a firmware update. Packet routing information is also lost after a reset. Route discoveries are necessary for DigiMesh unicasts involving the updated node as a source, destination, or intermediate node.

Because explicit API Tx frames can be addressed to a local node (accessible via the SPI or UART) or a remote node (accessible over the RF port) the same process can be used to update firmware on a device in either case.

Regulatory information

FCC (United States) 208
ISED (Innovation, Science and Economic Development Canada) 223
ACMA (Australia) 224
RSM (New Zealand) 224
Brazil (Anatel) 225
IFETEL (Mexico) 228

FCC (United States)

These RF modules comply with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

In order to operate under Digi's FCC Certification, integrators must comply with the following regulations:

1. The integrator must ensure that the text provided with this device (in the labeling requirements section that follows) is placed on the outside of the final product and within the final product operation manual.
2. The device may only be used with antennas that have been tested and approved for use with this device; refer to FCC antenna certifications.

OEM labeling requirements

WARNING! The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product enclosure that displays the text shown in the figure below.

The following text is the required FCC label for OEM products containing the XBee-PRO SX RF Module:

Contains FCC ID: MCQ-XBPSX
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.

The following text is the required FCC label for OEM products containing the XBee SX RF Module:

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

FCC notices

IMPORTANT: These RF modules have been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

IMPORTANT: Integrators must test final product to comply with unintentional radiators (FCC sections 15.107 & 15.109) before declaring compliance of their final product to Part 15 of the FCC rules.

IMPORTANT: These RF modules have been certified for remote and base radio applications. If the module will be used for portable applications, the device must undergo SAR testing.

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

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

Increase the separation between the equipment and receiver, Connect equipment and receiver to outlets on different circuits, or Consult the dealer or an experienced radio/TV technician for help.

RF exposure statement

This statement must be included as a CAUTION statement in integrator product manuals.

WARNING! This equipment is approved only for mobile and base station transmitting devices. Antenna(s) used for this transmitter must be installed to provide a separation distance of at least 34 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

FCC antenna certifications

WARNING! This device has been tested with the antennas listed in the tables of this section. When integrated into products, fixed antennas require installation preventing end users from replacing them with non-approved antennas. Antennas not listed in the tables must be tested to comply with FCC Section 15.203 (unique antenna connectors) and Section 15.247 (emissions).

Fixed base station and mobile applications

Digi devices are pre-FCC approved for use in fixed base station and mobile applications. When the antenna is mounted at least 34 cm from nearby persons, the application is considered a mobile application.

Portable applications and SAR testing

When the antenna is mounted closer than 34 cm to nearby persons, then the application is considered "portable" and requires an additional test be performed on the final product. This test is called Specific Absorption Rate (SAR) testing and measures the emissions from the device and how they affect the person.

RF exposure statement

This statement must be included as a CAUTION statement in integrator product manuals.

WARNING! This equipment is approved only for mobile and base station transmitting devices. Antenna(s) used for this transmitter must be installed to provide a separation distance of at least 34 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

XBee-PRO SX antenna options (30 dBm maximum RF power)

The following tables cover the antennas that are approved for use with the XBee-PRO SX RF Module. If applicable, the tables show the required cable loss between the device and the antenna.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

Dipole antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Connector	Gain	Required antenna cable loss	Application
A09-HSM-7 [1]	Straight half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed / mobile
A09-HASM-675	Articulated half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed / mobile
A09-HABMM-P5I	Swivel half wave with 5" pigtail	MMCX	2.1 dBi	0.4 dB	Fixed / mobile
A09-HBMM-P5I	Straight half-wave with 6" pigtail	MMCX	2.1 dBi	0.4 dB	Fixed / mobile
A09-HASM-7*	Articulated half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed
A09-HRSM*	Right angle half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed
A09-HG*	Glass mounted half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed
A09-HATM*	Articulated half-wave	RPTNC	2.1 dBi	0.4 dB	Fixed
A09-H*	Half-wave dipole	RPSMA	2.1 dBi	0.4 dB	Fixed

Yagi antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

[1] Installers should apply additional torque to screw on the antenna.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-Y6NF*	2 element Yagi	6.1 dBi	N	2.0 dB	Fixed/mobile
A09-Y7NF*	3 element Yagi	7.1 dBi	N	3.0 dB	Fixed/mobile
A09-Y8NF	4 element Yagi	8.1 dBi	N	4.0 dB	Fixed/mobile
A09-Y9NF*	4 element Yagi	9.1 dBi	N	5.0 dB	Fixed/mobile
A09-Y10NF*	5 element Yagi	10.1 dBi	N	6.0 dB	Fixed/mobile
A09-Y11NF	6 element Yagi	11.1 dBi	N	7.0 dB	Fixed/mobile
A09-Y12NF*	7 element Yagi	12.1 dBi	N	8.0 dB	Fixed/mobile
A09-Y13NF*	9 element Yagi	13.1 dBi	N	9.0 dB	Fixed/mobile
A09-Y14NF*	14 element Yagi	14.0 dBi	N	9.9 dB	Fixed/mobile
A09-Y6TM*	2 element Yagi	6.1 dBi	RPTNC	2.0 dB	Fixed/mobile
A09-Y7TM*	3 element Yagi	7.1 dBi	RPTNC	3.0 dB	Fixed/mobile
A09-Y8TM*	4 element Yagi	8.1 dBi	RPTNC	4.0 dB	Fixed/mobile
A09-Y9TM*	4 element Yagi	9.1 dBi	RPTNC	5.0 dB	Fixed/mobile
A09-Y10TM-P10I	5 element Yagi	10.1 dBi	RPTNC	6.0 dB	Fixed/mobile
A09-Y11TM*	6 element Yagi	11.1 dBi	RPTNC	7.0 dB	Fixed/mobile
A09-Y12TM*	7 element Yagi	12.1 dBi	RPTNC	8.0 dB	Fixed/mobile
A09-Y13TM*	9 element Yagi	13.1 dBi	RPTNC	9.0 dB	Fixed/mobile
A09-Y14TM*	14 element Yagi	14.0 dBi	RPTNC	9.9 dB	Fixed/mobile

