



XBee[®] Wi-Fi RF Module

S6B

User Guide

XBee Wi-Fi RF Module S6B User Guide

Part number 90002180 N

Revision	Date	Description
B to J	March 2013 to October 2014	Various operation, specification and instructional updates
K	January 2015	Added revision table; added ETSI updates; general document updates
L	September 2015	Fixed an error related to instructions for DIO13/DOUT. Added note to the Pull Direction in the AT command table. Updated to new template.
M	September 2015	Added serial data interface and serial data throughput to the RF specifications.
N	April 2016	Updated Device Cloud and XCTU information. Explained the use of ZigBee frames in the firmware. Added New Zealand certification information.

Product documentation

To find up-to-date documentation for all Digi products, visit www.digi.com/documentation.

To provide feedback on this documentation, send your comments to techcomm@digi.com.

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.

© 2016 Digi International. 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 warranties online, visit www.digi.com/howtobuy/terms.

Customer support

Digi offers multiple technical support plans and service packages to help our customers get the most out of their Digi product. For information on Technical Support plans and pricing, please contact us at 952.912.3456 or visit www.digi.com/support.

If you have a customer account, sign in to the Customer Support Web Portal at www.digi.com/support/eservice

Contents

Overview

XBee Wi-Fi module specifications	7
General specifications	7
RF specifications	8
RF data rates	9
Receiver sensitivity	10
RF transmit power - typical	11
EVM - maximum output power - typical	12
Electrical specifications	13
Serial communication specifications	14
Design notes	19
Power supply	19
Through-hole device keepout area	21
Surface-mount device keepout area	22

RF module operation

Serial communication	26
UART communications	26
SPI communications	27
Serial buffers	29
Serial interface protocols	30
Modes of operation	32
Idle mode	32
Transmit mode	32
Command mode	32

802.11 bgn networks

Infrastructure networks	37
Ad hoc networks	37

XBee IP services

XBee application service	41
Local host	41
TX64 and RX64 API frames	42
Network client	43

Sending configuration commands	43
Sending the serial data command to XBee	46
Sending over-the-air firmware upgrades	48
Transparent mode	49

Sleep

Using sleep mode: UART	51
Using sleep mode: SPI	52
Sleep options	52
AP associated sleep	52
Pin sleep mode	52
Cyclic sleep mode	53
Deep sleep (non-associated sleep)	53
Pin sleep mode	53
Cyclic sleep mode	53

Advanced application features

XBee analog and digital I/O lines	55
I/O sampling	57
Soft AP mode	60
Enable Soft AP mode	60
Use Soft AP mode	60
Wi-Fi protected setup	61
Commissioning button	62
Connection indicators	62
Working with flash memory	69
Over-the-air firmware upgrades	70
Distributing the new application	70
Verifying the new application	71
Installing the application	71
Things to remember	71

Configure and query devices with Device Cloud

Use XCTU to enable Device Cloud	72
Configure the device	72
Output control	73
IO command bits	73
Send I/O samples to Device Cloud	74
View I/O samples in Device Cloud	75
Update the firmware	75
Send data requests	75
The device request process	75
About the device request and frame ID	76
Populate and send a Device Request frame (0xB9)	76
Transparent mode data	77
Send data to Device Cloud	77
Send files	78
Send binary data points	78
Receive data from Device Cloud	78

API operation

API frame specifications	80
API UART and SPI exchanges	84
AT commands	84
Transmitting and receiving RF data	84
Remote AT commands	84
API frames	84
TX (Transmit) request: 64-Bit	85
Remote AT command request	85
AT command	87
AT command-queue parameter value	87
ZigBee transmit packet	88
ZigBee explicit transmit packet	89
ZigBee remote AT command	91
Transmit (TX) request: IPv4	92
Send data request	93
Device response	94
Rx (Receive) packet: 64-bit	94
Remote command response	95
AT command response	96
Transmission status	97
Modem status	98
ZigBee TX status	99
IO data sample RX indicator	100
ZigBee receive packet	102
Explicit ZigBee receive packet	103
ZigBee remote AT command response	104
RX (Receive) packet: IPv4	105
Send data response	106
Device request	107
Device response status	107
Frame error	108

Command reference tables

Addressing	109
Networking Commands	112
Security commands	113
RF interfacing commands	113
Serial interfacing	114
I/O settings	116
Output Control	121
Diagnostics interfacing	124
AT command options	126
Sleep commands	126
Execution commands	127

Module support

XCTU configuration tool	129
Serial firmware updates	129
Regulatory compliance	129

Agency certifications

United States FCC	130
Europe (CE)	138
OEM labeling requirements	138
Declarations of conformity	139
Approved antennas	139
Canada (IC)	139
Labeling requirements	140
Transmitters with detachable antennas	140
Australia (RCM)/New Zealand (R-NZ)	141
Brazil (ANATEL)	141

Manufacturing information for surface-mount XBee

Recommended solder reflow cycle	142
Recommended footprint	143
Common footprint for through-hole and surface mount	145
Rework	146

Glossary of terms

Definitions	147
-------------------	-----

Overview

The XBee® Wi-Fi RF module provides wireless connectivity to end-point devices in 802.11 bgn networks. Using the 802.11 feature set, these modules are interoperable with other 802.11 bgn devices, including devices from other vendors. With XBee, users can have their 802.11 bgn network up and running in a matter of minutes.

The XBee Wi-Fi modules are compatible with other devices that use 802.11 bgn technology. These include Digi external 802.11x devices like the ConnectPort products and the Digi Connect Wi-SP, as well as embedded products like the ConnectCore series and Digi Connect series of products. Learn more about these products at www.digi.com/products.

XBee Wi-Fi module specifications

General specifications

Specification	XBee Wi-Fi through-hole	XBee Wi-Fi surface mount
Dimensions	0.960 x 1.297 (2.438cm x 3.294cm)	0.866 x 1.330 in (2.200 x 3.378 cm)
Operating Temperature	-30 to 85° C	
Antenna Options	PCB Antenna, U.FL Connector, RPSMA Connector, or Integrated Wire	PCB Antenna, U.FL Connector, or RF Pad

RF specifications

Specification	XBee Wi-Fi Through-hole		XBee Wi-Fi Surface Mount	
Frequency	ISM 2.4-2.5GHz			
Number of Channels	13			
Adjustable Power	Yes			
Wi-Fi Standards	802.11 b, g, and n			
Transmit Power Output (Average)	Up to +16 dBm +13 dBm for Europe/Australia and New Zealand/Brazil (See table below)			
FCC/IC Test Transmit Power Range (Peak)	802.11b	2.73 to 26.81 dBm	802.11b	2.08 to 26.13 dBm
	802.11g	7.87 to 28.52 dBm	802.11g	7.15 to 27.72 dBm
	802.11n (800 ns GI)	8.03 to 28.75 dBm	802.11n (800 ns GI)	7.02 to 27.89 dBm
	802.11n (400 ns GI)	8.04 to 28.64 dBm	802.11n (400 ns GI)	7.33 to 28.20 dBm
RF Data Rates	1 Mb/s to 72.22 Mb/s (See table below)			
Serial Data Interface	UART up to 1 Mb/s, SPI up to 6 MHz			
Serial Data Throughput	UART up to 320 Kb/s, SPI up to 1 Mb/s			
Receiver Sensitivity (25° C, <10% PER)	-93 to -71 dBm (See table below)			

RF data rates

RF data rates			
Standard	Data rates (Mb/s)		
802.11b	1, 2, 5.5, 11		
802.11g	6, 9, 12, 18, 24, 36, 48, 54		
Standard	MCS index	Data rates (Mb/s)	
		800 ns guard interval	400 ns guard interval
802.11n	0	6.5	7.22
	1	13	14.44
	2	19.5	21.67
	3	26	28.89
	4	39	43.33
	5	52	57.78
	6	58.5	65
	7	65	72.22

Receiver sensitivity

Receiver sensitivity (25 °C, < 10% PER)		
Standard	Data rate	Sensitivity (dBm)
802.11b	1 Mb/s	-93
	2 Mb/s	-91
	5.5 Mb/s	-90
	11 Mb/s	-87
802.11g	6 Mb/s	-91
	9 Mb/s	-89
	12 Mb/s	-88
	18 Mb/s	-86
	24 Mb/s	-83
	36 Mb/s	-80
	48 Mb/s	-76
	54 Mb/s	-74
802.11n	MCS 0 6.5/7.22 Mb/s	-91
	MCS 1 13/14.44 Mb/s	-88
	MCS 2 19.5/21.67 Mb/s	-85
	MCS 3 26/28.89 Mb/s	-82
	MCS 4 39/43.33 Mb/s	-78
	MCS 5 52/57.78 Mb/s	-74
	MCS 6 58.5/65 Mb/s	-73
	MCS 7 65/72.22 Mb/s	-71

RF transmit power - typical

RF transmit power (Average)			
Standard	Data rate	Power (dBm)	
		North America/ Japan	Europe/Australia & New Zealand/Brazil
802.11b	1 Mb/s	16	13
	2 Mb/s		
	5.5 Mb/s		
	11 Mb/s		
802.11g	6 Mb/s	16	13
	9 Mb/s		
	12 Mb/s		
	18 Mb/s		
	24 Mb/s		
	36 Mb/s		
	48 Mb/s	14	13
	54 Mb/s		
802.11n	MCS 0 6.5/7.22 Mb/s	15	13
	MCS 1 13/14.44 Mb/s		
	MCS 2 19.5/21.67 Mb/s		
	MCS 3 26/28.89 Mb/s		
	MCS 4 39/43.33 Mb/s		
	MCS 5 52/57.78 Mb/s		
	MCS 6 58.5/65 Mb/s	14	13
	MCS 7 65/72.22 Mb/s		

EVM - maximum output power - typical

EVM (25 °C, max output power)		
Standard	Data rate	EVM (dB)
802.11b	1 Mb/s	-40
	2 Mb/s	-40
	5.5 Mb/s	-38
	11 Mb/s	-36
802.11g	6 Mb/s	-18
	9 Mb/s	-20
	12 Mb/s	-21
	18 Mb/s	-22
	24 Mb/s	-22
	36 Mb/s	-23
	48 Mb/s	-25
	54 Mb/s	-26
802.11n	MCS 0 6.5/7.22 Mb/s	-19
	MCS 1 13/14.44 Mb/s	-21
	MCS 2 19.5/21.67 Mb/s	-22
	MCS 3 26/28.89 Mb/s	-24
	MCS 4 39/43.33 Mb/s	-25
	MCS 5 52/57.78 Mb/s	-25
	MCS 6 58.5/65 Mb/s	-26
	MCS 7 65/72.22 Mb/s	-28

Electrical specifications

Specification	XBee Wi-Fi		
Supply voltage	3.14 - 3.46 VDC		
Operating Current (transmit, max output power)	802.11b	1 Mb/s	309 mA
		2 Mb/s	
		5.5 Mb/s	
		11 Mb/s	
	802.11g	6 Mb/s	271 mA
		9 Mb/s	
		12 Mb/s	
		18 Mb/s	
		24 Mb/s	
		36 Mb/s	
		48 Mb/s	225 mA
		54 Mb/s	
	802.11n	MCS 0 6.5/7.22 Mb/s	260 mA
		MCS 1 13/14.44 Mb/s	
		MCS 2 19.5/21.67 Mb/s	
MCS 3 26/28.89 Mb/s			
MCS 4 39/43.33 Mb/s			
MCS 5 52/57.78 Mb/s			
MCS 6 58.5/65 Mb/s		217 mA	
MCS 7 65/72.22 Mb/s		184 mA	
Operating current (Receive)	100mA		
Deep sleep current	6 μ A @25 °C		
Associated sleep current	2 mA asleep, 100 mA awake. For more information, see AP associated sleep on page 52.		

Serial communication specifications

The XBee Wi-Fi RF modules support both UART (Universal Asynchronous Receiver/Transmitter) and SPI slave mode (Serial Peripheral Interface in slave mode only) serial connections.

UART

Specification UART pins	XBee Wi-Fi Through-hole Module pin number	XBee Wi-Fi Surface Mount Module pin number
DIO13/DOUT	2	3
DIO14/DIN	3	4
DIO7/nCTS	12	25
DIO6/nRTS	16	29

For more information on UART operation see [UART communications](#) on page 26.

SPI

Specification SPI pins	XBee Wi-Fi Through-hole Module pin number	XBee Wi-Fi Surface Mount Module pin number
DIO2/SPI_SCLK	18	14
DIO3/SPI_nSSEL	17	15
DIO4/SPI_MOSI	11	16
DIO12/SPI_MISO	4	17
DIO1/SPI_nATTN	19	12

For more information on SPI operation see [SPI communications](#) on page 27.

GPIO specifications

The XBee Wi-Fi modules have 14 (Through-hole version) and 20 (Surface Mount version) GPIO (General Purpose Input Output) ports available. Those available will depend on the module configuration as some GPIO ports are consumed by serial communication, etc.

See [I/O sampling](#) on page 57 for more information on configuring and using GPIO ports.

Electrical specification for GPIO pads

Parameter	Condition	Min	Max	Units
Input Low Voltage			0.3VDD	V
Input High Voltage		0.7VDD		V
Output high Voltage relative to VDD	Sourcing 2 mA, VDD=3.3 V	85		%
Output low voltage relative to VDD	Sinking 2 mA, VDD=3.3 V		15	%
Output fall time	2 mA drive strength and load capacitance CL=350-600pF.	20+0.1CL	250	ns
I/O pin hysteresis (VIOTHR+ - VIOTHR-)	VDD = 3.14 to 3.46 V	0.1VDD		V
Pulse width of pulses to be removed by the glitch suppression filter		10	50	ns

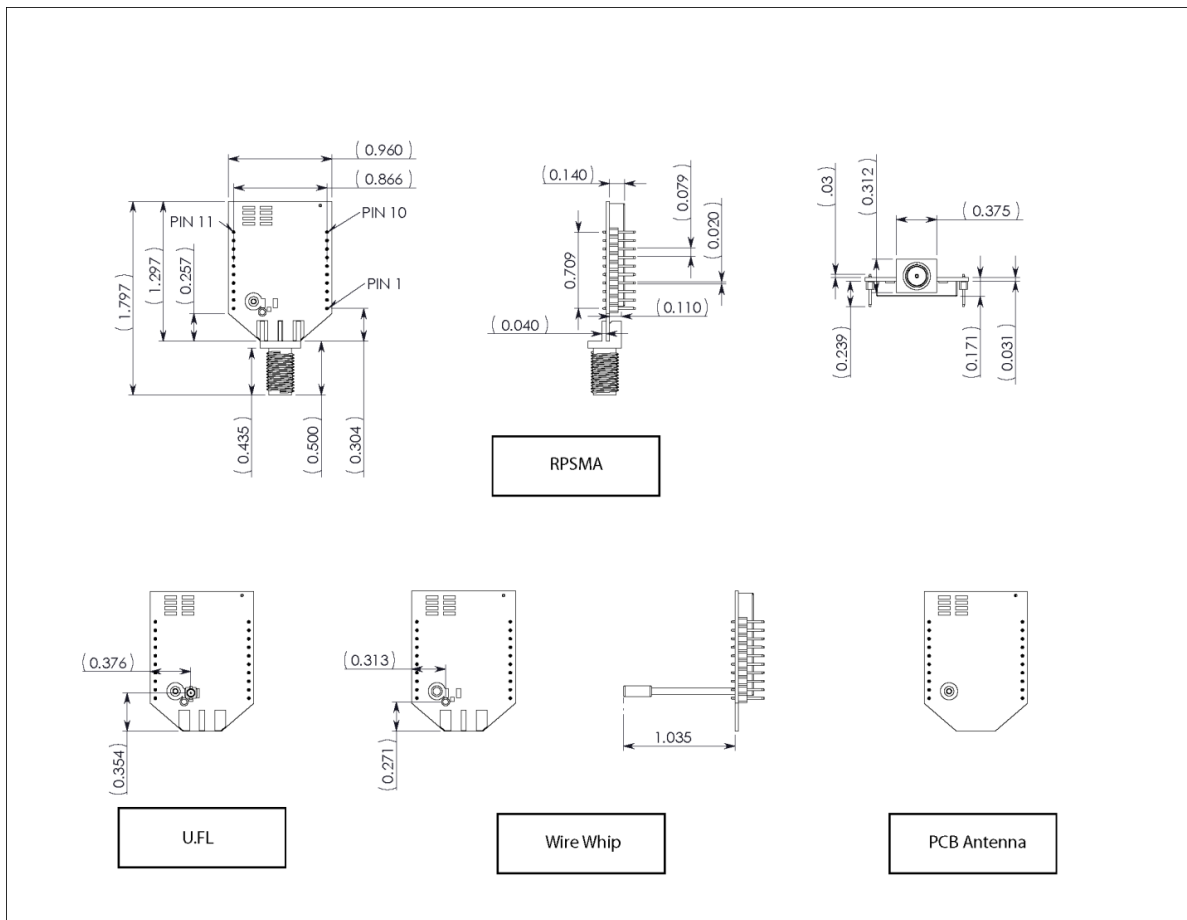
Agency approvals

Specification	XBee Wi-Fi Through-hole	XBee Wi-Fi Surface Mount
United States (FCC Part 15.247)	FCC ID: MCQ-XBS6B	FCC ID: MCQ-S6BSM
Industry Canada (IC)	IC: 1846A-XBS6B	IC: 1846A-S6BSM
Europe	CE	CE
Australia	RCM	RCM
New Zealand	R-NZ	R-NZ
Brazil	ANATEL: 2672-13-1209	ANATEL: 2672-13-1209
Japan	R210-101056	R210-101057

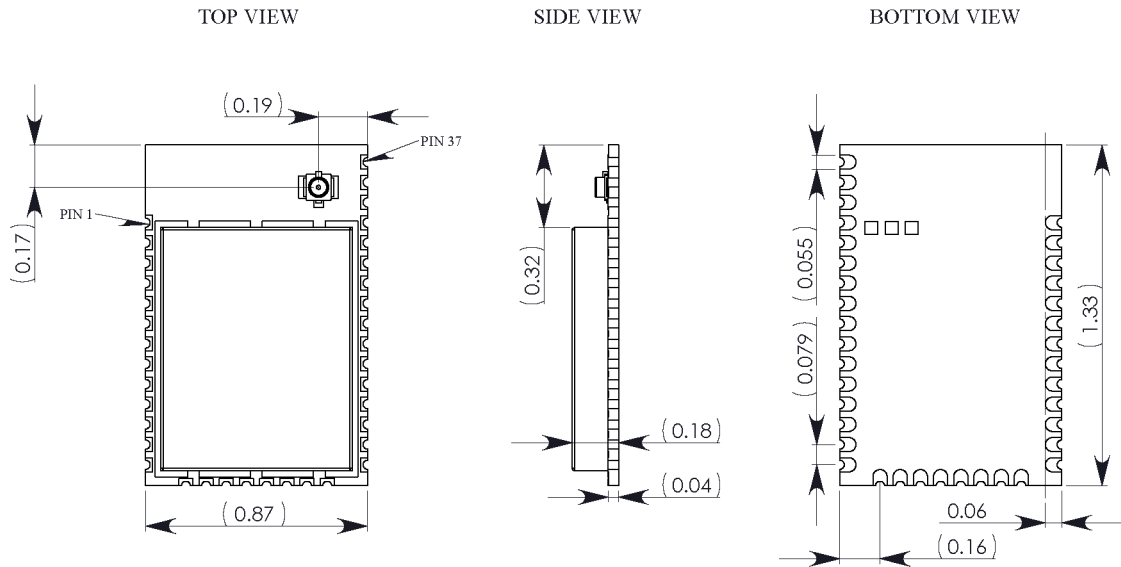
For details about FCC Approval (USA), see [United States FCC](#) on page 130. Systems that contain XBee Wi-Fi modules inherit Digi Certifications.

Mechanical drawings

Through-hole version



Surface Mount version



Pin signals

Pin assignment for the XBee Wi-Fi Through-hole module

(Low-asserted signals are distinguished with a lower case n before the signal name.)

Pin #	Name	Direction	Default state	Description
1	VCC	-	-	Power Supply
2	DIO13/DOUT	Both	Output	UART Data out
3	DIO14/DIN/ $\overline{\text{CONFIG}}$	Both	Input	UART Data In
4	DIO12/SPI_MISO	Both	Disabled	GPIO/ SPI slave out
5	$\overline{\text{RESET}}$	Input	Input	Module Reset
6	DIO10/RSSI PWM/PWM0	Both	Output	RX signal strength indicator/GPIO
7	DIO11/PWM1	Both	Disabled	GPIO
8	reserved	-	-	Do Not Connect
9	DIO8/ $\overline{\text{DTR}}$ /SLEEP_RQ	Both	Input	Pin Sleep Control line /GPIO
10	GND	-	-	Ground
11	DIO4/SPI_MOSI	Both	Disabled	GPIO/SPI slave In
12	DIO7/ $\overline{\text{CTS}}$	Both	Output	Clear-to-Send Flow Control/GPIO

Pin #	Name	Direction	Default state	Description
13	DIO9/ON_SLEEP	Both	Output	Module Status Indicator/GPIO
14	VREF	-	-	Not connected
15	DIO5/ASSOCIATE	Both	Output	Associate Indicator/GPIO
16	DIO6/RTS	Both	Input	Request-to-Send Flow Control/GPIO
17	DIO3/AD3/SPI_SSEL	Both	Disabled	Analog Input/GPIO/SPI Slave Select
18	DIO2/AD2/SPI_CLK	Both	Disabled	Analog Input/GPIO/SPI Clock
19	DIO1/AD1/SPI_ATT \bar{N}	Both	Disabled	Analog Input/GPIO/SPI Attention
20	DIO0/AD0/CB	Both	Disabled	Analog Input/Commissioning Button/GPIO

Pin assignment for the XBee Wi-Fi Surface Mount module

(Low-asserted signals are distinguished with a lower case n before the signal name.)

Pin #	Name	Direction	Default state	Description
1	GND	-	-	Ground
2	VCC	-	-	Power Supply
3	DIO13/DOUT	Both	Output	UART Data Out
4	DIO14/DIN/CONF $\bar{I}G$	Both	Input	UART Data In
5	DIO12	Both	Disabled	GPIO
6	RESE \bar{T}	Input	Input	Module Reset
7	DIO10/ RSSI PWM/PWM0	Both	Output	RX signal strength indicator/GPIO
8	DIO11/PWM1	Both	Disabled	GPIO
9	Reserved	-	-	Do Not Connect
10	DIO8/DTR/SLEEP_RQ	Both	Input	GPIO
11	GND	-	-	Ground
12	DIO19/SPI_ATT \bar{N}	Both	Output	GPIO/SPI Attention
13	GND	-	-	Ground
14	DIO18/SPI_CLK	Both	Input	GPIO/SPI Clock
15	DIO17/SPI_SSEL	Both	Input	GPIO/SPI Slave Select
16	DIO16/SPI_SI	Both	Input	GPIO/SPI Slave In
17	DIO15/SPI_SO	Both	Output	GPIO/SPI Slave Out
18	Reserved	-	-	Do Not Connect

Pin #	Name	Direction	Default state	Description
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	Both	Disabled	GPIO
25	DIO7/ $\overline{\text{CTS}}$	Both	Output	Clear-to-Send Flow Control/ GPIO
26	DIO9/On_ $\overline{\text{SLEEP}}$	Both	Output	Module Status Indicator/GPIO
27	VREF	-	-	Not connected
28	DIO5/ASSOC	Both	Output	Associate Indicator/GPIO
29	DIO6/ $\overline{\text{RTS}}$	Both	Input	Request-to-Send Flow Control/ GPIO
30	DIO3/AD3	Both	Disabled	Analog Input/GPIO
31	DIO2/AD2	Both	Disabled	Analog Input/GPIO
32	DIO1/AD1	Both	Disabled	Analog Input/GPIO
33	DIO0/AD0/CB	Both	Disabled	Analog Input/Commissioning Button/GPIO
34	Reserved	-	-	Do Not Connect
35	GND	-	-	Ground
36	RF	Both	-	RF IO for RF Pad Variant
37	Reserved	-	-	Do Not Connect

Design notes

XBee modules are designed to be self sufficient and do not specifically require any external circuitry other than the recommended pin connections described below. The following sections discuss general design guidelines that are recommended for help in troubleshooting and building a robust design.

Power supply

Poor power supply can lead to poor radio performance, especially if the supply voltage is not kept within tolerance or is excessively noisy. To help reduce noise, 1 μ F and 8.2pF capacitors are recommended to be placed as near to pin 1 on the PCB as possible. If using a switching regulator for your power supply, switching frequencies above 500 kHz are preferred. Power supply ripple should be limited to a maximum 50mV peak to peak.

Recommended pin connections

The only required pin connections are VCC, GND, and either DOUT and DIN or SPI_CLK, SPI_nSSEL, SPI_MOSI, and SPI_MISO. To support serial firmware updates, VCC, GND, DOUT, DIN, nRTS, and nDTR should be connected.

All unused pins should be left disconnected. All inputs on the radio can be pulled high with 40k internal pull-up resistors using the PR software command. No specific treatment is needed for unused outputs.

For applications that need to ensure the lowest sleep current, inputs should never be left floating. Use internal or external pull-up or pull-down resistors, or set the unused I/O lines to outputs. The deep sleep (pin sleep) current specification can be achieved using a standard XBee Interface Board with the XBee Wi-Fi module's pull-up and pull-down resistors configured as default.

Other pins may be connected to external circuitry for convenience of operation. For example, the Associate signal (through-hole pin 15 / surface mount pin 28) and the On_nSLEEP signal (through-hole pin 13 / surface mount pin 26) will change level or behavior based on the state of the module.

Board layout

When designing the host PCB, be sure to account for the module dimensions as shown in the mechanical drawings section. See [Manufacturing information for surface-mount XBee](#) on page 142 for recommended footprints and required keepout areas. Use good design practices when connecting Power and Ground, making those traces wide enough to comfortably support the maximum currents or using planes if possible.

