# XBee® Wi-Fi RF Module

S6B

User Guide

# Revision history—90002180

| Revision | Date | Description |
|---|---|---|
| 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. |
| P | May 2017 | Added information on the Associate LED. Revised the manual. Changed Device Cloud to Remote Manager. Updated frame 0x20 Transmit options bit description. |
| R | June 2017 | Modified regulatory and certification information as required by RED (Radio Equipment Directive). |

## Trademarks and copyright



## Disclaimers

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

## Warranty

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

www.digi.com/howtobuy/terms

## Send comments

**Documentation feedback**: To provide feedback on this document, send your comments to techcomm@digi.com.

## Customer support

**Digi Technical 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, contact us at +1 952.912.3444 or visit us at www.digi.com/support.

# Contents

## Modes



## Sleep modes



## 802.11 bgn networks

## IP services



## I/O support



## Wi-Fi Protected Setup (WPS)



## General Purpose Flash Memory

## Configure the XBee Wi-Fi RF Module using Digi Remote Manager



## API operation

## AT commands

## Regulatory information

## Manufacturing information

# XBee Wi-Fi RF Module User Guide

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 Wi-Fi RF Module, you can have an 802.11 bgn network up and running in a matter of minutes.

The XBee Wi-Fi RF 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.

## Applicable firmware and hardware

This manual supports the following firmware:

- x202x

It supports the following hardware:

- S6B

# Technical specifications

# General specifications

The following table describes the general specifications for the devices.

| Specification | XBee Wi-Fi through-hole | XBee Wi-Fi surface-mount |
|---|---|---|
| Dimensions | 2.438 cm x 2.761 cm (0.960 in x 1.087 in) | 2.200 x 3.378 cm (0.866 x 1.330 in) |
| 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 characteristics

The following table provides the RF characteristics for the device.

| Specification | XBee Wi-Fi through-hole | | XBee Wi-Fi surface-mount | |
|---|---|---|---|---|
| Frequency | Industrial, scientific and medical (ISM) 2.4 - 2.5 GHz | | | |
| Number of channels | 13 | | | |
| Adjustable power | Yes | | | |
| Wi-Fi standards | 802.11 b, g, and n | | | |
| Transmit power output (average) | Up to +16 dBm<br>+13 dBm for Europe/Australia and New Zealand/Brazil; see RF transmit power - typical | | | |
| 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 RF data rates | | | |
| 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 Receiver sensitivity | | | |

# RF data rates

The following table provides the RF data rates for the device.

| 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 |

| Receiver sensitivity (25 °C, < 10% PER)<br>Standard | Data rate | Sensitivity (dBm) |
|---|---|---|
| 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

The following table provides the average RF transmit power for the device.

| Standard | Data rate | Power (dBm) | |
|---|---|---|---|
| | | North America/Japan | Europe/Australia and 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 | | |

| Standard | Data rate | Power (dBm) | |
|---|---|---|---|
| | | North America/Japan | Europe/Australia and New Zealand/Brazil |
| 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 | 8.5 | 8.5 |

# Error vector magnitude (EVM) maximum output power - typical

The following table shows the EVM at 25 °C, maximum 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 |

| Standard | Data rate | EVM (dB) |
|---|---|---|
| 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

The following table provides the electrical specifications for the XBee Wi-Fi RF Module.

| Specification | XBee Wi-Fi |
|---|---|
| Supply voltage | 3.14 - 3.46 VDC |

| Specification | XBee Wi-Fi | | |
|---|---|---|---|
| Operating current (transmit, maximum 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) | 100 mA | | |
| Deep sleep current | 6 µA @ 25 °C | | |
| Associated sleep current | 2 mA asleep, 100 mA awake. For more information, see AP Associated Sleep mode. | | |

# Serial communication specifications

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

### UART pin assignments

| Specifications | Device pin number | |
|---|---|---|
| UART pins | XBee (surface-mount) | XBee (through-hole) |
| DIO13/DOUT | 3 | 2 |
| DIO14/DIN | 4 | 3 |
| DIO7/$\overline{\text{CTS}}$ | 25 | 12 |
| DIO6/$\overline{\text{RTS}}$ | 29 | 16 |

For more information on UART operation, see UART data flow.

### SPI pin assignments

| Specifications | Device pin number | |
|---|---|---|
| SPI pins | XBee (surface-mount) | XBee (through-hole) |
| DIO2/SPI_SCLK | 14 | 18 |
| DIO3/SPI_$\overline{\text{SSEL}}$ | 15 | 17 |
| DIO4/SPI_MOSI | 16 | 11 |
| DIO12/SPI_MISO | 17 | 4 |
| DIO1/SPI_$\overline{\text{ATTN}}$ | 12 | 19 |

For more information on SPI operation, see SPI communications.

## GPIO specifications

The XBee Wi-Fi RF Modules have 14 (through-hole version) and 20 (surface-mount version) General Purpose Input Output (GPIO) ports available. The exact list depends on the module configuration, as some GPIO pads are used for purposes such as serial communication.

See I/O sampling for more information on configuring and using GPIO ports. The following table provides the electrical specifications for the GPIO pads.

| Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|
| Input low voltage | | | 0.3 VDD | V |
| Input high voltage | | 0.7 VDD | | 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 | % |

| Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|
| Output fall time | 2 mA drive strength and load capacitance CL= 350 - 600 pF. | 20 +0.1 CL | 250 | ns |
| I/O pin hysteresis (VIOTHR+ - VIOTHR-) | VDD = 3.14 to 3.46 V | 0.1 VDD | | V |
| Pulse width of pulses to be removed by the glitch suppression filter | | 10 | 50 | ns |

# Regulatory conformity summary

This table describes the agency approvals for the devices.

| Country | 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) | Yes | Yes |
| 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 Regulatory information.

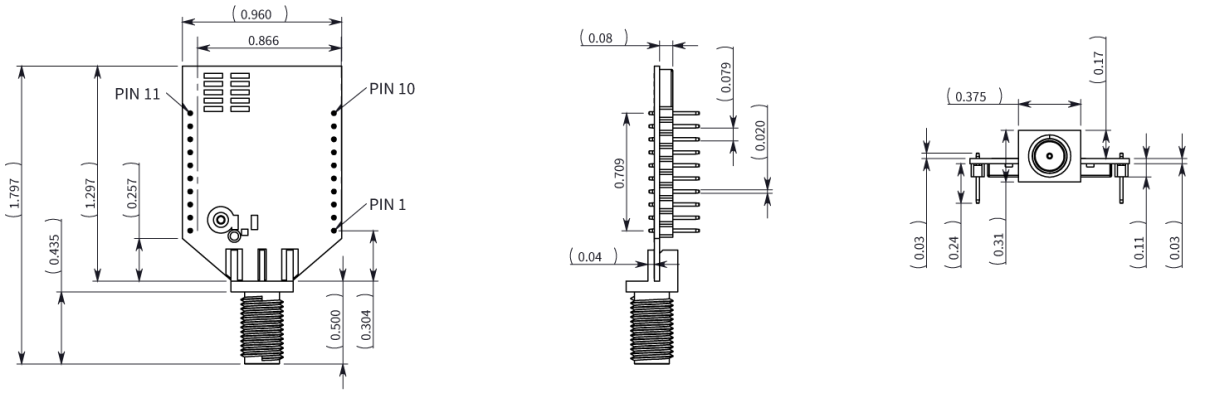
# Hardware

# Mechanical drawings

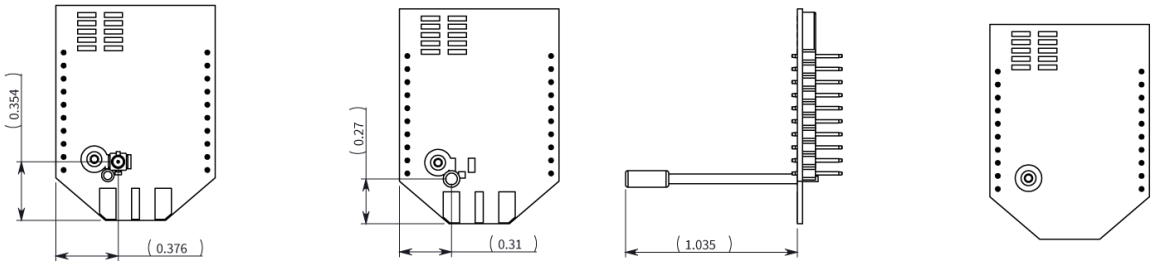
The following figures show the mechanical drawings for the XBee Wi-Fi RF Module. The drawings do not show antenna options. All dimensions are in inches.

## Through-hole device

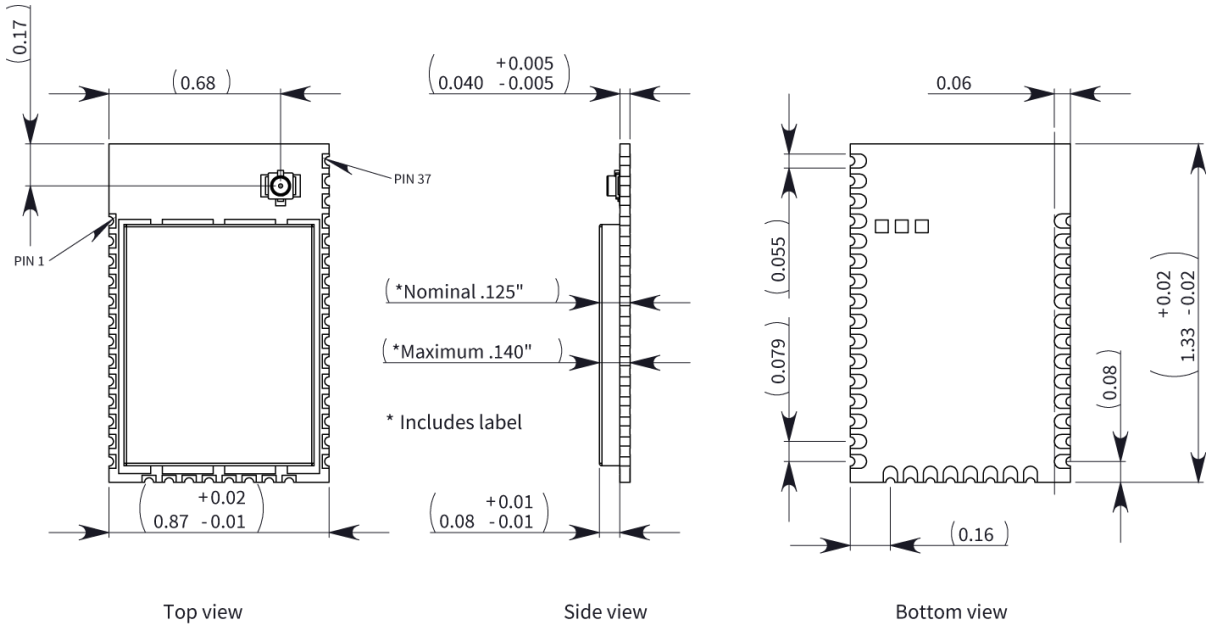
RPSMA




U.FL Wire whip PCB antenna



### Surface-mount device




# Pin signals

The following table describes the pin assignments for the through-hole device. A horizontal line above the signal name indicates low-asserted signals.

| 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 |

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

The following table describes the pin assignments for the surface-mount device. A horizontal line above the signal name indicates low-asserted signals.

| Pin # | Name | Direction | Default state | Description |
|---|---|---|---|---|
| 1 | GND | - | - | Ground |
| 2 | VCC | - | - | Power supply |
| 3 | DIO13/DOUT | Both | Output | UART data out |
| 4 | DIO14/DIN/$\overline{\text{CONFIG}}$ | Both | Input | UART data in |
| 5 | DIO12 | Both | Disabled | GPIO |
| 6 | $\overline{\text{RESET}}$ | 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/$\overline{\text{DTR}}$/SLEEP_RQ | Both | Input | GPIO |
| 11 | GND | - | - | Ground |
| 12 | DIO19/SPI_$\overline{\text{ATTN}}$ | Both | Output | GPIO/SPI attention |
| 13 | GND | - | - | Ground |
| 14 | DIO18/SPI_CLK | Both | Input | GPIO/SPI clock |
| 15 | DIO17/SPI_$\overline{\text{SSEL}}$ | Both | Input | GPIO/SPI slave select |

| Pin # | Name | Direction | Default state | Description |
|---|---|---|---|---|
| 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 |
| 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 I/O for RF pad variant |
| 37 | Reserved | - | - | Do not connect |

# Design notes

The XBee devices do not specifically require any external circuitry specific connections for proper operation. However, there are some general design guidelines that we recommend for help in troubleshooting and building a robust design.