Omni-directional base station antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F0NF*	Fiberglass base station	0 dBi	N	-	Fixed
A09-F1NF*	Fiberglass base station	1.0 dBi	N	-	Fixed
A09-F2NF-M	Fiberglass base station	2.1 dBi	N	-	Fixed
A09-F3NF*	Fiberglass base station	3.1 dBi	N	-	Fixed
A09-F4NF*	Fiberglass base station	4.1 dBi	N	-	Fixed
A09-F5NF-M	Fiberglass base station	5.1 dBi	N	-	Fixed
A09-F6NF*	Fiberglass base station	6.1 dBi	N	0.9 dB	Fixed
A09-F7NF*	Fiberglass base station	7.1 dBi	N	1.9 dB	Fixed
A09-F8NF-M	Fiberglass base station	8.1 dBi	N	2.9 dB	Fixed
A09-F0SM*	Fiberglass base station	0 dBi	RPSMA	-	Fixed
A09-F1SM*	Fiberglass base station	1.0 dBi	RPSMA	-	Fixed
A09-F2SM*	Fiberglass base station	2.1 dBi	RPSMA	-	Fixed
A09-F3SM*	Fiberglass base station	3.1 dBi	RPSMA	-	Fixed
A09-F4SM*	Fiberglass base station	4.1 dBi	RPSMA	-	Fixed
A09-F5SM*	Fiberglass base station	5.1 dBi	RPSMA	-	Fixed
A09-F6SM*	Fiberglass base station	6.1 dBi	RPSMA	0.9 dB	Fixed
A09-F7SM*	Fiberglass base station	7.1 dBi	RPSMA	1.9 dB	Fixed
A09-F8SM*	Fiberglass base station	8.1 dBi	RPSMA	2.9 dB	Fixed
A09-F0TM*	Fiberglass base station	0 dBi	RPTNC	-	Fixed
A09-F1TM*	Fiberglass base station	1.0 dBi	RPTNC	-	Fixed
A09-F2TM*	Fiberglass base station	2.1 dBi	RPTNC	-	Fixed

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F3TM*	Fiberglass base station	3.1 dBi	RPTNC	-	Fixed
A09-F4TM*	Fiberglass base station	4.1 dBi	RPTNC	-	Fixed
A09-F5TM*	Fiberglass base station	5.1 dBi	RPTNC	-	Fixed
A09-F6TM*	Fiberglass base station	6.1 dBi	RPTNC	0.9 dB	Fixed
A09-F7TM*	Fiberglass base station	7.1 dBi	RPTNC	1.9 dB	Fixed
A09-F8TM*	Fiberglass base station	8.1 dBi	RPTNC	2.9 dB	Fixed
A09-W7*	Wire base station	7.1 dBi	RPN	1.9 dB	Fixed
A09-W7SM*	Wire base station	7.1 dBi	RPSMA	1.9 dB	Fixed
A09-W7TM*	Wire base station	7.1 dBi	RPTNC	1.9 dB	Fixed

Dome antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-D3PNF*	Omnidirectional permanent mount	3.0 dBi	N	0.4 dB	Fixed/mobile
A09-D3NF*	Omnidirectional magnetic mount	3.0 dBi	N	0.4 dB	Fixed/mobile
A09-D3PTM*	Omnidirectional permanent mount	3.0 dBi	RPTNC	0.4 dB	Fixed/mobile
A09-D3PSM*	Omnidirectional permanent mount	3.0 dBi	RPSMA	0.4 dB	Fixed/mobile

Monopole antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-QRAMM	3" Quarter wave wire	2.1 dBi	MMCX	-	Fixed/mobile
A09-QRSM-2.1*	Quarter wave 2.1" right angle	3.3 dBi	RPSMA	0.4 dB	Fixed/mobile
A09-QW*	Quarter wave wire	1.9 dBi	Permanent	-	Fixed/mobile
A09-QSM-3*	Quarter wave straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QSM-3H*	Heavy duty quarter wave straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QBMM-P6I*	Quarter wave w/ 6" pigtail	1.9 dBi	MMCX	-	Fixed/mobile
A09-QHSM-2*	2" straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QHRSM-2*	2" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QHRSM-170*	1.7" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QRSM-380*	3.8" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QAPM-520*	5.2" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QSPM-3*	3" straight screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QAPM-3*	3" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QAPM-3H*	3" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile

XBee SX antenna options (13 dBm maximum RF power)

The following tables cover the antennas that are approved for use with the XBee SX RF Module. If applicable, the tables show the required cable loss between the device and the antenna.