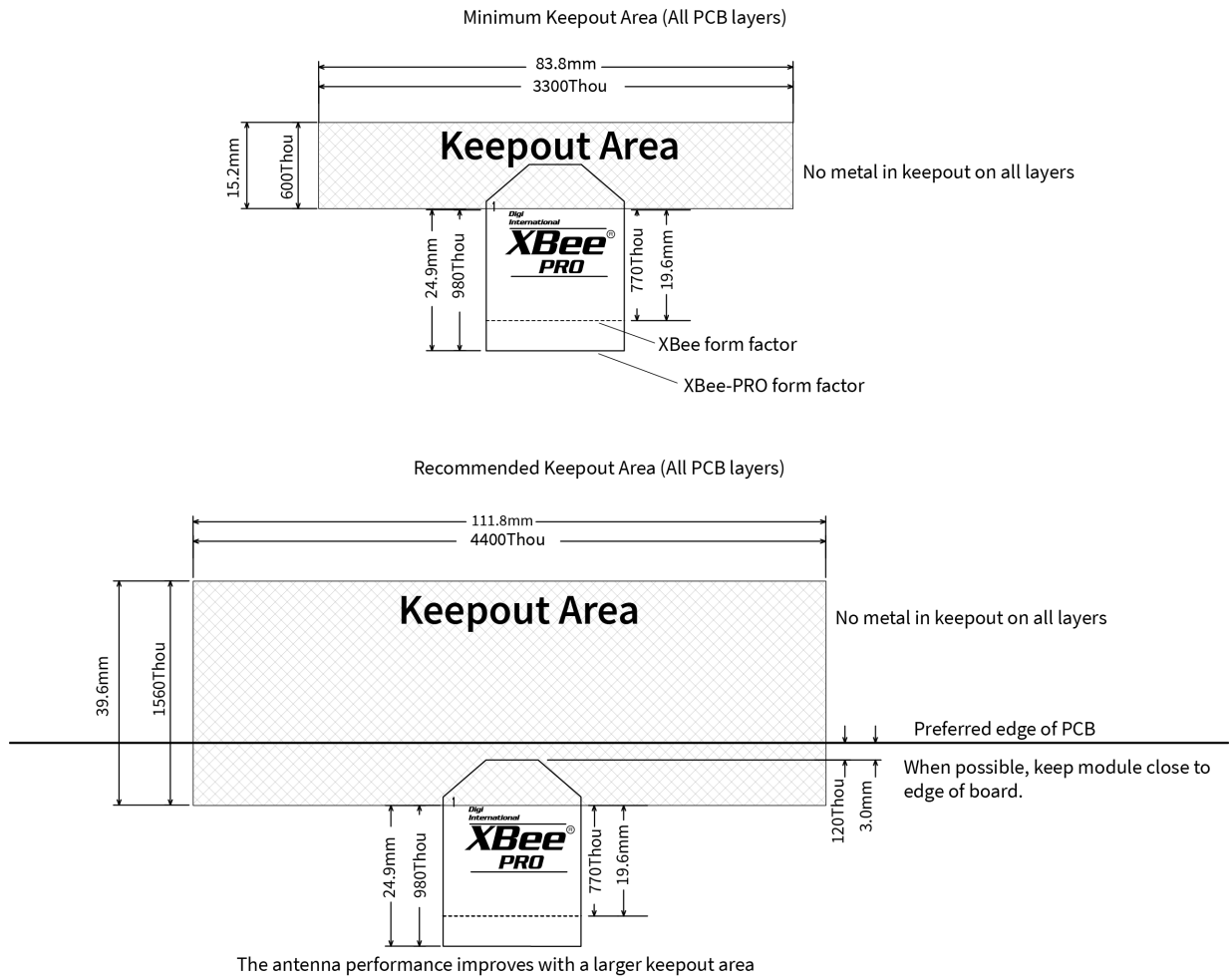
In addition to mechanical considerations, care should be taken in the choice of antenna and antenna location. Most antennas radiate perpendicular to the direction they point. Thus, a vertical antenna emits across the horizon. Metal objects near internal or external antennas may cause reflections and reduce the antenna's ability to radiate efficiently. Antennas should reside above or away from any metal objects, including batteries, tall electrolytic capacitors or metal enclosures. If using a metal enclosure, the antenna should be located externally (using an integral antenna in a metal enclosure will greatly reduce the range). Range may also be affected by metal objects between transmitting and receiving antennas. Some objects that are often overlooked are metal poles, metal studs or beams in structures, concrete (it is usually reinforced with metal rods), metal enclosures, vehicles, elevators, ventilation ducts, refrigerators, and microwave ovens.

Design notes for PCB antenna modules

XBee modules with an embedded PCB antenna should not have any ground planes or metal objects above or below the module at the antenna location. The module should not be placed in a metal enclosure, which may greatly reduce the range. It should be placed at the edge of the PCB to which it is mounted. The ground, power and signal planes should be vacant immediately below the antenna section.

The following two drawings illustrate important recommendations for designing with the PCB Antenna module using the Through-hole and Surface Mount XBee modules, respectively. It should be noted that the Surface Mount PCB antenna module should not be mounted on the RF Pad footprint described in the next section because that footprint requires a ground plane within the keepout area.

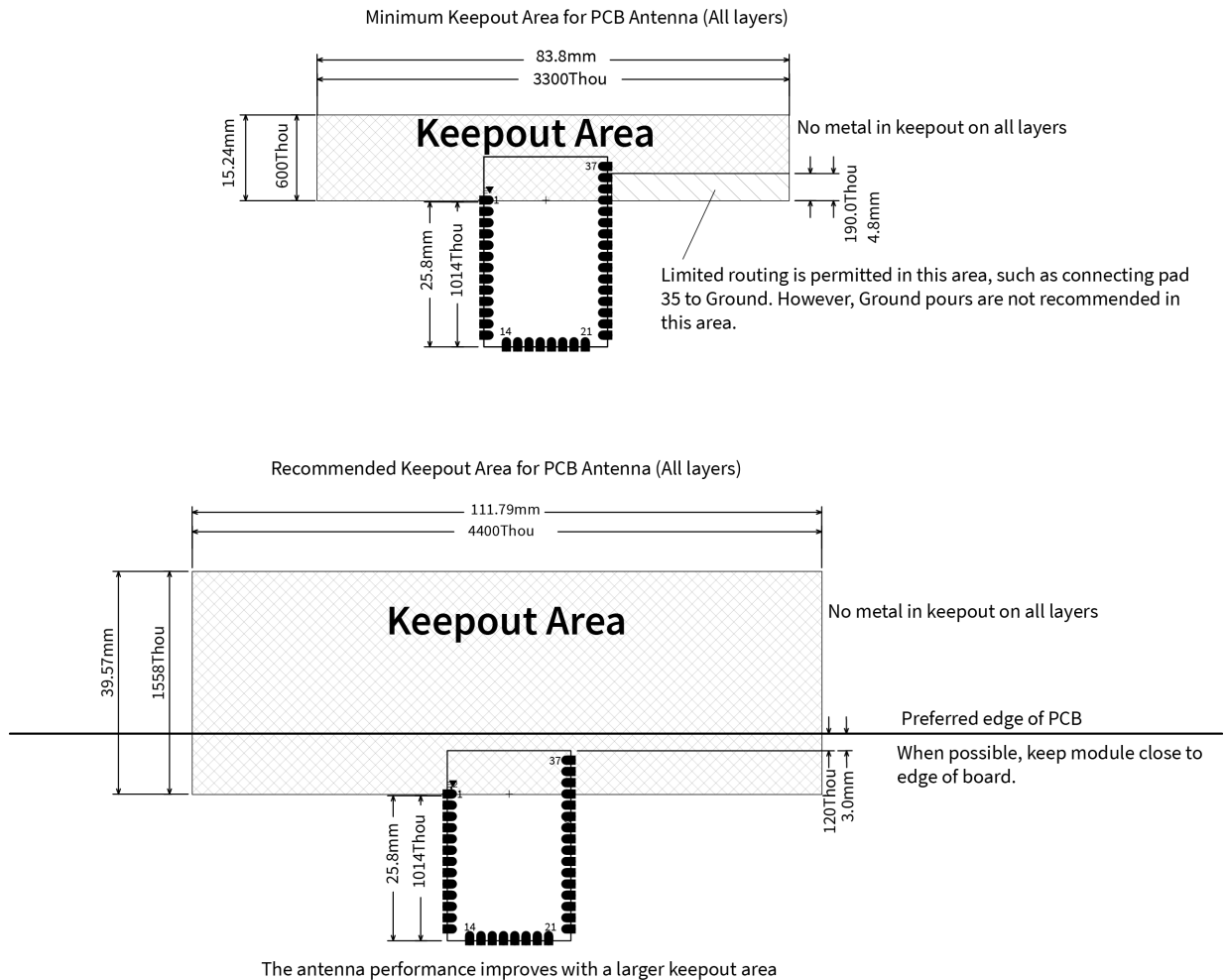
Through-hole device keepout area



Notes:

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

Surface-mount device keepout area



Notes:

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

Design notes for RF pad

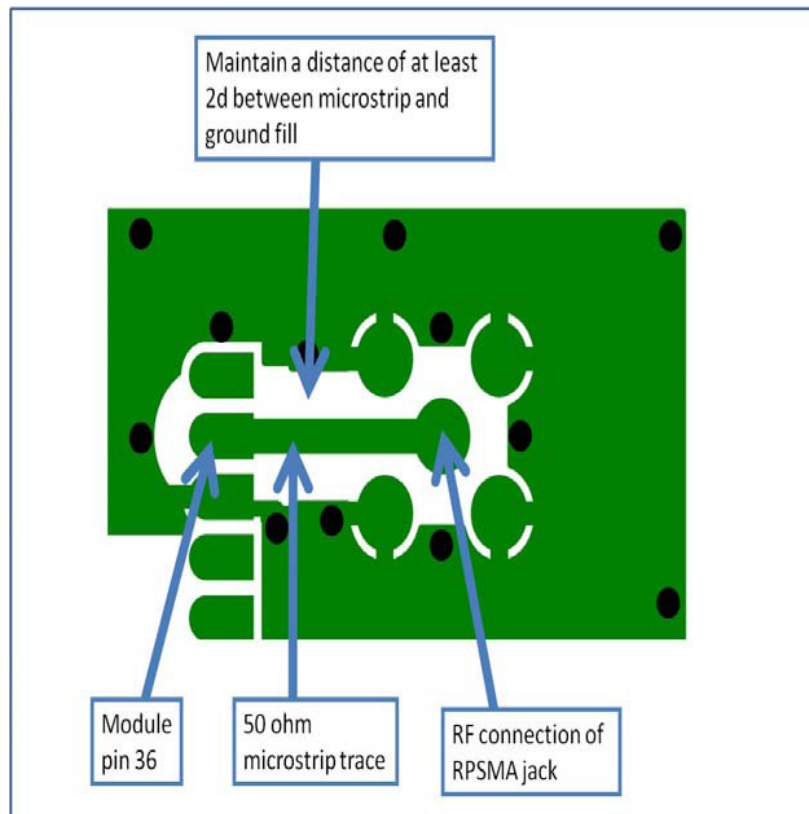
The RF Pad is a soldered antenna connection. The RF signal travels from pin 36 on the module to the antenna through an RF trace transmission line on the PCB. Note that any additional components between the module and antenna will violate modular certification. The RF trace should have a controlled impedance of 50 ohms. We recommend using a microstrip trace, although coplanar waveguide may also be used if more isolation is needed. Microstrip generally requires less area on the PCB than coplanar waveguide. Stripline is not recommended because sending the signal to different PCB layers can introduce matching and performance problems.

It is essential to follow good design practices when implementing the RF trace on a PCB. The following figures show a layout example of a host PCB that connects an RF Pad module to a right angle, through-hole RPSMA jack. 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 module to the RF pin of the RPSMA connector.

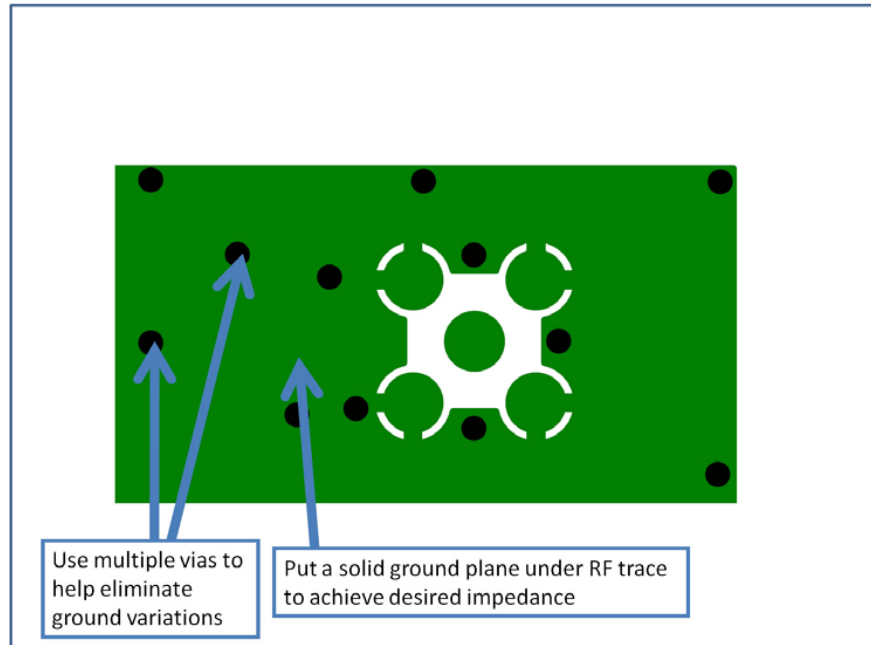
The RF trace's width determines the impedance of the transmission line with relation to the ground plane. Many online tools can estimate this value, although the PCB manufacturer should be consulted for the exact width. Assuming $d=0.025''$, and that the dielectric has a relative permittivity of 4.4, the width in this example will be approximately $0.045''$ for a 50 ohm trace. This trace width is a good fit with the module footprint's $0.060''$ pad width. Using a trace wider than the pad width is not recommended, and using a very narrow trace (under $0.010''$) can cause unwanted RF loss. The length of the trace is minimized 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. Any ground fill on the top layer should be spaced at least twice the distance d (in this case, at least $0.050''$) from the microstrip to minimize their interaction.

Implementing these design suggestions will help ensure that the RF Pad module performs to its specifications.

PCB layer 1 of RF pad layout example



PCB layer 2 of RF pad layout example



Mounting considerations – XBee Wi-Fi Through-hole

XBee Through-hole modules were designed to mount into a receptacle (socket) and therefore do not require any soldering when mounting to a board. XBee interface boards provided in XBee Wi-Fi Development Kits have two ten pin receptacles for connecting the module.

The receptacles used on Digi development boards are manufactured by Century Interconnect. Several other manufacturers provide comparable mounting solutions; however, Digi currently uses the following receptacles:

- Through-hole single-row receptacles - Samtec P/N: MMS-110-01-L-SV (or equivalent)
- Through-hole single-row receptacles - Mill-Max P/N: 831-43-0101-10-001000
- Surface-mount double-row receptacles - Century Interconnect P/N: CPRMSL20-D-0-1 (or equivalent)
- Surface-mount single-row receptacles - Samtec P/N: SMM-110-02-SM-S

We also recommend printing an outline of the module on the board to indicate the orientation the module should be mounted.

RF module operation

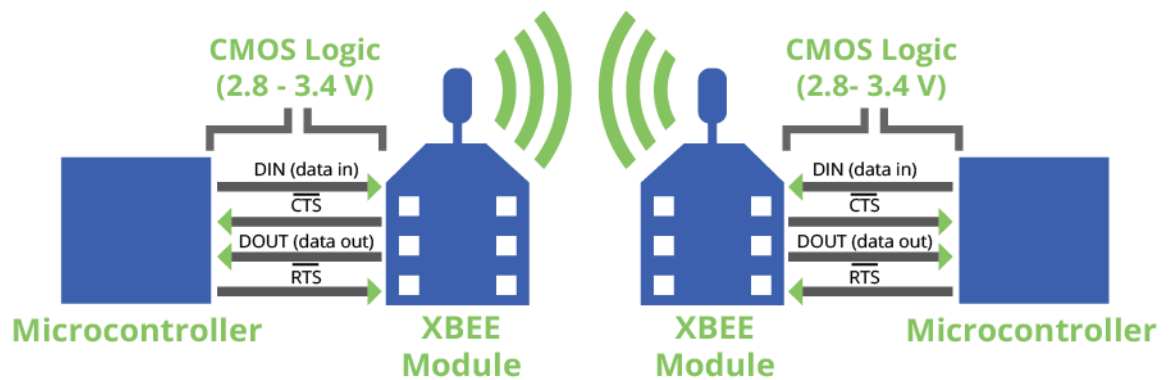
Serial communication

The XBee RF modules interface to a host device through a logic-level asynchronous serial port, or a Serial Peripheral Interface (SPI) port. Through its serial ports, the module can communicate with any logic and voltage compatible UART or SPI; or through a level translator to any serial device (for example: through a RS-232 or USB interface board).

UART communications

UART data flow

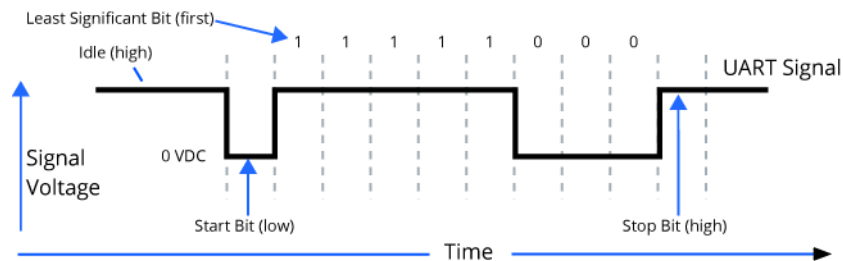
Devices that have a UART interface can connect directly to the pins of the RF module as shown in the following figure. The figure shows the system data flow in a UART-interfaced environment. Low-asserted signals are distinguished with a horizontal line over the signal name.



UART serial data

Data enters the module UART through the DIN pin as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.



Serial communications depend on the two UARTs (the microcontrollers and the RF modules) to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).

The UART baud rate, parity, and stop bits settings on the XBee module can be configured with the BD, NB, and SB commands respectively. For details, see [Command reference tables](#) on page 109.

In the rare case that a radio has been configured with the UART disabled, the module may be recovered to the UART operation by holding DIN low at reset time. As always, DIN forces a default configuration on the UART at 9600 baud and it will bring up the module in command mode on the UART port. Appropriate commands can then be sent to the module to configure it for UART operation. If those parameters are written, then the module will come up with the UART enabled, as desired on the next reset.

SPI communications

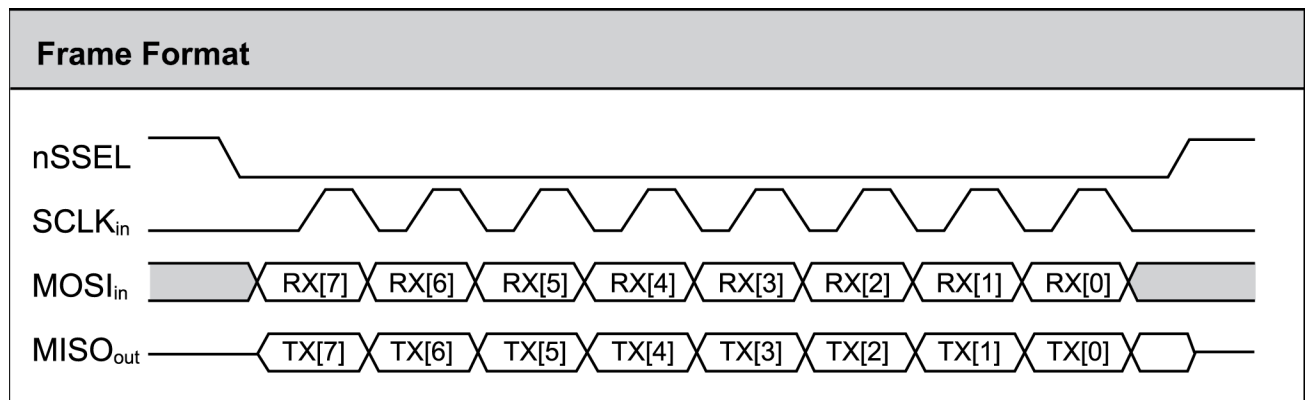
The XBee Wi-Fi module supports SPI communications in the slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The SPI port uses the following signals on the XBee:

- SPI_MOSI (Master Out, Slave In) – inputs serial data from the master
- SPI_MISO (Master In, Slave Out) – outputs serial data to the master
- SPI_SCLK (Serial Clock) – clocks data transfers on MOSI and MISO
- SPI_nSSEL (Slave Select) – enables serial communication with the slave
- SPI_nATTN(Attention) – alerts the master that the slave has data queued to send. The XBee module will assert this pin as soon as data is available to send to the SPI master and it will remain asserted until the SPI master has clocked out all available data.

In this mode the following apply:

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

Frame format for SPI communications



SPI mode is chip to chip communication. Digi does not supply SPI communication option on the Device Development Evaluation Boards.

On the through hole modules, SPI mode can be forced by holding DIO13/DOUT low while resetting the module until SPI_nATTN asserts. By this means, the XBee Wi-Fi module will disable the UART and go straight into SPI communication mode. Once configuration is completed, a modem status frame is queued by the module to the SPI port which will cause the SPI_nATTN line to assert. The host can use this to determine that the SPI port has been configured properly. This method internally forces the configuration to provide full SPI support for the following parameters:

- D1

Note This parameter will only be changed if it is at a default of zero when method is invoked.

- D2
- D3
- D4
- P2

As long as a WR command is not issued, these configuration values will revert back to previous values after a power on reset. If a WR command is issued while in SPI mode, these same parameters will be written to flash. After a reset, parameters that were forced and then written to flash become the mode of operation. If the UART is disabled and the SPI is enabled in the written configuration, then the module will come up in SPI mode without forcing it by holding DOUT low. If both the UART and the SPI are enabled at the time of reset, then output will go to the UART until the host sends the first input. If that first input comes on the SPI port, then all subsequent output will go to the SPI port and the UART will be disabled. If the first input comes on the UART, then all subsequent output will go to the UART and the SPI will be disabled. Note that once a serial port (UART or SPI) has been selected, all subsequent output will go to that port, even if a new configuration is applied. The only way to switch the selected serial port is to reset the module.

On SMT modules, forcing DOUT low at the time of reset has no effect. To use SPI mode on the SMT modules, assert the SPI_nSSEL pin (15) low after reset and before any UART data is input.

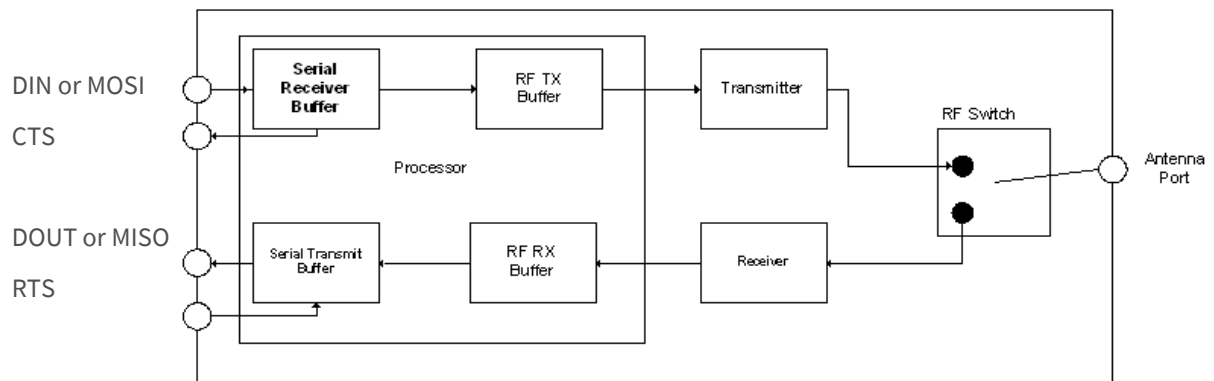
When the slave select (SPI_nSSEL) signal is asserted by the master, SPI transmit data is driven to the output pin SPI_MISO, and SPI data is received from the input pin SPI_MOSI. The SPI_nSSEL pin has to be asserted to enable the transmit serializer to drive data to the output signal SPI_MISO. A rising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

If the output buffer is empty, the SPI serializer transmits the last valid bit repeatedly, which may be either high or low. Otherwise, the module formats all output in API mode 1 format. For details, see [API operation](#) on page 80. The attached host is expected to ignore all data that is not part of a formatted API frame.

Serial buffers

The XBee modules maintain buffers to collect received serial and RF data, which is illustrated in the figure below. The serial receive buffer collects incoming serial characters and holds them until they can be processed. The serial transmit buffer collects data that is received via the RF link that will be transmitted out the UART or SPI port.

Internal data flow diagram



Serial receive buffer

When serial data enters the RF module through the DIN Pin (or the MOSI pin), the data is stored in the serial receive buffer until it can be processed. Under certain conditions, the module may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the module such that the serial receive buffer would overflow, then the new data will be discarded. If the UART is in use, this can be avoided by the host side honoring CTS flow control.

Serial transmit buffer

When RF data is received, the data is moved into the serial transmit buffer and sent out the UART or SPI port. If the serial transmit buffer becomes full and system buffers are also full, then the entire RF data packet is dropped. Whenever data is received faster than it can be processed and transmitted out the serial port, there is a potential of dropping data, even in TCP mode.

UART flow control

The nRTS and nCTS module pins can be used to provide RTS and/or CTS flow control. CTS flow control provides an indication to the host to stop sending serial data to the module. RTS flow control allows the host to signal the module to not send data in the serial transmit buffer out the UART. RTS and CTS flow control are enabled using the D6 and D7 commands.

nCTS flow control

The FT command allows the user to specify how many bytes of data can be queued up in the serial transmit buffer before the module asserts CTS low. The serial receive buffer can hold up to 2100 bytes, but FT cannot be set any larger than 2083 bytes, leaving 17 bytes that can be sent by the host before the data is dropped.

By default, FT is 2035 (0x7F3), which allows the host to send 65 bytes to the module after the module asserts CTS before the data is dropped.

In either case, CTS will not be re-asserted until the serial receive buffer has FT-17 or less bytes in use.

nRTS Flow Control

If RTS flow control is enabled (D6 command), data in the serial transmit buffer will not be sent out the DOUT pin as long as nRTS is de-asserted (set high). The host device should not de-assert nRTS for long periods of time to avoid filling the serial transmit buffer. If an RF data packet is received, and the serial transmit buffer does not have enough space for all of the data bytes, the entire RF data packet will be discarded.

Note If RTS flow control is enabled and the XBee is sending data out the UART when nRTS is de-asserted (set high), the XBee could send up to 4 characters out the UART to clear its FIFO after nRTS is de-asserted. This implies that the user needs to de-assert nRTS by the time its receive capacity is within 4 bytes of full.

Serial interface protocols

The XBee modules support both transparent and API (application programming interface) serial interfaces.

Transparent operation

When operating in transparent mode, the modules act as a serial line replacement. All UART data received is queued up for RF transmission. When RF data is received, the data is sent out through the UART. The module configuration parameters are configured using the AT command mode interface. Note that transparent operation is not an option when using SPI.

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

- No serial characters are received for the amount of time determined by the RO parameter. If RO is zero, data is packetized as soon as it is received, without delay. If RO is non-zero, the data is packetized after RO character times of no transitions on the DIN pin. However, if the time required for RO characters is less than 100 microseconds, then DIN must still be idle for at least 100 microseconds, which is the minimal idle time required for packetizing packets at any baud rate.
- The Command Mode Sequence (GT + CC + GT) is received. Any character buffered in the serial receive buffer before the sequence is packetized and transmitted before command mode is entered.
- The maximum number of characters that will fit in an RF packet is received.

API operation

API operation is an alternative to transparent operation. The frame-based API extends the level to which a host application can interact with the networking capabilities of the module. When in API mode, all data entering and leaving the UART or SPI is contained in frames that define operations or events within the module.

Transmit Data Frames (received through the DIN or SPI_MOSI pin) include:

- RF Transmit Data Frame
- Local commands (equivalent to AT commands)
- Remote commands to be sent to another radio

Receive Data Frames (sent out the DOUT or SPI_MISO pin) include:

- RF-received data frames
- Local command responses
- Remote command responses
- I/O samples from a remote radio
- Event notifications such as transmission status, reset, associate, disassociate, etc.

The API provides an alternative means of configuring modules and of routing data at the local host application layer. A local host application can send data frames to the module that contain address and payload information instead of using command mode to modify addresses. The module will send data frames to the application containing status packets; as well as source, and payload information from received data packets. The API operation option facilitates many operations such as the examples cited below:

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

A comparison of transparent and API operation

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

Transparent operation features	
Simple interface	All received serial data is transmitted unless the module is in command mode.
Easy to support	It is easier for an application to support transparent operation and command mode.
API operation features	
Easy to manage data transmissions to multiple destinations	Transmitting RF data to multiple remotes only requires changing the address in the API frame. This Process is much faster than transparent operation where the application must enter AT command mode, change the address, exit command mode, and then transmit data. Each API transmission can return a transmit status frame indicating the success or reason for failure
Received data frames indicate the sender's address	All received RF data API frames indicate the source address.
Advanced Networking diagnostics	API frames can provide indication of IO samples from remote modules, transmission status messages, and local radio status messages.
Remote Configuration	Set/read configuration commands can be sent to remote modules to configure them as needed using the API.

As a general rule of thumb, API firmware is recommended when a module:

- Sends RF data to multiple destinations
- Sends remote configuration commands to manage modules in the network
- Receives IO samples from remote modules
- Receives RF data packets from multiple modules, and the application needs to know which module sent which packet
- Needs to use the send data request and device request features of Device Cloud

If the above conditions do not apply, (e.g. in a sensor node, or a simple application) then transparent operation might be suitable. It is acceptable to use a mixture of modules running API mode and transparent mode in a network.