## Power supply

A poor power supply can lead to poor device performance, especially if you do not keep the supply voltage within tolerance or if it is excessively noisy. To help reduce noise, place a 1.0 µF and 8.2 pF capacitor as near as possible to pin 1 on the PCB. If you are using a switching regulator for the power

supply, switch the frequencies above 500 kHz. Limit the power supply ripple to a maximum 50 mV peak to peak.

## Pin connection recommendations

The only required pin connections are VCC, GND, and either DOUT and DIN or SPI_CLK, SPI_SSEL, SPI_MOSI, and SPI MISO. To support serial firmware updates, you should connect VCC, GND, DOUT, DIN, RTS, and DTR.

Leave all unused pins disconnected. Use the **PR** command to pull all of the inputs on the device high using 40 k internal pull-up resistors. You do not need a specific treatment for unused outputs.

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

You can connect other pins to external circuitry for convenience of operation. For example, the Associate signal (through-hole pin 15 / surface-mount pin 28) and the ON_SLEEP signal (through-hole pin 13 / surface-mount pin 26) will change level or behavior based on the state of the device.

## Board layout

When designing the host PCB, account for the device dimensions shown in Mechanical drawings. See Manufacturing information for the 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.

## Antenna performance

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

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

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

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

- vehicles
- elevators
- ventilation ducts
- refrigerators
- microwave ovens
- batteries
- tall electrolytic capacitors

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

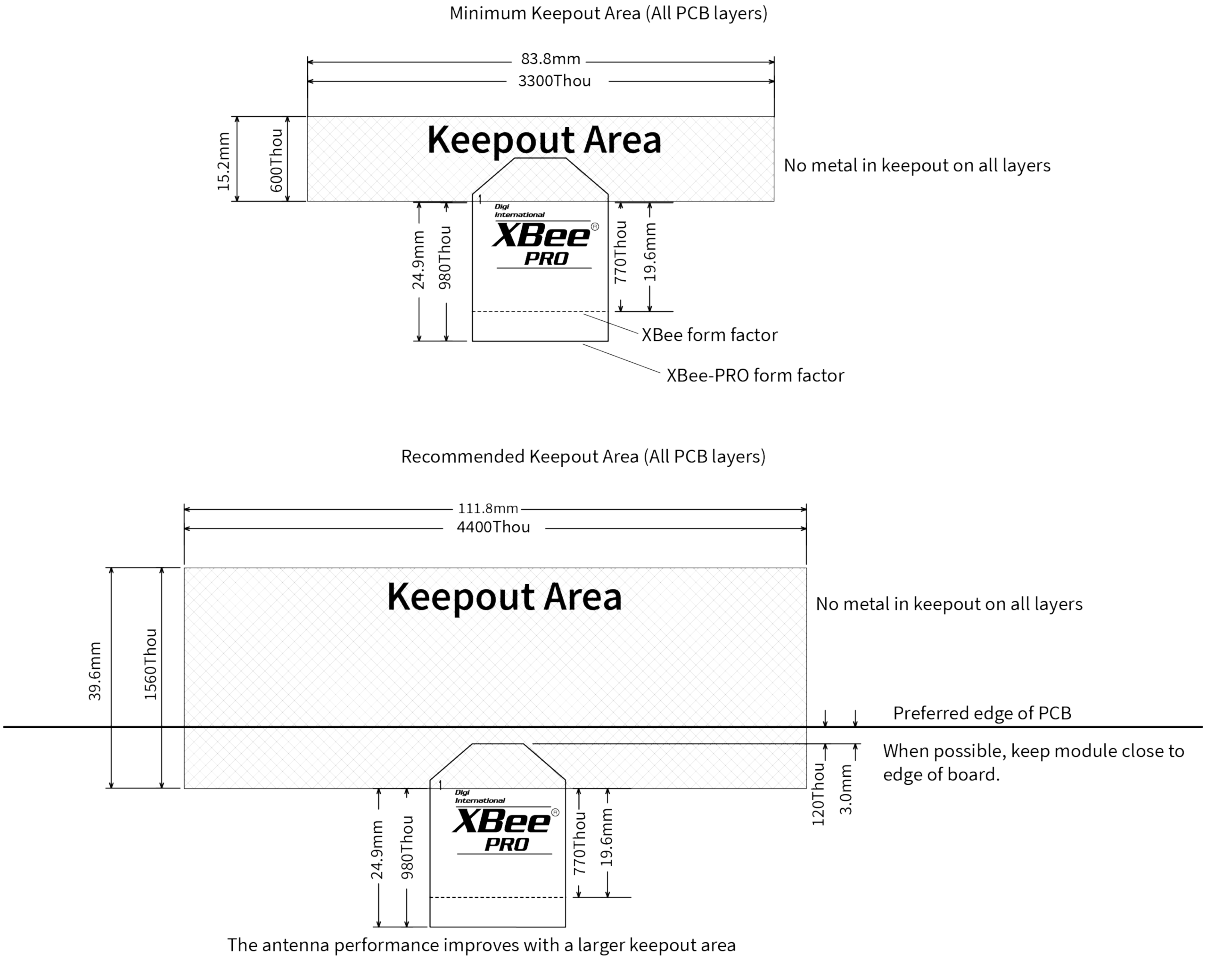
Do not place any ground planes or metal objects above or below the antenna.

For the best results, mount the device at the edge of the host PCB. Ensure that the ground, power, and signal planes are vacant immediately below the antenna section.

## *Keepout area*

The following drawings show important recommendations for designing with the PCB antenna module using the through-hole and surface-mount devices. Do not mount the surface-mount PCB antenna module on the RF Pad footprint because that footprint requires a ground plane within the keepout area.

**Through-hole keepout**

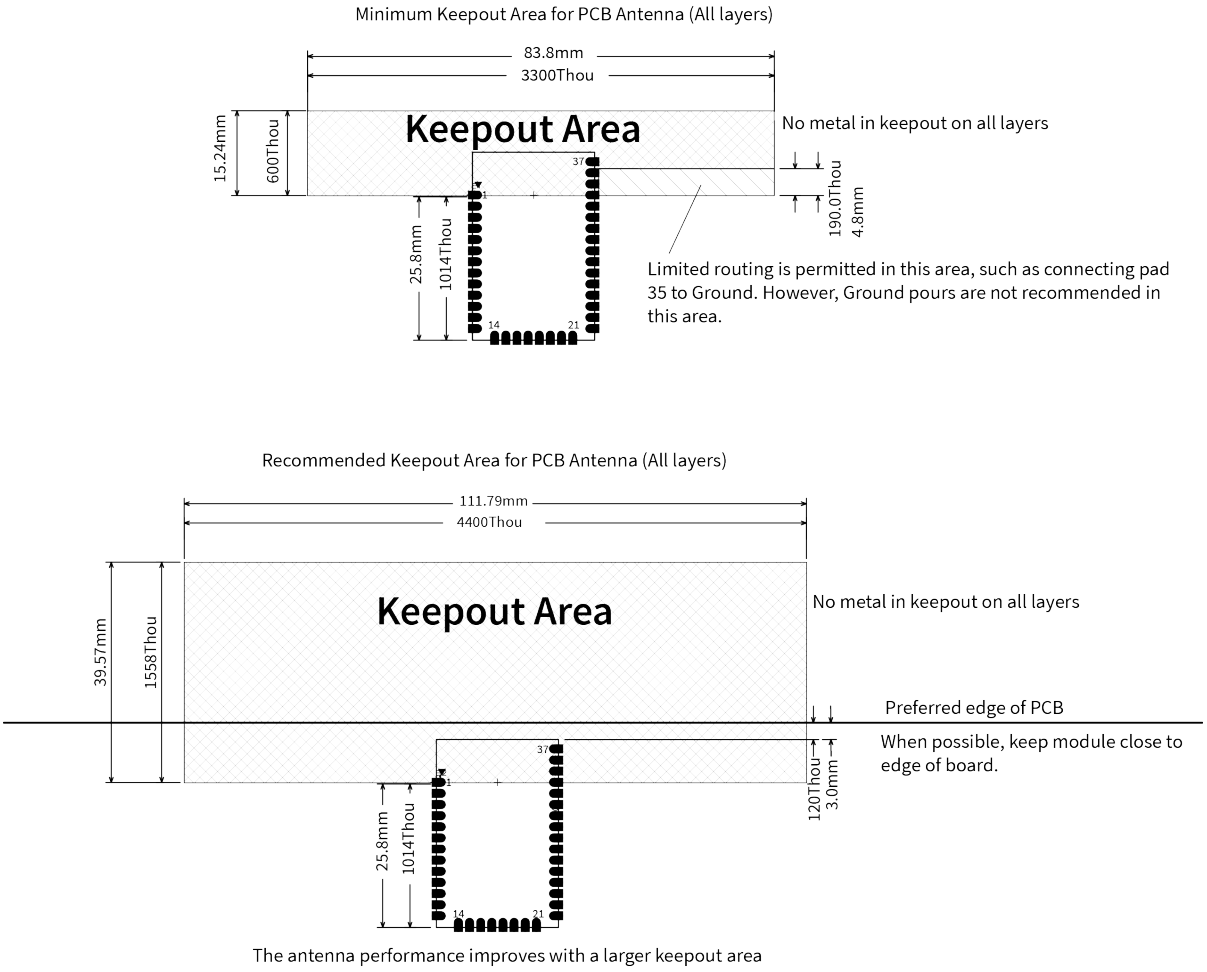



***Notes***

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 keepout**




***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 Micro RF pad devices

The RF pad is a soldered antenna connection. The RF signal travels from pin 33 on the device to the antenna through an RF trace transmission line on the PCB. Any additional components between the

device and antenna violates modular certification. The controlled impedance for the RF trace is 50 Ω.