Dipole antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Connector	Gain	Required antenna cable loss	Application
A09-HSM-7 [1]	Straight half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed/mobile
A09-HASM-675	Articulated half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed/mobile
A09-HABMM-P5I	Swivel half wave with 5" pigtail	MMCX	2.1 dBi	0.4 dB	Fixed/mobile
A09-HBMM-P5I	Straight half-wave with 6" pigtail	MMCX	2.1 dBi	0.4 dB	Fixed/mobile
A09-HASM-7*	Articulated half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed
A09-HRSM*	Right angle half-wave	RPSMA	2.1 dBi	0.4 dB	Fixed
A09-HG*	Glass mounted half-wave	RPSMA	2.1dBi	0.4 dB	Fixed
A09-HATM*	Articulated half-wave	RPTNC	2.1 dBi	0.4 dB	Fixed
A09-H*	Half-wave dipole	RPSMA	2.1 dBi	0.4 dB	Fixed

Yagi antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

[1] Installers should apply additional torque to screw on the antenna.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-Y6NF*	2 element Yagi	6.1 dBi	N	-	Fixed/mobile
A09-Y7NF*	3 element Yagi	7.1 dBi	N	-	Fixed/mobile
A09-Y8NF	4 element Yagi	8.1 dBi	N	-	Fixed/mobile
A09-Y9NF*	4 element Yagi	9.1 dBi	N	-	Fixed/mobile
A09-Y10NF*	5 element Yagi	10.1 dBi	N	-	Fixed/mobile
A09-Y11NF	6 element Yagi	11.1 dBi	N	-	Fixed/mobile
A09-Y12NF*	7 element Yagi	12.1 dBi	N	-	Fixed/mobile
A09-Y13NF*	9 element Yagi	13.1 dBi	N	-	Fixed/mobile
A09-Y14NF*	10 element Yagi	14.1 dBi	N	-	Fixed/mobile
A09-Y14NF-ALT*	12 element Yagi	14.1 dBi	N	-	Fixed/mobile
A09-Y15NF	13 element Yagi	15.1 dBi	N	0.7 dB	Fixed/mobile
A09-Y15NF-ALT*	15 element Yagi	15.1 dBi	N	0.7 dB	Fixed/mobile
A09-Y6TM*	2 element Yagi	6.1 dBi	RPTNC	-	Fixed/mobile
A09-Y7TM*	3 element Yagi	7.1 dBi	RPTNC	-	Fixed/mobile
A09-Y8TM*	4 element Yagi	8.1 dBi	RPTNC	-	Fixed/mobile
A09-Y9TM*	4 element Yagi	9.1 dBi	RPTNC	-	Fixed/mobile
A09-Y10TM-P10*	5 element Yagi	10.1 dBi	RPTNC	-	Fixed/mobile
A09-Y11TM*	6 element Yagi	11.1 dBi	RPTNC	-	Fixed/mobile
A09-Y12TM*	7 element Yagi	12.1 dBi	RPTNC	-	Fixed/mobile
A09-Y13TM*	9 element Yagi	13.1 dBi	RPTNC	-	Fixed/mobile
A09-Y14TM*	10 element Yagi	14.1 dBi	RPTNC	-	Fixed/mobile

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-Y14TM-ALT*	12 element Yagi	14.1 dBi	RPTNC	-	Fixed/mobile
A09-Y15TM*	13 element Yagi	15.1 dBi	RPTNC	0.7 dB	Fixed/mobile
A09-Y15TM-P10I	15 element Yagi	15.1 dBi	RPTNC	0.7 dB	Fixed/mobile

Omni-directional base station antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F0NF*	Fiberglass Base Station	0 dBi	N	-	Fixed
A09-F1NF*	Fiberglass Base Station	1.0 dBi	N	-	Fixed
A09-F2NF-M*	Fiberglass Base Station	2.1 dBi	N	-	Fixed
A09-F3NF*	Fiberglass Base Station	3.1 dBi	N	-	Fixed
A09-F4NF*	Fiberglass Base Station	4.1 dBi	N	-	Fixed
A09-F5NF-M	Fiberglass Base Station	5.1 dBi	N	-	Fixed
A09-F6NF*	Fiberglass Base Station	6.1 dBi	N	-	Fixed
A09-F7NF*	Fiberglass Base Station	7.1 dBi	N	-	Fixed
A09-F8NF-M	Fiberglass Base Station	8.1 dBi	N	0.7 dB	Fixed
A09-F0SM*	Fiberglass Base Station	0 dBi	RPSMA	-	Fixed
A09-F1SM*	Fiberglass Base Station	1.0 dBi	RPSMA	-	Fixed
A09-F2SM*	Fiberglass Base Station	2.1 dBi	RPSMA	-	Fixed
A09-F3SM*	Fiberglass Base Station	3.1 dBi	RPSMA	-	Fixed
A09-F4SM*	Fiberglass Base Station	4.1 dBi	RPSMA	-	Fixed