Modes of operation

Idle mode

When not receiving or transmitting data, the RF module is in Idle Mode. The module shifts into the other modes of operation under the following conditions:

- Transmit Mode (serial data in the serial receive buffer is ready to be packetized)
- Receive Mode (valid RF data is received through the antenna)
- Sleep Mode
- Command Mode (Command Mode Sequence is issued)

Transmit mode

When serial data is received and is ready to be packetized, the RF module will exit Idle Mode and attempt to transmit the data. The destination address determines which node(s) will receive the data. Receive Mode

If a valid RF packet is received, the data is transferred to the serial transmit buffer.

Command mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming serial characters are interpreted as commands. See [API operation](#) on page 80 for an alternate means of configuring modules, which is the only method available for SPI mode. (Command Mode is unavailable when using the SPI interface.)

AT command mode

To enter AT Command mode, send the 3-character command sequence “+++” and observe guard times before and after the command characters. [Refer to the “Default AT Command Mode Sequence” below.]

Default AT Command Mode Sequence (for transition to Command Mode):

- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]
- Input three plus characters (“+++”) within one second [CC (Command Sequence Character) parameter = 0x2B.]
- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]

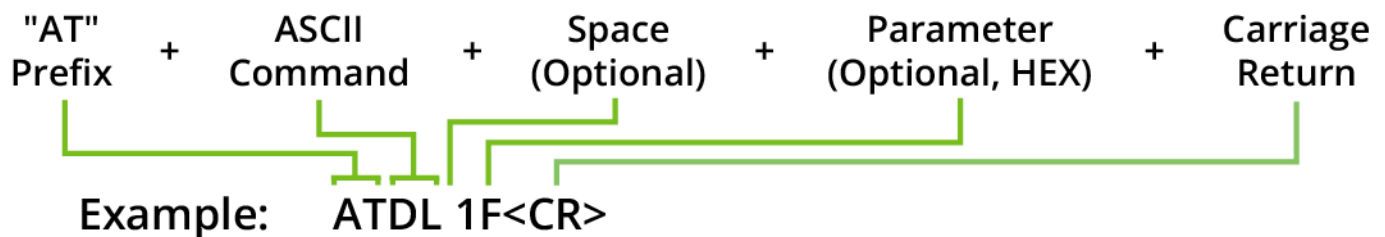
Once the AT command mode sequence has been issued, the module sends an "OK\r" out the UART. The "OK\r" characters can be delayed if the module has not finished transmitting received serial data.

When command mode has been entered, the command mode timer is started (CT command), and the module is able to receive AT commands on the UART.

All of the parameter values in the sequence can be modified to reflect user preferences.

Note Failure to enter AT Command Mode is most commonly due to baud rate mismatch. By default, the BD (Baud Rate) parameter = 3 (9600 b/s).

Send AT commands and parameters using the syntax shown below:



To read a parameter value stored in the RF module’s register, omit the parameter field.

The preceding example would change the RF module baud rate to 7, which would allow operation at 115,200b/s. To store the new value to non-volatile (long term) memory, subsequently send the WR (Write) command.

For modified parameter values to persist in the module’s registry after a reset, changes must be saved to non-volatile memory using the WR (Write) Command. Otherwise, parameters are restored to previously saved values after the module is reset.

Multiple AT Commands can be sent at a time when separated by a comma in Command Mode (e.g. ATSH,SL).

Command response

When a command is sent to the module, the module will parse and execute the command. Upon successful execution of a command, the module returns an “OK” message. If execution of a command results in an error, the module returns an “ERROR” message.

Applying command changes

Any changes made to the configuration command registers through AT commands will not take effect until the changes are applied. For example, sending the BD command to change the baud rate will not change the actual baud rate until changes are applied. Changes can be applied in one of the following ways:

- The AC (Apply Changes) command is issued.

- AT command mode is exited.

To exit AT command mode, do one of the following:

- Send the ATCN (Exit Command Mode) command (followed by a carriage return).
- If no valid AT Commands are received within the time specified by CT (Command Mode Timeout) Command, the RF module automatically returns to Idle Mode.

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, see [Command reference tables](#) on page 109.

Configuration mode

The user may not always know the parameters with which the XBee module is configured. If those parameters affect the means by which command mode is entered (and the parameters were previously written to non-volatile memory), then command mode is not available to either read the parameters or to set them to known values. This makes configuration of the XBee difficult unless the user can successfully guess the configuration to allow entry into command mode. A common example of this problem is when the UART baud rate is unknown. In this case, the “+++” sequence to enter command mode would not be recognized due to a baud rate mismatch, preventing entry into command mode.

Forcing entry into configuration mode

To overcome this issue, the XBee may be forced into command mode with a known configuration as follows: While holding DIN low (a.k.a. asserting the break key), reset the module. Rather than coming up in transparent mode, which is normal, it will come up in command mode and issue the OK prompt with the following default parameters applied for operation while in command mode:

- UART enabled (P3=1, P4=1)—only set for SPI-enabled modules.
- 9600 baud rate (BD=3)
- One stop bit (SB=0)
- No parity (NB=0)
- Three character times with no change on DIN before transmission (RO=3)
- No RTS flow control (D6=0)
- CTS flow control (D7=1)
- 65 characters left in transmission buffer before CTS is turned off (FT)
- ‘+’ is used for command mode character (CC=0x2b)
- One second guard time (GT=0x3e8)
- Ten second command mode timeout (CT=0x64).

If the configuration mode is left without setting any parameters (i.e. without changing parameter values), then all parameters will revert to their previous unknown state after exiting command mode. Also, any values queried will return the previously written settings rather than the temporarily applied default settings described above.



When the need arises to recover from an unknown configuration to a known configuration, the user should do the following:

- 1 Set up the interface to the XBee to match the default configuration as described above.
- 2 Press and hold DIN low while resetting the XBee module.
- 3 Release DIN (let it be pulled high) so that UART data may be received.

- 4 At the OK prompt, enter the desired configuration settings. (If desired, configuration settings which were unknown may be read before setting them in this state.)
- 5 Write the desired configuration to non-volatile memory using the WR command.
- 6 Set up the interface to the XBee to match the configuration just written to non-volatile memory.
- 7 Optionally, reset the module and then begin operation in the new mode.

Use XCTU to enter Configuration mode

XCTU is designed to support a forced configuration on a UART interface using the following. XCTU does not work over an SPI interface directly.

- 1 Connect an asynchronous serial port of the PC (either RS-232 or USB) to the development board into which the XBee device is plugged.
- 2 Start XCTU.
- 3 If you know the serial configuration of the XBee device, use the following steps to add it manually:
 - a. Click the **Add a radio module** icon in the toolbar  .
 - b. Select the serial port to which the radio module is connected (or enter its name manually) and configure the serial settings to match the default configuration. This radio module is not programmable, so do not select the checkbox next to The radio module is programmable.
 - c. Click **Finish**. If the settings were correctly configured, it should now be displayed in the list. If not, an action required dialog asks you to reset the module. After resetting, XCTU should add the module.
- 4 If you do not know the serial configuration of the XBee module, the port it is connected to, or you want to add more than one module, you can use the discovery method to add the radio module(s) with the following steps:
 - a. Click the **Discover radio modules** icon at the top left in the toolbar  .
 - b. Select the serial ports that you want to use to find the radio modules, then click **Next**.
 - c. Check all the settings you want to try in the discovery process to discover radio modules. The estimated discovery time is listed below the settings. The more settings you add to the discovery process, the longer the discovery process.
 - d. Click **Finish** to start the discovery process. As XCTU finds modules, they appear in the discovery process dialog box. To stop the process at any time, click **Stop**. For example, you can stop the process if the module(s) you were looking for are already found.
 - e. Check the radio modules to add to the list and click the **Add selected devices** button.
- 5 The radio module(s) appear under the **Radio Modules** section on the left of the display.
- 6 To configure the settings, click the radio module that you want to configure on the left pane. The **Radio Configuration** section on the right updates.
- 7 Enter the desired configuration under the **Radio Configuration** section. All parameters that you change but do not write are highlighted green on the right side. All written parameters that are not the default values are highlighted blue. If you set an invalid value, the parameter is highlighted red along with a description explaining why it is invalid. All default values are not highlighted.
- 8 When you are done entering the parameters, click the **Write** button at the top of the **Radio Configuration** section.
- 9 When the write is completed, all of the settings on the radio module are updated.

10 Click the **Terminal** tab on the upper right of the toolbar and begin normal transparent operation.

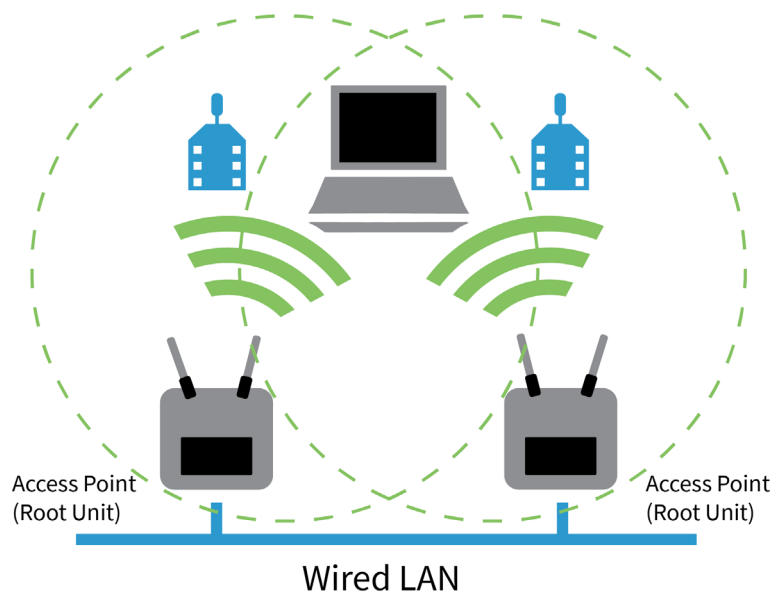
Sleep mode

Sleep modes allow the RF module to enter states of low power consumption when not in use. The XBee Wi-Fi modules support both pin sleep (sleep mode entered on pin transition) and cyclic sleep (module sleeps for a fixed time). For both pin sleep and cyclic sleep the sleep level may be either deep sleep or associated sleep. See [Sleep](#) on page 51 for more information.

802.11 bgn networks

Infrastructure networks

The main type of wireless network will involve a number of wireless devices (called stations) talking through a master wireless device known as an **Access Point** (AP for short, or STA for short). This type of setup is called an **Infrastructure or BSS** (Basic Service Set) network. Most wireless networks are of this type. An example of an infrastructure wireless network is shown below:



Infrastructure wireless Network. By default, the module operates as a STA in the infrastructure network, which means it associates to an AP and all data to and from the module goes through that AP.

If CE is configured to 1, the module takes the position of an AP in the network, allowing STA device to associate to the module operating in what is called Soft AP mode.

Ad hoc networks

Wireless devices can get on a wireless network without an access point. This is called an **Ad Hoc or IBSS** (Independent Basic Service Set) network.

Note Ad hoc networks are point to point: **there can only be two nodes in the network**, a creator and a joiner. Set up the creator first, and then the joiner.

Ad hoc creator

Set up the following parameters for the creator:

- AH1 designates the node as an Ad hoc creator.
- MA1 specifies static IP addresses. (No DHCP is supported in Ad Hoc mode.)
- EE0 specifies no security. (Security is not available in Ad Hoc mode.)
- CH may be any channel from 1 to 0xB.
- ID sets the SSID, which is any string of choice, as long as it isn't the same as another SSID in the vicinity.
- MY sets IP address of creator node.
- DL specifies IP address of joiner node.
- MK sets IP mask for both of the above addresses.

Ad hoc joiner

Set up the following parameters for the joiner:

- AH0 designates the node as an Ad hoc joiner.
- MA1 specifies static IP addresses. (No DHCP is supported in Ad Hoc mode.)
- EE0 specifies no security. (Security is not available in Ad Hoc mode.)
- ID sets the SSID, which must match the ID of the creator. Problems arise if it matches the SSID of an access point in the vicinity.
- MY sets IP address of joiner node.
- DL specifies IP address of creator node.
- MK sets IP mask for both of the above addresses.

Network basics

Clients will need to join the wireless network before they can send data across it. This is called Association. In order for a device to associate it must know the following items about the desired wireless network:

- **SSID:** the name of the wireless network.
- **Encryption:** if and how the network encrypts or scrambles its data.
- **Authentication:** how and if the network requires its members to —prove their identity.
- **Channel:** what channel (frequency range) the wireless network uses.

Once a device is associated it can send and receive data from other associated devices on the same network. When the client is done or needs to leave, it then can Dis-associate and be removed from the wireless network.

XBee Wi-Fi standards

The XBee Wi-Fi module will operate in three of the available 802.11 standards.

802.11 b

The 802.11b standard was approved in July 1999 and can be considered the second generation. 802.11b operates in the 2.4 GHz frequency ISM band. The data rate is from 1 to 11 Mb/s.

802.11 g

The 802.11g standard was approved in 2003. It provides a maximum data rate of 54 Mb/s. In addition, the standard is also fully backwards-compatible with existing 802.11b wireless networks.

802.11 n

The 802.11n standard was approved in 2009. It provides for data rates up to 300Mb/s. The XBee Wi-Fi module uses the single stream n mode with 20MHz bandwidth and is capable of up to 72.2 Mb/s over the air in n mode.

Encryption

Encryption is a method of scrambling a message that makes it unreadable to unwanted parties, adding a degree of secure communications. There are different protocols for providing encryption, and the XBee Wi-Fi module supports WPA, WPA2, and WEP.

Authentication

Authentication deals with proving the identity of the wireless device attempting to associate with the network. There are different methods of doing this. The XBee Wi-Fi module supports Open and Shared Key authentication in WEP mode and it only supports shared key authentication in WPA and WPA2 modes.

Open

Open Authentication is when the access point simply accepts the wireless devices identity without verifying or proving it. The benefits to this is simplicity and compatibility (all devices can do it). In this mode, which is only available when using WEP, a connection to the access point occurs even if the WEP key is wrong. However, no real communication can occur because of mismatched keys. If DHCP is configured, it will fail too, causing the AI indicator to get stuck in the AI 41 state.

If, on the other hand, the AP is configured for shared key authentication, no connection will occur with an incorrect WEP key. Instead, AI will get stuck in the FF state, indicating scanning. Although shared key authentication sounds better, it exposes a big security flaw with WEP. The challenge text, its encrypted result, and a success/failure result are passed in the clear and can easily be caught over the air to determine the WEP key.

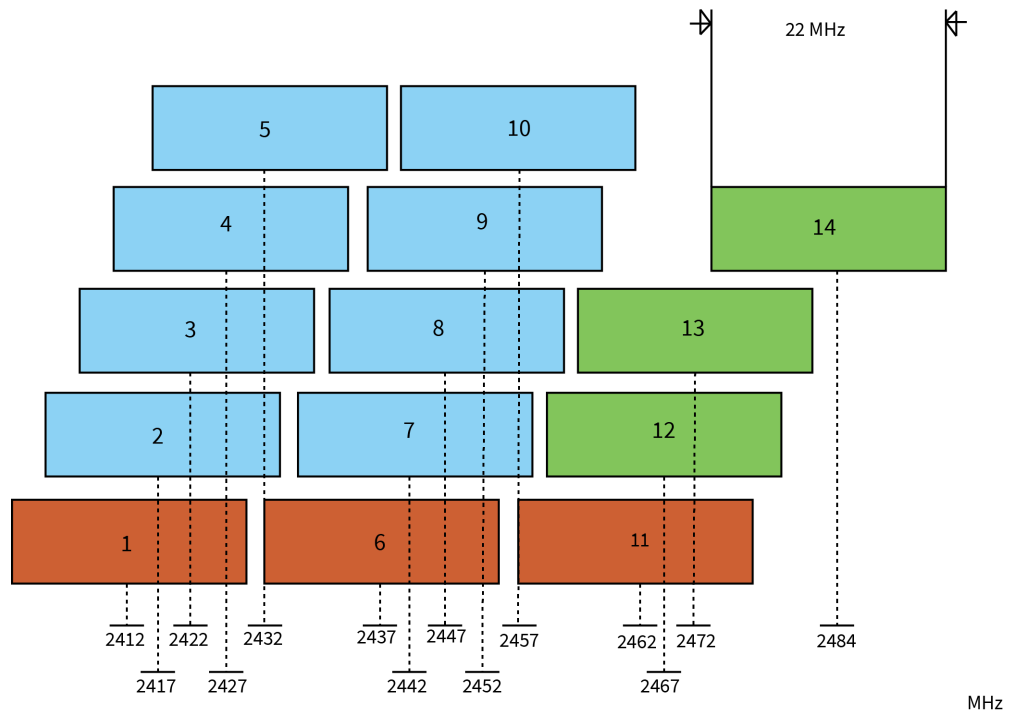
Shared key

Shared Key is when the wireless devices must present the proper key to get on the network. Although Shared Key has more security than Open Authentication it should not be considered secure. One of the benefits of Shared Key Authentication is simplicity.

Channels

The XBee Wi-Fi modules operate in the 2412-2472 MHz range. The frequency range is broken down into 13 channels. Data is transmitted on a channel by radio frequencies over a certain frequency range. In order to avoid bad performance caused by the overlapping (“collision”) of channel frequencies in a wireless LAN environment, it is very important that the channels of neighboring access points are selected accordingly.

The center frequencies of the 13 possible channels range from 2412 to 2472 MHz, with each channel being 22 MHz wide and centered in 5 MHz intervals. This means that only 3 channels (1, 6, and 11) in North America are not subject to overlapping.



XBee IP services

The XBee provides services using IP (Internet Protocol) for XBee and other clients on the network. IP services provide functionality to allow XBee configuration and direct serial port access. There are two XBee services:

- XBee Application Service
- Serial Communication Service

XBee application service

This service primarily provides for XBee configuration. It also provides API compatibility for customers who have designed around other XBees. It uses UDP to transfer packets to and from port number 0xBEE. Packets are optionally acknowledged by the service but retries are not available. An extra header is added to the packet data to define commands for configuration and serial data transfer. The following sections describe how this service can be accessed from a local host or network client. C0 and DE are used to configure source and destination ports for the serial communication service. The XBee application service uses hard coded port 0xBEE for both source and destination and there is no option to configure another port.

Note Do not configure C0 and/or DE to 0xBEE to use the XBee application service. Doing so causes an error (AI=42), and the transceiver will neither send nor receive data.

Local host

From a local host this functionality is accessed through XBee API frames. There are remote AT command frames as well as transmission frames. The API frames are listed as follows:

- TX request: 64-bit (TX64)
- RX indicator: 64-bit (RX64) (This frame is generated by the XBee module.)
- Remote AT command
- General Purpose Memory command

TX64 and RX64 API frames

The intent of the XBee transmit and receive 64-bit API frames is to provide a standardized set of API frames to use for a point to multipoint network—a closed network of XBee Wi-Fi modules. The format of these frames has been standardized to work with other XBee products, such as the API frames of the 802.15.4 module.

Note The XBee Wi-Fi module cannot communicate with an XBee 802.15.4 module.

Transmitting data

The local host uses the TX64 frame to send data to another XBee using this service. When the frame is received through the serial port the XBee converts the contents of the frame to a serial data transfer command as defined by the XBee application service.

Receiving data

A received Serial data transfer command will go to the serial port. The mode of the serial port will determine the format of the data. When in API mode the data will be sent to the host using the RX 64-bit frame.

Note It is not recommended to use this service to send data to a network client. Use the serial communication service.

Remote AT command configuration

The Remote AT command frame is used to change configuration on a remote XBee. See [ZigBee remote AT command](#) on page 91 for more information.

Firmware upgrades

To perform firmware upgrades from the local host, send ZigBee explicit API frames (type 0x11) to the IP address of the desired node with cluster ID 0x23. For details about the format of the explicit frames, see [Advanced application features](#) on page 55. For details about the sequence of operations to follow for firmware upgrades, see [Sleep](#) on page 51.

Frame type 0x11 is the "ZigBee explicit transmit packet". It has this name because it has fields that are uniquely ZigBee, such as endpoint and cluster ID. To perform firmware upgrades the same on all XBee devices, Digi chose this frame type, even though all XBee devices (including the XBee Wi-Fi RF Module) do not support the ZigBee standard.

The XBee Wi-Fi RF Module also supports other ZigBee frame types. These are frame types 0x10, 0x17, 0x8B, 0x90, 0x91, and 0x97. These frame types do not allow XBee Wi-Fi RF Modules to execute the ZigBee protocol and they do not allow OTA compatibility between the two protocols. If you want to implement these API frame types on your host program, you can continue to use them, even with the XBee Wi-Fi RF Module. If you are developing a new host program for the XBee Wi-Fi RF Module, you should not use these frame types because the WiFi frame types (0x20, and 0xB0) make more sense and have less API overhead.

Frame 0x11 is required across all protocols for firmware updates.

Network client

This port is accessed by sending a packet from the client using the UDP protocol on port 0xBEE. Data sent to this port must have an additional header preceding the data. See the following table for the header description.

Field name	Offset	Field length	Description
Number1	0	2	Can be any random number
Number2	2	2	Number1 ^ 0x4242 (Exclusive OR of Number1 and constant 0x4242)
PacketID	4	1	Reserved for later use (0 for now)
EncPad	5	1	Reserved for later use (0 for now)
Command ID	6	1	0x00 = Data 0x02 = Remote Command 0x03 = General Purpose Memory Command 0x04 = I/O Sample 0x80 = Data Acknowledgment 0x82 = Response to remote command 0x83 = Response to General Purpose Memory Command
Command options	7	1	bit 0 – encrypted if set (Reserved for later use) bit 1 – set to request an ACK bits 2:7 - unused (Set to 0 for forward compatibility.)

All of the commands and command responses detailed below are preceded with the above application header.

Sending configuration commands

AT commands can be sent to the XBee Wi-Fi module from a network client. The following packet structure demonstrates how to query the SSID from a network client:

Packet fields	Offset	Example	Description	
Application header	Number1	0	0x4242	
	Number2	2	0x0000	Number1 ^ Number2 = 0x4242
	Packet ID	4	0x00	Reserved for later use (0 for now)
	Encryption pad	5	0x00	
	Command ID	6	0x02	
	Command options	7	0x00	Options are not available for this command

Packet fields		Offset	Example	Description
Command-specific data	Frame ID	8	0x01	
	Configuration options	9	0x02	0 – Queue command parameter. Must send AC command or use apply changes option to apply changes. 2 – Apply changes to all changed commands
	AT command	10	0x49 (I)	Command Name - Two ASCII characters that identify the AT command
Command-specific data		11	0x44(D)	
	Parameter value	12		If present, indicates the requested parameter value to set the given command. If no characters present, command is queried.

The response is sent back to the host with the following bytes.

Packet fields		Offset	Example	Description
Application header	Number1	0	0x4242	
	Number2	2	0x0000	Number1 ^ Number2 = 0x4242
	Packet ID	4	0x00	Reserved for later use (0 for now)
	Encryption pad	5	0x00	
	Command ID	6	0x82	Indicates remote AT command response
	Command options	7	0x00	Options not available for this response

Packet fields		Offset	Example	Description
Command-specific data	Frame ID	8	0x01	Copied from the command
	AT Command	9	0x49 (I)	Command Name - Two ASCII characters that identify the AT command
		10	0x44(D)	
	Status	11	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
	Parameter Value	12	0x41 'A'	Data in binary or ASCII format, based on the command. For the ID command, the data is in ASCII format. If the command was set, then this field is not returned.
		13	0x63 'c'	
		14	0x63 'c'	
		15	0x65 'e'	
		16	0x73 's'	
		17	0x73 's'	
		18	0x50 'p'	
		19	0x6F 'o'	
		20	0x69 'i'	
21		0x6E 'n'		
22	0x74 't'			

Sending the serial data command to XBee

Using this service to send data out the serial port is not required. Most users choose to use the Serial Communication Service (see below) for sending data from a network client. One reason to use the XBee Application Service to send the serial data command from a network client is to receive an acknowledgment when sending a UDP packet.

The client can request an acknowledgment from the XBee but must wait to receive the acknowledgment before sending the next packet. The client is responsible for retransmissions due to missed acknowledgments. When resending packets, duplicates can be received at the destination due to a successful serial data command and a failed acknowledgment packet. The host in this case must be able to handle duplicate packets. The following packet structures are examples of sending data and receiving an acknowledgment using the XBee application service:

Serial data command

Packet fields		Offset	Example	Description
Application header	Number1	0	0x4242	
	Number2	2	0x0000	Number1 ^ Number2 = 0x4242
	Packet ID	4	0x00	Reserved for later use (0 for now)
	Encryption pad	5	0x00	
	Command ID	6	0x00	Indicates transmission data
	Command options	7	0x02	Request acknowledgment
Command-specific data	Serial data	8	0x48 'H'	Can be up to 1492 bytes. Data will be sent out the XBee's serial port.
		9	0x65 'e'	
		10	0x6C 'l'	
		11	0x6C 'l'	
		12	0x6F 'o'	

Serial data command acknowledgment - if requested

Packet fields		Offset	Example	Description
Application Header	Number1	0	0x4242	
	Number2	2	0x0000	Number1 ^ Number2 = 0x4242
	Packet ID	4	0x00	Reserved for later use (0 for now)
	Encryption Pad	5	0x00	
	Command ID	6	0x80	Indicates data acknowledgment
	Command Options	7	0x0	Options not available for this response
Command Specific Data	Serial Data	8		No command specific data

Receiving I/O sampled data

Sample data generated by the module will be sent to the address configured by the DL commands. This data can be sent to another XBee or to a network client. It will be sent using UDP from the 0xBEE port as with other XBee Application services. Sample data will be received by the client as follows:

Frame fields		Offset	Example	Description
Application header	Number1	0	0x4242	
	Number2	2	0x0000	Number1 ^ Number2 = 0x4242
	Packet ID	4	0x00	Reserved for later use (0 for now)
	Encryption pad	5	0x00	
	Command ID	6	0x04	Indicates I/O sample data
	Command options	7	0x00	Options not available for this response
Command-specific data	Number of Samples	8	0x01	Indicates one sample set
	Digital Mask	MSB 9	0x01	Bit Mask. Each bit represents an enabled DIO line starting with DIO0 at bit 0.
		LSB 10	0x01	
Analog Mask	11	0x02	Bit Mask. Each bit represents an enabled ADC starting with ADC0 at bit 0. This selects ADC1 for analog sampling.	

Frame fields		Offset	Example	Description
Command-specific data	Digital Sample	MSB 12	0x00	This field is only present if at least one DIO line is enabled in the digital mask specified above. Each bit represents a DIO line. Start with bit 0 for DIO0.
		LSB 13	0x01	
	Analog Sample	MSB 14	0x02	0x200 indicates that reading is half of VREF. For a default VREF of 2.5V, 0x200 represents 1.25 volts on ADC1 in this example.
		LSB 15	0x00	

Sending over-the-air firmware upgrades

A network client can also use the XBee IP services to send a firmware upgrade to the module. This is done by sending a frame formatted with an application header, followed by a GPM header, followed by GPM data. The format of the application header is given above. See [Sleep](#) on page 51 for details about GPM headers format options. Make sure each GPM header is preceded by an application header. The following table shows an example of the final step of a firmware upgrade process.

Serial communication service

Packet fields		Offset	Example	Description
Application header	Number1	0	0x4242	This is an easy number to create an accepted frame.
	Number2	2	0x0000	$\text{Number1} \wedge \text{Number2} = 0x4242$ (This is an easy way to send a frame that software will not reject.)
	Packet ID	4	0x00	Reserved for later use (0 for now)
	EncPad	5	0x00	
	Command ID	6	0x03	General Purpose Memory Command
	Command Options	7	0x00	Don't request an acknowledgment
Command-specific data	GPM_CMD_ID	8	0x06	Firmware verify and install command
	GPM_OPTIONS	9	0x00	Reserved for later use (0 for now)
	GPM_BLOCK_NUM	10	0x00	
	GPM_START_INDEX	12	0x00	
	GPM_NUM_BYTES	14	0x0000	
	GPM_DATA	16		This field is unused for this command

The serial communication service connects an IP port to the serial peripheral (UART or SPI) of the XBee. No additional formatting or header is required and data will be transferred between the RF hardware and Serial Communication hardware as received. The IP ports are configured using the C0 and DE commands. Note that port 0xBEE is reserved for the XBee Application Service and should not be used for the Serial Communication Service. The behavior of this service varies based on the mode of the serial port and is discussed in the following sections.