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

Following good design practices is essential when implementing the RF trace on a PCB. Consider the following points:

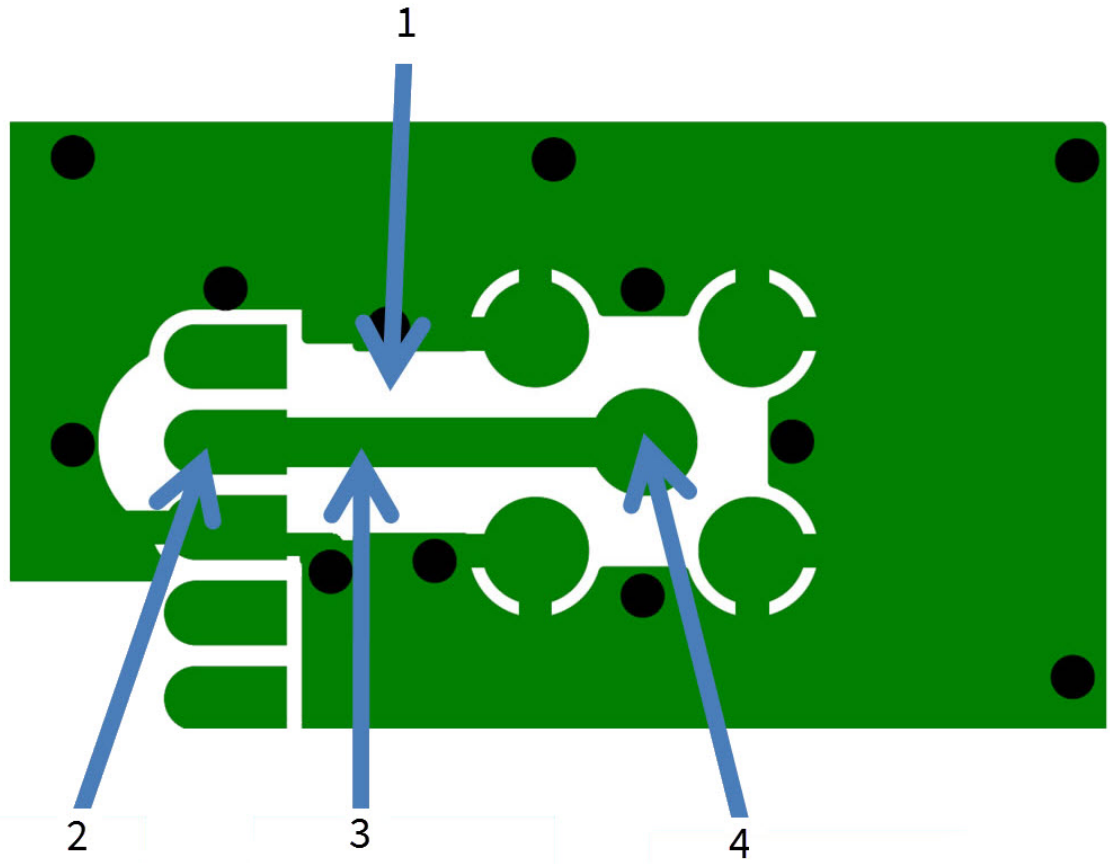
- Minimize the length of the trace by placing the RPSMA jack close to the device.
- Connect all of the grounds on the jack and the device to the ground planes directly or through closely placed vias.
- Space any ground fill on the top layer at least twice the distance d (in this case, at least 0.028") from the microstrip to minimize their interaction.

Additional considerations:

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

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

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

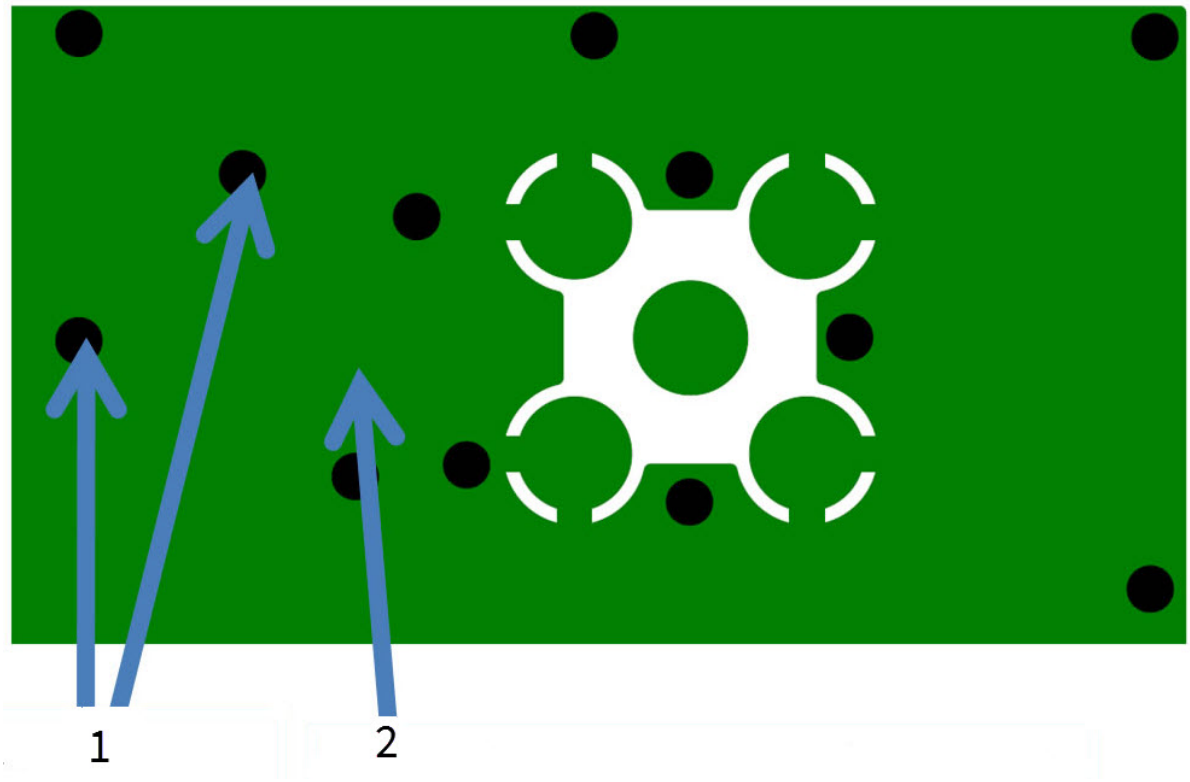



| Number | Description |
|---|---|
| 1 | Maintain a distance of at least 2 d between microstrip and ground fill. |
| 2 | Device pin 33. |
| 3 | 50 Ω microstrip trace. |
| 4 | RF connection of RPSMA jack. |

The width in this example is approximately 0.025 in for a 50 Ω trace, assuming d = 0.014 in, and that the dielectric has a relative permittivity of 4.4. This trace width is a good fit with the device footprint's 0.335" pad width.

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

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




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

# Mounting considerations

We design the through-hole module to mount into a receptacle so that you do not have to solder the module when you mount it to a board. The interface boards provided in the XBee Wi-Fi Development Kit has two ten-pin receptacles for connecting the module.

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

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

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

# Operation

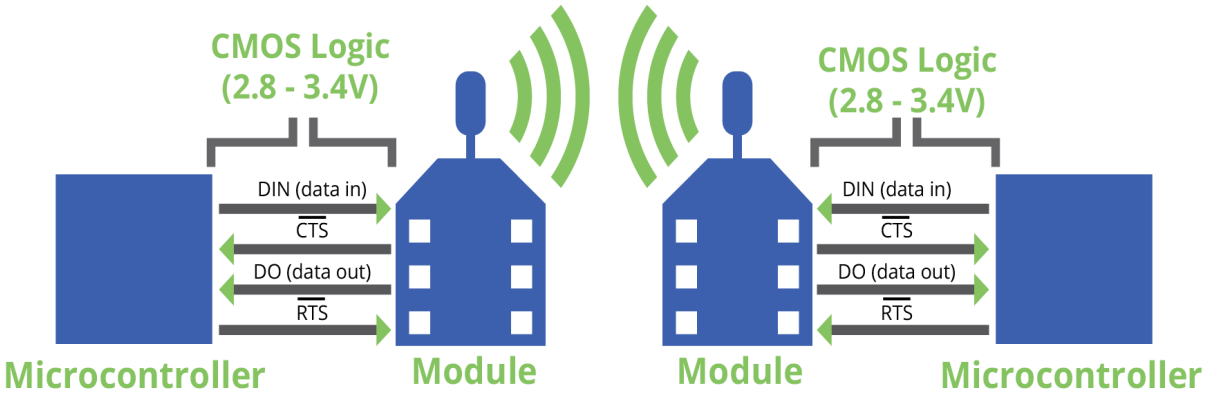
# Serial interface

The XBee Wi-Fi RF Module interfaces to a host device through a serial port. The device can communicate through its serial port with:

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

# UART data flow

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

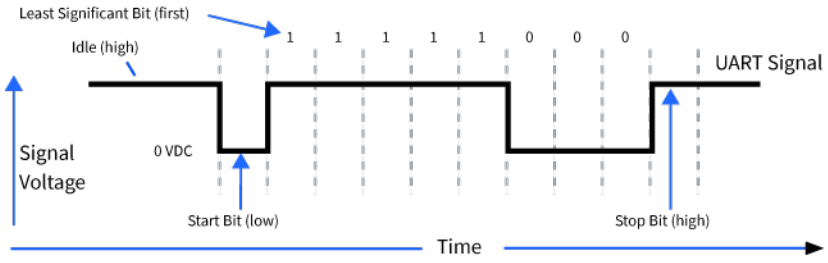



# Serial data

A device sends data to the XBee Wi-Fi RF Module's UART through TH pin 3/SMT pin 4 (DIN) as an asynchronous serial signal. When the device is not transmitting data, the signals should idle high.

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

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




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

In the rare case that a device has been configured with the UART disabled, you can recover the device to UART operation by holding DIN low at reset time. DIN forces a default configuration on the UART at 9600 baud and it brings the device up in Command mode on the UART port. You can then send the appropriate commands to the device to configure it for UART operation. If those parameters are written, the device comes up with the UART enabled on the next reset.

# SPI communications

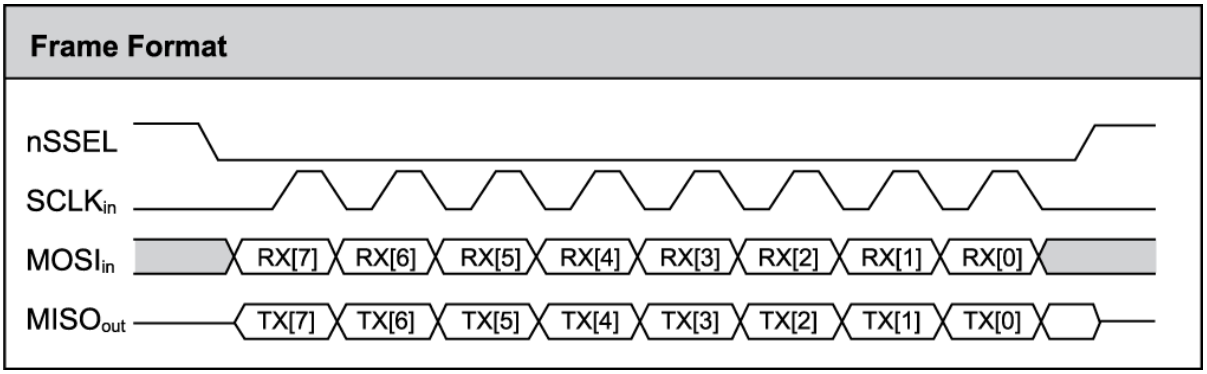
The XBee Wi-Fi RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.

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

In this mode:

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

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




SPI mode is chip to chip communication. We do not supply a SPI communication option on the device development evaluation boards.

## Select the SPI port

On the through-hole devices, you can force SPI mode by holding DOUT/DIO13 low while resetting the device until SPI_ATTN asserts. This causes the device to disable the UART and go straight into SPI communication mode. Once configuration is complete, the device queues a modem status frame to the SPI port, which causes the SPI_ATTN line to assert. The host can use this to determine that the SPI port is configured properly. This method forces the configuration to provide full SPI support for the following parameters:

- **D1** (This parameter will only be changed if it is at a default of zero when the method is invoked.)
- **D2**
- **D3**
- **D4**
- **P2**

As long as the host does not issue a **WR** command, these configuration values revert to previous values after a power-on reset. If the host issues a **WR** command while in SPI mode, these same parameters are written to flash. 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 device comes 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 goes to the UART until the host sends the first input. If that first input comes on the SPI port, then all subsequent output goes to the SPI port and the UART is disabled. If the first input comes on the UART, then all subsequent output goes to the UART and the SPI is disabled.