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F5SM*	Fiberglass Base Station	5.1 dBi	RPSMA	-	Fixed
A09-F6SM*	Fiberglass Base Station	6.1 dBi	RPSMA	-	Fixed
A09-F7SM*	Fiberglass Base Station	7.1 dBi	RPSMA	-	Fixed
A09-F8SM*	Fiberglass Base Station	8.1 dBi	RPSMA	0.7 dB	Fixed
A09-F0TM*	Fiberglass Base Station	0 dBi	RPTNC	-	Fixed
A09-F1TM*	Fiberglass Base Station	1.0 dBi	RPTNC	-	Fixed
A09-F2TM*	Fiberglass Base Station	2.1 dBi	RPTNC	-	Fixed
A09-F3TM*	Fiberglass Base Station	3.1 dBi	RPTNC	-	Fixed
A09-F4TM*	Fiberglass Base Station	4.1 dBi	RPTNC	-	Fixed
A09-F5TM*	Fiberglass Base Station	5.1 dBi	RPTNC	-	Fixed
A09-F6TM*	Fiberglass Base Station	6.1 dBi	RPTNC	-	Fixed
A09-F7TM*	Fiberglass Base Station	7.1 dBi	RPTNC	-	Fixed
A09-F8TM*	Fiberglass Base Station	8.1 dBi	RPTNC	0.7 dB	Fixed
A09-W7*	Wire Base Station	7.1 dBi	RPN	-	Fixed
A09-W7SM*	Wire Base Station	7.1 dBi	RPSMA	-	Fixed
A09-W7TM*	Wire Base Station	7.1 dBi	RPTNC	-	Fixed

Dome antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-D3PNF*	Omnidirectional permanent mount	3.0 dBi	N	0.4 dB	Fixed/mobile

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-D3NF*	Omnidirectional magnetic mount	3.0 dBi	N	0.4 dB	Fixed/mobile
A09-D3PTM*	Omnidirectional permanent mount	3.0 dBi	RPTNC	0.4 dB	Fixed/mobile
A09-D3PSM*	Omnidirectional permanent mount	3.0 dBi	RPSMA	0.4 dB	Fixed/mobile

Monopole antennas

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-QRAMM	3" Quarter wave wire	2.1 dBi	MMCX	-	Fixed/mobile
A09-QRSM-2.1*	Quarter wave 2.1" right angle	3.3 dBi	RPSMA	0.4 dB	Fixed/mobile
A09-QW*	Quarter wave wire	1.9 dBi	Permanent	-	Fixed/mobile
A09-QSM-3*	Quarter wave straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QSM-3H*	Heavy duty quarter wave straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QBMM-P6I*	Quarter wave w/ 6" pigtail	1.9 dBi	MMCX	-	Fixed/mobile
A09-QHSM-2*	2" straight	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QHRSM-2*	2" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QHRSM-170*	1.7" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QRSM-380*	3.8" right angle	1.9 dBi	RPSMA	-	Fixed/mobile
A09-QAPM-520*	5.2" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QSPM-3*	3" straight screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QAPM-3*	3" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile
A09-QAPM-3H*	3" articulated screw mount	1.9 dBi	Permanent	-	Fixed/mobile

FCC publication 996369 related information

In publication 996369 section D03, the FCC requires information concerning a module to be presented by OEM manufacturers. This section assists in answering or fulfilling these requirements.

2.1 General

No requirements are associated with this section.

2.2 List of applicable FCC rules

This module conforms to FCC Part 15.247.

2.3 Summarize the specific operational use conditions

Certain approved antennas require attenuation for operation. For the XBee SX, see XBee SX antenna options (13 dBm maximum RF power). For the XBee-PRO SX, see XBee-PRO SX antenna options (30 dBm maximum RF power).

Host product user guides should include the antenna table if end customers are permitted to select antennas.

2.4 Limited module procedures

Not applicable.

2.5 Trace antenna designs

While it is possible to build a trace antenna into the host PCB, this requires at least a Class II permissive change to the FCC grant which includes significant extra testing and cost. If an embedded trace or chip antenna is desired contact a Digi sales representative for information on how to engage with a lab to get the modified FCC grant.

2.6 RF exposure considerations

For RF exposure considerations see RF exposure statement, XBee-PRO SX antenna options (30 dBm maximum RF power) and XBee SX antenna options (13 dBm maximum RF power).

Host product manufacturers need to provide end-users a copy of the “RF Exposure” section of the manual: RF exposure statement.

2.7 Antennas

A list of approved antennas is provided for the SX product. For the XBee SX, see XBee SX antenna options (13 dBm maximum RF power). For the XBee-PRO SX, see XBee-PRO SX antenna options (30 dBm maximum RF power).

2.8 Label and compliance information

Host product manufacturers need to follow the sticker guidelines outlined in OEM labeling requirements.

2.9 Information on test modes and additional testing requirements

Contact a Digi sales representative for information on how to configure test modes for the XBee SX and XBee-PRO SX modules.

2.10 Additional testing, Part 15 Subpart B disclaimer

All final host products must be tested to be compliant to FCC Part 15 Subpart B standards. While the XBee SX and XBee-PRO SX modules were tested to be complaint to FCC unintentional radiator standards, FCC Part 15 Subpart B compliance testing is still required for the final host product. This testing is required for all end products, and XBee SX/XBee-PRO SX Part 15 Subpart B compliance does not affirm the end product's compliance.

See FCC notices for more details.

ISED (Innovation, Science and Economic Development Canada)

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Labeling requirements

XBee SX

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product must display the following text:

Contains Model XBSX Radio, IC: 1846A-XBSX

The integrator is responsible for its product to comply with IC ICES-003 and FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

XBee-PRO SX

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product must display the following text:

Contains Model XBPSX Radio, IC: 1846A-XBPSX

The integrator is responsible for its product to comply with IC ICES-003 and FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Transmitters for detachable antennas

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the tables in FCC antenna certifications with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device. The required antenna impedance is 50 ohms.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Detachable antennas

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres

utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire àl'établissement d'une communication satisfaisante.