Transparent mode

In transparent mode, only one port is available, and that port may be either UDP or TCP, depending on the configuration specified in the IP command. Data received on the serial port is packetized and sent to the RF port and data received on the RF port is sent to the serial port without any formatting of the data. For details about how data is packetized, see [Transparent operation](#) on page 30.

UDP

When the IP command is configured for UDP, serial data is sent to the IP address specified by DL and it is sent to the UDP port specified by DE. The source of the packet is defined by the C0 command. No connection is established

TCP

When the IP command is configured for TCP, only one connection is allowed at a time. If a transmission is attempted while a TCP connection exists, the data will be sent on that connection, ignoring the DL and DE parameters. This connection can be initiated by a local host or by a network client.

A local host initiates a connection by sending data to the serial port. A connection will be created based on the DL (IP address) and DE (destination port) commands. However, if DL is a broadcast address, then UDP will be used, ignoring the TCP configuration.

A network client establishing a TCP connection to the XBee will use the port defined by the C0 command. When established any data sent by the local host will not create a new connection based on DL and DE, but rather the existing connection will be utilized.

API mode

API mode allows specification of protocol, (UDP or TCP), destination address and port, and source port for transmission.

UDP mode

If UDP mode is specified in the Transmit IPv4 frame, no connection is made to the destination address and port. Instead, the data is packetized and sent directly, providing the source port matches the local port specified by the C0 command.

TCP mode

In API mode, multiple TCP connections are allowed simultaneously. A TCP connection is fully defined by these four entities:

- Local IP address
- Local port number
- Remote IP address
- Remote port number

When an IPv4 Transmission frame is sent to the module, it specifies a destination address and port. To send data on an existing TCP connection, the destination address and port given in the API frame must match the remote address and port of an existing TCP connection. Note that the search for a matching connection ignores the source port number given in the API frame. This means that only one TCP connection is allowed per remote port.

The source port matters in the event that a matching TCP connection is not found. If it is 0, then an attempt is made to create a new connection prior to sending the data. If not, the data is dropped with an error.

For purposes of the following discussion, a client requests a TCP connection of a server and a server accepts a TCP connection request from a client.

As a client, the best strategy is to specify a source port of 0 and a destination port to match the listening socket (C0) of the receiving module or the server port for any other network device. This way, if a connection is not found, a new one is created.

As a server, the best strategy is to swap the source and destination ports found in the IPv4 receive packet and place them in the response, which is an IPv4 transmit packet. This allows the response to be sent back on the same socket as the received data. If, on the other hand the data is sent to the listening socket of the other module rather than to the source socket given in the IPv4 receive packet, then an extra socket is created. While this still works, it unnecessarily creates an extra socket connection.

Sleep

The XBee Wi-Fi module supports two different sleep modes.

- Pin Sleep
- Cyclic Sleep

In addition the sleep mode current draw can be modified with the following sleep options.

- AP Associated Sleep
- Deep Sleep

Pin sleep allows an external microcontroller to determine when the XBee should sleep and when it should wake by using either the SleepRq pin (default) or the SPI_nSSEL pin. In contrast, cyclic sleep allows the sleep period and wake times to be configured through the use of AT commands. The module can stay associated to the access point or can enter a deeper sleep and associate to the access point for each sleep/wake occurrence. The sleep mode is configurable with the SM and SO commands.

Besides the four sleep modes mentioned above, each of them operate a little differently based on the serial interface (UART or SPI).

Using sleep mode: UART

When the serial interface is UART, the On/nSleep pin is used to indicate that the module is entering sleep mode, unless it is configured for a different usage. (See [Command reference tables](#) on page 109). If D9 is configured for On/nSleep, then it is driven low when asleep and high when awake, whether using pin sleep or cyclic sleep.

If CTS hardware flow control is enabled (D7 command), the CTS pin is de-asserted (high) when entering sleep to indicate that serial data should not be sent to the module. The module will not respond to serial or RF data when it is sleeping. Applications that utilize the UART are encouraged to observe CTS flow control in any of the sleep modes. When the XBee wakes from sleep with flow control enabled, the CTS pin is asserted (low).

If using pin sleep, D8 must be configured for SleepRq (See [Command reference tables](#) on page 109) to put the module to sleep. Otherwise, there is no sleep at all, meaning the module will always stay awake in full power mode. When D8 is configured for SleepRq, the host should drive SLEEP_RQ high to put the module to sleep, and the host should drive SLEEP_RQ low to wake up the module.

Using sleep mode: SPI

When the serial interface is SPI, SPI_nATTN is used as an attention indicator to tell the SPI master when it has data to send. Since SPI only operates in API mode, it will assert SPI_nATTN and send out a modem status indicator after initialization. The host can use this to know when the radio is ready to operate as a SPI slave. Since the function of SPI_nATTN is to indicate when the XBee has data to send to the host, it may legitimately be driven high or low while the module is awake.

When using the SPI, either SleepRq or SPI_nSSEL may be used for pin sleep. If D8 is configured as a peripheral (1), then it will be used for pin sleep. If not, and SPI_nSSEL is configured as a peripheral (which it must be to enable SPI operation), then SPI_nSSEL is used for pin sleep.

Using SPI_nSSEL for pin sleep has the advantage of requiring one less physical pin connection to implement pin sleep on SPI. It has the disadvantage of putting the radio to sleep whenever the SPI master negates SPI_nSSEL, even if that wasn't the intent. Therefore, if the user can control SPI_nSSEL, whether or not data needs to be transmitted, then sharing the pin may be a good option. It makes the SleepRq pin available for another purpose, or it simply requires one less pin to the SPI interface.

Sleep options

AP associated sleep

This option allows the module to sync up with beacons sent from the AP which contains the DTIM (Delivery Traffic Indication Message). The DTIM indicates when broadcast and multicast data will be sent on the network. This property is configured on the AP and is typically configured as the number of beacons between each beacon with DTIM. The current draw in associated sleep mode varies significantly. When the module is awake it draws approximately 100 mA. When it is asleep, it draws approximately 2 mA. Total current draw increases when the DTIM rate is higher and it decreases when the DTIM rate is lower on the access point.

The sleep modes are described as follows with this option enabled.

Pin sleep mode

UART data can be received in pin sleep mode, whether or not the host asserts the SleepRq pin. For example, if RF data is received by the module while SleepRq is asserted, the module will wake up long enough to send the data out the UART and then immediately resume sleeping. Note that if wake host is configured, the module will assert the appropriate I/O lines (indicating that it is awake), then wait for wake host timer to expire, then output the data, and then immediately resume sleeping.

In this mode, when SleepRq is asserted the module will power down the Wi-Fi circuitry. When SleepRq is de-asserted, the Wi-Fi circuitry is powered up. This causes the module to associate to the access point for each wake event. If the module was associated when it went to sleep, it should be ready to transmit data as soon as the On/Sleep pin indicates that the module is awake. If the module was not associated when it went to sleep, the host must wait until the module is associated before a transmission can occur. (In API mode, a modem status frame will be received when the module becomes associated. Outside of API mode, the AI command must be used to determine when the module is associated.)

SPI operation is similar except that the radio asserts nATTN when data becomes available and then the local host is expected to assert SPI_nSSEL and to provide a clock until the data available is sent out.

When the local UART host needs to send data it de-asserts SleepRq. Once the appropriate status I/O lines are asserted (CTS and/or On/nSleep) the module is ready to accept data. However data will be queued and not sent until the next DTIM.

When the local SPI host needs to send data it asserts SPI_nSSEL. If SPI_nSSEL is being used for pin sleep, asserting SPI_nSSEL is enough to awaken the module to receive the incoming data. But, if SleepRq is being used to control sleep, then SPI_nSSEL must be asserted and SleepRq must be de-asserted to awaken the module to receive the data. This wakes up the module, which then accepts the incoming data; however, data is queued and not sent until the next DTIM.

Cyclic sleep mode

The module remains associated to the AP and sleeps based on the SP parameter. After SP expires, the module awakens for 30 milliseconds to check for data from the AP and to allow the host to send data or commands. This time is factored in as part of the overall ST time. When data is received or sent within 30 ms, the module remains awake for ST time and any further activity does not restart this time. When no data is received or sent within 30 ms, the module resumes sleep immediately, without waiting for ST time-out.

Deep sleep (non-associated sleep)

This option allows the Wi-Fi circuitry to be powered down resulting in the lowest sleep current (about 6 μ A) but at the expense of losing packets received during the time the module is asleep. This is because the access point will behave like the module is in full power mode while it is asleep and it will not hold back packets until the module wakes up.

Pin sleep mode

In this mode when SleepRq is asserted the module will power down the Wi-Fi circuitry. When SleepRq is de-asserted the Wi-Fi circuitry is powered up. This causes the module to associate to the access point for each wake event. If the module was associated when it went to sleep, it should be ready to transmit data as soon as the On/Sleep pin indicates that the module is awake. If the module was not associated when it went to sleep, the host must wait until the module is associated before a transmission can occur. (In API mode, a modem status frame will be received when the module becomes associated. Outside of API mode, the AI command must be used to determine when the module is associated.)

Cyclic sleep mode

In this mode the module will enter and exit sleep based on the SP, ST, and SA commands. SP specifies the sleep time and ST specifies the wake time of the module after it is associated. SA specifies the maximum time to wait for association before starting the ST timer. If SA expires before the association process completes, then the module will sleep anyway. When it awakens from this state, then it will start the SA timer again to seek to establish association.

Under normal conditions, SA is used for a time out for the first association following reset and ST is used for short wake cycles thereafter. To conserve battery power, SA should be long enough for association and ST should be as short as possible for the application.

Sampling data using sleep modes

Data can be sampled when waking from any sleep mode by enabling an ADC or digital input and setting IR appropriately with respect to ST to obtain the desired number of samples.

Sample rate (ATIR)

If multiple samples are wanted during the wake period then IR can be used. This provides ST/IR+1 samples. Each sample is sent separately.

Wake host

Wake host parameter (ATWH) delays UART and sample data from being initiated until the timer has expired. This allows the host to wake up before receiving data or a sensor to power up before an I/O sample is taken.

Digital outputs and special function outputs such as ON_SLEEP and CTS are not affected by WH. This is to allow these signals to be used to wake up devices.

Note that for deep sleep, both WH must be expired and the module must be associated before I/O samples are taken.

Advanced application features

XBee analog and digital I/O lines

XBee Wi-Fi firmware supports a number of analog and digital I/O pins that are configured through software commands. Analog and digital I/O lines can be set or queried. The following tables list the configurable I/O pins and the corresponding configuration commands.

Through-hole module				
Pin name(s)	Module pin	AT cmd	Command range	Default value
DIO1/AD1/ SPI_nATTN	19	D1	0-5	0
DIO2/AD2/SPI_CLK	18	D2	0-5	0
DIO3/AD3/SPI_nSSEL	17	D3	0-5	0
DIO4/SPI_MOSI	11	D4	0-1,3-5	0
DIO5/ASSOCIATE	15	D5	0-1,3-5	1
DIO6/nRTS	16	D6	0-1,3-5	0
DIO7/nCTS	12	D7	0-1,3-7	1
DIO8/nDTR/SLEEP_RQ	9	D8	0-1,3-5	1
DIO9/On_nSLEEP	13	D9	0-1,3-5	1
DIO10/RSSI PWM/PWM0	6	P0	0-5	1
DIO11/PWM1	7	P1	0,2-5	0
DIO12/SPI_MISO	4	P2	0-1,3-5	0
DIO13/DOUT	2	P3	0-1	1
DIO14/DIN/nCONFIG	3	P4	0-1	1

SMT module				
DIO1/AD1	32	D1	0,2-5	0
DIO2/AD2	31	D2	0,2-5	0
DIO3/AD3	30	D3	0,2-5	0
DIO4	24	D4	0,3-5	0
DIO5/ASSOCIATE	28	D5	0-1,3-5	1
DIO6/nRTS	29	D6	0-1,3-5	0
DIO7/nCTS	25	D7	0-1,3-7	1
DIO8/nDTR/SLEEP_RQ	10	D8	0-1,3-5	1
DIO9/On_nSLEEP	26	D9	0-1,3-5	1
DIO10/RSSI PWM/PWM0	7	P0	0-5	1
DIO11/PWM1	8	P1	0,2-5	0
DIO12	5	P2	0,3-5	0
DIO13/DOUT	3	P3	0-1	1
DIO14/DIN/nCONFIG	4	P4	0-1	1
DIO15/SPI_MISO	17	P5	0-1,4-5	1
DIO16/SPI_MOSI	16	P6	0-1,4-5	1
DIO17/SPI_nSSEL	15	P7	0-1,4-5	1
DIO18/SPI_CLK	14	P8	0-1,4-5	1
DIO19/SPI_nATTN	12	P9	0-1,4-6	1

I/O configuration

To enable an analog or digital I/O function on one or more XBee module pin(s), the appropriate configuration command must be issued with the correct parameter. After issuing the configuration command, changes must be applied on the module for the I/O settings to take effect. Pull-up/down resistors can be set for each digital input line using the PR command. The PR value enables or disables the pull-up/down resistors, and the PD command determines if a pull-up or pull-down is used. Note that internal pull-up/down resistors are not available for digital output pins, analog input pins, or for disabled pins. See [Command reference tables](#) on page 109 for information on these commands.

Pin command parameter	Description
0	Disabled
1	Peripheral control

Pin command parameter	Description
2	Analog input or PWM output
3	Data in monitored
4	Data out default low
5	Data out default High
6	RS485 enable low
7	RS485 enable high
>7	Unsupported

I/O sampling

The XBee modules have the ability to monitor and sample the analog and digital I/O lines. I/O samples indicate the current state of I/O lines. These samples may be output on the local (serial) port, transmitted to a remote XBee module, or sent to Device Cloud.

There are three ways to obtain I/O samples, either locally or remotely:

- Queried Sampling
- Periodic Sampling
- Change Detection Sampling.

IO sample data is formatted as shown in the table below.

Bytes	Name	Description
1	Sample Sets	Number of sample sets in the packet. (Always set to 1.)
2	Digital Channel mask	Digital IO line on the module. bit 0 = DIO0 bit 1 = DIO1 bit 2 = DIO2 bit 3 = DIO3 bit 4 = DIO4 bit 5 = DIO5 bit 6 = DIO6 bit 7 = DIO7 bit 8 = DIO8 bit 9 = DIO9 bit 10 = DIO10 bit 11 = DIO11 bit 12 = DIO12 For example, a digital channel mask of 0x002F means DIO0 1, 2, 3, and 5 are enabled as digital IO.

Bytes	Name	Description
1	Analog Channel Mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. <ul style="list-style-type: none"> • bit 0 = AD0 • bit 1 = AD1 • bit 2 = AD2 • bit 3 = AD3
Variable	Sampled Data Set	If any digital I/O lines are enabled, the first two bytes of the data set indicate the state of all enabled digital IO. Only digital channels that are enabled in the Digital Channel Mask bytes have any meaning in the sample set. If no digital IO is enabled on the module, these 2 bytes will be omitted. Following the digital IO data (if any), each enabled analog channel will return 2 bytes. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3.

The sampled data set includes 2 bytes of digital I/O data only if one or more I/O lines on the module are configured as digital I/O. If no pins are configured as digital I/O, these 2 bytes are omitted.

The digital I/O data is only relevant if the same bit is enabled in the digital I/O mask.

Analog samples are 10 bit values and aligned on a 16 bit boundary. The analog reading is scaled such that 0x0000 represents 0V, and 0x3FF = VREF. VREF may be either 1.25 volts or 2.5 volts based on the setting of the AV command, where 2.5 volts is the default. The analog inputs on the module are capped at 0x3FF. Analog samples are returned in order starting with AD0 and finishing with AD3. Only enabled analog input channels return data as shown in the example below.

To convert the A/D reading to mV, do the following:

$$AD \text{ (mV)} = (A/D \text{ reading (converted to decimal)} * VREF) / 1023 \text{ where } VREF \text{ may be } 1250 \text{ or } 2500$$

Assuming that AV is set to the default value, the reading in the sample frame represents voltage inputs of 2385.14 mV (0x3D0) and 713.59 mV (0x124) for AD0 and AD1 respectively.

Queried sampling

The IS command can be sent to a module locally, or to a remote module using the API remote command frame (see [Command reference tables](#) on page 109 for details). When the IS command is sent and at least one I/O line is enabled as an input or an output, the receiving device samples all enabled digital I/O and analog input channels and returns an I/O sample. When no I/O line is enabled, IS will return error. If IS is sent locally, the I/O sample is sent out the UART or SPI port. If the IS command was received as a remote command, the I/O sample is sent over-the-air to the module that sent the IS command.

If the IS command is issued in command mode, the module returns a carriage return-delimited list containing the above-listed fields. If the IS command is issued in API mode, the module returns an API command response packet with the I/O data included in the command data portion of the response frame.

The following table shows an example of the fields in an IS response.

Example	Sample AT response
0x01	[1 sample set]
0x0C0C	[Digital Inputs: DIO 2, 3, 10, 11 selected]
0x03	[Analog Inputs; A/D 0,1]
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 the XBee module to take an I/O sample and transmit it to a remote module at a periodic rate. The periodic sample rate is set by the IR command. If IR is set to 0 or there are no active I/O lines, periodic sampling is disabled. For all other values of IR, data will be sampled after IR milliseconds have elapsed and transmitted to a remote module. However, the module cannot keep up with transmitting an I/O sample more often than every three milliseconds. Therefore, when IR is set to 1 or 2, many samples are lost.

When Device Cloud is enabled (see [Networking Commands](#) on page 112), samples are sent as a data stream. See [Transparent mode data](#) on page 77 to learn how to view the data streams. When DO bits 0 and 3 are both set (0x09), I/O samples are sent to the Device Cloud and to DL.

When Device Cloud is not enabled, the I/O sample is sent to the address specified by the DL command. When DL points to another XBee module, that module must have API mode enabled. Otherwise, the data will be dropped by the receiving module and not sent out the serial port.

When DL points to a network client, the I/O sample is sent to that network client. See [XBee IP services](#) on page 41 for the format of I/O samples sent to a network client.

IR can be used with sleep. A module will transmit periodic I/O samples at the IR rate until the module resumes sleeping. Even if the IR rate is set longer than the ST defined wake time, at least one I/O sample will still be sent before the module returns to sleep because it sends one immediately upon wake up. If it is not desired that a sample is sent every wake cycle, the IF command can be used to configure how many wake cycles should elapse before sending I/O samples at the IR rate.

Change detection sampling

Modules can be configured to transmit a data sample immediately whenever a monitored digital I/O pin changes state. (Change detect sampling cannot be triggered by an enabled analog input.) The IC command is a bitmask that can be used to set which digital I/O lines should be monitored for a state change. If one or more bits in IC is set, an I/O sample will be transmitted as soon as a state change is observed in one of the monitored digital IO lines. Change detection samples are transmitted to the IPv4 address specified by DL, to Device Cloud, or to both, depending on the setting of the DO command. Viewing I/O samples on the remote XBee or Device Cloud is the same for change detect sampling as it is for periodic sampling.

Example: Configure the following I/O settings on the XBee

Configure DIO1/AD1 as a digital input with pull-up resistor enabled

Configure DIO2/AD2 as an analog input

Configure DIO4 as a digital output, driving high.

To configure DIO1/AD1 as an input, issue the ATD1 command with a parameter of 3 ("ATD13"). To enable pull-up resistors on the same pin, the PR command should be issued with bit 3 set (e.g. ATPR8, ATPR1FFF, etc.). The ATD2 command should be issued with a parameter of 2 to enable the analog input ("ATD22"). Finally, DIO4 can be set as an output, driving high by issuing the ATD4 command with a parameter value of 5 ("ATD45").

After issuing these commands, changes must be applied before the module I/O pins will be updated to the new states. The AC or CN commands can be issued to apply changes (e.g. ATAC).

RSSI PWM

The XBee module features an RSSI/PWM pin (DIO10) that, if enabled, will adjust the PWM output to indicate the signal strength of the last received packet. The P0 command is used to enable the RSSI pulse width modulation (PWM) output on the pin. If P0 is set to 1, the RSSI/PWM pin will output a pulse width modulated signal where the frequency is adjusted based on the received signal strength of the last packet. When a data packet is received, if P0 is set to enable the RSSI/PWM feature, the RSSI PWM output is adjusted based on the link margin of the last packet. The RSSI/PWM output will be enabled for a time based on the RP command. Each time an RF packet is received, the RSSI/PWM output is adjusted based on the link margin of the new packet, and the RSSI timer is reset. If the RSSI timer expires, the RSSI/PWM pin is driven low. RP is measured in 100ms units and defaults to a value of 40 (4 seconds). If running on the XBIB development board, DIO10 is connected to the RSSI LEDs, which may be interpreted as follows:

PWM duty cycle	Number of LEDs turned on	Link margin
79.39% or more	3	30 db or more
62.42% to 79.39%	2	20 db to 30 db
45.45% to 62.42%	1	10 db to 20 db
Less than 45.45%	0	Less than 10 db or no reception for RP time

Soft AP mode

The XBee Wi-Fi module can operate in Soft AP mode, (aka Wi-Fi Direct). In this mode the XBee module emulates an access point (AP) rather than a station (STA). This allows another Wi-Fi client device (STA) to connect to the XBee module directly without requiring a separate AP. WPA2 security is available in Soft AP mode, but not WPA or WEP security. By default, Soft AP operates with no security.

Enable Soft AP mode

The module operates in Soft AP mode in two different ways: provisioning mode and pass through mode. These two Soft AP modes are enabled differently.

Pass through mode is enabled by setting CE to 1, which is not the default configuration. When CE is 1, it overrides parameters for provisioning mode.

Provisioning mode is enabled by default. It may be disabled by clearing bit 1 of DO. The other requirement to enable provisioning mode is that SSID must be NULL. Again, SSID is NULL by default and it may be forced to NULL by issuing the NR command.

Use Soft AP mode

When the module is operating in Soft AP provisioning mode, it is waiting for a connection from a STA device. ATAI will be 0x23 because the SSID is not configured. The STA device needs to support Wi-Fi and it needs to have an HTTP browser operating on TCP port 80. Some examples of devices that might connect to the module operating in Soft AP mode are smart phones, tablets, and laptop computers.

The connecting STA device should scan for an AP. The XBee module will advertise an SSID of 'xbee<MAC>' where <MAC> is the 6 byte MAC address of the XBee module formatted as follows: xbee-XXXXXXXXXX where each X represents a hex digit. The STA needs to connect to that SSID, and then open a browser by entering 192.168.1.10 into the address bar.

This will bring up the web page from the XBee module to allow the module to be configured as desired. The primary purpose of this feature is to configure the device to connect to the desired access point with the desired security settings. The secondary purpose is to configure any other parameters.

The current value of each configuration field is displayed on the web page. The user can enter the desired parameters and then go to the bottom of the page and press the apply button. The selected parameters will be written to the non-volatile memory on the module. However, the network access parameters are only written if a valid set of parameters is entered. The module tests the validity of those parameters by attempting to connect to the given access point. When the module is operating in Soft AP pass through mode, HTTP on port 80 is not used and it operates the same as it would in STA mode with a few exceptions:

- Only one device may connect and the connecting device must be operating in STA mode.
- A TCP listening socket on the port specified by C0 is open to accept connections, but no UDP listening socket is available.
- ID specifies the SSID sent by the module in the beacon, but if ID is NULL, it will advertise an SSID based on the MAC address of the module. For details, see [Soft AP mode](#) on page 60.

The following are operational notes:

- The Soft AP web page is designed to provide the same configuration options as are available on the XCTU program.
- The web page is divided into sections that may expand/collapse by clicking show/hide. By default, the only section that is expanded is the Network Access section.
- All fields preceded with 0x must be hex values.
- Do not programmatically configure the module in Soft AP mode because it is subject to change. Note that AI will be 0x23 while in Soft AP provisioning mode (because ID is NULL), but AI will be 0 in Soft AP pass through mode as soon as the module is ready to accept a connection from a STA device. This is true whether or not ID is null.

Wi-Fi protected setup

The XBee Wi-Fi module may also be configured using Wi-Fi Protected Setup (WPS). WPS allows for easy establishment of a secure wireless network because security parameters are learned from a nearby access point without having to enter them manually. The module only supports WPS with the push button method. (There are security concerns with using WPS with the pin method because the security information is passed in the clear.)

How to enable WPS

WPS is enabled by default, but it is disabled if SSID is configured (ATID is not NULL) or if the module is connected in Soft AP mode. To use WPS with the commissioning button, it must be enabled by configuring ATD0 to 1.

How to use WPS

WPS is invoked by pressing the commissioning button twice (if ATD0 is set to 1) or by entering the CB 2 command. Then, a corresponding WPS button must be pressed on a nearby WPS-capable AP, which allows the security parameters to be exchanged and the connection to the AP to occur.

Security

Pre-shared key mode (PSK, also known as Personal mode) is designed for home and small office networks that don't require the complexity of an 802.1X authentication server. Each wireless network device encrypts the network traffic

using a 256 bit key. This key may be entered either as a string of 64 hexadecimal digits, or as a passphrase of 8 to 63 printable ASCII characters. If ASCII characters are used, the 256 bit key is calculated by applying the PBKDF2 key derivation function to the passphrase, using the SSID as the salt and 4096 iterations of HMAC-SHA1.

Commissioning button

The commissioning button is used for setting up modules for the first time. It can only be used if D0 is set to 1. It provides two different services:

- Two button presses in fast sequence invoke WPS (see [Wi-Fi protected setup](#) on page 61).
- Four button presses in fast sequence force the module into provisioning mode by clearing the SSID and security parameters. It also ensures that soft AP mode is enabled. Note that after the four button press clears the security parameters, they are NOT written. A separate ATWR command must be entered, if desired.

The commissioning button operations may also be invoked with the ATCB 2 and ATCB 4 commands for two button presses and four button presses, respectively.

Connection indicators

In addition to AI and the associated LED indicator, this software includes two other indicators as described below:

TCP connection indicator

In transparent mode, only one TCP connection is allowed and DIO12 (aka CD) can be configured to indicate whether or not that TCP socket is connected. This is enabled by configuring P2 to 6. When so configured, DIO12 outputs a low signal when the TCP socket is connected and it outputs a high signal when the TCP socket is disconnected. Obviously, the high signal remains when operating in UDP mode because there will never be a TCP connection.

Device Cloud connection indicator

AI and the associate LED indicate when the module is fully associated with the access point, but there is another level of connectivity provided by the DI command that tells whether or not the TCP socket to the Device Cloud is connected. The values defined for DI are as follows:

- 0 = connected to the Device Cloud
- 1 = Configured, but not yet associated to AP
- 2 = Associated to AP, but not yet connected to the Device Cloud
- 3 = Disconnecting from the Device Cloud
- 4 = Not configured to connect to the Device Cloud

When DI is either 2 or 3, the Associate LED has a different blink pattern that looks like this:



Where the low signal means LED off and the high signal means LED on.

The normal association LED signal alternates evenly between high and low as shown below:



General purpose flash memory

The XBee Wi-Fi RF modules provide 160 4096-byte blocks of flash memory which can be read and written by the user application. This memory provides a non-volatile data storage area which can be used for a multitude of purposes.

Some common uses of this data storage include: storing logged sensor data, buffering firmware upgrade data for a host microcontroller, or 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 upgrade file for over-the-air firmware upgrades of the XBee module itself.

Accessing general purpose flash memory

The GPM of a target node can be accessed from the XBee serial port or from a non-XBee network client.

Serial port access is done by sending explicit API frames to the MEMORY_ACCESS cluster ID (0x23) on the DIGI_DEVICE endpoint (0xE6) of the target node. (Explicit API frames have frame identifier 0x11; for details, see [API operation](#) on page 80.)

Access from a non-XBee network client is done by sending UDP frames to the target node on port 0x0BEE. The payload begins with an application header followed by the GPM header described below. (See [Network client](#) on page 43R to learn how to format the application header.)

The following header is used to generate a GPM command. It should be used whether using serial port access or network client access. For network client access, an application header needs to precede the GPM header. To keep things simple, this section is written from the perspective of serial port access, without the application header. Do not forget to precede each frame with an application header if using a network client for GPM access.