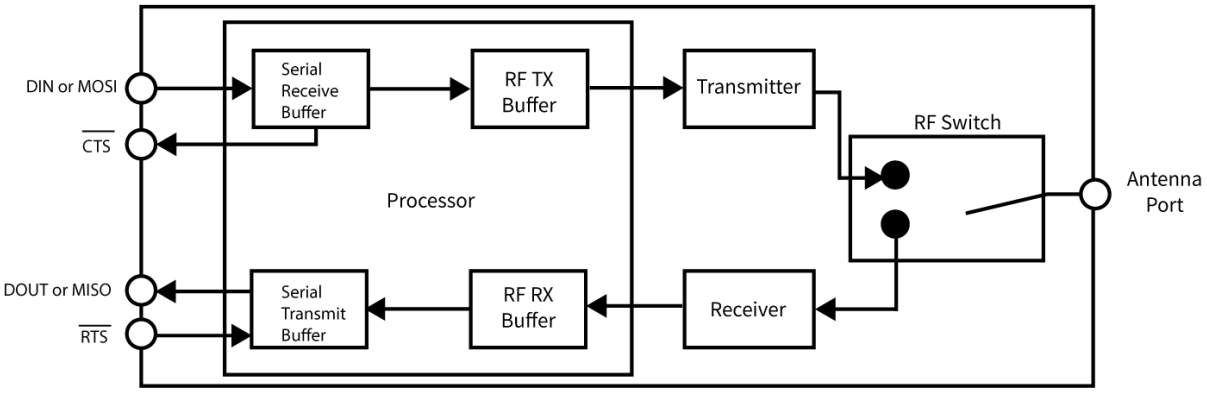
Once you select a serial port (UART or SPI), all subsequent output goes to that port, even if you apply a new configuration. The only way to switch the selected serial port is to reset the device. On surface-mount devices, forcing DOUT low at the time of reset has no effect. To use SPI mode on the SMT modules, assert the SPI_SSEL pin (15) low after reset and before any UART data is input.

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

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

# Serial buffers

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




## Serial receive buffer

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

## Serial transmit buffer

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

# UART flow control

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

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

### CTS flow control

The **FT** command allows you to specify how many bytes of data can be queued up in the serial transmit buffer before the device asserts CTS low. The serial receive buffer can hold up the 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 device after the device asserts CTS before the data is dropped. In either case, CTS is not re-asserted until the serial receive buffer has **FT**-17 or less bytes in use.

### RTS flow control

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

If the device sends data out the UART when RTS is de-asserted (set high) the device could send up to four characters out the UART port after RTS is de-asserted. This means your application needs to de-assert RTS by the time its receive capacity is within 4 bytes of full.

## The Commissioning Button

The XBee Wi-Fi RF Module supports a set of commissioning and LED functions to help you deploy and commission devices. These functions include the Commissioning Button definitions and the associated LED functions.

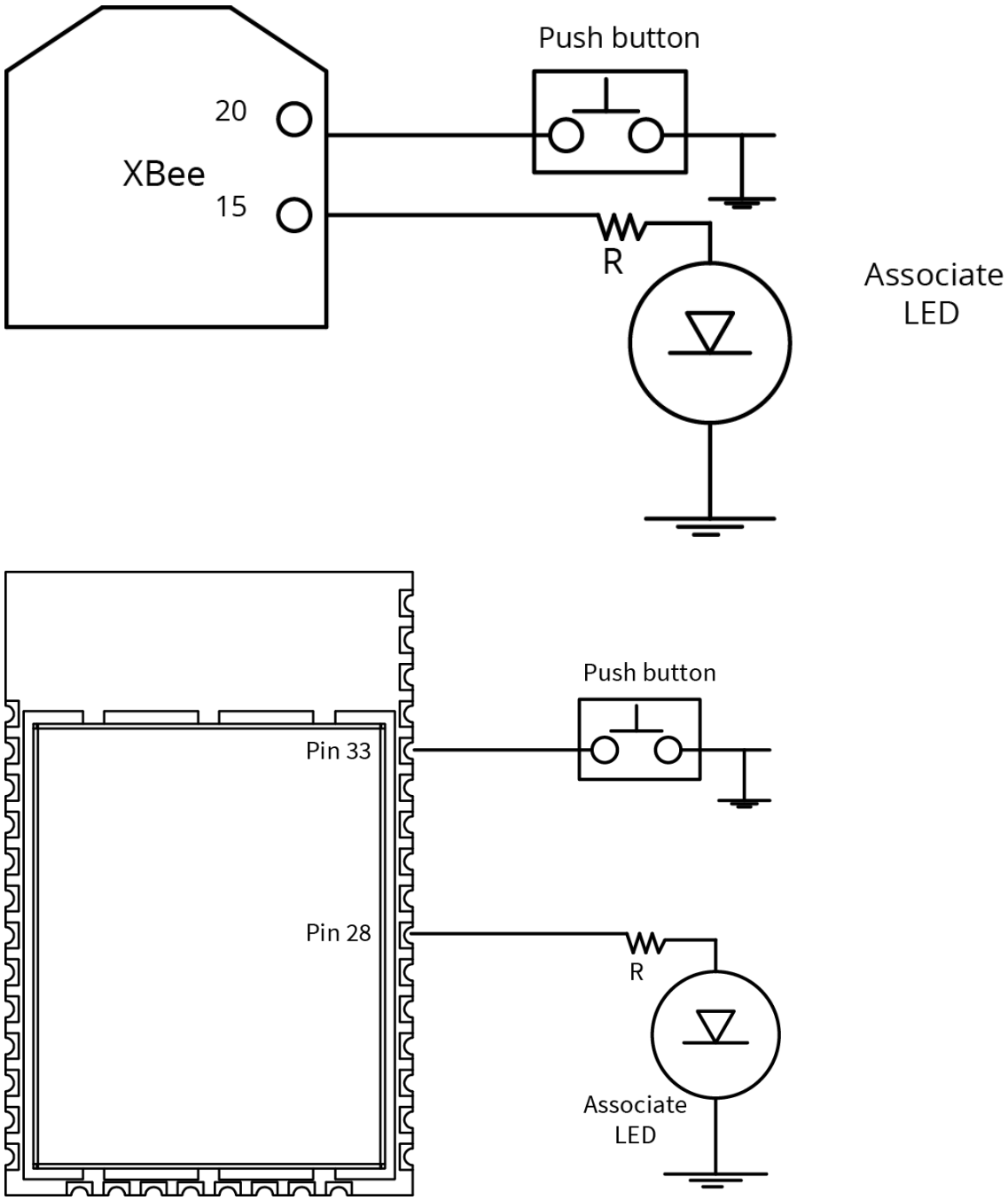
To enable the Commissioning Button functionality on TH pin 20/SMT pin 33, set DO command to 1. The functionality is enabled by default.

Use the **CB** command to simulate button presses in software. Send **CB** with a parameter set to the number of button presses to perform. For example, if you send **CB2**, the device performs the action(s) associated with two button presses, **CB4** is four button presses. See CB (Commissioning Pushbutton).

It provides two different services:

- Two button presses in fast sequence invoke WPS; see Wi-Fi Protected Setup (WPS).
- Four button presses in fast sequence force the device into Soft AP Provisioning mode by clearing the SSID and security parameters. It also ensures that Soft AP mode is enabled. After the four button presses clear the security parameters, they are NOT written. Send a separate WR (Write), if desired.

The following features can be supported in hardware. Connect a pushbutton and an LED to XBee Wi-Fi RF Module pins 33 and 28 (SMT), or pins 20 and 15 (TH) respectively to support Commissioning Button definitions and the associated LED functions.




## Connection indicators

There are four connection indicators in this software:

- AI (Association Indication)
- The Associate LED
- TCP connection indicator
- Remote Manager connection indicator

### The Associate LED

The Associate pin (TH pin 15/SMT pin 28) provides an indication of when the device associates with an access point (AP). To take advantage of these indications, connect an LED to the Associate pin.

To enable the Associate LED functionality, set D5 (DIO5 Configuration) to 1; it is enabled by default. If enabled, the Associate pin is configured as an output.

Use LT (Associate LED Blink Time) to override the blink rate of the Associate pin. If you set **LT** to 0, the device uses the default blink time: 250 ms.

### TCP connection indicator

In Transparent mode, only one TCP connection is allowed and you can configure DIO12 (also known as CD) to indicate whether or not that TCP socket is connected. To enable DIO12, set P2 (DIO12 Configuration) 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. The high signal remains when operating in UDP mode because there is never be a TCP connection.

### Remote Manager connection indicator

AI (Association Indication) and the Associate LED indicate when the device is fully associated with the access point (AP), but there is another level of connectivity provided by DI (Device Cloud Indicator) that tells whether or not the TCP socket to Digi Remote Manager is connected. The values defined for **DI** are:

- **0** = Connected to Remote Manager
- **1** = Configured, but not yet associated to AP
- **2** = Associated to AP, but not yet connected to Remote Manager
- **3** = Disconnecting from Remote Manager
- **4** = Not configured to connect to Remote Manager

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:

## Perform a serial firmware update

Serial firmware updates use the XBee bootloader which ships in all devices. This bootloader allows you to update the firmware. 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 TH pin 3/SMT pin 4, TH pin 9/SMT pin 10, and TH pin 16/SMT pin 29 (DIN, DTR, and RTS, respectively).

To force the XBee bootloader to run and load a new version of the firmware, at the time the device is reset:

1. Drive DIN low.
2. Drive $\overline{\text{DTR}}$ low.
3. Drive $\overline{\text{RTS}}$ high.

This method works even when the current firmware version does not support the firmware upgrade feature. XCTU can update firmware on the XBee Wi-Fi RF Module over the UART port, but not over the SPI port. Contact Digi support for details.

# Modes

# Serial modes

The firmware operates in several different modes. Two top-level modes establish how the device communicates with other devices through its serial interface: Transparent operating mode and API operating mode. Use the **AP** command to choose Serial mode. XBee Wi-Fi RF Modules use Transparent operation as the default serial mode.

The following modes describe how the serial port sends and receives data.

## Transparent operating mode

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

**Note** Transparent operating mode is not available when using the SPI interface; see SPI communications.

### *Serial-to-RF packetization*

The device buffers data in the serial receive buffer until one of the following causes the data to be packetized and transmitted:

- The device receives no serial characters for the amount of time determined by the **RO** (Packetization Timeout) parameter. If **RO** = 0, packetization begins when a character is received. 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 device receives the Command Mode Sequence (**GT** + **CC** + **GT**). Any character buffered in the serial receive buffer before the sequence is transmitted.
- The device receives the maximum number of characters that fits in an RF packet (100 bytes).

## API operating mode

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

For more information, see API mode overview.

### *Comparing Transparent and API modes*

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

| Feature | Description |
|---|---|
| **Transparent mode features** | |
| Simple interface | All received serial data is transmitted unless the device is in Command mode |
| Easy to support | It is easier for an application to support Transparent operation and Command mode |
| **API mode features** | |
| Easy to manage data transmissions to multiple destinations | Transmitting RF data to multiple remote devices only requires the application to change the address in the API frame. This process is much faster than in Transparent mode where the application must enter Command mode, change the address, exit Command mode, and then transmit data. |
| Each API transmission can return a transmit status frame indicating the success or reason for failure | Because acknowledgments are sent out of the serial interface, this provides more information about the health of the RF network and can be used to debug issues after the network has been deployed. |
| Received data frames indicate the sender's address | All received RF data API frames indicate the source address |
| Advanced addressing support | API transmit and receive frames can expose addressing fields including source and destination endpoints, cluster ID, and profile ID |
| Advanced networking diagnostics | API frames can provide indication of I/O samples from remote devices, and node identification messages. |
| Remote Configuration | Set/read configuration commands can be sent to remote devices to configure them as needed using the API |

We recommend API mode when a device:

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

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

## Command mode

Command mode is a state in which the firmware interprets incoming characters as commands. It allows you to modify or read the device’s firmware using parameters you can set using AT commands.

See API operation for an alternate means of configuring devices, which is the only method available for SPI mode.

Command mode is available on the UART interface in both Transparent and API modes. You cannot use the SPI interface to enter Command mode.

## Enter Command mode

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

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

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

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

You can customize the guard times and timeout in the device's configuration settings. For information on how to do this, see CC (Command Mode Character), CT (Command Mode Timeout) and GT (Guard Times).