ACMA (Australia)

Power requirements

Regulations in Australia stipulate a maximum of 30 dBm EIRP (Effective Isotropic Radiated Power). The EIRP equals the sum (in dBm) of power output, antenna gain and cable loss and cannot not exceed 30 dBm.

The EIRP formula for Australia is:

power output + antenna gain - cable loss <= 30 dBm

Note The maximum EIRP for the FCC (United States) and IC (Canada) is 36 dBm.

These modules comply with requirements to be used in end products in Australia. All products with EMC and radio communications must have a registered RCM mark. Registration to use the compliance mark will only be accepted from Australian manufacturers or importers, or their agent, in Australia. In order to have a RCM mark on an end product, a company must comply with a or b below:

a. have a company presence in Australia.
b. have a company/distributor/agent in Australia that will sponsor the import of the end product.

Contact Digi for questions related to locating a contact in Australia.

RSM (New Zealand)

Power requirements

No antenna with gain greater that 2.1 dBi (dipole) can be used with this radio in New Zealand.

These modules comply with requirements to be used in end products in New Zealand. All products with EMC and radio communications must have a registered R-NZ mark. Registration to use the compliance mark will only be accepted from manufacturers or importers, or their agent, in New Zealand. In order to have a R-NZ mark on an end product, a company must comply with a or b below:

a. have a company presence in New Zealand.
b. have a company/distributor/agent in New Zealand that will sponsor the import of the end product.

Brazil (Anatel)

ANATEL Brazil for XBee-PRO SX radio products (XBPSX)

The XBee SX-PRO RF modules (models noted 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): Digi Model: XBPSX

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

ANATEL

Agência Nacional de Telecomunicações

05774-16-01209

SX ANATEL Brazil for XK9X-DMS-1 XBee SX RF Module Dev Kit (XK9X-DMS-1)

The XBee SX RF modules (models noted 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): Digi Model: XK9X-DMS-1

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

ANATEL

Agência Nacional de Telecomunicações

01434-17-01209

ANATEL Brazil for XBee SX radio products (XBSX)

The XBee SX RF modules (models noted 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): Digi Model: XBSX

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados.

ANATEL

Agência Nacional de Telecomunicações

05776-16-01209

IFETEL (Mexico)

Mexico does not have a “modular” approval for radio products.

If XBee/XBee-PRO SX RF Modules are integrated into a product, the manufacturer must ensure compliance of the final product with all the requirements of Mexican regulations.

Mexico IFETEL

Manufacturer: Digi International

Country: USA

Brand: Digi

Tariff Code (HS): 8517-62-14

Model: XBPSX

IFETEL (IFT) number RCPDIXB19-2288 applies to these XBee-PRO SX radios:

- XBP9X-DMRS-001, XBP9X-DMUS-001

Model: XBSX

IFETEL (IFT) number RCPDIXB19-1819 applies to these XBee-PRO SX radios:

- XB9X-DMRS-001, XB9X-DMUS-001

OEM labeling requirements

WARNING! The Original Equipment Manufacturer (OEM) must ensure that Mexico IFT labeling requirements are met. This includes a clearly visible label on the outside of the final product enclosure that displays the text shown in the figure below.

The IFETEL number for the XBee-PRO SX product must be listed either on the end product, on the packaging, in the manual, or in the software with the following phrase:

> “*Este equipo contiene el módulo XBP9X con Número IFETEL*: RCPDIXB19-2288”

OR

> “Este equipo contiene el módulo XBP9X con IFT #: RCPDIXB19-2288”

The IFETEL number for the XBee SX product must be listed either on the end product, on the packaging, in the manual, or in the software with the following phrase:

> “*Este equipo contiene el módulo XB9X con Número IFETEL*: RCPDIXB19-1819”

OR

> “Este equipo contiene el módulo XB9X con IFT #: RCPDIXB19-1819”

The following paragraph must also be present in the User Manual for the end product:

> “La operación de este equipo está sujeta a las siguientes dos condiciones: (1) es posible que este equipo o dispositivo no cause interferencia perjudicial y (2) este equipo o dispositivo debe aceptar cualquier interferencia, incluyendo la que pueda causar su operación no deseada.”