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	Specific GPM commands are described below
1	1	GPM_OPTIONS	Command-specific option
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM. Ranges from 0 to 159 (0x9F).
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block
6	2*	GPM_NUM_BYTES	Then number of bytes in the GPM_DATA field, or in the case of a READ, the number of bytes requested
8	Varies	GPM_DATA	

*Multibyte parameters should be specified with big-endian byte ordering.

When a GPM command is sent to a radio via a unicast the receiving radio will unicast a response back to the requesting radio's source endpoint specified in the request packet. No response is sent for broadcast requests. If the source endpoint is set to the DIGI_DEVICE endpoint (0xE6) or explicit API mode is enabled on the requesting radio then a GPM response will be output as an explicit API RX indicator frame on the requesting node (assuming API mode is enabled.) The format of the response is very 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. Ranges from 0 to 159 (0x9F)

Byte offset in payload	Number of bytes	Field name	General field description
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block
6	2*	GPM_NUM_BYTES	Then number of bytes in the GPM_DATA field
8	Varies	GPM_DATA	

*Multi-byte parameters should be specified with big-endian byte ordering.

The following commands exist for interacting 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	
GPM_START_INDEX	
GPM_NUM_BYTES	
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 will send 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_OPTIONS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Indicates the number of GPM blocks available.
GPM_START_INDEX	Indicates the size of a GPM block in bytes.
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 should be specified for this command.

Example:

A **PLATFORM_INFO_REQUEST** sent to a radio 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. The ERASE command can also be used to erase all blocks of the GPM by setting the GPM_NUM_BYTES field to 0.

ERASE_RESPONSE (0x81):

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 should be specified for this command.

When an ERASE command request has been unicast to a node, that node will send 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	
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 should be specified for this command.

Example:

To erase flash block 42 of a target radio with serial number of 0x0013a200407402ac, an ERASE packet should be formatted 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. Ranges from 0 to 159 (0x9F).
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be written.
GPM_NUM_BYTES	Specifies number of bytes to write to GPM. If the command is ERASE_THEN_WRITE (0x03), the GPM is erased before it is written. For the WRITE (0x02) command, the area being written should have previously been erased. Note that if this parameter is zero, the command erases the entire GPM and writes nothing. 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. It is also important to remember that the number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the radio. The maximum payload size can be queried with the ATNP 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 will send 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.

Field name	Command-specific description
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	
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 should be specified for this command.

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):

The READ command can be used 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. Ranges from 0 to 159 (0x9F).
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 be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. It is also important to remember that the number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the radio. The maximum payload size can be queried 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 will send 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	
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):

The FIRMWARE_VERIFY and FIRMWARE_VERIFY_AND_INSTALL commands are used when remotely updating firmware on a module. Remote firmware upgrades are covered in detail in the next section. These commands check if the General Purpose Memory 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 module will reset and begin 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	These fields are unused for this command. Set to 0.
GPM_START_INDEX	
GPM_NUM_BYTES	
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 will send 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_OPTIONS	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	These fields are unused for this command. Set to 0.
GPM_START_INDEX	
GPM_NUM_BYTES	
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 will send 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 module will reset and begin using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_AND_INSTALL_RESPONSE (0x86).
GPM_OPTIONS	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	These fields are unused for this command. Set to 0.
GPM_START_INDEX	
GPM_NUM_BYTES	
GPM_DATA	This field is unused for this command.

Example:

To verify a firmware image previously loaded into the GPM on a target radio with serial number of 0x0013a200407402ac, a FIRMWARE_VERIFY packet should be formatted 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
```

Working with flash memory

When working with the General Purpose Memory, the user should be aware of a number of limitations associated with working with flash memory:

- Flash memory write operations are only capable of changing binary 1's to binary 0's. Only the erase operation can change binary 0's to binary 1's. For this reason it is usually necessary to erase a flash block before performing a write operation.
- A flash memory block must be erased in its entirety when performing an erase operation. A block cannot be partially erased.

- Flash memory has a limited lifetime. The flash memory on which the GPM is based is rated at 20,000 erase cycles before failure. Care must be taken to ensure that the frequency of erase/ write operations allows for the desired product lifetime. Digi's warranty will not cover products whose number of erase cycles has been exceeded.
- Over-the-Air firmware upgrades (described in the next section) require the entire GPM be erased. Any user data stored in the GPM will be lost during an over-the-air upgrade.

Over-the-air firmware upgrades

The XBee Wi-Fi RF modules provide two methods of updating the firmware on the module. Firmware can be updated locally via XCTU (a free testing and configuration utility provided by Digi) using the radio's serial port interface. Firmware can also be updated using the radios' RF interface (Over-the-Air Updating.)

The over-the-air firmware upgrading method provided is a robust and versatile technique which can be tailored to many different networks and applications. It has been engineered to be reliable and minimize disruption of normal network operations.

There are three phases of the over-the-air upgrade process: distributing the new application, verifying the new application, and installing the new application. In the following section the node which will be upgraded will be referred to as the target node. The node providing the update information will be referred to as the source node. In most applications the source node will be locally attached to a PC running update software.

Distributing the new application

The first phase of performing an over-the-air upgrade on a module is transferring the new firmware file to the target node. The new firmware image should be loaded in the target node's GPM prior to installation XBee Wi-Fi RF modules use an encrypted binary (.ebin) file for both serial and over-the-air firmware upgrades. These firmware files are available on the Digi Support website.

The contents of the .ebin file should be sent to the target radio using general purpose memory WRITE commands. The entire GPM should be erased 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:

If the size of the .ebin file is 217,088 bytes, then it could be sent to the module in 1024 byte blocks as follows:

CPT_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
0	0	1024	0 to 1023
0	1024	1024	1024 to 2047
0	2048	1024	2048 to 3071
0	3072	1024	3071 to 4095
1	0	1024	4096 to 5119
1	1024	1024	5120 to 6143
-	-	-	-
-	-	-	-
-	-	-	-

CPT_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
52	1024		214,016 to 215,039
52	2048		215,040 to 216,063
52	3072		216,064 to 217,087

Verifying 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 will report if the uploaded data is invalid. If the data is valid it will begin installing the application. No installation will take place on invalid data.

Installing the application

When the entire .ebin file has been uploaded to the GPM of the target node a FIRMWARE_VERIFY_AND_INSTALL command can be issued. Once the target receives the command it will verify the .ebin file loaded in the GPM. If it is found to be valid then the module will install the new firmware. This installation process can take up to 8 seconds. During the installation the module will be unresponsive to both serial and RF communication. To complete the installation the target module will reset. AT parameter settings which have not been written to flash (using the WR command) will be lost.

Things to remember

- The firmware upgrade process requires that the module resets itself. Because of this reset parameters which have not been written to flash will be lost after the reset. To avoid this, write all parameters with the WR command before doing a firmware upgrade.
- 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 module in either case.

Configure and query devices with Device Cloud

Use Device Cloud (<https://devicecloud.digi.com>) to perform the operations in this section. Each operation requires that you enable Device Cloud with the **DO** command and that you connect the XBee Wi-Fi RF Module to an access point that has an external Internet connection to allow access to Device Cloud.

Use XCTU to enable Device Cloud

To enable an XBee Wi-Fi RF Module connected to XCTU for Device Cloud:

1. Select the module in the **Radio Modules** list.
2. In the **Working area** of XCTU, type 1 in the **DO** (Device Options) command field; this enables Device Cloud for the module.
3. Click the Write button.

Configure the device

If your XBee devices are connected to Device Cloud, you can query and configure the XBee Wi-Fi RF Module through Device Cloud. To do this:

1. Log in to Device Cloud.
2. Click the **Device Management** tab.
3. Click the **XBee Networks** tab.
4. Double click the device you want to configure.
5. Click **Configuration**. A list of AT command types displays.
6. Click the category of AT commands that you want to modify, **Network Settings** for example.
7. Change the settings of the appropriate AT command(s).
8. Click the **Refresh** button to query the current configuration.
9. Click the **Save** button to save the current configuration changes. If the changes are valid, Device Cloud writes them to non-volatile memory and applies them.
10. If you want to apply these changes later, click the drop-down arrow on the **Save** button and select **Schedule**.
11. The **Save Device Properties** dialog allows you to schedule changes to connected devices to be either:

- a. **Immediate** (default)
 - b. **One-Time**: Schedule a one-time reboot at a specified date and time.
 - c. **Recurring**: Set a recurring reboot schedule, with start and end dates and times, as well as frequency.
 - d. **Schedule Offline**: Queue device reboot to occur the next time this device connects to Device Cloud.
12. Click the **Export** button to export the device's settings; the **Export Properties** dialog appears.
 13. Choose the **Export all** option (default) or **Export all except unique network and device identity properties**.
 14. Click **OK**. A message appears, asking whether you want to open or save the downloaded file.

Output control

You can find **Executable Commands** in **Device Management**. Select an XBee Wi-Fi RF Module and click **System Information**. The executable commands are the **IO** (Set Output Pins) and **OM** (Output Mask) commands.

1. Log in to Device Cloud.
2. Click the **Device Management** tab.
3. Click the **XBee Networks** tab.
4. Double click the device you want to configure.
5. Click **System Information**. A list of AT command types displays.
6. Click **Executable commands**.
7. In the **(IO) Set output pins** field, configure the pins that you want to set the output to the desired level. None of the pins are configured for output by default. The parameter given to the IO command is a bit map that specifies which IO lines are set to which levels; see [IO command bits](#) on page 73 for details on the bits.
8. In the **(OM) Set mask for IO command field** set the bits you want to control.
The **OM** command is an output mask that enables (1) or disables (0) the corresponding bit in the **IO** command. To control the output level of a pin with an **IO** command, you set the corresponding bit in the **OM** command and you configure the corresponding pin (for example DIO2 for bit 2) as an output low (4) or output high (5).
9. Send the **IO** command to set the output to the desired level. (0 sets it low and 1 sets it high.)

Each pin also has an associated timer to be used in conjunction with the **IO** command. The timer determines how long the **IO** command remains effective for each IO pin that is set to a level different than its configured value. If the timer is set to a default value of 0, then the **IO** command remains in effect until it is overridden by another **IO** command or you reset the module. Otherwise, the timer specifies the number of tenth second (100 ms) units that the module stays at the selected level before reverting to its configured level. The maximum value allowed is 6000 (0x1770), which allows for ten minutes.

The AT commands for these timers are **T0** through **T9** for pins DIO0 through DIO9 and **Q0** through **Q9** for pins DIO10 through DIO19. To modify these commands, click **Configuration > Input and Output Settings** and use the choice menu to modify the time.

IO command bits

The **IO** command sets the level of output pins to high or low. The parameter you give to the **IO** command is a bit map that specifies which IO pins are set to which levels. The following table shows the bits and the corresponding pin.

No pins are configured for output by default.

Bit	IO pin
0	DIO0
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	DIO5
6	DIO6
7	DIO7
8	DIO8
9	DIO9
10	DIO10
11	DIO11
12	DIO12
13	DIO13
14	DIO14
15	DIO15 (surface mount module only)
16	DIO16 (surface mount module only)
17	DIO17 (surface mount module only)
18	DIO18 (surface mount module only)
19	DIO19 (surface mount module only)

Send I/O samples to Device Cloud

To send I/O samples to Device Cloud:

1. Set **IR** to a non-zero value.
2. Activate at least one I/O line.
3. Enable Device Cloud.

If Device Cloud is not enabled, then I/O samples go to the address specified by the **DL** command. See [Periodic I/O sampling](#) on page 59.

View I/O samples in Device Cloud

1. In your Device Cloud account, click **Device Management > Devices**.
2. Double click the device.
3. Click **Home**.
4. Click the **View Device Streams** button.

or

1. Click **Data Services > Data Streams**.
2. In the **Stream** column, click the MAC address of the module/serial/0.

The data streams are organized by the serial number of the sending device and then by the signal line being monitored.

Update the firmware

The XBee Wi-Fi RF Module supports Device Cloud firmware updates. To perform a firmware update, use the following steps.

1. Download the updated firmware file for your device from the manufacturer's site.
2. In your Device Cloud account, click **Device Management > Devices**.
3. Select the first device you want to update.
4. To select multiple devices (must be of the same type), press the Control key and select additional devices.
5. Click **More** in the Devices toolbar and select **Update Firmware** from the Update category of the More menu. The Update Firmware dialog appears.
6. Click **Browse** to select the appropriate firmware file for the device. The file must have a .ebin extension, which indicates an encrypted binary file. This same file is normally zipped up with the .mxi file of each firmware release.
7. Click **Update Firmware**. The updated devices will automatically reboot when the updates are complete.

Send data requests

The Send Data request allows the host to use the XBee Wi-Fi RF Module to send a file to Device Cloud. To send a request, a host connected to the serial port of the device sends a Send Data Request (0x28) API frame. For details, see [Send data request](#) on page 93.

The device request process

If you have a host application connected to the serial port of the XBee Wi-Fi RF Module, you can enable Device Cloud to send a message to the host, as the following sequence shows.

1. In Device Cloud, send the Device Request frame (0xB9) to the host.
2. If the Device Request frame (0xB9) ID is non-zero, then after the Device Request frame (0xB9) goes out the serial port, the host has up to five seconds to send back a Device Response frame (0x2A).
3. If the host does not send a Device Response frame (0x2A), the XBee module sends a timeout response to Device Cloud.
4. After the host sends a Device Response frame (0x2A), the XBee module sends a Device Response Status frame (0xBA) to the host.

For information about the format of these frames, see:

- [Device request](#) on page 107
- [Device response](#) on page 94
- [Device response status](#) on page 107

About the device request and frame ID

The firmware uses two identifiers in these three frames to correlate the messages:

1. The device request ID identifies the device request
2. The frame ID identifies the device response

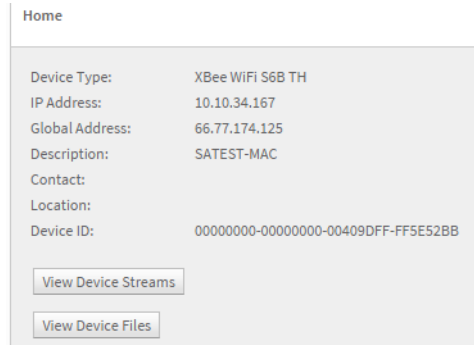
The host reads the Device Request frame (0xB9) ID when it is received on the serial port. If the device request ID is non-zero, it generates a device response containing that same device request ID. A mismatch causes an error. In addition to the Device Request frame (0xB9) ID, the Device Response frame (0x2A) that the host generates contains a frame ID. A frame ID of 0 instructs the XBee module not to send a Device Response Status frame (0xBA). A non-zero frame ID is a request for a Device Response Status frame (0xBA), which includes the designated frame ID. Therefore, a Device Request frame (0xB9) contains a device request ID, a Device Response frame (0x2A) contains a device request ID and a frame ID, and a Device Response Status frame (0xBA) contains only a frame ID.

Populate and send a Device Request frame (0xB9)

1. In Device Cloud, click **Documentation > API Explorer**.
2. Click **Examples > SCI > Data Service > Send request**. A sample XML file that you can modify appears as seen in the following code sample:

```
<sci_request version="1.0">
  <data_service>
    <targets>
      <device id="00000000-00000000-00000000-00000000"/>
    </targets>
    <requests>
      <device_request target_name="myTarget">
        my payload string
      </device_request>
    </requests>
  </data_service>
</sci_request>
```

3. Under **<targets>** type the MAC address of the XBee module in this format:
00000000-00000000-010203FF-FF040506
where 01 to 06 are the first through sixth bytes of the MAC address, respectively, and 00 and FF are literally 00 and FF. In Device Cloud, you can find the device MAC address in this format in the device's **Home** field.



4. Under **<requests>**, you can type a target name as desired, but any target name beginning with “XBee” (case insensitive) is reserved for use on the XBee module itself and will not be sent out the serial port.
5. Type the string that will be output in the device request. Both the target name and the device request string depend on your application and the XBee module passes these strings on, unmodified.
6. In the **HTTP Method field**, select the **POST** option button.
7. Click **Send**. The device response appears in the **Web Services Responses** pane.

Transparent mode data

The XBee Wi-Fi RF Module also supports Device Cloud transmissions and receptions in Transparent mode. Transparent data is sent to Device Cloud using the Send Data interface and Transparent data is received from Device Cloud using the Device Request interface. Some parts of those interfaces are lost due to not using the API interface.

Send data to Device Cloud

The module can send serial data to Device Cloud as files or as binary data points. To make this selection, use bit 4 of the **DO** command. If you set **DO** bit 4, then the serially-connected host sends Transparent data to Device Cloud as binary data points. Otherwise, it sends the data as a file. Files are static and permanent. Data streams are dynamic and you can graph them to understand changes.

Things you need to know to put serial data in Device Cloud

AP (API Enable) command:

- 0: Transparent Mode
- 1: API Mode without escapes
- 2: API Mode with escapes

EQ (Device Cloud Server FQDN) command:

The URL for Device Cloud must be my.devicecloud.com or devicecloud.digi.com

DO (Device Options) command:

- Bits 0-7: Encoded in Hex.
- 0: Enable Device Cloud (1), Disable (0).
- 1: Soft **AP** when **ID** = None. This has nothing to do with Device Cloud.
- 2: Send Transparent Data to Device Cloud (1), send to **DL** (0).
- 3: Send **IO** both to Device Cloud and **DL** (1), Device Cloud only (0).

4: Send Transparent data as a data stream (1), file data (0) – must have bit 2.

5: If bit 4 is 0, replace file (1), append to file (0).

Hence, the following values:

00: Do not use Device Cloud at all.

05: Send Transparent data to Device Cloud as a file which is appended (00000101).

`/ws/FileData/~00000000-00000000-00409DFF-FFxxxxx/serial/0`

25: Send transparent data to Device Cloud as a file which is overwritten (00100101)

`/ws/FileData/~00000000-00000000-00409DFF-FFxxxxx/serial/0`

15: Send transparent data to Device Cloud as a data point – encoded in BINARY

`/ws/v1/streams/inventory/00000000-00000000-00409DFF-FF5DB54F/serial/0`

`/ws/DataPoint/00000000-00000000-00409DFF-FF5DB54F/serial/0`

`/ws/DataStream/00000000-00000000-00409DFF-FF5DB54F/serial/0`

01: Connect to Device Cloud, but send Transparent data to the address and port of **DL/DE** using the protocol in IP.

This is convenient if you want to use the TCP or UDP method for posting at datapoint.

Send files

The file name that is written on Device Cloud is named “serial/0”. The file type will be “text/plain”. **DO** bit 5 selects whether to append to an existing file or to replace it. If replacing an existing file and the size of the data being sent exceeds the maximum frame size allowed (1400 bytes), then that frame will be broken up and only the last part will show up in the file because the last part will replace rather than append to the first part.

Send binary data points

To send binary data points to Device Cloud in Transparent mode: set **DO** to 0x15 or 0x17, which means:

- Set **DO** to bit 4.
- Set **DO** bit 0 to enable Device Cloud.
- Set **DO** bit 2 so Transparent data goes to Device Cloud.

To view the files:

1. Double click the device.
2. Click **Home**.
3. Click the **View Device Streams** button.
or
4. Click **Data Services > Data Streams**.
5. In the **Stream** column, click the MAC address of the module/serial/0.

Receive data from Device Cloud

Transparent data is received from Device Cloud using the Device Request interface if the module is operating in transparent mode (**AP** = 0) and Device Cloud is enabled with **DO** bit 0. Only the raw data will be seen on the serial

interface and the target string to which Device Cloud sent the data will not be available. You must use API mode to see the target string. In transparent mode, Device Cloud should not request a response because none will be given.

API operation

As an alternative to Transparent Operation, API (Application Programming Interface) Operations are available. API operation requires that communication with the module be done through a structured interface (data is communicated in frames in a defined order). The API specifies how commands, command responses and module status messages are sent and received from the module using a UART or SPI Data Frame.

Note that Digi may add new API frames to future versions of firmware, so build into your software interface the ability to filter out additional API frames with unknown Frame Types.

API frame specifications

Two API modes are supported and both can be enabled using the AP (API Enable) command. Use the following AP parameter values to configure the module to operate in a particular mode:

- AP = 1: API Operation
- AP = 2: API Operation (with escaped characters)

API operation (AP parameter = 1)

When this API mode is enabled (AP = 1), the UART or SPI data frame structure is defined as follows:

UART or SPI data frame structure:



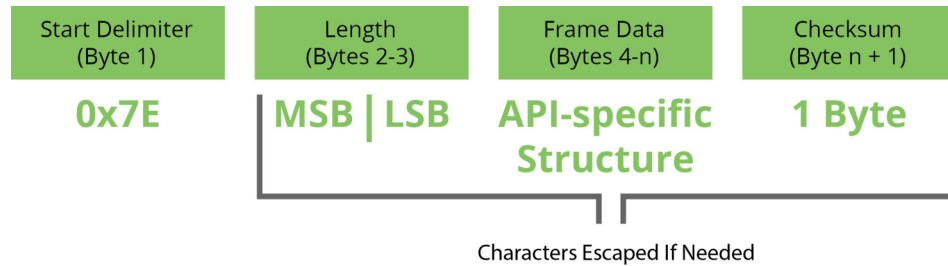
MSB = Most Significant Byte, LSB = Least Significant Byte

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

API operation with escape characters (AP parameter = 2)

When this API mode is enabled (AP = 2), SPI mode is not supported and the UART frame structure is:

UART Data Frame Structure with escape control characters:



MSB = Most Significant Byte, LSB = Least Significant Byte

Escape characters

When sending or receiving a UART data frame, specific data values must be escaped (flagged) so they do not interfere with the data frame sequencing. To escape an interfering data byte, insert 0x7D and follow it with the byte to be escaped XOR'd with 0x20.

Data bytes that need to be escaped:

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

Example: Raw UART data frame (before escaping interfering bytes)

```
0x7E 0x00 0x02 0x23 0x11 0xCB
```

0x11 needs to be escaped, which results in the following frame:

```
0x7E 0x00 0x02 0x23 0x7D 0x31 0xCB
```

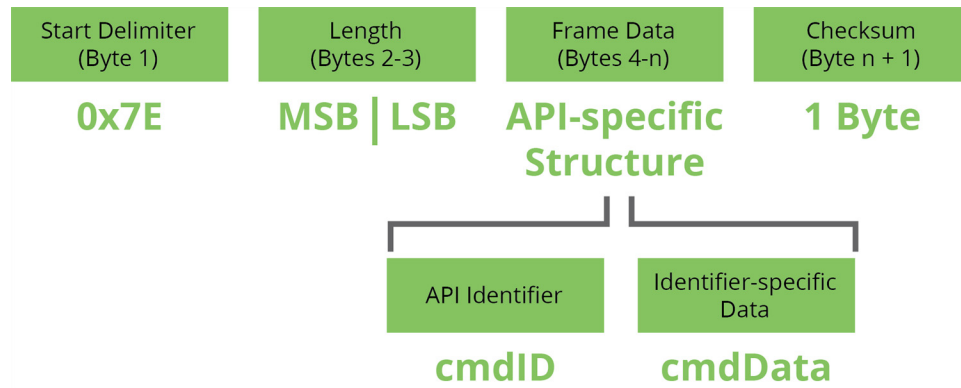
Note In the above example, the length of the raw data (excluding the checksum) is 0x0002 and the checksum of the non-escaped data (excluding frame delimiter and length) is calculated as: $0xFF - (0x23 + 0x11) = (0xFF - 0x34) = 0xCB$.

Length

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

Framed data

Frame data of the UART or SPI data frame forms an API-specific structure as follows:

UART or SPI data frame and API-specific structure:

The cmdID frame (API-identifier) indicates which API messages will be contained in the cmdData frame (Identifier-specific data). Note that multi-byte values are sent big endian. The XBee modules support the following API frames:

API frame names and values

API frame names	API ID
Tx64 Request	0x00
Remote Command Request	0x07
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Packet	0x10
ZigBee Explicit Transmit Packet	0x11
ZigBee Remote AT Command	0x17
TX IPv4	0x20
Send Data Request	0x28
Device Response	0x2A
Rx64 Indicator	0x80
Remote Command Response	0x87
AT Command Response	0x88
TX Status	0x89
Modem Status	0x8A
ZigBee TX Status	0x8B
IO Data Sample Rx Indicator	0x8F
ZigBee Receive Packet	0x90
Explicit ZigBee Receive Packet	0x91
ZigBee Remote AT Command Response	0x97
RX IPv4	0xB0
Send Data Response	0xB8

API frame names	API ID
Device Request	0xB9
Device Response Status	0xBA
Frame Error	0xFE

Checksum

To test data integrity, a checksum is calculated and verified on non-escaped data.

- **To calculate:** Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract the result from 0xFF.
- **To verify:** Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

API examples

Example: Create an API AT command frame to configure an XBee baud rate to 230,400 (set BD to 0x08).

The frame should look like (in hex):

```
7E 00 05 08 01 42 44 08 68
```

Where:

- 0x0005 = length excluding checksum
- 0x08 = AT Command API frame type
- 0x01 = Frame ID (set to non-zero value for transmit status)
- 0x4244 = AT Command ('BD')
- 0x08 = value to set command to
- 0x68 = Checksum

The checksum is calculated as $[0xFF - (0x08 + 0x01 + 0x42 + 0x44 + 0x08)]$

Example: Send a remote command to a module with the IP address 192.168.0.103 (C0 A8 00 67) to set DIO1/AD1 as a digital input (D1=3) and apply changes to force the IO update. The API remote command frame should look like (in hex):