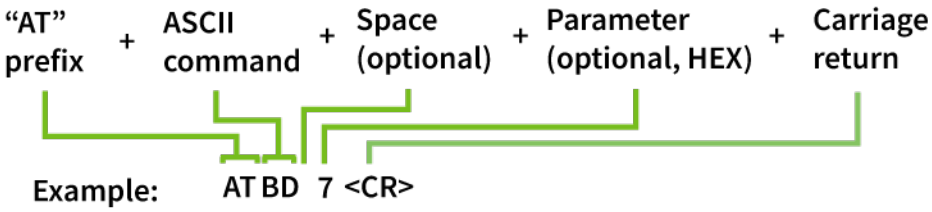
### Troubleshooting

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

## Send AT commands

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

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




The preceding example changes the device's baud rate to 7, which allows operation at 115,200 b/s.

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

### *Multiple AT commands*

You can send multiple AT commands at a time when they are separated by a comma in Command mode; for example, **ATSH**,**SL**.

#### *Parameter format*

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

#### Response to AT commands

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

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

### *Apply command changes*

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

1. Send the **AC** (Apply Changes) command.

   or:

2. Exit Command mode.

### *Exit Command mode*

1. Send the **CN** (Exit Command Mode) command followed by a carriage return.

   or:

2. If the device does not receive any valid AT commands within the time specified by **CT** (Command Mode Timeout), it returns to Idle Mode.

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

# Modes of operation

## Idle mode

When not receiving or transmitting data, the device is in Idle mode.

The device shifts into the other modes of operation under the following conditions:

- Transmit mode (serial data in the serial receive buffer is ready to be packetized).
- Receive mode (valid RF data received through the antenna).
- Sleep mode (Sleep mode condition is met).
- Command mode (Command mode sequence issued).

## Transmit mode

When the device receives serial data and is ready to packetize it, the device attempts to transmit the serial data. The destination address determines which node(s) will receive and send the data.

## Receive mode

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

## Configuration mode

You may not always know the parameters that the XBee Wi-Fi RF Module is configured with. If those parameters affect how the XBee Wi-Fi RF Module enters Command mode, and if 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 configuring the device difficult unless you can successfully guess the configuration to allow it to enter Command mode.

An example of this problem is when the UART baud rate is unknown. In this case, the **+++** sequence to enter Command mode is not recognized due to a baud rate mismatch, preventing the device from entering Command mode.

### *Force the device to enter Configuration mode*

To overcome the issue of unknown configuration parameters, you can force the XBee Wi-Fi RF Module into Command mode with a known configuration as follows:

While holding DIN low (asserting the break key), reset the device.

Rather than coming up in Transparent mode, which is normal, it comes up in Command mode and issues the **OK** prompt with the following default parameters applied for operation while in Command mode:

| Parameter setting | Meaning |
|---|---|
| **P3** = 1, **P4** = 1 | UART enabled—only set for SPI-enabled devices |
| **BD** = 3 | 9600 baud rate |
| **SB** = 0 | One stop bit |
| **NB** = 0 | No parity |
| **RO** = 3 | Three character times with no change on DIN before transmission |
| **D6** = 0 | No $\overline{\text{RTS}}$ flow control |
| **D7** = 1 | $\overline{\text{CTS}}$ flow control |
| **FT** | 65 characters left in transmission buffer before $\overline{\text{CTS}}$ is turned off |
| **CC** = 0x2b | **+** is used for Command mode character |
| **GT** = 0x3e8 | One second guard time |
| **CT** = 0x64 | Ten second Command mode timeout |

If the XBee Wi-Fi RF Module exits Configuration mode without changing any parameter values, then all parameters revert to their previous unknown state after it exits Command mode. Also, any values that you query return the previously written settings rather than the temporarily applied default settings described above.

### *Recover from an unknown configuration*

To recover from an unknown configuration to a known configuration, do the following:

1. Set up the interface to the XBee Wi-Fi RF Module to match the default configuration described in Force the device to enter Configuration mode.
2. Press and hold DIN low while resetting the XBee Wi-Fi RF Module.
3. Release DIN (let it be pulled high) so the device can receive UART data.
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. Use the **WR** command to write the desired configuration to non-volatile memory.
6. Set up the interface to the XBee Wi-Fi RF Module to match the configuration just written to non-volatile memory.
7. Optionally, reset the device and 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 instructions. XCTU does not work directly over a SPI interface.

1. Connect an asynchronous serial port of the PC (either RS-232 or USB) to the development board that the XBee Wi-Fi RF Module is plugged into.
2. Open XCTU.
3. To add your device to XCTU, see Add radio modules to XCTU in the XCTU User Guide.
4. The device(s) appear under the **Radio Modules** section on the left of the display.
5. To configure the settings, see Configure your modules in the XCTU User Guide.
6. When you are done entering the parameters, click the **Write module settings** button.

When the write is complete, all of the settings on the device are updated.

Click the **Consoles working mode** button on the toolbar and begin normal Transparent operation.

## Sleep mode

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

# Sleep modes

Sleep modes allow the device to enter states of low power consumption when not in use. The XBee Wi-Fi RF Module supports both pin sleep (sleep mode entered on pin transition) and cyclic sleep (device 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 modes for more information.

# Soft AP mode

The XBee Wi-Fi RF Module can operate in Soft AP mode, also known as Wi-Fi Direct. In this mode the XBee Wi-Fi RF Module emulates an access point (AP) rather than a station (STA). This allows another

Wi-Fi client device (STA) to connect to the XBee device 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 device operates in Soft AP mode in two different ways:

1. Provisioning mode
2. Pass through mode

You enable these two modes differently. To enable Pass through mode:

Set CE (Infrastructure Mode) to 1, which is not the default configuration. When **CE** is 1, it overrides parameters for Provisioning mode.

Provisioning mode is enabled by default. To disable it:

Clear bit 1 of DO command.

To enable Provisioning mode, SSID must be NULL. SSID is NULL by default and you can force it to NULL by issuing NR (Network Reset).

## Station (STA) connection in Soft AP Provisioning mode

When the device operates in Soft AP Provisioning mode, it waits for a connection from a STA device. Because the Service Set Identifier (SSID) is not configured, AI (Association Indication) is 0x23. The STA device must:

- Support Wi-Fi
- Have an HTTP browser operating on TCP port 80

Examples of devices that might connect to the XBee Wi-Fi RF Module operating in Soft AP mode are smart phones, tablets, and laptop computers.

The connecting STA device should scan for an **AP**. The XBee Wi-Fi RF Module advertises an SSID of:

xbee<MAC>

where <MAC> is the 6 byte MAC address of the XBee Wi-Fi RF Module formatted as follows:

xbee-XXXXXXXXXXXX

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 opens the webpage from the XBee Wi-Fi RF Module to allow you to configure it as desired. The primary purpose of this webpage is to configure the XBee Wi-Fi RF Module to connect to the desired access point with the desired security settings. The secondary purpose is to configure any other parameters.

## Use the webpage to configure a connected device

The webpage displays the current value of each configuration field.

1. Enter the desired parameters.
2. Click the **Apply** button at the bottom of the page.

The selected parameters are written to the non-volatile memory on the XBee Wi-Fi RF Module. However, the network access parameters are only written if you enter a valid set of parameters. The device tests the validity of those parameters by attempting to connect to the given access point.

- The Soft AP webpage provides the same configuration options that are available in XCTU.
- The webpage is divided into sections that expand or collapse by clicking **Show** or **Hide**. The only section expanded by default is the **Network Access** section.
- All fields proceeded with 0x must be hex values.

**Note** Do not programmatically configure the device in Soft AP mode because it is subject to change.

## Station (STA) connection in Soft AP Pass Through mode

When the XBee Wi-Fi RF Module operates in Soft AP Pass Through mode, it does not use HTTP on port 80 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 (Source Port) is open to accept connections, but no UDP listening socket is available.
- ID (SSID) specifies the SSID sent by the XBee Wi-Fi RF Module in the beacon, but if **ID** is NULL, it advertises an SSID based on the device's MAC address. For details, see Enable Soft AP mode.
- AI (Association Indication) is **0x23** while in Soft AP Provisioning mode (because **ID** is NULL), but **AI** is **0** in Soft AP Pass Through mode as soon as the XBee Wi-Fi RF Module is ready to accept a connection from a STA device. This is true whether or not **ID** is null.

# Sleep modes

# About sleep modes

The XBee Wi-Fi RF Module supports two different sleep modes:

- Pin Sleep
- Cyclic Sleep

In addition, you can modify the sleep mode current draw with the following sleep options:

- AP Associated Sleep
- Deep Sleep

Pin sleep allows an external microcontroller to determine when the XBee Wi-Fi RF Module should sleep and when it should wake by using either the SLEEP_RQ pin (default) or the SPI_SSEL pin. In contrast, cyclic sleep allows the sleep period and wake times to be configured using AT commands. The device 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.

Each of the sleep modes operate differently based on the serial interface (UART or SPI).

# Use the UART Sleep mode

When the serial interface is UART, the ON_SLEEP pin is used to indicate that the device is entering sleep mode, unless it is configured for a different usage. If you configure Sets or displays the DIO9 configuration (TH pin 13/SMT pin 26). for ON_SLEEP, then it is driven low when asleep and high when awake, whether using pin sleep or cyclic sleep.

If you enable CTS hardware flow control with D7 (DIO7 Configuration), the CTS pin is de-asserted (high) when entering sleep to indicate that serial data should not be sent to the device. The device will not respond to serial or RF data when it is sleeping. Applications that use the UART are encouraged to observe CTS flow control in any of the sleep modes. When the XBee Wi-Fi RF Module wakes from sleep with flow control enabled, the CTS pin is asserted (low).

If using pin sleep, you must configure D8 (DIO8 Configuration) for SLEEP_RQ to put the device to sleep. Otherwise, there is no sleep at all, meaning the device always stays awake in full power mode. When you configure **D8** for SLEEP_RQ, the host should drive SLEEP_RQ high to put the device to sleep, and the host should drive SLEEP_RQ low to wake up the device.

# Use the SPI Sleep mode

When the serial interface is SPI, SPI_ATTN 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 asserts SPI_ATTN and sends out a modem status indicator after initialization. The host can use this to know when the device is ready to operate as a SPI slave. Since the function of SPI_ATTN is to indicate when the device has data to send to the host, it may legitimately be driven high or low while the device is awake.

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

Using SPI_SSEL 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 device to sleep whenever the SPI master negates SPI_SSEL, even if that was not the intent. Therefore, if you can control SPI_SSEL, whether or not data needs to be transmitted, then sharing the pin may be a good option. It makes the SLEEP_RQ pin available for another purpose, or it simply requires one less pin to the SPI interface.

# AP Associated Sleep mode

This option allows the device to sync up with beacons sent from the Access Point (AP) which contains the Delivery Traffic Indication Message (DTIM). The DTIM indicates when broadcast and multicast data is 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 device 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 described in this user guide have this option enabled.

## Pin Sleep mode

UART data can be received in pin sleep mode, whether or not the host asserts the SLEEP_RQ pin. For example, if RF data is received by the device while SLEEP_RQ is asserted, the device wakes up long enough to send the data out the UART and then immediately resumes sleeping. If wake host is configured, the device asserts the appropriate I/O lines (indicating that it is awake), then waits for wake host timer to expire, then outputs the data, and then immediately resumes sleeping.

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

SPI operation is similar except that the device asserts $\overline{\text{ATTN}}$ when data becomes available and then the local host is expected to assert SPI_SSEL and to provide a clock until the data available is sent out.