PCB design and manufacturing

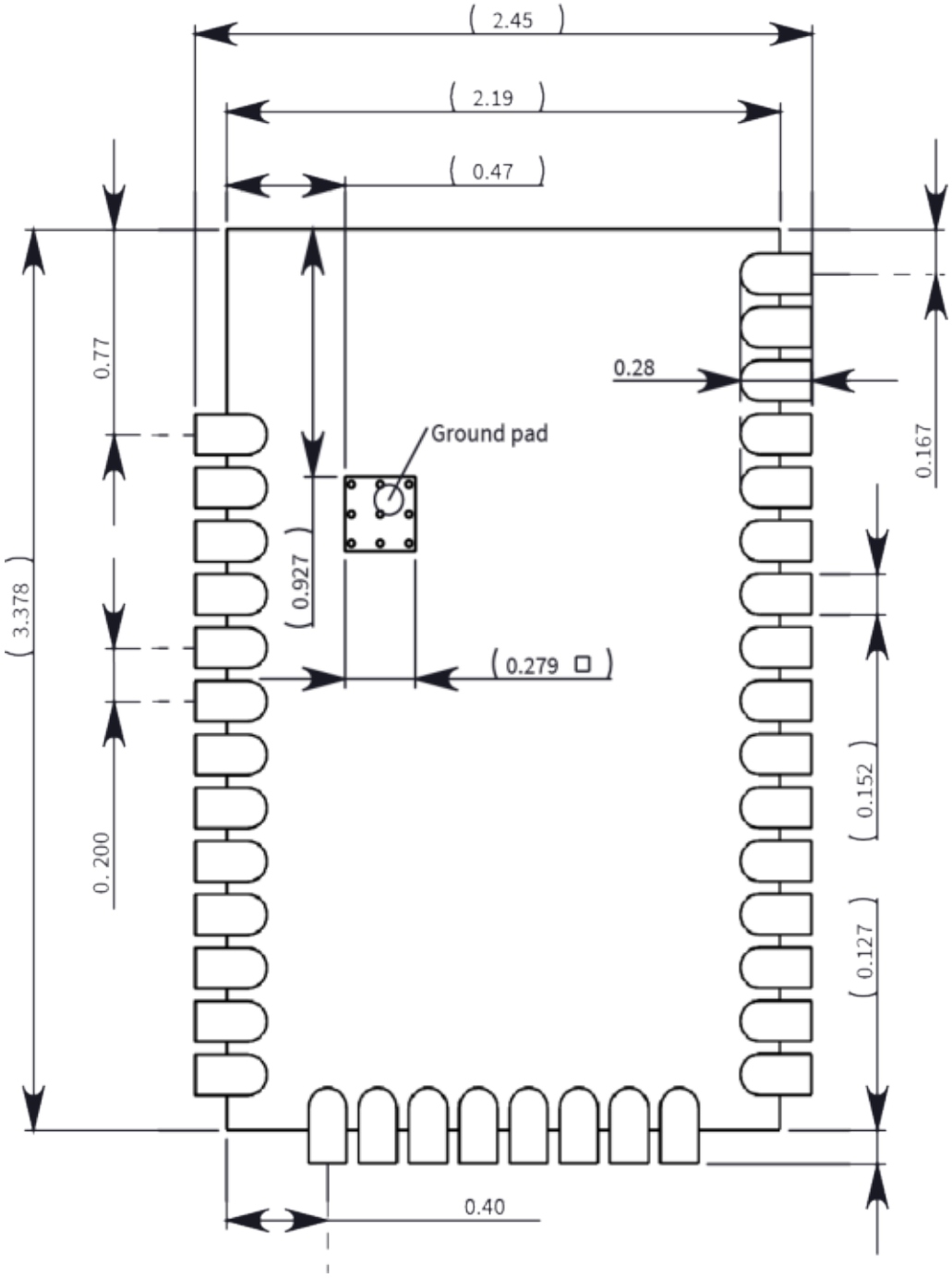
The XBee/XBee-PRO SX RF Module is designed for surface-mount on the OEM PCB. It has castellated pads to allow for easy solder attach inspection. The pads are all located on the edge of the module, so there are no hidden solder joints on these modules.

Recommended footprint and keepout 230
Design notes 232
Recommended solder reflow cycle 234
Flux and cleaning 235
Rework 235

Recommended footprint and keepout

We designed the XBee/XBee-PRO SX RF Module for surface-mounting on the OEM printed circuit board (PCB). It has castellated pads around the edges and one ground pad on the bottom. Mechanical drawings includes a detailed mechanical drawing.

We recommend that you use the following PCB footprint for surface-mounting. Dimensions are in centimeters.