```
7E 00 0E 07 01 00 00 00 00 C0 A8 01 64 02 44 31 03 B0
```

Where:

- 0x000E = length (14 bytes excluding checksum)
- 0x07 = Remote Command API frame type
- 0x01 = Frame ID
- 0x00000000 C0A80067 = Remote address (Pad first 4 bytes with 00)
- 0x02 = Apply Changes (Remote Command Options)
- 0x4431 = AT command ('D1')
- 0xB0 = Checksum

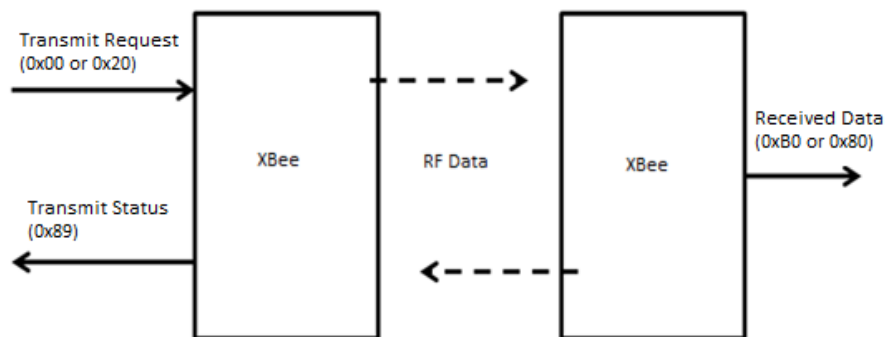
API UART and SPI exchanges

AT commands

The following image shows the API frame exchange that takes place at the UART or SPI when sending an AT command request to read or set a module parameter. The response can be disabled by setting the frame ID to 0 in the request.

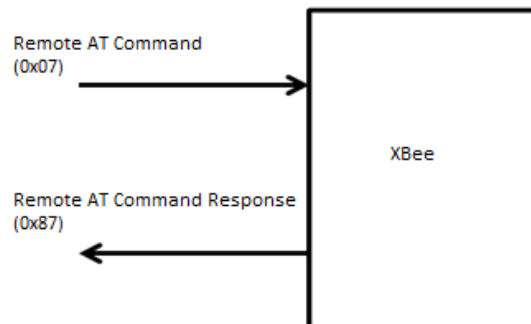
Transmitting and receiving RF data

The following image shows the API exchanges that take place at the UART or SPI when sending RF data to another module. The transmit status frame is always sent at the end of a data transmission unless the frame ID is set to 0 in the transmit request. If the packet cannot be delivered to the destination, the transmit status frame will indicate the cause of failure. The received data frame (0x80 or 0xB0) is set by the AP command.



Remote AT commands

The following image shows the API frame exchanges that take place at the UART or SPI when sending a remote AT command. A remote command response frame is not sent out the UART or SPI if the remote module does not receive the remote command.



API frames

The following sections illustrate the types of frames encountered while using the API.

TX (Transmit) request: 64-Bit

Frame Type: 0x00

This frame type uses the XBee Application Service. This command allows for software compatibility with other XBee module such as the 802.15.4 module.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x0D	
API frame-specific data	API frame identifier	3	0x00	
	Frame ID	4	0x01	
	64-bit destination address	5	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. IP address is in hex. The address in this example is 192.168.0.100. For sending a broadcast, use 0xFF 0xFF 0xFF 0xFF. A MAC address may also be in the lower 6 bytes of this field. But if the MAC address doesn't match the module's own MAC address, then this field will be interpreted as an IP address, as described above.
			0x00	
			0x00	
			0x00	
			0xC0	
			0xA8	
			0x00	
0x64				
TX options	13	0x00	0x01 – Disable ACK All other bits must be set to 0.	
Data	14	0x1516	Max is 1392 bytes. Data will be sent to the XBee application service port.	
Checksum			0x07	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Remote AT command request

Frame Type: 0x07

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

Example: Send a remote command to query the DL register on a remote module. In this example, the IP address of the remote is 192.168.0.100.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x0D	
API frame-specific data	API frame identifier	3	0x07	
	Frame ID	4	0x01	
	64-bit destination address	5	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. IP address is in hex. The address in this example is 192.168.0.100. A MAC address may also be in the lower 6 bytes of this field. But if the MAC address doesn't match the module's own MAC address, then this field will be interpreted as an IP address, as described above.
		6	0x00	
		7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x00	
		12	0x64	
	Command options	13	0x02	0x02 – Apply changes on the remote. If not set then the AC command must be sent or the last remote command sent must set this option.
	AT command	MSB 14	0x44(D)	Command Name - Two ASCII characters that identify the AT command
LSB 15		0x4C(L)		
Parameter value	--	--	If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.	
Checksum		16	0x99	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

AT command

Frame Type: 0x08

Used to query or set module parameters on the local module. This API command applies changes after executing the command. (Changes made to module parameters take effect once changes are applied.) The API example below illustrates an API frame when modifying the NI parameter value of the module.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x05	
API frame-specific data	API frame identifier	3	0x08	
	Frame ID	4	0x01	
	AT command	MSB 5	0x4E(N)	Command Name - Two ASCII characters that identify the AT command
		LSB 6	0x49(I)	
Parameter value	-	-	If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.	
Checksum		7	0x5E	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

AT command-queue parameter value

Frame Type: 0x09

This API type allows module parameters to be queried or set. In contrast to the “AT Command” API type, new parameter values are queued and not applied until either the “AT Command” (0x08) API type or the AC (Apply Changes) command is issued. Register queries (reading parameter values) are returned immediately.

Example: Send a command to change the baud rate (BD) to 115200 baud, but don't apply changes yet. (Module will continue to operate at the previous baud rate until changes are applied.)

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x05	
API frame-specific data	API frame identifier	3	0x09	
	Frame ID	4	0x01	

Frame fields		Offset	Example	Description
API frame-specific data	AT command	MSB 5	0x42 (B)	Command Name - Two ASCII characters that identify the AT command
		LSB 6	0x44 (D)	
	Parameter value	7	0x07	If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
Checksum		8	0x68	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

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

ZigBee transmit packet

Frame Type: 0x10

This frame type is only provided for software compatibility with other XBee modules. Frame type 0x20 is recommended for data transmissions from this module. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x13	
API frame-specific data	API frame identifier	3	0x10	
	Frame ID	4	0x01	Correlates request with a later TX STATUS frame (0x8B). If 0, no TX STATUS frame will be sent
	64-bit source address	5	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. IP address is in hex. The example uses address 192.168.0.130. A MAC address may also be in the lower 6 bytes of this field. But if the MAC address doesn't match the module's own MAC address, then this field will be interpreted as an IP address, as described above.
		6	0x00	
		7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x01	
12	0x82			

Frame fields		Offset	Example	Description
API frame-specific data	Reserved	13	0xFF	Unused placeholders
		14	0xFE	
		15	0x00	
	Options	16	0x00	0x01 – Disable ACK All other bits must be set to 0
	RF data	17	0x48 ‘H’	Up to 1392 bytes of data.
		18	0x65 ‘e’	
		19	0x6C ‘l’	
20		0x6C ‘l’		
21		0x6F ‘o’		
Checksum		22	0x12	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

ZigBee explicit transmit packet

Frame Type: 0x11

This frame type is provided for software compatibility with other XBee modules and for sending GPM requests. If neither of these is required, then frame type 0x20 is recommended for data transmissions from this module. An example of a GPM request is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and checksum
		LSB 2	0x1C	
API frame-specific data	API frame identifier	3	0x11	
	Frame ID	4	0x01	Correlates request with a later TX STATUS frame (0x8B). If 0, no TX STATUS frame will be sent.
	64-bit source address	5	0x00	Align IP address to low 32-bits of the field. The other bytes are set to 0. IP address is in hex. This example uses address 192.168.1.130. A MAC address may also be in the lower 6 bytes of this field. But if the MAC address doesn't match the module's own MAC address, then this field will be interpreted as an IP address, as described above.
6		0x00		

Frame fields		Offset	Example	Description
API frame-specific data	64-bit source address	7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x01	
		12	0x82	
	Reserved	13	0xFF	Unused placeholders
		14	0xFE	
		15	0xE6	
	Destination endpoint	16	0xE6	Digi Device Object
	Cluster ID	17	0x00	Memory Access Cluster ID
		18	0x23	
	Reserved	19	0xC1	Unused placeholders
		20	0x05	
		21	0x00	
	Options	22	0x00	0x01 – Disable ACK - All other bits must be set to 0.
	RF data	23	0x00	Up to 1392 bytes of data.
		24	0x00	
		25	0x00	
		26	0x00	
		27	0x00	
		28	0x00	
		29	0x00	
		30	0x00	
Checksum	31	0x50	Add this value to sum of bytes from byte 3 to here such that result = 0xff.	

ZigBee remote AT command

Frame Type: 0x17

This frame type is only provided for software compatibility with other XBee modules. Frame type 0x07 is recommended for sending remote commands from this module. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x13	
API frame-specific data	API frame identifier	3	0x17	
	Frame ID	4	0x01	
	64-bit source address	5	0x00	Align IP address to low 32-bits of the field. The other bytes are set to 0. IP address is in hex. This example uses address 192.168.1.130
		6	0x00	
		7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x01	
		12	0x82	
	Reserved	13	0xFF	Unused placeholders
		14	0xFE	
	Command options	15	0x02	0x02 – Apply changes on remote. If not set, then AC command must be sent or the last remote command sent must set this option.
	AT command	16	0x44 ‘D’	Two ASCII characters representing command name (DL in this case).
		17	0x4C ‘L’	
	Parameter value	18	0xC0	Sets DL to 192.168.1.140 (Parameter value field doesn’t exist on a query.)
19		0xA8		
20		0x01		
21		0x8C		
Checksum		22	0x78	Add this value to sum of bytes from byte 3 to here such that result = 0xff.

Transmit (TX) request: IPv4

Frame Type: 0x20

This frame type utilizes the serial data service. The frame gives greater control to the application over the IP setting for the data.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and checksum
		LSB 2	0x11	
API frame-specific data	API frame identifier	3	0x20	
	Frame ID	4	0x01	Set to a value that will be passed back in the Tx Status frame. 0 disables the Tx Status frame.
	IPv4 32-bit destination address	MSB 5	0xC0	Use 0xFFFFFFFF for broadcast when protocol is UDP. The address in the example is for a destination of 192.168.0.100
		6	0xA8	
		7	0x00	
		8	0x64	
	16-bit destination port	MSB 9	0x26	UDP or TCP port number
		LSB 10	0x16	
	16-bit source port	MSB 11	0x26	UDP or TCP port number
		LSB 12	0x16	To send a UDP packet, this must match the port number of the listening port as specified by C0. To send a TCP packet on a new connection, this must be 0.
Protocol	13	0x00	0 = UDP, 1= TCP - Protocol use for the transmitted data	
Transmit options bitfield	14	0x00	Bit field: BIT 1 = 1 - Terminate socket after tx complete 0 - Leave socket open (use TCP timeout). Ignore bit for UDP packets. All other bits are reserved and should be 0.	

Frame fields		Offset	Example	Description
API frame-specific data	RF data	15	0x48('H')	Up to 1400 bytes of data. This is 8 bytes more than the max size reported by NP command because no application header is needed.
		16	0x65('e')	
		17	0x6C('l')	
		18	0x6C('l')	
		19	0x6F('o')	
Checksum		20	0xA6	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Send data request

Frame Type: 0x28

This frame type is used to send a file of the given name and type to Device Cloud.

Frame fields		Offset	Example	Description
Start		0	0x7E	
Length		1-2	0x0033	Number of bytes between the length and the checksum
API frame-specific data	API frame identifier	3	0x28	
	Frame ID	4	0x55	Identifies the frame for send data response. If 0, then no send data response status will be received.
	Path length	5	0x08	Length of path and file name
	Path	6-13	TestFile	Path and file name
	Content type length	14	0x0A	Length of target string (up to 16 bytes)
	Content type	15-24	Text/plain	Indicates file type, e.g. text/plain, text/xml, or application/json
	Transport	25	0x00	Must be 0 to indicate TCP
	Options	26	0x00	0—overwrite 1—archive 2—append 3—transient data (do not store)
Data	27-53	abcdefghi		
Checksum		54	0x49	0xFF minus the 8 bit sum of bytes 3-52 of this frame.

Device response

Frame Type: 0x2A

This frame type is sent to the serial port by the host in response to the device request (0xB9). It should be sent within five seconds to avoid a timeout error.

Frame fields		Offset	Example	Description
Start		0	0x7E	
Length		1-2	0x0009	Number of bytes between the length and the checksum
API frame-specific data	API frame identifier	3	0x2A	
	Frame ID	4	0x01	Identifies the frame for the device response status. If 0, then no device response status will be received.
	Device request ID	5	0x00	This number should match the device request ID in the device request. Otherwise, an error will occur. If Device Request ID was 0 in the 0xB9 frame, then this Device Response (0x2A) frame is not expected from the serial port.
	Reserved	6	0x00	Must be 0 for now.
	Data	7-11	Hello	The particular data for the device response is
Checksum		12	0xE0	0xFF minus the 8 bit sum of bytes 3-11 of this frame

Rx (Receive) packet: 64-bit

Frame Type: 0x80

This frame type is used by XBee when RF data is received using the XBee application service. It allows for software compatibility with other XBee modules such as 802.15.4. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
API frame-specific data	API Frame Identifier	3	0x80	

Frame fields		Offset	Example	Description
API frame-specific data	64-bit source address	4	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. IP address is in hex. The example uses address 192.168.0.103
		5	0x00	
		6	0x00	
		7	0x00	
		8	0xC0	
		9	0xA8	
		10	0x00	
		11	0x67	
	RSSI	12	0x2E	RSSI in terms of dB above sensitivity (link margin)
	Options	13	0x00	None currently defined
	RF data	14	0x48 'H'	Up to 1392 bytes of data
15		0x65 'e'		
16		0x6C 'l'		
17		0x6C 'l'		
18		0x6F 'o'		
Checksum	19	0x8E	0xFF - the 8 bit sum of bytes from offset 3 to this byte	

Remote command response

Frame Type: 0x87

If a module receives a remote command response RF data frame in response to a Remote AT Command Request, the module will send a Remote AT Command Response message out the UART or SPI.

Example: If a remote command is sent to a remote module with an IP address of 192.168.0.103 to set the D1 parameter to 3 (digital input), the response is shown in the example API frame in the table below.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x0D	

Frame fields		Offset	Example	Description
API frame-specific data	API frame identifier	3	0x87	
	Frame ID	4	0x01	
	64-bit responder address	5	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. Value is in hex. In this example the IP address is 192.168.0.103.
		6	0x00	
		7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x00	
		12	0x67	
AT command	MSB 13	0x44 (D)	Command Name - Two ASCII characters that identify the AT command	
	LSB 14	0x31 (1)		
Status	15	0x00	0 = OK 1 = ERROR 2 = Invalid dommand 3 = Invalid parameter 4 = Tx failure	
Parameter value		-	If present, indicates value of the requested parameter. If not present, this is not a response to a query command.	
Checksum		16	0xFF minus the 8 bit sum of bytes from offset 3 to this byte	

AT command response

Frame Type: 0x88

In response to an AT Command message, the module will send an AT Command Response message. Some commands will send back multiple frames (for example, the AS (Active Scan) command).

Example: Suppose the BD parameter is changed on the local module with a frame ID of 0x01. If successful (parameter was valid), the response below would be received.

Frame fields	Offset	Example	Description
Start delimiter	0	0x7E	

Frame fields		Offset	Example	Description
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x05	
API frame-specific data	API Frame Identifier	3	0x88	
	Frame ID	4	0x01	
	AT Command	MSB 5	0x42 (B)	Command Name - Two ASCII characters that identify the AT command
		LSB 6	0x44 (D)	
Status		0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter	
	Parameter Value	7		Register data in binary format. If the register was set, then this field is not returned, as in this example.
Checksum		8	0xF0	0xFF minus the 8 bit sum of bytes from offset 3 to this byte

Transmission status

Frame Type: 0x89

RF transmission status messages are sent from the module in response to transmission attempts.

Example: The following API frame is returned when a successful transmission occurs on an API transmission using frame ID 01.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x03	
API frame-specific data	API frame identifier	3	0x89	
	Frame ID	4	0x01	Identifies the frame for which status is being reported. This number corresponds with the Frame ID provided in the transmission. If that frame ID was 0, then this frame will not be generated.

Frame fields		Offset	Example	Description
API frame-specific data	Status	5	0x00	0x00 = Success 0x03 = Transmission was purged because it was attempted before stack was completely up. 0x04 = Physical error occurred on the interface with the Wi-Fi transceiver. 0x21 = TX64 transmission timed out awaiting an acknowledgment from the remote module. 0x32 = Resource Error. Either buffers or sockets were depleted, preventing a transmission from occurring. 0x74 = Message not sent because it was too long 0x76 = Attempt to create a client socket failed 0x77 = TCP connection to given IP address and port doesn't exist. Source port is non-zero so that a new connection is not attempted. 0x78 = Source port on a UDP transmission doesn't match a listening port on the transmitting module.
Checksum		6	0x75	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Note New transmission status codes may be added in future firmware releases.

Modem status

Frame Type: 0x8A

RF module status messages are sent from the module in response to specific conditions.

Example: The following API frame is returned when a module is powered on in API mode.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x02	
API frame-specific data	API Frame Identifier	3	0x8A	

Frame fields		Offset	Example	Description
	Status	4	0x00	0 = Hardware reset or power up 1 = Watchdog timer reset 2 = Joined 3 = No longer joined to access point 0x0E = Device Cloud connected 0x0F = Device Cloud disconnected
Checksum		5	0x75	0xFF minus the 8 bit sum of bytes from offset 3 to this byte

Note New modem status codes may be added in future firmware releases.

ZigBee TX status

Frame Type: 0x8B

This frame type is only provided for software compatibility with other XBee modules. Frame type 0x89 is normally sent in response to transmissions. This frame type is sent in response to ZigBee (0x10) and ZigBee explicit (0x11) transmissions. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x07	
API frame-specific data	API frame identifier	3	0x8B	
	Frame ID	4	0x01	Frame ID of the correlating transmission
	Reserved	5	0xFF	Hard coded values can be ignored
		6	0xFE	
	7	0x00		

Frame fields		Offset	Example	Description
API frame-specific data	Status	8	0x00	0x00 = Success 0x03 = Transmission was purged because it was attempted before stack was completely up. 0x04 = Physical error occurred on the interface with the Wi-Fi transceiver. 0x21 = Transmission timed out awaiting an acknowledgment from the remote device. 0x32 = Resource Error; Either buffers or sockets were depleted, preventing a transmission from occurring. 0x74 = Message not sent because it was too long 0x76 = Attempt to create a client socket failed
	Reserved	9	0x00	Hard coded value can be ignored
Checksum		10	0x76	Add this value to sum of bytes from byte 3 to here such that result = 0xff.

IO data sample RX indicator

Frame Type: 0x8F

When the module receives an IO sample frame from a remote module, it sends the sample out the UART or SPI using this frame type. Only modules running API mode will be able to receive IO samples.

Example: The following is the IO sample response from a radio at IP address 192.168.0.103 reporting one active DIO (DIO8) and one active analog input (AN1).

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x13	
API frame-specific data	API frame identifier	3	0x8F	
	64-bit source address	4	0x00	Align IP address to low 32-bits of the field. The other bytes set to 0. IP address is in hex. The example uses address 192.168.0.103
		5	0x00	
		6	0x00	

Frame fields		Offset	Example	Description
API frame-specific data	64-bit source address	7	0x00	
		8	0xC0	
		9	0xA8	
		10	0x00	
		11	0x67	
	RSSI in terms of link margin	12	0x2E	
	Receive options	13	0x00	None currently defined
	Number of samples	14	0x01	Number of sample sets included in the payload. (Always set to 1)
	Digital Channel Mask*	MSB 15	0x01	Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any). In this example DIO8 is active.
		LSB 16	0x00	
	Analog Channel Mask**	17	0x81	Bitmask field that indicates which analog IO lines on the remote have sampling enabled (if any). The most significant bit signals that the Vcc value is included in the frame. In this example Analog input 1 and Vcc are active.
Digital Samples (if included)	MSB 18	0x00	If the sample set includes any digital IO lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital IO lines. DIO lines that do not have sampling enabled return 0. The bits in these 2 bytes map the same as they do in the Digital Channels Mask field. In this example, DIO8 has value 0.	
	LSB 19	0x00		
Analog Sample	MSB 20	0x03	If the sample set includes any analog input lines (Analog Channel Mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from DIO0/AD0 to DIO3/AD3.	
	LSB 21	0xB5		
Checksum		22	0x38	0xFF - 8 bit sum of bytes from offset 3 to this byte.

*	N/A	N/A	N/A	CD/DIO12	PWM/DIO11	RSSI/DIO10	N/A	N/A
	CTS/DIO7	RTS/DIO6	ASSOC DIO5	DIO4	AD3/DIO3	AD2/DIO2	AD1/DIO1	AD0/DIO0
**	Supply Voltage	N/A	N/A	N/A	AD3	AD2	AD1	AD0

ZigBee receive packet

Frame Type: 0x90

This frame type is used by XBee when RF data is received using the XBee application service and AO is set to 0. It is not generally used, but it allows for software compatibility with other XBee modules if desired. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x11	
API frame-specific data	API frame identifier	3	0x90	
	64-bit source address	4	0x00	Align IP address to low 32-bits of the field. The other bytes are set to 0. IP address is in hex. This example uses address 192.168.0.103
		5	0x00	
		6	0x00	
		7	0x00	
		8	0xC0	
		9	0xA8	
		10	0x00	
		11	0x67	
	Reserved	12	0xFF	Unused placeholder
		13	0xFE	
	Options	14	0x00	Bit 1: Broadcast packet
	RF Data	15	0x48 'H'	Up to 1392 bytes of data
		16	0x65 'e'	
		17	0x6C 'l'	
18		0x6C 'l'		
19		0x6F 'o'		
Checksum		20	0xAF	Add this value to sum of bytes from byte 3 to here such that result = 0xff

Explicit ZigBee receive packet

Frame Type: 0x91

This frame type is used by XBee when RF data is received using the XBee application service and AO is set to 1. Even when AO is not 1, this frame is also used for GPM response frames, see [Sleep](#) on page 51. See the following table for an example of this frame type.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x19	
API frame-specific data	API Frame Identifier	3	0x91	
	64-bit source address	4	0x00	Align IP address to low 32-bits of the field. The other bytes are set to 0. IP address is in hex. This example uses address 192.168.0.103
		5	0x00	
		6	0x00	
		7	0x00	
		8	0xC0	
		9	0xA8	
		10	0x00	
		11	0x67	
	Reserved	12	0xFF	Unused placeholder
		13	0xFE	
	Source endpoint	14	0xE6	Digi device object endpoint
	Destination endpoint	15	0xE6	Digi device object endpoint
	Cluster ID	16	0x00	Memory access cluster ID
		17	0x23	
	Profile ID	18	0xC1	Digi profile ID
		19	0x05	
Options	20	0x00		

Frame fields		Offset	Example	Description
API frame-specific data	RF data	21	0x80	Response to platform info request, indicating 160 GPM blocks available. Each block size is 4096 bytes.
		22	0x00	
		23	0xA0	
		24	0x10	
		25	0x00	
		26	0x00	Response to platform info request, indicating 160 GPM blocks available. Each block size is 4096 bytes.
		27	0x00	
Checksum		28	0xBD	Add this value to sum of bytes from byte 3 to here such that result = 0xff.

ZigBee remote AT command response

Frame Type: 0x97

This frame type is only provided for software compatibility with other XBee modules. It is used to generate a response to the ZigBee Remote AT Command (0x17). Normally, Remote AT command (0x07) is used instead with a remote command response of 0x87. An example of this frame type is given below:

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x0F	
API frame-specific data	API frame identifier	3	0x97	
	Frame ID	4	0x01	
	64-bit source address	5	0x00	The address of the remote radio returning this response. Align IP address to low 32-bits of the field. The other bytes set to 0. Value is in hex. In this example the IP address is 192.168.0.103
		6	0x00	
		7	0x00	
		8	0x00	
		9	0xC0	
		10	0xA8	
		11	0x00	
		12	0x67	

Frame fields		Offset	Example	Description
API frame-specific data	Reserved	13	0xFF	Unused placeholder
		14	0xFE	
	AT command	15	0x44 'D'	Name of the Command
		16	0x4C 'L'	
	Command status	17	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
Command data			If present, indicates the value of the requested parameter value. If not present, this is not a response to a query command.	
Checksum		18	0x0B	Add this value to sum of bytes from byte 3 to here such that result = 0xff.

RX (Receive) packet: IPv4

Frame Type: 0xB0

This frame is used by XBee when RF data is received using the Serial Data service on the port defined by the C0 command.

Example: When a module in API mode receives an IPv4 transmission, it will produce an RX notification (0xB0) and send it out the UART or SPI. This example is the response to a UDP transmission to IP address 192.168.0.103 with data 'Hello' from the source address 192.168.0.104.

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
API frame-specific data	API frame identifier	3	0xB0	
	IPv4 32-bit source address	MSB 4	0xC0	The address in the example is for a source address of 192.168.0.104
		5	0xA8	
		6	0x00	
		7	0x68	
	16-bit destination port	MSB 8	0x26	Same value as the C0 command.
LSB 9		0x16		

Frame fields		Offset	Example	Description
API frame-specific data	16-bit source port	MSB 10	0x26	
		LSB 11	0x16	
	Protocol	MSB 12	0x00	0 = UDP, 1= TCP - Protocol use for the transmitted data
	Status	13	0x00	Reserved
	RF data	14	0x48 'H'	Up to 1400 bytes of data. This is 8 bytes more than the max size reported by NP command because no application header is needed.
		15	0x65 'e'	
16		0x6C 'l'		
17		0x6C 'l'		
18	0x6F 'o'			
Checksum		19	0x13	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Send data response

Frame Type: 0xB8

This frame type is sent out the serial port in response to the send data request, providing its frame ID is non-zero.

Frame fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		1-2	0x0003	Number of bytes between the length and the checksum
API frame-specific data	API frame identifier	3	0xB8	
	Frame ID	4	0x55	Identifies the frame ID of the corresponding send data request.
	Status	5	0x00	0x00= Success 0x01= Bad Request 0x02= Response unavailable 0x03= Device Cloud Error 0x40= Unknown Error
Checksum		6	0xF2	0xFF minus 8-bit sum of bytes 3-5 of this frame.

Device request

Frame Type: 0xB9

This frame type is sent out the serial port when the XBee module receives a valid device request from Device Cloud.