When the local UART host needs to send data it de-asserts SLEEP_RQ. Once the appropriate status I/O lines are asserted (CTS and/or ON_SLEEP) the device 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_$\overline{\text{SSEL}}$. If SPI_$\overline{\text{SSEL}}$ is being used for pin sleep, asserting SPI_SSEL is enough to awaken the device to receive the incoming data. But, if SLEEP_RQ is being used to control sleep, then SPI_SSEL must be asserted and SLEEP_RQ must be de-asserted to awaken the device to receive the data. This wakes up the device, which then accepts the incoming data; however, data is queued and not sent until the next DTIM.

## Cyclic Sleep mode

The device remains associated to the Access Point (AP) and sleeps based on the SP (Sleep Period) parameter. After **SP** expires, the device wakes 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 device 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 device resumes sleep immediately, without waiting for **ST** time-out.

# Deep Sleep (Non-Associated Sleep) mode

This option allows the Wi-Fi circuitry to be powered down resulting in the lowest sleep current (about 6 µA) but at the expense of losing packets received during the time the device is asleep. This is

because the Access Point behaves like the device is in full power mode while it is asleep and it will not hold back packets until the device wakes up.

### Pin Sleep mode

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

### Cyclic Sleep mode

In this mode the device enters and exits sleep based on the SP (Sleep Period), ST (Wake Time), and SA commands. **SP** specifies the sleep time and **ST** specifies the wake time of the device 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 device sleeps anyway. When it wakes from this state, then it starts 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.

## Use sleep modes to sample data

To sample data when waking from any sleep mode:

1. Enable an ADC or digital input.
2. Set Set or read the I/O sample rate to enable periodic sampling. appropriately with respect to ST (Wake Time) to obtain the desired number of samples.

If you want multiple samples during the wake period then user **IR**. This provides **ST**/**IR**+1 samples. Each sample is sent separately.

WH (Wake Host) 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.

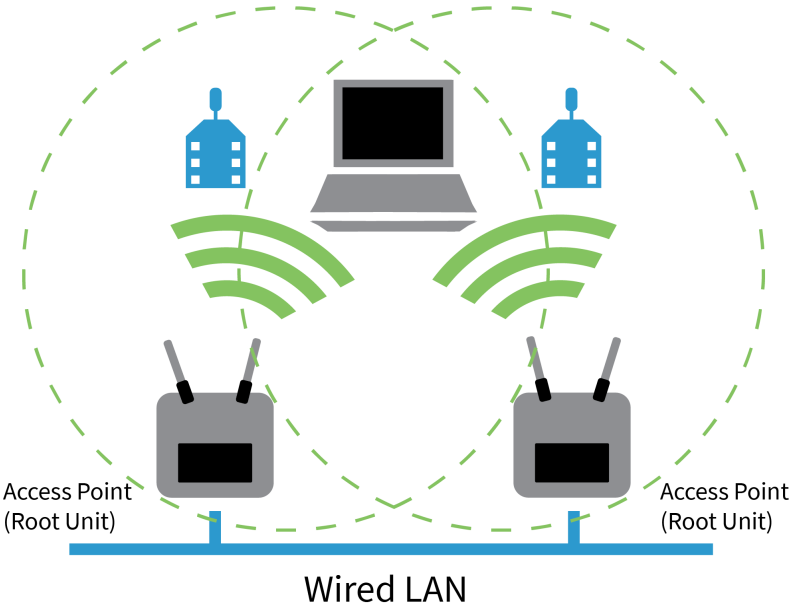
Digital outputs and special function outputs such as ON_$\overline{\text{SLEEP}}$ and $\overline{\text{CTS}}$ are not affected by **WH**. This is to allow these signals to be used to wake up devices.

For deep sleep, **WH** must be expired and the device must be associated before I/O samples are taken.

# 802.11 bgn networks

# Infrastructure networks

The main type of wireless network involve a number of wireless devices called stations talking through a master wireless device known as an access point (AP) or station (STA). This type of setup is called an Infrastructure or Basic Service Set (BSS) network. Most wireless networks are of this type. The following illustration is an example of an infrastructure wireless network.




## Infrastructure Wireless Network

By default, the XBee Wi-Fi RF Module operates as a STA in the infrastructure network, which means it associates to an AP and all data to and from the device goes through that AP.

If you configure CE (Infrastructure Mode) to 1, the device takes the position of an AP in the network, allowing STA devices to associate to the XBee Wi-Fi RF Module operating in Soft AP mode.

# Ad Hoc networks

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

## Set Ad Hoc creator parameters

Set the following parameters for the creator:

| Parameter | Function |
|---|---|
| **AH1** | Designates the node as an Ad Hoc creator. |
| **MA1** | Specifies the static IP addresses. No DHCP is supported in Ad Hoc mode. |

| Parameter | Function |
|---|---|
| **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 is not the same as another SSID in the vicinity. |
| **MY** | Sets the IP address of the creator node. |
| **DL** | Specifies the IP address of the joiner node. |
| **MK** | Sets the IP mask for both of the above addresses. |

### Set Ad Hoc joiner parameters

Set the following parameters for the joiner:

| Parameter | Function |
|---|---|
| **AH0** | Designates the node as an Ad Hoc joiner. |
| **MA1** | Specifies the 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 the IP address of the joiner node. |
| **DL** | Specifies the IP address of the creator node. |
| **MK** | Sets the IP mask for both of the above addresses. |

## Network basics

Clients 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.

## 802.11 standards

The XBee Wi-Fi RF Module operates in three of the available 802.11 standards, they are:

**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 300 Mb/s. The XBee Wi-Fi module uses the single stream n mode with 20 MHz 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 RF 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 RF Module supports Open and Shared Key authentication in WEP mode and it only supports shared key authentication in WPA and WPA2 modes.

## Open authentication

Open authentication is when the access point simply accepts the wireless devices identity without verifying or proving it. The benefits to this are 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 fails 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 occurs with an incorrect WEP key. Instead, **AI** gets 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.

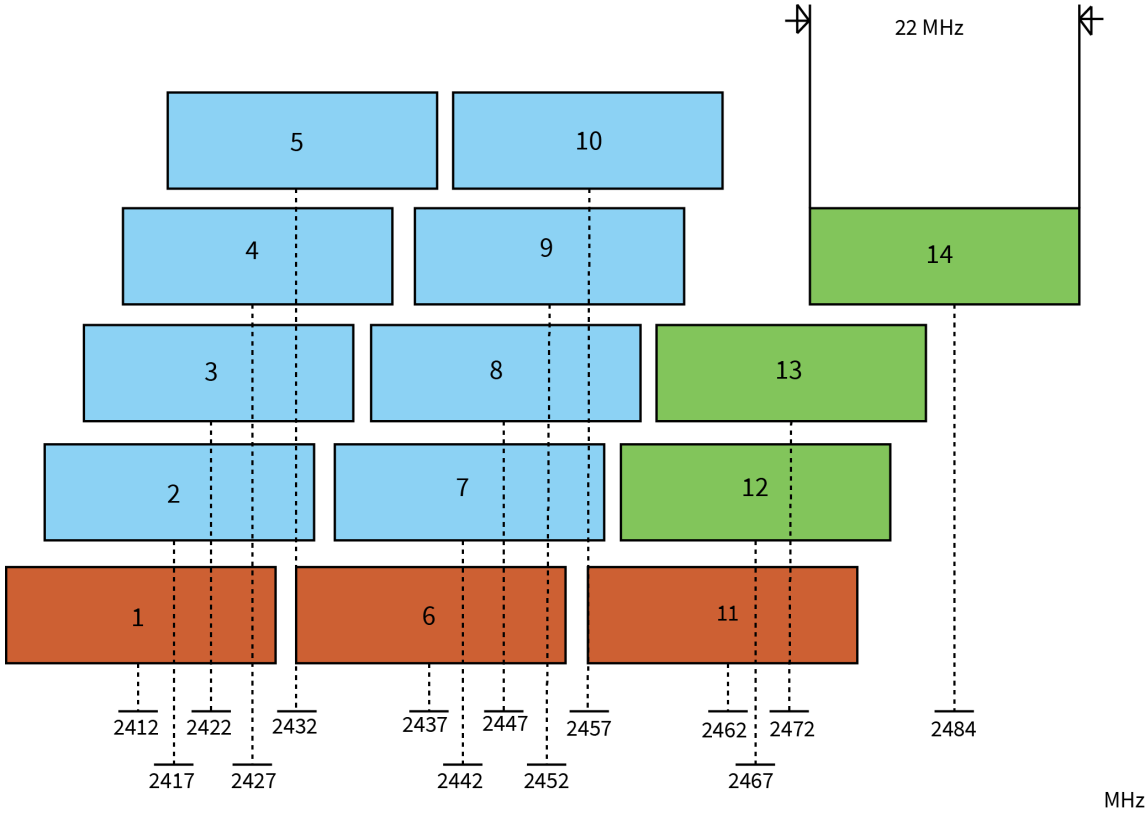
## Shared Key

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 RF Module operates in the 2412 - 2472 MHz range. The frequency range is broken down into thirteen 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 you select the channels of neighboring access points accordingly.

The center frequencies of the thirteen 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.



# IP services

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

# XBee Application Service

This service primarily provides for XBee device configuration. It also provides API compatibility for customers who have designed around other XBee devices. 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.

Access the service from a local host or network client. Use C0 (Serial Communication Service Port) and DE command to configure the 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 access

From a local host, access XBee Application Service using API frames. There are remote AT command frames as well as transmission frames. The API frames are:

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

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

### *Transmit and receive data*

#### Transmit data

The local host uses the TX64 frame to send data to another XBee device using this service. When the device receives the frame through the serial port it converts the contents of the frame to a serial data transfer command as defined by the XBee Application Service.

#### Receive data

A received serial data transfer command goes to the serial port. The serial port's mode determines the data format. When in API mode, the data is sent to the host using the RX 64-bit frame.

**Note** We do not recommended using this service to send data to a network client. Use the serial communication service.

### *Change configuration on a remote device*

Use the Remote AT command frame to change configuration on a remote device. See Remote AT Command Request - 0x07 for more information.

### Update the firmware

To perform firmware updates 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 sequence of operations to follow for firmware updates, see Update the firmware over-the-air.

ZigBee Explicit Transmit Packet - 0x11 has this name because it has fields that are uniquely ZigBee, such as endpoint and cluster ID. To perform firmware updates the same on all XBee devices, we 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 access

To access this port, send 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's 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<br>0x02 = Remote Command<br>0x03 = General Purpose Memory Command<br>0x04 = I/O Sample<br>0x80 = Data Acknowledgment<br>0x82 = Response to remote command<br>0x83 = Response to General Purpose Memory Command |
| Command options | 7 | 1 | bit 0 – encrypted if set (Reserved for later use)<br>bit 1 – set to request an ACK<br>bits 2:7 - unused (Set to 0 for forward compatibility.) |

All of the commands and command responses that follow are preceded with this application header.

### *Send configuration commands*

A network client can send AT commands to the XBee Wi-Fi RF Module. 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 | Indicates remote AT command. |
| | Command options | 7 | 0x00 | Options are not available for this command. |
| 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.<br>2 – Apply changes to all changed commands. |
| | AT command | 10 | 0x49 (I) | Command Name - Two ASCII characters that identify the AT command. |
| | | 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. |
| 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<br>1 = ERROR<br>2 = Invalid Command<br>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) | |

### *Send the serial data command*

Using the IP service to send data out the serial port is not required. Most users choose to use the Serial Communication Service to send 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 device 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 device'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. |

### Receive I/O sample data

Sample data generated by the device is sent to the address configured by the **DL** commands. This data can be sent to another XBee device or to a network client. It is sent using UDP from the 0xBEE port as with other XBee Application services. Sample data is 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. |
| | 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 2.5 V, 0x200 represents 1.25 volts on ADC1 in this example. |
| | | LSB 15 | 0x00 | |

### Send over-the-air firmware updates

A network client can also use the XBee IP services to send a firmware update to the device. 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 provided in Network client access. See General Purpose Flash Memory 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 update process.

| Packet fields | | Offset | Example | Description |
|---|---|---|---|---|
| Application header | Number1 | 0 | 0x4242 | This is an easy number to create an accepted frame. |
| | Number2 | 2 | 0x0000 | Number1 ^ 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 | Do not 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. |

# Serial Communication Service

The Serial Communication Service connects an IP port to the serial peripheral (UART or SPI) of the XBee Wi-Fi RF Module. No additional formatting or header is required and data is transferred between the RF hardware and Serial Communication hardware as received.