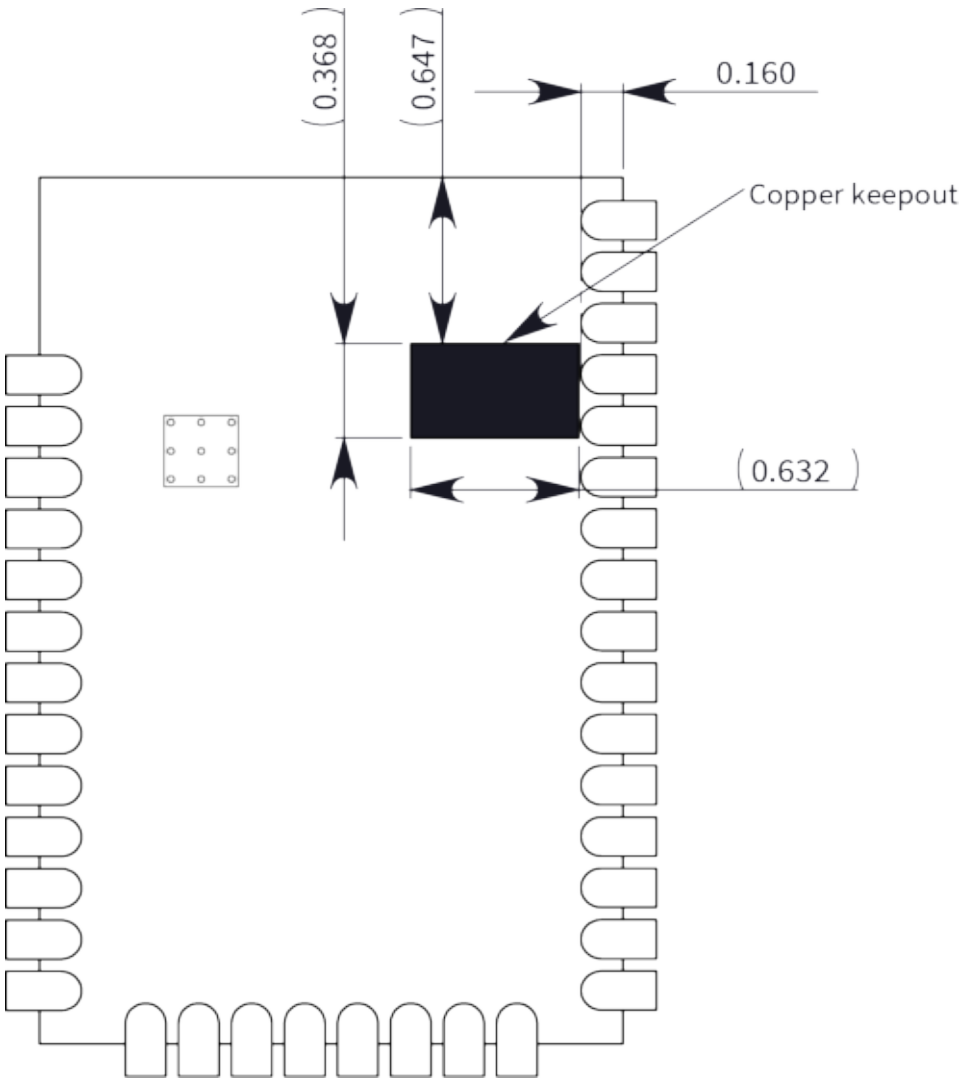
The recommended footprint includes an additional ground pad that you must solder to the corresponding pad on the device. This ground pad transfers heat generated during transmit mode away from the device's power amplifier. The pad must connect through vias to a ground plane on the host PCB. Connecting to planes on multiple layers will further improve the heat transfer performance and we recommend doing this for applications that will be in transmit mode for sustained periods. We recommend using nine 0.030 cm diameter vias in the pad as shown. Plug vias with epoxy or solder mask them on the opposite side to prevent solder paste from leaking through the holes during reflow. Do not mask over the ground pad.

Note The ground pad is unique to the XBee/XBee-PRO XTC and SX modules. This footprint is not compatible with other SMT XBees.

Although the underside of the device is mostly coated with solder mask, we recommend that you leave the copper layer directly below the device open to avoid unintended contacts. Most importantly, copper or vias must not interfere with the three exposed RF test points on the bottom of the device shown in the following keepout drawing. Observe the copper keepout on all layers of the host PCB, to avoid the possibility of capacitive coupling that could impact RF performance.

Match the solder footprint to the copper pads, but you may need to adjust it depending on the specific needs of assembly and product standards. We recommend a stencil thickness of 0.15 mm (0.005 in). Place the component last and set the placement speed to the slowest setting.

The following drawing show the SMT footprint, with the required copper keepout (all layers). Dimensions are in centimeters.



Design notes

The following guidelines help to ensure a robust design.

Host board design

A good power supply design is critical for proper device operation. If the supply voltage is not kept within tolerance, or is excessively noisy, it may degrade device performance and reliability. To help reduce noise, we recommend placing both a 1 uF and 100 pF capacitor as near to VCC as possible. If you use a switching regulator, we recommend switching frequencies above 500 kHz and you should limit power supply ripple to a maximum 50 mV peak to peak.

As with all PCB designs, make power and ground traces thicker than signal traces and make them able to comfortably support the maximum current specifications. Ground planes are preferable.

Improve antenna performance

The choice of antenna and antenna location is important for optimal performance. In general, antenna elements radiate perpendicular to the direction they point. Thus a vertical antenna, such as a dipole, emit across the horizon.

Metal objects near the antenna cause parasitic coupling and detuning, preventing the antenna from radiating efficiently. Metal objects between the transmitter and receiver can also block the radiation path or reduce the transmission distance, so position external antennas away from them as much as possible. Some objects that are often overlooked are:

- Metal poles
- Metal studs or beams in structures
- Concrete (reinforced with metal rods)
- Metal enclosures
- Vehicles
- Elevators
- Ventilation ducts
- Large appliances
- Batteries
- Tall electrolytic capacitors

RF pad version

The RF pad is a soldered antenna connection. The RF signal travels from pin on the module to the antenna through a single ended RF transmission line on the PCB. This line should have a controlled impedance of 50 Ω.