Frame fields		Offset	Example	Description
Start		0	0x7E	
Length		1-2	0x0026	Number of bytes between the length and the checksum
API frame-specific data	API frame identifier	3	0xB9	
	Device request ID	4	0x01	Identifies the device request. (If 0, then no response is required.)
	Transport	5	0x00	Placeholders. Values can be ignored.
	Flags	6	0x00	
	Target length	7	0x08	Length of target string
	Target string	8-15	myTarget	String required by the host side, e.g. a file name.
Data	16-40	A message for serial host		
Checksum		41	0xC6	0xFF minus the 8 bit sum of bytes 3-40 of this frame. this frame.

Device response status

Frame Type: 0xBA

This frame type is sent to the serial port after the serial port sends a device response (frame type 0x2A).

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		1-2	0x0003	Number of bytes between the length and the checksum

Frame fields		Offset	Example	Description
API frame-specific data	API frame identifier	3	0xBA	
	Frame ID	4	0x01	Identifies the frame for which status is being reported. Corresponds to the frame ID in the device response.
	Status	5	0x00	0x00 = Success 0x20 = Device Request canceled by user 0x21 = Session timed out 0x40 = Unknown Error
Checksum		6	0x44	0xFF minus the 8 bit sum of bytes 3-5 of this frame.

Frame error

Frame Type: 0xFE

This frame type is sent to the serial port for any type of frame error.

Note This frame may be sent out the serial port in addition to a TX status response (frame type 0x2A).

Frame fields		Offset	Example	Description
Start delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x02	
API frame-specific data	API frame identifier	3	0xFE	
	Status	4	0x07	0x02 = Invalid frame type. 0x03 = Invalid frame length. 0x04 = Erroneous Checksum on last frame. 0x05 = payload of last API frame was too big to fit into a buffer. 0x06 = string entry was too big on last API frame sent. 0x07 = Wrong state to receive frame (e.g. a device response was sent out without first receiving a device request.) 0x08 = Device request ID of device response didn't match the number in the request.
Checksum		6	0xFA	0xFF minus the 8 bit sum of bytes 3-4 of this frame.

Command reference tables

Addressing

AT command	Name and description	Parameter range	Default
EQ	Device Cloud Server FQDN. Set / Read fully qualified domain name of Device Cloud server	Valid FQDN (fully qualified domain name). May be up to 64 characters long.	https://devicecloud.digi.com
LA	Lookup IP Address of FQDN. Perform a DNS lookup of the given FQDN and output its IP address. When the command is issued in API mode, the IP address is formatted in binary. In all other cases (e.g. command mode), the format is dotted decimal notation.	Valid FQDN. May be up to 64 characters long.	-
PG	Ping an IP address. Ping given IP address and indicate the response time or an error indication on failure. Response will always be a string.	Valid IPv4 address in dotted decimal notation either dotted decimal notation or binary format.	-
NS	DNS Address. Set / Read address of DNS server. When reading API mode, the format is binary. In all other cases (e.g. command mode), the read format is dotted decimal notation.	Valid IPv4 address in dotted decimal notation	208.67.222.22 (address of openDNS)
DL	Destination Address Low. Set / Get the 32 bits of the IPv4 destination address. When setting, the format may be either dotted decimal (e.g. 192.168.0.100) or binary (e.g. C0A80064). When reading in API mode, the format is binary. In all other cases (e.g. command mode), the read format is dotted decimal notation.	0.0.0.0 – 255.255.255.255	255.255.255.255

AT command	Name and description	Parameter range	Default
MY	IP Network Address. Read the 32-bit network address of the module when using DHCP. Set / Read values when using static IP address. When setting, the format may be either dotted decimal (e.g. 192.168.0.100) or binary (e.g. C0A80064). When reading in API mode, the format is binary. In all other cases (e.g. command mode) the read format is dotted decimal notation.	0.0.0.0 – 255.255.255.255	0.0.0.0
MK	IP Address Mask. This command is read-only when DHCP is enabled and it is read / write when using static IP addresses. When setting, the format may be either dotted decimal (e.g. 255.255.255.0) or binary (e.g. FFFFFFF0). When reading in API mode, the format is binary. In all other cases (e.g. command mode), the read format is dotted decimal notation.	0.0.0.0 – 255.255.255.255	0.0.0.0
GW	Gateway IP address. This command is read-only when DHCP is enabled and it is read / write when using static IP addresses. When setting, the format may be either dotted decimal (e.g. 192.168.0.1) or binary (e.g C0A80001). When reading in API mode, the format is binary. In all other cases (e.g. command mode), the read format is dotted decimal notation.	0.0.0.0 – 255.255.255.255	0.0.0.0
SH	Serial Number High. Read the high 16 bits of the module's unique 48-bit address.	0 - 0xFFFFFFFF [read-only]	factory-set
SL	Serial Number Low. Read the low 32 bits of the module's unique 48-bit address.	0 - 0xFFFFFFFF [read-only]	factory-set
NI	Node Identifier. Stores a string identifier. The register only accepts printable ASCII data. In AT Command Mode, a string cannot start with a space. A carriage return ends the command. Command will automatically end when maximum bytes for the string have been entered.	20-Byte printable ASCII string	ASCII space character (0x20)
DE	Destination Port. Set / Get destination UDP/TCP port value.	0 - 0xFFFF	0x2616
KP	Device Cloud Descriptor. Description of the module that is displayed on Device Cloud	Up to 20 ASCII characters	ASCII space character (0x20)
KC	Device Cloud Contact. Contact information for the module that is displayed on Device Cloud	Up to 20 ASCII characters	ASCII space character (0x20)
KL	Device Cloud Location. Location of the module that is displayed on Device Cloud	Up to 20 ASCII characters	ASCII space character (0x20)

AT command	Name and description	Parameter range	Default
C0	Serial Communication Service Port. Set / Get port number used to provide the serial communication service. Data sent to this port comes from the serial port of the module. The protocol used is set by the IP command when UART is in transparent mode.	0 – 0xFFFF	0x2616
DD	Device Type Identifier. Stores a device type value. This value can be used to differentiate different XBee-based devices. Digi reserves the range 0 - 0xFFFFFFFF.	0-0xFFFFFFFF	0x90000
NP	Maximum RF Payload Bytes. This value returns the maximum number of RF payload bytes that can be sent in a transmission Note: NP returns a hexadecimal value. (e.g. if NP returns 0x54, this is equivalent to 84 bytes). Also note that the maximum payload is 8 bytes more than the value in the NP parameter, when using the native IPv4 frames, because an application header does not precede the payload.	0 - 0xFFFF	[read-only]

Networking Commands

AT command	Name and description	Parameter range	Default
DO	<p>Device options. Set/Read device options</p> <p>Bit 0 – Enable Device Cloud</p> <p>Bit 1 – Enable SoftAP when ID is NULL</p> <p>Bit 2 – Enable sending transparent data to Device Cloud</p> <p>Bit 3 – Send I/O samples to both Device Cloud and to DL if Device Cloud is enabled.</p> <p>Bit 4 – Send transparent data as binary data points rather than to a file.</p> <p>Bit 5 – Replace a Device Cloud file (1) rather than append to a file (0).</p> <p>Bits 6, 7- Reserved; should be 0.</p> <p>Note In transparent mode, if DO is 0x25 and then over 1400 bytes are sent at once, the Device Cloud file is overwritten twice, losing the first 1400 bytes.</p>	0x03F	1
ID	SSID. Set/read the SSID of the access point, which may be up to 31 ASCII characters	Up to 31 bytes of printable ASCII	NULL
AH	Network Type. Set/read network type. Network types supported are Infrastructure (using an access point) and Ad hoc (IBSS).	0-IBSS Joiner 1-IBSS Creator 2 - Infrastructure	2
IP	IP Protocol. Set / Read the protocol used for the serial communication service. This is the port used by the C0 command.	0 – UDP 1 - TCP	0
MA	IP Addressing Mode. Set / Read the IP addressing mode.	0 – DHCP 1 – Static	0
TM	TCP timeout. Set / Read the timeout for connection on TCP client sockets. If 0, socket closes immediately after data sent.	0-0xFFFF [x 100 msec]	0x64
TS	TCP Server Socket Timeout. Set / Read the timeout for connection on a TCP server socket. This is a socket whose connection was initiated at the other end.	0 x000A– 0xFFFF * 100 ms.	0x0258 (1 minute)
CE	Infrastructure Mode. Selects AP mode (1) or STA mode (2). For more information, see Enable Soft AP mode on page 60.	1 - Soft AP Mode 2 - STA Mode	2

Security commands

AT command	Name and description	Parameter range	Default
EE	Encryption Enable. Set/Read the encryption enable setting.	0 – No security 1 – WPA 2 – WPA2 3 – WEP	0
PK	Security Key. Set the security key used for WEP, WPA, and WPA2 security. This command is write only; PK cannot be read. Note: The PK parameter cannot include a comma because a comma is used to signify the start of the next command when sending multiple commands at a time when in command mode. A comma can be included when PK is set through an API frame.	8-63 ASCII or 64 hexadigit characters for WPA and WPA2. WEP keys can be either 40 bits or 104 bits. 40-bit WEP keys are entered with 5 ASCII characters or 10 hex characters. 104-bit WEP keys are entered with 13 ASCII characters or 26 hex characters.	

RF interfacing commands

AT command	Name and description	Parameter range	Default
PL	Power Level. Select/Read the power level at which the RF module transmits conducted power.	0 – 0 dBm 1 – 5 dBm 2 – 10 dBm 3 – 15 dBm 4 – Max power	4
CH	Channel. Read the channel number of the access point or 0xFF if not associated. Channel can be set when AH is configured for Ad hoc creator mode. Note: When using Ad hoc mode, not all channels are available in all countries. It is the responsibility of the installer to use the appropriate channels.	1-0xB	[read only]

Serial interfacing

AT command	Name and description	Parameter range	Default
AP	API Enable. Enable API Mode.	0 = Transparent mode 1 = API-enabled 2 = API-enabled (with escaped control characters)	1
AO	API Output Options. Indicates the type of frame to output when data is received on the IP services port	0=ZigBee Rx 1=Explicit ZigBee Rx 2=RX64	2 (RX64)
BD	Interface Data Rate. Set / Read the serial interface data rate for communication between the module serial port and host. Any value above 0x0A is interpreted as an actual baud rate. When a value above 0x0A is sent, the closest interface data rate represented by the number is stored in the BD register.	1 - 7 (standard baud rates) 1 = 2400 b/s 2 = 4800 3 = 9600 4 = 19200 5 = 38400 6 = 57600 7 = 115200 8 = 230400 9 = 460,800 0xA = 921,600 0X5B9 - 0X5B8D80 (non-standard rates up to 6 Mb/s)	3
NB	Serial Parity. Set / Read the serial parity setting on the module.	0 = No parity 1 = Even parity 2 = Odd parity	0
SB	Stop Bits. Set / Read the number of stop bits for the UART.	0 = 1 stop bit 1 = 2 stop bits	0
RO	Packetization Timeout. Set / Read number of character times of inter-character silence required before packetization. Set (RO=0) to transmit characters as they arrive instead of buffering them into one RF packet. Regardless of how small RO is, the inter-character silence required to trigger a transmission of the data is 100 usec.	0 - 0xFF [x character times]	3

AT command	Name and description	Parameter range	Default
FT	Flow Control Threshold. De-assert CTS when FT bytes are in the UART receive buffer	0x11 – 0x823	0x7F3
D7	DIO7 Configuration. Select/Read options for the DIO7 line of the RF module.	0 = Disabled 1 = CTS Flow Control 3 = Digital input 4 = Digital output, low 5 = Digital output, high 6 = RS-485 transmit enable (low enable) 7 = RS-485 transmit enable (high enable)	1
D6	DIO6 Configuration. Configure options for the DIO6 line of the RF module.	0 = Disabled 1 = RTS flow control 3 = Digital input 4 = Digital output, low 5 = Digital output, high	0

I/O settings

AT command	Name and description	Parameter range	Default
IS	Force Sample. Forces a read of all enabled digital and analog input lines. If no lines are enabled for digital or analog input, an error is returned.	-	-
IR	IO Sample Rate. Set / Read the IO sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital IO functionality enabled (see the Command reference tables on page 109). The sample rate is measured in milliseconds. WARNING: If IR is set to 1 or 2, the module will not keep up and many samples will be lost.	0-0xFFFF (x 1 ms)	0 – no sampling
IC	IO Digital Change Detection. Set / Read the digital IO pins to monitor for changes in the IO state. IC works with the individual pin configuration commands (D0-D9, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate IO sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0.	0 - 0xFFFF	0
IF	Sample from Sleep Rate. The number of sleep cycles that must elapse between periodic I/O samples. This allows I/O samples to be taken only during some wake cycles. During those cycles I/O samples are taken at the rate specified by IR. IR can be 0, which causes only one sample to be taken.	1-0xFF (1 gives you a sample every sleep cycle)	1
P0	DIO10 Configuration. Select / Read function for the DIO10 line of the RF module.	0 = Disabled, 1 = PWM RSSI Output 2 = PWM0 Output 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	1
P1	DIO11 Configuration. Select / Read function for the DIO11 line of the RF module.	0 = Disabled 2 = PWM1 Output 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	0

AT command	Name and description	Parameter range	Default
P2	DIO12 Configuration. Select / Read function for the DIO12 line of the RF module.	0 = Disabled 1 = SPI_MISO* 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high, 6 = TCP connection indicator	0
P3	DOUT. Enables or disables output on UART port	0 = Disabled 1 = Enabled	1
P4	DIN. Enables or disables input on UART port	0 = Disabled 1 = Enabled	1
P5**	DIO15 Configuration. Select / Read function for the DIO15 line of the RF module.	0 = Disabled 1 = SPI_MISO 4 = Digital output, default low 5 = Digital output, default high	1
P6**	DIO16 Configuration. Select / Read function for the DIO16 line of the RF module.	0 = Disabled 1 = SPI_MOSI 4 = Digital output, default low 5 = Digital output, default high	1
P7**	DIO17 Configuration. Select/Read function for the DIO17 line of the RF module.	0 = Disabled 1 = SPI_nSSEL 4 = Digital output, default low 5 = Digital output, default high	1
P8**	DIO18 Configuration. Select/Read function for the DIO18 line of the RF module.	0 = Disabled 1 = SPI_CLK 4 = Digital output, default low 5 = Digital output, default high	1
P9**	DIO19 Configuration. Select/Read function for the DIO19 line of the RF module.	0 = Disabled 1 = SPI_nATTN 4 = Digital output, default low 5 = Digital output, default high 6 = UART data present indicator	1

AT command	Name and description	Parameter range	Default
D0	DIO0/AD0/ CB Configuration. Select/Read function for DIO0/AD0/ CB.	0 = Disabled 1 = Commissioning Button 2 = Analog input 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	1
D1	DIO1/AD1 Configuration. Select/Read function for DIO1/AD1	0 - Disabled 1 = SPI_nATTN* 2 = Analog input 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	1
D2	DIO2/AD2 Configuration. Select/Read function for DIO2/AD2	0 = Disabled 1 = SPI _CLK* 2 = Analog input 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	0
D3	DIO3/AD3 Configuration. Select/Read function for DIO3/AD3	0 = Disabled 1 = SPI Slave Select* 2 = Analog input 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	0
D4	DIO4 Configuration. Select/Read function for DIO4	0 = Disabled 1 = SPI_MOSI* 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	0

AT command	Name and description	Parameter range	Default
D5	DIO5 Configuration. Select/Read function for DIO5	0 = Disabled 1 = Associated LED 3 = Digital input 4 = Digital output, default low 5 = Digital output, default high	1
D8	DIO8 Configuration. Select/Read function for DIO8	0 = Disabled 1 = SleepRq 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	1
D9	DIO9 Configuration. Select/Read function for DIO9	0 = Disabled 1 = On/Sleep indicator 3 = Digital input, monitored 4 = Digital output, default low 5 = Digital output, default high	1
LT	Assoc LED Blink Time. Set / Read the Associate LED blink time. If the Associate LED functionality is enabled (D5 command), this value determines the on and off blink times for the LED when the module has joined a network. If LT = 0, the default blink rate of 250ms is used. For all other LT values, LT is measured in 10ms.	0, 0x14 - 0xFF (200 - 2550 ms)	0

AT command	Name and description	Parameter range	Default
PR	<p>Pull-up Resistor. Set / Read the bit field that configures the internal resistor status for the digital input lines. Note that internal pull-up/down resistors are not available for digital output pins, analog input pins, or for disabled pins.</p> <p>"1" specifies the resistor is enabled. "0" specifies no resistor.</p> <p>The PD command specifies whether the resistor is pull-up or pull-down. Pin numbers are listed with the TH module pin first, followed by the SMT module pin (e.g. pin 11/24 indicates pin 11 on the through-hole module and pin 24 on the surface mount module.)</p> <p>Bits:</p> <ul style="list-style-type: none"> 0 – DIO4 (Pin 11/24) 1 – DIO3 / AD3 (Pin 17/30) 2 – DIO2 / AD2 (Pin 18/31) 3 – DIO1/AD1 (Pin 19/32) 4 – DIO0 / AD0 (Pin 20/33) 5 – DIO6 / RTS (Pin 16/29) 6 – DIO8 / nDTR / Sleep Request (Pin 9/10) 7 – DIN / Config (Pin 3/4) 8 – DIO5 / Associate (Pin 15/28) 9 – DIO9 / On/Sleep (Pin 13/26) 10 – DIO12 (Pin 4/5) 11 – DIO10 / PWM RSSI/PWM0 (Pin 6/7) 12 – DIO11/PWM1 (Pin 7/8) 13 – DIO7 / CTS (Pin 12/25) 14 – DIO13/DOUT (pin2/3) 15 – DIO15 (pin NA/17) 16 – DIO16 (Pin NA/16) 17 – DIO17 (Pin NA/15) 18 – DIO18 (Pin NA/14) 19 – DIO19 (Pin NA/12) 	<p>0 - 0x7FFF (TH)</p> <p>0-0xFFFFF (SMT)</p>	<p>0x7FFF (TH)</p> <p>0xFFFFF (SMT)</p>
PD	<p>Pull Direction. Set / Read resistor direction for the corresponding bits set in PR (1 = pull up, 0 = pull down). If the bit is not set in PR, then PD is unused. Note that resistors are not applied to disabled lines.</p>	<p>0 – 0x7FFF on TH</p> <p>0 – 0xFFFFF on SMT</p>	<p>0x7FFF on TH</p> <p>0xFFFFF on SMT</p>
DS	<p>Drive Strength. Set/Read the output drive strength (output amperes) for DIO lines. Bits are mapped the same as the PR and PD commands. If the bit is set, the drive strength is 6mA. Otherwise, it is 2mA.</p>	<p>0 – 0x7FFF on TH</p> <p>0 – 0xFFFFF on SMT</p>	<p>0</p>

AT command	Name and description	Parameter range	Default
AV	Analog Voltage Reference. Set/Read the analog voltage reference. This specifies the volts for an analog reading of 0x03ff, where a reading of 0x200 indicates a voltage input that is half of VREF. VREF may be one of these two values: 0 – 1.25Volts 1 – 2.5 Volts	0-1	1
M0	PWM0 Duty cycle. Sets the duty cycle of PWM0 for P0=2, where a value of 0x200 is a 50% duty cycle.	0 – 0x03FF	0
M1	PWM1 Duty cycle. Sets the duty cycle of PWM1 for P1=2, where a value of 0x200 is a 50% duty cycle.	0 – 0x03FF	0

* indicates that the option is available on the TH module, but not the SMT module

** indicates that the command is available on the SMT module, but not the TH module

Output Control

AT command	Name and description	Parameter range	Default
IO	Set Output Pins. Set output pins to the designated level. Bit 0 corresponds with DIO0, bit 1 with DIO1, up to bit 19 that corresponds with DIO19. See Output control on page 73 for a functional description.	0 to 0x7fff on TH 0 to 0xffff on SMT	-
OM	Output Mask. Sets the output mask for the IO command. If a bit is set, then the corresponding bit in the IO command is enabled. If it is clear, then that same bit has no effect in the IO command.	0 to 0x7fff on TH 0 to 0xffff on SMT	0x7fff on TH 0xffff on SMT
T0	Set time to hold DIO0. Sets how long an output level programmed by bit 0 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T1	Set time to hold DIO1. Sets how long an output level programmed by bit 1 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T2	Set time to hold DIO2. Sets how long an output level programmed by bit 2 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0

AT command	Name and description	Parameter range	Default
T3	Set time to hold DIO3. Sets how long an output level programmed by bit 3 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T4	Set time to hold DIO4. Sets how long an output level programmed by bit 4 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T5	Set time to hold DIO5. Sets how long an output level programmed by bit 5 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T6	Set time to hold DIO6. Sets how long an output level programmed by bit 6 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T7	Set time to hold DIO7. Sets how long an output level programmed by bit 7 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T8	Set time to hold DIO8. Sets how long an output level programmed by bit 8 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
T9	Set time to hold DIO9. Sets how long an output level programmed by bit 9 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x100 ms)	0
Q0	Set time to hold DIO10. Sets how long an output level programmed by bit 10 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q1	Set time to hold DIO11. Sets how long an output level programmed by bit 11 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0

AT command	Name and description	Parameter range	Default
Q2	Set time to hold DIO12. Sets how long an output level programmed by bit 12 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q3	Set time to hold DIO13. Sets how long an output level programmed by bit 13 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q4	Set time to hold DIO14. Sets how long an output level programmed by bit 14 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q5**	Set time to hold DIO15. Sets how long an output level programmed by bit 15 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q6**	Set time to hold DIO16. Sets how long an output level programmed by bit 16 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q7**	Set time to hold DIO17. Sets how long an output level programmed by bit 17 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q8**	Set time to hold DIO18. Sets how long an output level programmed by bit 18 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0
Q9**	Set time to hold DIO19. Sets how long an output level programmed by bit 19 of the IO command is held in the selected state before reverting to its configured level. See Output control on page 73 for a functional description.	0 – 0x1770 (x 100 ms)	0

** indicates that the option is available on the SMT module, but not the TH module.

Diagnostics interfacing

AT command	Name and description	Parameter range	Default
VR	<p>Firmware Version. Read firmware version for the module.</p> <p>The firmware version returns 4 hexadecimal values (2 bytes) "ABCD". Digits ABC are the main release number and D is the revision number from the main release. "B" is a variant designator where 0 means standard release.</p>	0 - 0xFFFF [read-only]	Factory-set
HV	<p>Hardware Version. Read the hardware version for the module.</p> <p>This command can be used to distinguish among different hardware platforms. The upper byte returns a value that is unique to each module type. The lower byte indicates the hardware revision.</p> <p>XBee Wi-Fi modules return 0x1Fxx for the HV command.</p>	0 - 0xFFFF [read-only]	Factory-set
HS	<p>Hardware Series. Indicates the hardware series number of the module. This module should indicate 0x601 for S6B.</p>		
AI	<p>Association Indication. Read information regarding last node join request:</p> <p>0x00 - Successfully joined an access point, established IP addresses and IP listening sockets</p> <p>0x01 - Wi-Fi transceiver initialization in progress.</p> <p>0x02 - Wi-Fi transceiver initialized, but not yet scanning for access point</p> <p>0x13 - Disconnecting from access point</p> <p>0x23 - SSID not configured</p> <p>0x24 - Encryption key invalid (either NULL or invalid length for WEP)</p> <p>0x27 - SSID was found, but join failed. 0x40- Waiting for WPA or WPA2 Authentication</p> <p>0x41 - Module joined a network and is waiting for IP configuration to complete, which usually means it is waiting for a DHCP provided address</p> <p>0x42 - Module is joined, IP is configured, and listening sockets are being set up.</p> <p>0xFF - Module is currently scanning for the configured SSID.</p> <p>Note: New non-zero AI values may be added in later firmware versions. Applications should read AI until it returns 0x00, indicating a successful startup.</p>	0 - 0xFF [read-only]	-

AT command	Name and description	Parameter range	Default
DI	<p>Device Cloud indicator</p> <p>0 = Device cloud is connected</p> <p>1 = Initial state</p> <p>2 = Attempting to connect to Device Cloud</p> <p>3 = Disconnecting from Device Cloud</p> <p>4 = Device cloud not configured</p>	0 – 4	
AS	<p>Active Scan. Scan for access points in the vicinity.</p> <p>This command may be issued in command mode or in API mode. In either case, the following information is returned for each access point found:</p> <p>02 - Indicates scan type of 802.11 in this format unique to S6B</p> <p>CH - Channel number in use by access point</p> <p>ST - Security type where: 00=open, 01=WPA, 02=WPA2, and 03=WEP</p> <p>LM - Link Margin (Signal strength in dB above sensitivity)</p> <p>ID - SSID of access point found</p> <p>When this command is issued in command mode, the above record is displayed, one per line for each access point found. Readable ASCII characters are outputs with a carriage return and each field on a new line.</p> <p>When it is issued in API mode, each record (i.e. each access point) outputs a separate AT command response of type 0x88 with the above fields in binary format. The command will terminate with a null AT Command Response Packet.</p> <p>In AT command mode, the AS command will terminate with an additional carriage return.</p> <p>Note that this command is not available as a remote command. Also note that this command gives an error if associated to an access point. AI must be 0x23 for this command to work which may be achieved by first using the NR command.</p>	-	-
TP	<p>Temperature. Read temperature of module in degrees Celsius.</p>	-30 to 85C	-
CK	<p>Configuration Code. Read the configuration code associated with the current AT command configuration.</p>	2 bytes	-

AT command	Name and description	Parameter range	Default
%V	Supply Voltage. Read supply voltage in millivolt units.	3.1 to 3.5V	-
LM	Link Margin. Reads the received signal strength (RSSI) in terms of dB units above sensitivity; reports 0xff until the first reception after connection to access point.	0 – 0xFF	

AT command options

AT command	Name and description	Parameter range	Default
CT	Command Mode Timeout. Set / Read the period of inactivity (no valid commands received) after which the RF module automatically exits AT Command Mode and returns to Idle Mode. This time period can be up to ten minutes.	2 - 0x1770 [x 100 ms]	0x64 (100d)