Use C0 (Serial Communication Service Port) and DE command to configure the IP ports. 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.

## 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 IP (IP Protocol). 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 operating mode.

## UDP

When IP (IP Protocol) is configured for User Datagram Protocol (UDP), serial data is sent to the IP address specified by DL (Destination Address) and it is sent to the UDP port specified by DE command. The source of the packet is defined by C0 (Serial Communication Service Port). No connection is established

## TCP

When IP (IP Protocol) is configured for Transmission Control Protocol (TCP), only one connection is allowed at a time. If a transmission is attempted while a TCP connection exists, the data is sent on that connection, ignoring the DL (Destination Address) and DE command 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 is created based on the **DL (IP address)** and **DE** commands. However, if **DL** is a broadcast address, then UDP is used, ignoring the TCP configuration.

A network client establishing a TCP connection to the XBee Wi-Fi RF Module uses the port defined by C0 (Source Port). When established, any data sent by the local host does not create a new connection based on **DL** and **DE**, but rather uses the existing connection.

## API mode

API mode allows you to specify the protocol (UDP or TCP) destination address, destination port, and source port for transmission.

## UDP mode

If you specify UDP mode in the Transmit (TX) Request: IPv4 - 0x20, 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 C0 (Serial Communication Service Port).

## 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 Transmit (TX) Request: IPv4 - 0x20 is sent to the device, 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. 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 (Serial Communication Service Port)] of the receiving device 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 the data is sent to the listening socket of the other device 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.

# I/O support

# Analog and digital I/O lines

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

## Through-hole device

| Pin names | Device pin | AT command | Command range | Default value |
|---|---|---|---|---|
| DIO1/AD1/ SPI_$\overline{\text{ATTN}}$ | 19 | **D1** | 0-5 | 0 |
| DIO2/AD2/SPI_CLK | 18 | **D2** | 0-5 | 0 |
| DIO3/AD3/SPI_$\overline{\text{SSEL}}$ | 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/$\overline{\text{RTS}}$ | 16 | **D6** | 0-1, 3-5 | 0 |
| DIO7/$\overline{\text{CTS}}$ | 12 | **D7** | 0-1, 3-7 | 1 |
| DIO8/$\overline{\text{DTR}}$/SLEEP_RQ | 9 | **D8** | 0-1, 3-5 | 1 |
| DIO9/On_$\overline{\text{SLEEP}}$ | 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/$\overline{\text{CONFIG}}$ | 3 | **P4** | 0-1 | 1 |

## Surface-mount device

| Pin names | Device pin | AT command | Command range | Default value |
|---|---|---|---|---|
| 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/$\overline{\text{RTS}}$ | 29 | **D6** | 0-1, 3-5 | 0 |
| DIO7/$\overline{\text{CTS}}$ | 25 | **D7** | 0-1, 3-7 | 1 |

| Pin names | Device pin | AT command | Command range | Default value |
|---|---|---|---|---|
| DIO8/$\overline{\text{DTR}}$/SLEEP_RQ | 10 | **D8** | 0-1, 3-5 | 1 |
| DIO9/On_$\overline{\text{SLEEP}}$ | 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/$\overline{\text{CONFIG}}$ | 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_$\overline{\text{SSEL}}$ | 15 | **P7** | 0-1, 4-5 | 1 |
| DIO18/SPI_CLK | 14 | **P8** | 0-1, 4-5 | 1 |
| DIO19/SPI_$\overline{\text{ATTN}}$ | 12 | **P9** | 0-1, 4-6 | 1 |

# Configure I/O functions

To enable an analog or digital I/O function on one or more device pin(s):

1. Issue the appropriate configuration command with the correct parameter.
2. Apply the changes on the device for the I/O settings to take effect.

You can use PR (Pull-up Resistor) to set pull-up/down resistors for each digital input line. The **PR** value enables or disables the pull-up/down resistors, and PD (Pull Direction) determines if a pull-up or pull-down is used. Internal pull-up/down resistors are not available for digital output pins, analog input pins, or for disabled pins.

| Pin command parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Peripheral control |
| 2 | Analog input or PWM output |
| 3 | Data in monitored |
| 4 | Data out default low |
| 5 | Data out default High |
| 6 | RS-485 enable low |
| 7 | RS-485 enable high |
| >7 | Unsupported |

# I/O sampling

The XBee Wi-Fi RF Module has 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 device, or sent to Remote Manager.

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

- Queried sampling
- Periodic sampling
- Change detection sampling

I/O sample data is formatted as shown in following table.

| Bytes | Name | Description |
|---|---|---|
| 1 | Sample Sets | Number of sample sets in the packet. Always set to 1. |
| 2 | Digital channel mask | Digital I/O line on the device.<br>bit 0 = DIO0<br>bit 1 = DIO1<br>bit 2 = DIO2<br>bit 3 = DIO3<br>bit 4 = DIO4<br>bit 5 = DIO5<br>bit 6 = DIO6<br>bit 7 = DIO7<br>bit 8 = DIO8<br>bit 9 = DIO9<br>bit 10 = DIO10<br>bit 11 = DIO11<br>bit 12 = DIO12<br>For example, a digital channel mask of 0x002F means DIO0 1, 2, 3, and 5 are enabled as digital I/O. |
| 1 | Analog channel mask | Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel.<br>bit 0 = AD0<br>bit 1 = AD1<br>bit 2 = AD2<br>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 I/O. 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 device, these two bytes will be omitted.<br>Following the digital I/O data (if any), each enabled analog channel will return two bytes. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3. |

The sampled data set includes two bytes of digital I/O data only if one or more I/O lines on the device are configured as digital I/O. If no pins are configured as digital I/O, these two 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 0 V, and 0x3FF = VREF. VREF may be either 1.25 V or 2.5 V based on the setting of AV (Analog Voltage Reference), where 2.5 V is the default. The analog inputs on the device 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 (mV) = (A/D reading (converted to decimal) * VREF) / 1023 where VREF may be 1250 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

IS (Force Sample) can be sent to a device locally, or to a remote device using the API remote command frame. 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** returns an 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 device that sent the **IS** command.

If the **IS** command is issued in Command mode, the device returns a carriage return-delimited list containing the above-listed fields. If the **IS** command is issued in API mode, the device 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 Wi-Fi RF Module to take an I/O sample and transmit it to a remote device at a periodic rate. The periodic sample rate is set by Set or read the I/O sample rate to enable periodic sampling. . 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 is sampled after **IR** milliseconds have elapsed and transmitted to a remote device. However, the device 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 Remote Manager is enabled (see Network commands), samples are sent as a data stream. See Transparent mode data 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 Remote Manager and to **DL**.

When Remote Manager is not enabled, the I/O sample is sent to the address specified by DL (Destination Address). When **DL** points to another device, that device must have API mode enabled. Otherwise, the data is dropped by the receiving device and is not sent out the serial port.

When **DL** points to a network client, the I/O sample is sent to that network client. See Network client access for the format of I/O samples sent to a network client.

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

# Change detection sampling

Devices 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. IC (Digital Change Detection) 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 is 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 (Destination Address), to Remote Manager, or to both, depending on the setting of DO command. Viewing I/O samples on the remote device or Remote Manager is the same for change detect sampling as it is for periodic sampling.

## Example

Configure the following I/O settings on the XBee Wi-Fi RF Module:

1. To configure DIO1/AD1 as a digital input, issue D1 (DIO1/AD1 Configuration) with a parameter of 3 (**ATD13**).
2. To enable pull-up resistors on the same pin, issue PR (Pull-up Resistor) with bit 3 set (for example **ATPR8**, **ATPR1FFF**, and so forth).
3. To configure DIO2/AD2 as an analog input, issue D2 (DIO2/AD2 Configuration) with a parameter of 2 to enable the analog input (**ATD22**).
4. To configure DIO4 as a digital output, driving high, issue D4 (DIO4/AD4 Configuration) with a parameter value of 5 (**ATD45**).
5. After issuing these commands, apply the changes so the device's I/O pins update to the new states. Issue AC (Apply Changes) or CN (Exit Command mode) to apply the changes.

# RSSI PWM

The XBee Wi-Fi RF Module features an RSSI/PWM pin (DIO10) that, if enabled, adjusts the PWM output to indicate the signal strength of the last received packet. Use P0 (DIO10 Configuration) to enable the RSSI pulse width modulation (PWM) output on the pin. If **P0** is set to 1, the RSSI/PWM pin outputs a PWM 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 is enabled for a time based on RP (RSSI PWM Timer). 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 100 ms 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 |

# Wi-Fi Protected Setup (WPS)

The XBee Wi-Fi RF Module may 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 device 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.

## Enable WPS

WPS is enabled by default, but it is disabled if SSID is configured [ID (SSID) is not NULL] or if the device is connected in Soft AP mode.

To use WPS with the Commissioning button, enable it by configuring D0 (DIO0/AD0/ CB Configuration) to 1.

## Use WPS

To invoke WPS:

1. Press the Commissioning button twice if D0 (DIO0/AD0/ CB Configuration) is set to 1.

or

1. Enter **CB2** to use the CB (Commissioning Pushbutton) command.
2. Then, you must press a corresponding WPS button on a nearby WPS-capable access point (AP) , which allows the security parameters to be exchanged and the connection to the AP to occur.

## Pre-shared key (PSK) mode security

Pre-shared key (PSK) mode, also known as Personal mode, is designed for home and small office networks that do not require the complexity of an 802.1X authentication server. Each wireless network device encrypts the network traffic using a 256 bit key. You can enter this key either as a string of 64 hexadecimal digits, or as a passphrase of 8 to 63 printable ASCII characters. If you use ASCII characters, 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.

# General Purpose Flash Memory

# General Purpose Flash Memory

XBee Wi-Fi RF Modules provide 160 4096-byte blocks of flash memory that an application can read and write to. This memory provides a non-volatile data storage area that an application uses for many purposes. Some common uses of this data storage include:

- Storing logged sensor data
- Buffering firmware update data for a host microcontroller
- Storing and retrieving data tables needed for calculations performed by a host microcontroller

The General Purpose Memory (GPM) is also used to store a firmware update file for over-the-air firmware updates of the device itself.

# Work with flash memory

When working with the General Purpose Memory, observe the following limitations:

- Flash memory write operations are only capable of changing binary 1s to binary 0s. Only the erase operation can change binary 0s to binary 1s. For this reason, you should erase a flash block before performing a write operation.
- When performing an erase operation, you must erase the entire flash memory block—you cannot erase parts of a flash memory block.
- Flash memory has a limited lifetime. The flash memory on which the GPM is based is rated at 20,000 erase cycles before failure. Take care to ensure that the frequency of erase/write operations allows for the desired product lifetime. Digi's warranty does not cover products that have exceeded the allowed number of erase cycles.
- Over-the-air firmware upgrades erase the entire GPM. Any user data stored in the GPM will be lost during an over-the-air upgrade.

# Access General Purpose Flash Memory

The GPM of a target node can be accessed from the XBee Wi-Fi RF Module's serial port or from a non-XBee network client.

To access the GPM of a target node locally or over-the-air, send commands to the MEMORY_ACCESS cluster ID (0x23) on the DIGI_DEVICE endpoint (0xE6) of the target node using explicit API frames. Explicit API frames have frame identifier 0x11. For a description of Explicit API frames, see API operation.

Access from a non-XBee network client is performed 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 access to learn how to format the application header.