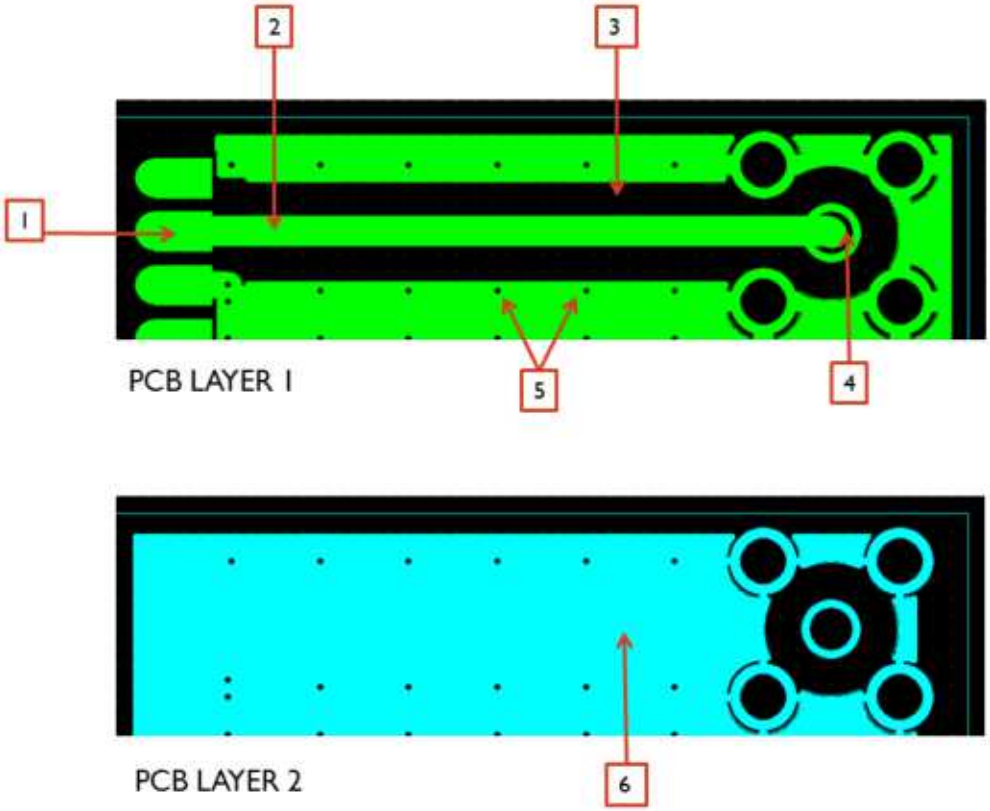
For the transmission line, we recommend either a microstrip or coplanar waveguide trace on the PCB. We provide a microstrip example below, because it is simpler to design and generally requires less area on the host PCB than coplanar waveguide.

We do not recommend using a stripline RF trace because that requires routing the RF trace to an inner PCB layer, and via transitions can introduce matching and performance problems.

The following figure shows a layout example of a microstrip connecting an RF pad module to a through-hole RPSMA RF connector.

- The top two layers of the PCB have a controlled thickness dielectric material in between. The second layer has a ground plane which runs underneath the entire RF pad area. This ground plane is a distance *d*, the thickness of the dielectric, below the top layer.
- The top layer has an RF trace running from pin 36 of the device to the RF pin of the RPSMA connector. The RF trace's width determines the impedance of the transmission line with relation to the ground plane. Many online tools can estimate this value, although you should consult the PCB manufacturer for the exact width. Assuming d = 0.025 in, and that the dielectric has a relative permittivity of 4.4, the width in this example will be approximately 0.045 in for a 50 Ω trace. This trace width is a good fit with the module footprint's 0.060 in pad width.

We do not recommend using a trace wider than the pad width, and using a very narrow trace can cause unwanted RF loss. You can minimize the length of the trace by placing the RPSMA jack close to the module. All of the grounds on the jack and the module are connected to the ground planes directly or through closely placed vias. Space any ground fill on the top layer at least twice the distance *d* (in this case, at least 0.050 in) from the microstrip to minimize their interaction.



Number	Description
1	XBee pin 36
2	50 Ω microstrip trace
3	Back off ground fill at least twice the distance between layers 1 and 2
4	RF connector
5	Stitch vias near the edges of the ground plane
6	Pour a solid ground plane under the RF trace on the reference layer

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

Recommended solder reflow cycle

The following table provides the recommended solder reflow cycle. The table shows the temperature setting and the time to reach the temperature; it does not show the cooling cycle.

Time (seconds)	Temperature (degrees C)
30	65
60	100
90	135
120	160
150	195
180	240
210	260

The maximum temperature should not exceed 260 °C.

The SX device will reflow during this cycle, and therefore must not be reflowed upside down. Take care not to jar the device while the solder is molten, as this can remove components under the shield from their required locations.

The device has a Moisture Sensitivity Level (MSL) of 3. When using this product, consider the relative requirements in accordance with standard IPC/JEDEC J-STD-020.

In addition, note the following conditions:

a. Calculated shelf life in sealed bag: 12 months at < 40 °C and < 90% relative humidity (RH).
b. Environmental condition during the production: 30 °C /60% RH according to IPC/JEDEC J-STD-033C, paragraphs 5 through 7.
c. The time between the opening of the sealed bag and the start of the reflow process cannot exceed 168 hours if condition b) is met.
d. Baking is required if conditions b) or c) are not met.
e. Baking is required if the humidity indicator inside the bag indicates a RH of 10% more.
f. If baking is required, bake modules in trays stacked no more than 10 high for 4-6 hours at 125 °C.

Flux and cleaning

We recommend that you use a “no clean” solder paste in assembling these devices. This eliminates the clean step and ensures that you do not leave unwanted residual flux under the device where it is difficult to remove. In addition:

- Cleaning with liquids can result in liquid remaining under the device or in the gap between the device and the host PCB. This can lead to unintended connections between pads.
- The residual moisture and flux residue under the device are not easily seen during an inspection process.

Rework

Once you mount the device, do not perform rework on the SX device (for example, removing it from the host PCB).

CAUTION! Any modification to the device voids the warranty coverage and certifications.