CN	Exit Command Mode. Explicitly exit the module from AT Command Mode. Note: Whether command mode is exited using the CN command or by CT timing out, changes are applied upon exit.	-	-
GT	Guard Times. Set the required period of silence before and after the Command Sequence Characters of the AT Command Mode Sequence (GT + CC + GT). The period of silence is used to prevent inadvertent entrance into AT Command Mode.	2 - 0x0576 [x 1 ms] (max of 1.4 seconds)	0x3E8 (1000d)
CC	Command Mode Character. Set/read the command mode character used between guard times of the AT Command Mode Sequence (GT + CC + CC + CC + GT). This sequence allows the module to enter into AT Command Mode.	0 - 0xFF	0x2B (‘+’ ASCII)

Sleep commands

AT command	Name and description	Parameter range	Default
SM	Sleep Mode. Sets the sleep mode on the RF module. Sleep mode is also affected by the SO command, option bit 6. See Sleep on page 51 for a full explanation of the various sleep modes.	0 = No sleep 1 = Pin sleep 4 = Cyclic sleep 5 = Cyclic sleep, pin wake	0
SP	Sleep Period. This value determines how long the module will sleep at a time, up to 24 hours or 86,400 seconds. This corresponds to 0x83d600 in 10ms units.	1 - 0x83D600 x 10ms	0xC8 (2 seconds)

AT command	Name and description	Parameter range	Default
SO Command	<p>Sleep Options. Configure options for sleep. Unused option bits should be set to 0. Sleep options include:</p> <p>0x40 – Stay associated with AP during sleep. Draw more current during sleep with this option enabled, but also avoid data loss.</p> <p>0x100 – For cyclic sleep, ST specifies the time before returning to sleep. With this bit set, new receptions from either the serial or the RF port do not restart the ST timer. Current implementation does not support this bit being turned off.</p>	0 - 0x01FF	0x100
WH	<p>Wake Host. Set/Read the wake host timer value. If the wake host timer is set to a non-zero value, this timer specifies a time (in millisecond units) that the module should allow after waking from sleep before sending data out the UART or transmitting an IO sample. If serial characters are received, the WH timer is stopped immediately.</p>	0 - 0xFFFF (x 1ms)	0
ST	<p>Wake Time. Wake time for cyclic modes. New data does not refresh the timer. However, if there is data to transmit or receive after ST expires, those actions occur before the module goes to sleep. Max wake time is 3600 seconds.</p>	0x1 – 0x36EE80 (x 1 ms)	0x7D0
SA	<p>Association Timeout. Time to wait for association before entering deep sleep. (Wakeup from deep sleep is much faster if association occurs before going to sleep.)</p>	0x1 – 0x36EE80 (x1 ms)	0x2710 (10 seconds)

Execution commands

Where most AT commands set or query register values, execution commands cause an action to be executed on the module. Execution commands are executed immediately and do not require changes to be applied.

AT command	Name and description	Parameter range	Default
AC	<p>Apply Changes. Applies changes to all command registers causing queued command register values to be applied. For example, changing the serial interface rate with the BD command will not change the UART interface rate until changes are applied with the AC command. The CN command and 0x08 API command frame also apply changes.</p>	-	-
WR	<p>Write. Write parameter values to non-volatile memory so that parameter modifications persist through subsequent resets. Note: Once WR is issued, no additional characters should be sent to the module until after the "OK\r" response is received. The WR command should be used sparingly to preserve flash.</p>	-	-
RE	<p>Restore Defaults. Restore module parameters to factory defaults.</p>	-	-
FR	<p>Software Reset. Reset module. Responds immediately with an OK status, and then performs a software reset about 2 seconds later.</p>	-	-

AT command	Name and description	Parameter range	Default
NR	<p>Network Reset. Reset network layer. For Wi-Fi, this means to disassociate from the access point and set SSID to NULL, thereby preventing the node from immediately establishing the same connection with the same access point. This command also clears security settings (EE and PK).</p> <p>Note that NR and NR0 both do the same thing and may be used interchangeably.</p>	0	-
CB	<p>Commissioning Button.</p> <p>2 = WPS push button configuration</p> <p>4 = Force Provisioning in Soft AP mode by issuing an NR command.</p> <p>See Commissioning button on page 62 for more details.</p>	2, 4	-

Module support

This section provides customization information for the XBee Wi-Fi module. In addition to providing an extremely flexible and powerful API, the XBee module is a robust development platform that has passed FCC and European testing.

XCTU configuration tool

Digi provides the XCTU tool for configuring module parameters and updating firmware. The XCTU tool has the capability to do the following:

- Update firmware on a local module (requires USB or serial connection)
- Read or write module configuration parameters on a local
- Save and load configuration profiles containing customized settings.
- Contact Digi support for more information about the XCTU.

Serial firmware updates

Serial firmware updates make use of the XBee bootloader which ships in all modules. This bootloader allows firmware to be updated. Normally, the running application can be told to invoke the bootloader through a command from XCTU. If that command is not available in the currently loaded firmware, the bootloader includes a modified entry mechanism using pins 3, 9, and 16 (DIN, nDTR, and nRTS, respectively). By driving DIN low, nDTR low, and nRTS high at the time the module is reset, the XBee bootloader is forced to run, allowing a new version of firmware to load. This method works even when the current firmware version does not support the firmware upgrade feature.

The XCTU program can update firmware on the XBee module over the UART port, but not currently over the SPI port. Contact Digi support for details.

Regulatory compliance

XBee modules are certified for FCC and IC operation on all 11 channels (1-11) allowable, and European certified for all European 13 channels (1-13) allowable.

Agency certifications

United States FCC

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference and (2) this device must accept any interference received, including interference that may cause undesired operation.

The XBee Wi-Fi Module complies with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

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

1. The system integrator must ensure that the text on the module label is placed on the outside of the final product.
2. XBee Wi-Fi Module may only be used with antennas that have been tested and approved for use with this module (for details, see [Antennas approved for use with the XBee Wi-Fi Through-hole Module](#) on page 131 and [Antennas approved for use with the XBee Wi-Fi Surface Mount Module](#) on page 135).

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.

Required FCC Label for OEM products containing the XBee Wi-Fi S6B Through-hole Module:

Contains FCC ID: MCQ-XBS6B

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.

Required FCC Label for OEM products containing the XBee Wi-Fi S6B Surface Mount Module:

Contains FCC ID: MCQ-S6BSM

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 integrator is responsible for its product to comply with FCC Part 15, Sub. B - Unintentional Radiators.

FCC Notices

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

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

IMPORTANT: The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, the module must undergo SAR testing.

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

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Re-orient or relocate the receiving antenna, Increase the separation between the equipment and receiver, Connect equipment and receiver to outlets on different circuits, or Consult the dealer or an experienced radio/TV technician for help.

FCC-Approved Antennas (2.4 GHz)

The XBee Wi-Fi Module can be installed utilizing antennas and cables constructed with non-standard connectors (RPSMA, RPTNC, etc.). An adapter cable may be necessary to attach the XBee connector to the antenna connector.

The modules are FCC approved for fixed base station and mobile applications. If the antenna is mounted at least 20cm (8 in.) from nearby persons, the application is considered a mobile application. Antennas not listed in the table must be tested to comply with FCC Section 15.203 (Unique Antenna Connectors) and Section 15.247 (Emissions). XBee Wi-Fi RF Modules have been approved for use with all the antennas listed in the tables below. (Cable-loss is required when using gain antennas as shown below.) Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

Antennas approved for use with the XBee Wi-Fi Through-hole Module

Integrated antennas							
Part Number	Type (Description)	Gain	Application	Min Separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
29000294	Integral PCB antenna	-0.5 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-QI	Monopole (Integrated Whip)	1.5 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A

Dipole antennas							
Part Number	Type (Description)	Gain	Application	Min Separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-HASM-450	Dipole (Half-wave articulated RPSMA-4.5")	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-HABSM	Dipole (Articulated RPSMA)	2.1 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave bulkhead mount U.FL s/ 5" pigtail)	2.1 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA-5.25")	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A

Omni-directional antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-F2NF	Omni-Directional (Fiberglass base station)	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-F3NF	Omni-Directional (Fiberglass base station)	3.0 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-F5NF	Omni-Directional (Fiberglass base station)	5.0 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-F8NF	Omni-Directional (Fiberglass base station)	8.0 dBi	Fixed	2 m	N/A	0.4 dB	0.4 dB

Omni-directional antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-F9NF	Omni-Directional (Fiberglass base station)	9.5 dBi	Fixed	2 m	0.4 dB	2.4 dB	2.4 dB
A24-F10NF	Omni-Directional (Fiberglass base station)	10 dBi	Fixed	2 m	0.9 dB	2.9 dB	2.9 dB
A24-F12NF	Omni-Directional (Fiberglass base station)	12 dBi	Fixed	2 m	2.9 dB	4.9 dB	4.9 dB
A24-F15NF	Omni-Directional (Fiberglass base station)	15 dBi	Fixed	2 m	5.9 dB	7.9 dB	7.9 dB
A24-W7NF	Omni-Directional (base station)	7.2 dBi	Fixed	2 m	N/A	0.1 dB	0.1 dB
A24-M7NF	Omni-directional (Mag-mount base station)	7.2 dBi	Fixed	2 m	N/A	0.1 dB	0.1 dB

Panel class antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-P8SF	Flat Panel	8.5 dBi	Fixed	2 m	N/A	1.4 dB	1.4 dB
A24-P8NF	Flat Panel	8.5 dBi	Fixed	3 m	N/A	1.4 dB	1.4 dB
A24-P13NF	Flat Panel	13 dBi	Fixed	4 m	3.9 dB	5.9 dB	5.9 dB
A24-P14NF	Flat Panel	14 dBi	Fixed	5 m	4.9 dB	6.9 dB	6.9 dB
A24-P15NF	Flat Panel	15 dBi	Fixed	2 m	5.9 dB	7.9 dB	7.9 dB
A24-P16NF	Flat Panel	16 dBi	Fixed	2 m	6.9 dB	8.9 dB	8.9 dB
A24-19NF	Flat Panel	19 dBi	Fixed	2 m	9.9 dB	11.9 dB	11.9 dB

Yagi class antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-Y6NF	Yagi (6 element)	8.8 dBi	Fixed	2 m	N/A	1.7 dB	1.7 dB
A24-Y7NF	Yagi (7 element)	9.0 dBi	Fixed	2 m	N/A	1.9 dB	1.9 dB
A24-Y9NF	Yagi (9 element)	10.0 dBi	Fixed	2 m	0.9 dB	2.9 dB	2.9 dB
A24-Y10NF	Yagi (10 element)	11.0 dBi	Fixed	2 m	1.9 dB	3.9 dB	3.9 dB
A24-Y12NF	Yagi (12 element)	12.0 dBi	Fixed	2 m	2.9 dB	4.9 dB	4.9 dB
A24-Y13NF	Yagi (13 element)	12.0 dBi	Fixed	2 m	2.9 dB	4.9 dB	4.9 dB
A24-Y15NF	Yagi (15 element)	12.5 dBi	Fixed	2 m	3.4 dB	5.4 dB	5.4 dB
A24-Y16NF	Yagi (16 element)	13.5 dBi	Fixed	2 m	4.4 dB	6.4 dB	6.4 dB
A24-Y16RM	Yagi (16 element, RPSMA connector)	13.5 dBi	Fixed	2 m	4.4 dB	6.4 dB	6.4 dB
A24-Y18NF	Yagi (18 element)	15.0 dBi	Fixed	2 m	5.9 dB	7.9 dB	7.9 dB

Antennas approved for use with the XBee Wi-Fi Surface Mount Module

Integrated antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
31000005-01	Integral PCB antenna	0 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-QI	Monopole (Integrated Whip)	1.5 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A

Dipole antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-HASM-450	Dipole (Half-wave articulated RPSMA-4.5")	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-HABSM	Dipole (Articulated RPSMA)	2.1 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-HABUF-P5I	Dipole (Half-wave bulkhead mount U.FL s/ 5" pigtail)	2.1 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-HASM-525	Dipole (Half-wave articulated RPSMA-5.25")	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A

Omni-directional antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-F2NF	Omni-Directional (Fiberglass base station)	2.1 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-F3NF	Omni-Directional (Fiberglass base station)	3.0 dBi	Fixed/Mobile	20 cm	N/A	N/A	N/A
A24-F5NF	Omni-Directional (Fiberglass base station)	5.0 dBi	Fixed	20 cm	N/A	N/A	N/A
A24-F8NF	Omni-Directional (Fiberglass base station)	8.0 dBi	Fixed	2 m	N/A	N/A	1.5 dB
A24-F9NF	Omni-Directional (Fiberglass base station)	9.5 dBi	Fixed	2 m	N/A	1.5 dB	1.0 dB
A24-F10NF	Omni-Directional (Fiberglass base station)	10 dBi	Fixed	2 m	0.5 dB	2.0 dB	2.5 dB
A24-F12NF	Omni-Directional (Fiberglass base station)	12 dBi	Fixed	2 m	2.5 dB	4.0 dB	4.5 dB
A24-F15NF	Omni-Directional (Fiberglass base station)	15 dBi	Fixed	2 m	5.5 dB	7.0 dB	7.5 dB
A24-W7NF	Omni-Directional (base station)	7.2 dBi	Fixed	2 m	N/A	N/A	N/A
A24-M7NF	Omni-directional (Mag-mount base station)	7.2 dBi	Fixed	2 m	N/A	N/A	N/A

Panel class antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-P8SF	Flat Panel	8.5 dBi	Fixed	2 m	N/A	0.5 dB	0.9 dB
A24-P8NF	Flat Panel	8.5 dBi	Fixed	3 m	N/A	0.5 dB	0.9 dB
A24-P13NF	Flat Panel	13 dBi	Fixed	4 m	3.5dB	5.0 dB	5.5 dB
A24-P14NF	Flat Panel	14 dBi	Fixed	5 m	4.5dB	6.0 dB	6.5 dB
A24-P15NF	Flat Panel	15 dBi	Fixed	2 m	5.5dB	7.0 dB	7.5 dB
A24-P16NF	Flat Panel	16 dBi	Fixed	2 m	6.5dB	8.0 dB	8.5 dB
A24-19NF	Flat Panel	19 dBi	Fixed	2 m	9.5dB	11.0 dB	11.5 dB

Yagi class antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-Y6NF	Yagi (6 element)	8.8 dBi	Fixed	2 m	N/A	0.8 dB	1.2 dB
A24-Y7NF	Yagi (7 element)	9.0 dBi	Fixed	2 m	N/A	1.0 dB	1.5 dB
A24-Y9NF	Yagi (9 element)	10.0 dBi	Fixed	2 m	0.5 dB	2.0 dB	2.5 dB
A24-Y10NF	Yagi (10 element)	11.0 dBi	Fixed	2 m	1.5 dB	3.0 dB	3.5 dB
A24-Y12NF	Yagi (12 element)	12.0 dBi	Fixed	2 m	2.5 dB	4.0 dB	4.5 dB
A24-Y13NF	Yagi (13 element)	12.0 dBi	Fixed	2 m	2.5 dB	4.0 dB	4.5 dB
A24-Y15NF	Yagi (15 element)	12.5 dBi	Fixed	2 m	3.0 dB	4.5 dB	5.0 dB

Yagi class antennas							
Part number	Type (Description)	Gain	Application	Min separation	Minimum cable loss/power reduction/attenuation required		
					b mode	g mode	n mode
A24-Y16NF	Yagi (16 element)	13.5 dBi	Fixed	2 m	4.0 dB	5.5 dB	6.0 dB
A24-Y16RM	Yagi (16 element, RPSMA connector)	13.5 dBi	Fixed	2 m	4.0 dB	5.5 dB	6.0 dB
A24-Y18NF	Yagi (18 element)	15.0 dBi	Fixed	2 m	5.5 dB	7.0 dB	7.5 dB

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

RF Exposure



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

The preceding statement must be included as a CAUTION statement in OEM product manuals in order to alert users of FCC RF Exposure compliance.

Europe (CE)

The XBee Wi-Fi RF Module has been certified for use in several European countries. For a complete list, refer to www.digi.com.

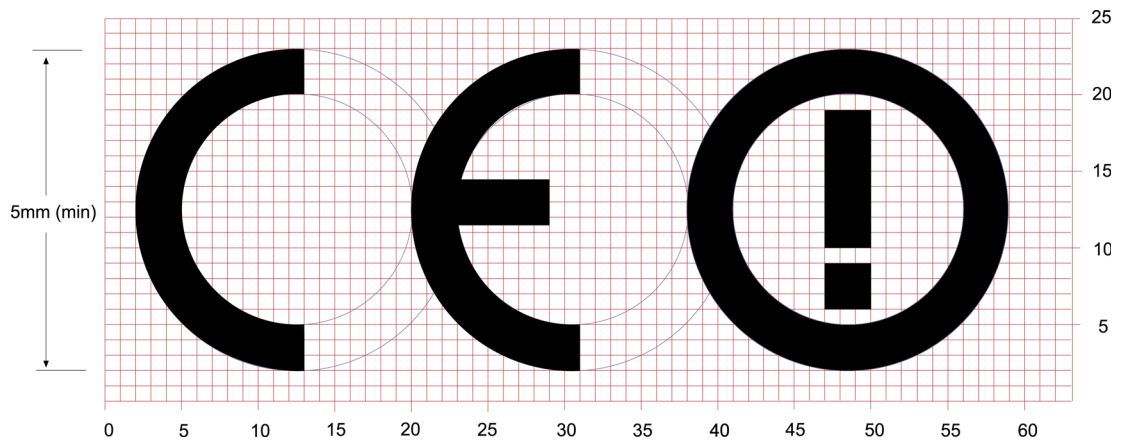
If the module is incorporated into a product, the manufacturer must ensure compliance of the final product to the European standards for their product. A Declaration of Conformity must be issued for each of these standards and kept on file as described in Annex II of the R&TTE Directive and other applicable European Directives.

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

OEM labeling requirements

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

CE Labeling Requirements



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

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

Declarations of conformity

Digi has issued Declarations of Conformity for the XBee RF Modules concerning emissions, EMC and safety. Files can be obtained by contacting Digi Support.

Important Note:

Digi does not list the entire set of standards that must be met for each country. Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. Refer to the radio regulatory agency in the desired countries of operation for more information.

Approved antennas

When integrating high-gain antennas, European regulations stipulate EIRP power maximums. The following antennas are approved for use with the XBee Wi-Fi Module:

- Dipole (2.1 dBi, Omni-directional, Articulated RPSMA, Digi part number A24-HABSM)
- PCB Antenna (0 dBi)
- Wire Whip Antenna (1.5 dBi)

Canada (IC)

This device complies with Industry Canada licence-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

Labeling requirements for Industry Canada are similar to those of the FCC. A clearly visible label on the outside of the final product enclosure must display the following text.

XBee Wi-Fi Through-hole:

Contains Model XBEE56B Radio, IC: 1846A-XBS6B

XBee Wi-Fi Surface Mount:

Contains Model S6BSM Radio, IC: 1846A-S6BSM

The integrator is responsible for its product to comply with IC ICES-003 & FCC Part 15, Sub. B - Unintentional Radiators. ICES-003 is the same as FCC Part 15 Sub. B and Industry Canada accepts FCC test report or CISPR 22 test report for compliance with ICES-003.

Transmitters with detachable antennas

This radio transmitter (IC: 1846A-XBS6B and IC: 1846A-S6BSM) has been approved by Industry Canada to operate with the antenna types listed in the tables above with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Le présent émetteur radio (IC: 1846A-XBS6B et IC: 1846A-S6BSM) a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Detachable antenna

Under Industry Canada regulations, this radio transmitter may 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 (e.i.r.p.) 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 à l'établissement d'une communication satisfaisante.

Australia (RCM)/New Zealand (R-NZ)

These modules comply with requirements to be used in end products in Australia and New Zealand. All products with EMC and radio communications must have registered RCM and R-NZ marks. Registration to use the compliance mark will only be accepted from Australia or New Zealand manufacturers or importers, or their agents.

In order to have a RCM or R-NZ mark on an end product, a company must comply with a or b below.

- a. have a company presence in Australia or New Zealand.
- b. have a company/distributor/agent in Australia or New Zealand that will sponsor the importing of the end product.

Contact Digi for questions related to locating a contact in Australia and New Zealand.

Brazil (ANATEL)

These modules 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):

Anatel homologation number to be stamped on product label:

ANATEL: 2672-13-1209

Anatel homologation label to be stamped on the user manual.

modelo Xbee S6B e modelo S6BSM



Resolution 506 warning to be stamped on the user manual:

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

Manufacturing information for surface-mount XBee

The surface-mount XBee 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 that there are no hidden solder joints.

Recommended solder reflow cycle

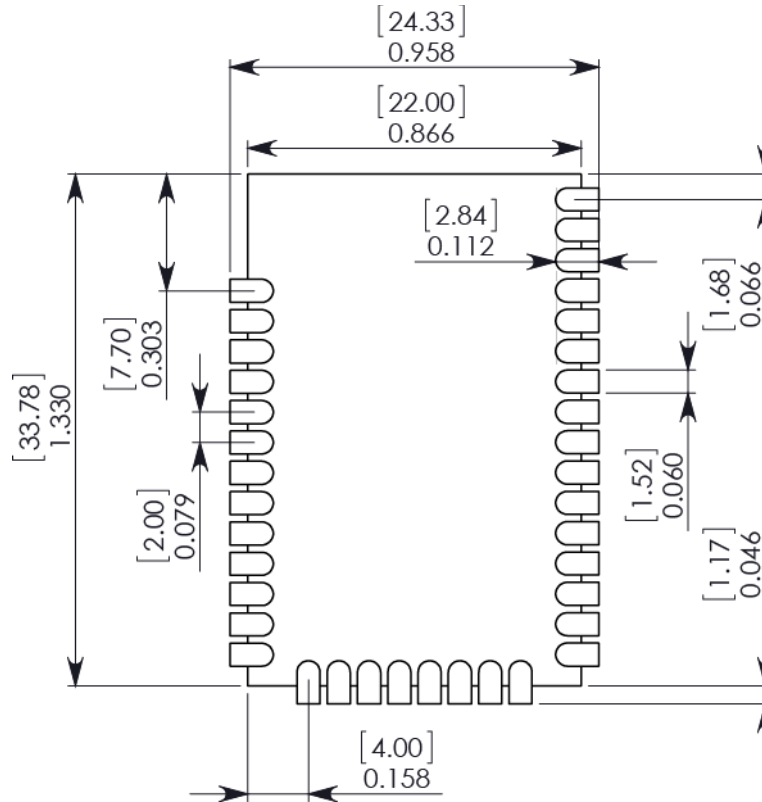
The recommended solder reflow cycle is shown below. The chart shows the temperature setting and the time to reach the temperature. The cooling cycle is not shown.

Time (seconds)	Temperature (°C)
30	65
60	100
90	135
120	160
150	195
180	240
210	260

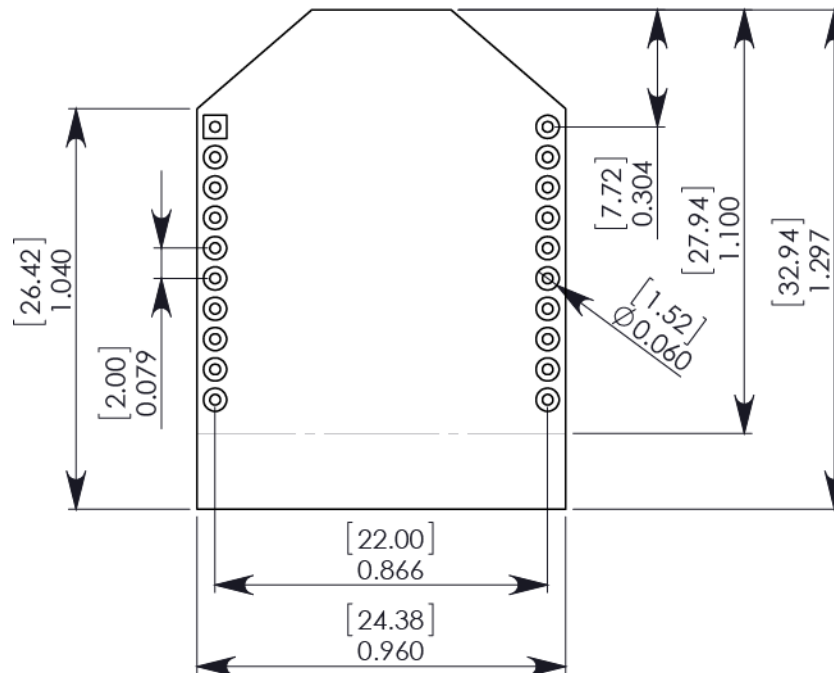
The maximum temperature should not exceed 260 degrees Celsius. The module will reflow during this cycle, and therefore must not be reflowed upside down. Care should be taken not to jar the module while the solder is molten, as parts inside the module can be removed from their required locations. Hand soldering is possible and should be done in accordance with approved standards. This module has a Moisture Sensitivity Level (MSL) of 3.

Recommended footprint

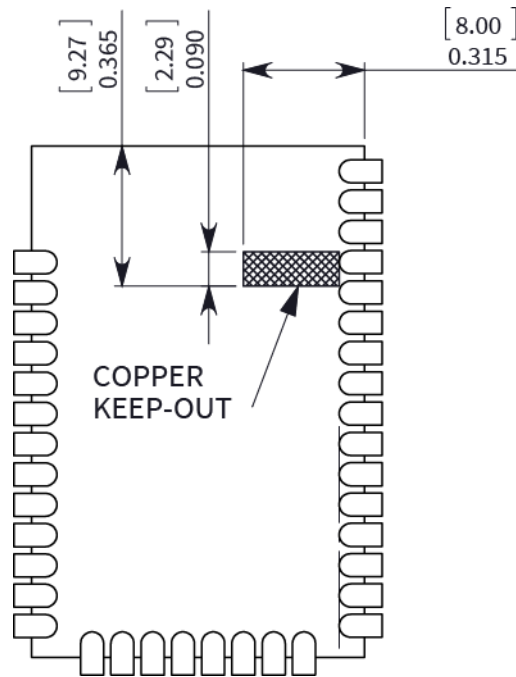
We recommend that you use the PCB footprints in the following drawings for surface mounting. The first drawing shows the surface mount module.



The following image shows the through-hole module.

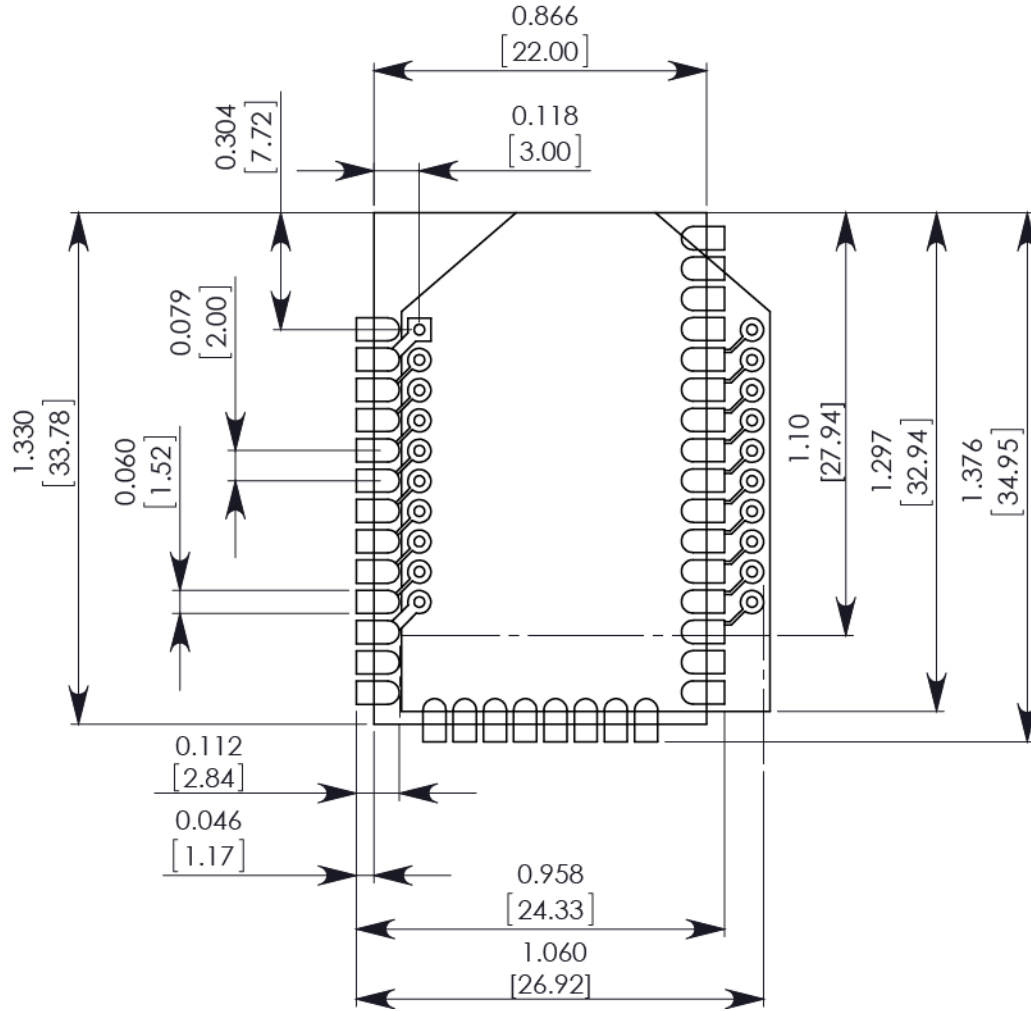


Match the solder footprint to the copper pads. You may need to adjust the footprint depending on the specific needs of assembly and product standards. The underside of the module is mostly coated with solder resist, but we recommend that you leave the copper layer directly below the module open to avoid unintended contacts. Copper or vias must not interfere with the three exposed RF test points on the bottom of the module (see below). These modules have a ground plane in the middle on the back side for shielding purposes, which can be affected by copper traces directly below the module.



Common footprint for through-hole and surface mount

We designed a common footprint that allows either module to be attached to a PCB. The following drawing shows this layout. The round holes in the drawing are for the through-hole version, and the semi-oval pads are for the surface mount version. Pin 1 of the through-hole version connects with pin 2 of the surface mount. Use the diagonal traces to connect the pins and the layout will work for both modules.



Flux and cleaning

We recommend that you use a “no clean” solder paste in assembling these modules. This eliminates the clean step and ensure unwanted residual flux is not left under the module where it is difficult to remove. In addition:

- Cleaning with liquids can result in liquid remaining under the shield or in the gap between the module and the OEM PCB. This can lead to unintended connections between pads on the module.
- The residual moisture and flux residue under the module are not easily seen during an inspection process.

Factory recommended best practice is to use a “no clean” solder paste to avoid the issues above and ensure proper module operation.

Rework

Rework should never be performed on the module itself. The module has been optimized to give the best possible performance, and reworking the module itself will void warranty coverage and certifications. We recognize that some customers will choose to rework and void the warranty; the following information is given as a guideline in such cases to increase the chances of success during rework, though the warranty is still voided. The module may be removed from the OEM PCB by the use of a hot air rework station, or hot plate. Care should be taken not to overheat the module. During rework, the module temperature may rise above its internal solder melting point and care should be taken not to dislodge internal components from their intended positions.

Glossary of terms

Definitions

Local Host

A device that is electrically connected to an XBee. Typically this is a microcontroller connected to the serial pins of the module.

MAC address

A unique network identifier. All network devices are required to have their own unique MAC address. The MAC address is on a sticker on your Digi device server. The number is displayed as 12 hexadecimal digits, usually starting with 00:40:9D.

Network Client

A device that communicates with an XBee through the 802.11 network.

Static IP address assignment

The process of assigning a specific IP address to a device. Contrast with assigning a device through Dynamic Host Configuration Protocol (DHCP), or Automatic Private IP Addressing (APIPA or Auto-IP).

TCP

See [Transmission Control Protocol \(TCP\)](#) on page 147.

Temporal Key Integrity Protocol (TKIP)

Part of the IEEE 802.11i encryption standard for wireless LANs. TKIP is the next generation of the Wired Equivalent Privacy (WEP), which is used to secure 802.11 wireless LANs. TKIP provides per-packet key mixing, a message integrity check and re-keying mechanism, and addresses several design shortcomings of the original WEP.

Transmission Control Protocol (TCP)

A set of rules (protocol) used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. While IP handles the actual delivery of the data, TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet. For example, when an HTML file is sent to you from a Web server, the Transmission Control Protocol (TCP) program layer in that server divides the file into one or more packets, numbers the packets, and then forwards them individually to the IP program layer. Although each packet has the same destination IP address, it may get routed differently through the network. At the other end (the client program in your computer), TCP reassembles the individual packets and waits until they have arrived to forward them to you as a single file.

TCP is known as a connection-oriented protocol, which means that a connection is established and maintained until such time as the message or messages to be exchanged by the application programs at each end have been exchanged. TCP is responsible for ensuring that a message is divided into the packets that IP manages and for reassembling the packets back into the complete message at the other end. In the Open Systems Interconnection (OSI) communication model, TCP is in layer 4, the Transport Layer.

UDP

See [User Datagram Protocol \(UDP\)](#) on page 148.

User Datagram Protocol (UDP)

A communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol (TCP) and, together with IP, is sometimes referred to as UDP/IP. Like the Transmission Control Protocol, UDP uses the Internet Protocol to actually get a data unit (called a datagram) from one computer to another. Unlike TCP, however, UDP does not provide the service of dividing a message into packets (datagrams) and reassembling it at the other end. Specifically, UDP does not provide sequencing of the packets in which the data arrives, nor does it guarantee delivery of data. This means that the application program that uses UDP must be able to make sure that the entire message has arrived and is in the right order.

Network applications that want to save processing time because they have very small data units to exchange (and therefore very little message reassembling to do) may prefer UDP to TCP. The Trivial File Transfer Protocol (TFTP) uses UDP instead of TCP.

UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact.

Wi-Fi Protected Access (WPA)

A data encryption / user authentication method for 802.11 wireless LANs. WPA uses the Temporal Key Integrity Protocol (TKIP).

Wired Equivalency Protocol (WEP)

A security algorithm that uses an RC4 stream cipher, but which has multiple known flaws.

WPA

See [Wi-Fi Protected Access \(WPA\)](#) on page 148.

WPA2/802.11i

WPA with AES-based encryption (CCMP)