Use the following header to generate a GPM command. Use it whether you use serial port access or network client access. For network client access, an application header needs to precede the GPM header. This section describes the perspective of serial port access, without the application header. Do not forget to precede each frame with an application header if you use 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 in detail in the topics that follow. |
| 1 | 1 | GPM_OPTIONS | Command-specific options. |
| 2 | 2* | GPM_BLOCK_NUM | The block number addressed in the GPM.. Ranges from 0 to 159 (0x9F). |
| 4 | 2* | GPM_START_INDEX | The byte index within the addressed GPM block. |
| 6 | 2* | GPM_NUM_BYTES | The number of bytes in the GPM_DATA field, or in the case of a READ, the number of bytes requested. |
| 8 | varies | GPM_DATA | |
| * Specify multi-byte parameters with big-endian byte ordering. | | | |

When a device sends a GPM command to another device via a unicast, the receiving device sends a unicast response back to the requesting device's source endpoint specified in the request packet. It does not send a response for broadcast requests. If the source endpoint is set to the DIGI_DEVICE endpoint (0xE6) or Explicit API mode is enabled on the requesting device, then the requesting node outputs a GPM response as an explicit API RX indicator frame (assuming it has API mode enabled).

The format of the response is similar to the request packet:

| Byte offset in payload | Number of bytes | Field name | General field description |
|---|---|---|---|
| 0 | 1 | GPM_CMD_ID | This field is the same as the request field. |
| 1 | 1 | GPM_STATUS | Status indicating whether the command was successful. |
| 2 | 2* | GPM_BLOCK_NUM | The block number addressed in the GPM.. Ranges from 0 to 159 (0x9F). |
| 4 | 2* | GPM_START_INDEX | The byte index within the addressed GPM block. |
| 6 | 2* | GPM_NUM_BYTES | The number of bytes in the GPM_ DATA field. |
| 8 | varies | GPM_DATA | |
| * Specify multi-byte parameters with big-endian byte ordering. | | | |

# General Purpose Flash Memory commands

This section provides information about commands that interact with GPM:

## PLATFORM_INFO_REQUEST (0x00)

A PLATFORM_INFO_REQUEST frame can be sent to query details of the GPM structure.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to PLATFORM_INFO_REQUEST (0x00). |
| GPM_OPTIONS | This field is unused for this command. Set to 0. |
| GPM_BLOCK_NUM | This field is unused for this command. Set to 0. |
| GPM_START_INDEX | This field is unused for this command. Set to 0. |
| GPM_NUM_BYTES | This field is unused for this command. Set to 0. |
| GPM_DATA | No data bytes should be specified for this command. |

## PLATFORM_INFO (0x80)

When a PLATFORM_INFO_REQUEST command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to PLATFORM_INFO (0x80). |
| GPM_STATUS | A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time. |
| GPM_BLOCK_NUM | Indicates the number of GPM blocks available. |
| GPM_START_INDEX | Indicates the size, in bytes, of a GPM block. |
| GPM_NUM_BYTES | The number of bytes in the GPM_DATA field. For this command, this field will be set to 0. |
| GPM_DATA | No data bytes are specified for this command. |

### *Example*

A PLATFORM_INFO_REQUEST sent to a device with a serial number of 0x0013a200407402AC should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 00 00 0000 0000 0000 24

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 80 00 0077 0200 0000 EB

## ERASE (0x01)

The ERASE command erases (writes all bits to binary 1) one or all of the GPM flash blocks. You can also use the ERASE command to erase all blocks of the GPM by setting the GPM_NUM_BYTES field to 0.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to ERASE (0x01). |
| GPM_OPTIONS | There are currently no options defined for the ERASE command. Set this field to 0. |
| GPM_BLOCK_NUM | Set to the index of the GPM block that should be erased. When erasing all GPM blocks, this field is ignored (set to 0). |
| GPM_START_INDEX | The ERASE command only works on complete GPM blocks. The command cannot be used to erase part of a GPM block. For this reason GPM_START_INDEX is unused (set to 0). |
| GPM_NUM_BYTES | Setting GPM_NUM_BYTES to 0 has a special meaning. It indicates that every flash block in the GPM should be erased (not just the one specified with GPM_BLOCK_NUM). In all other cases, the GPM_NUM_BYTES field should be set to the GPM flash block size. |
| GPM_DATA | No data bytes are specified for this command. |

## ERASE_RESPONSE (0x81)

When an ERASE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to ERASE_RESPONSE (0x81). |
| GPM_STATUS | A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time. |
| GPM_BLOCK_NUM | Matches the parameter passed in the request frame. |
| GPM_START_INDEX | Matches the parameter passed in the request frame. |
| GPM_NUM_BYTES | The number of bytes in the GPM_DATA field. For this command, this field will be set to 0. |
| GPM_DATA | No data bytes are specified for this command. |

### *Example*

To erase flash block 42 of a target radio with serial number of 0x0013a200407402ac format an ERASE packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 01 00 002A 0000 0200 37

Assuming all transmissions were successful, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 81 00 002A 0000 0000 39

## WRITE (0x02) and ERASE_THEN_WRITE (0x03)

The WRITE command writes the specified bytes to the GPM location specified. Before writing bytes to a GPM block it is important that the bytes have been erased previously. The ERASE_THEN_WRITE command performs an ERASE of the entire GPM block specified with the GPM_BLOCK_NUM field prior to doing a WRITE.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to WRITE (0x02) or ERASE_THEN_WRITE (0x03). |
| GPM_OPTIONS | There are currently no options defined for this command. Set this field to 0. |
| GPM_BLOCK_NUM | Set to the index of the GPM block that should be written. 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 | Set to the number of bytes specified in the GPM_DATA field. 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.<br>**Note** If this parameter is zero, the command erases the entire GPM and writes nothing.<br>Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. The maximum payload size can be queried with the **NP** command. |
| GPM_DATA | The data to be written. |

## WRITE _RESPONSE (0x82) and ERASE_THEN_WRITE_RESPONSE (0x83)

When a WRITE or ERASE_THEN_WRITE command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to WRITE_RESPONSE (0x82) or ERASE_THEN_WRITE_RESPONSE (0x83) |
| GPM_STATUS | A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time |
| GPM_BLOCK_NUM | Matches the parameter passed in the request frame |
| GPM_START_INDEX | Matches the parameter passed in the request frame |

| Field name | Command-specific description |
|---|---|
| GPM_NUM_BYTES | The number of bytes in the GPM_DATA field. For this command, this field will be set to 0 |
| GPM_DATA | No data bytes are specified for these commands |

**Example**

To write 15 bytes of incrementing data to flash block 22 of a target radio with serial number of 0x0013a200407402ac a WRITE packet should be formatted as follows (spaces added to delineate fields):

7E 002B 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 02 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C5

Assuming all transmissions were successful and that flash block 22 was previously erased, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 82 00 0016 0000 0000 4C

## READ (0x04)

You can use the READ command to read the specified number of bytes from the GPM location specified. Data can be queried from only one GPM block per command.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to READ (0x04). |
| GPM_OPTIONS | There are currently no options defined for this command. Set this field to 0. |
| GPM_BLOCK_NUM | Set to the index of the GPM block that should be read. 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. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. You can query the maximum payload size with the **NP** AT command. |
| GPM_DATA | No data bytes should be specified for this command. |

## READ_RESPONSE (0x84)

When a READ command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to READ_RESPONSE (0x84). |
| GPM_STATUS | A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time. |
| GPM_BLOCK_NUM | Matches the parameter passed in the request frame. |
| GPM_START_INDEX | Matches the parameter passed in the request frame. |
| GPM_NUM_BYTES | The number of bytes in the GPM_DATA field. |
| GPM_DATA | The bytes read from the GPM block specified. |

### *Example*

To read 15 bytes of previously written data from flash block 22 of a target radio with serial number of 0x0013a200407402ac a READ packet should be formatted as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 C0 04 00 0016 0000 000F 3B

Assuming all transmissions were successful and that flash block 22 was previously written with incrementing data, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 0029 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 84 00 0016 0000 000F 0102030405060708090A0B0C0D0E0F C3

## FIRMWARE_VERIFY (0x05) and FIRMWARE_VERIFY_AND_INSTALL (0x06)

Use the FIRMWARE_VERIFY and FIRMWARE_VERIFY_AND_INSTALL commands when remotely updating firmware on a device. For more inoformation about firmware updates. These commands check if the GPM contains a valid over-the-air update file. For the FIRMWARE_VERIFY_AND_INSTALL command, if the GPM contains a valid firmware image then the device resets and begins using the new firmware.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to FIRMWARE_VERIFY (0x05) or FIRMWARE_ VERIFY_AND_INSTALL (0x06) |
| GPM_OPTIONS | There are currently no options defined for this command. Set this field to 0. |
| GPM_BLOCK_NUM | This field is unused for this command. Set to 0. |
| GPM_START_INDEX | This field is unused for this command. Set to 0. |
| GPM_NUM_BYTES | This field is unused for this command. Set to 0. |
| GPM_DATA | This field is unused for this command |

## FIRMWARE_VERIFY_RESPONSE (0x85)

When a FIRMWARE_VERIFY command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to FIRMWARE_VERIFY_RESPONSE (0x85) |
| GPM_STATUS | A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. A 0 in the least significant bit indicates the GPM does contain a valid firmware image. All other bits are reserved at this time. |
| GPM_BLOCK_NUM | This field is unused for this command. Set to 0. |
| GPM_START_INDEX | This field is unused for this command. Set to 0. |
| GPM_NUM_BYTES | This field is unused for this command. Set to 0. |
| GPM_DATA | This field is unused for this command |

## FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86)

When a FIRMWARE_VERIFY_AND_INSTALL command request has been unicast to a node, that node sends a response in the following format to the source endpoint specified in the requesting frame only if the GPM memory does not contain a valid image. If the image is valid, the device resets and begins using the new firmware.

| Field name | Command-specific description |
|---|---|
| GPM_CMD_ID | Should be set to FIRMWARE_VERIFY_AND_INSTALL_RESPONSE (0x86). |
| GPM_STATUS | A 1 in the least significant bit indicates the GPM does not contain a valid firmware image. All other bits are reserved at this time. |
| GPM_BLOCK_NUM | This field is unused for this command. Set to 0. |
| GPM_START_INDEX | This field is unused for this command. Set to 0. |
| GPM_NUM_BYTES | This field is unused for this command. Set to 0. |
| GPM_DATA | This field is unused for this command. |

**Example**

To verify a firmware image previously loaded into the GPM on a target device with serial number 0x0013a200407402ac, format a FIRMWARE_VERIFY packet as follows (spaces added to delineate fields):

7E 001C 11 01 0013A200407402AC FFFE E6 E6 0023 C105 00 00 05 00 0000 0000 0000 1F

Assuming all transmissions were successful and that the firmware image previously loaded into the GPM is valid, the following API packets would be output the source node's serial interface:

7E 0007 8B 01 FFFE 00 00 00 76

7E 001A 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 85 00 0000 0000 0000 5F

# Update the firmware over-the-air

This section provides instruction on how to update your firmware using wired updates and over-the-air updates.

## Over-the-air firmware updates

There are two methods of updating the firmware on the device. You can update the firmware locally with XCTU using the device's serial port interface. You can also update firmware using the device's RF interface (over-the-air updating.)

The over-the-air firmware update method provided is a robust and versatile technique that you can tailor to many different networks and applications. OTA updates are reliable and minimize disruption of normal network operations.

In the following sections, we refer to the node that will be updated as the target node. We refer to the node providing the update information as the source node. In most applications the source node is locally attached to a computer running update software.

There are three phases of the over-the-air update process:

1. Distribute the new application
2. Verify the new application
3. Install the application

## Distribute the new application

The first phase of performing an over-the-air update on a device is transferring the new firmware file to the target node. Load the new firmware image in the target node's GPM prior to installation. XBee Wi-Fi RF Modules use an encrypted binary (.ebin) file for both serial and over-the-air firmware updates. These firmware files are available on the Digi Support website and via XCTU.

Send the contents of the .ebin file to the target device using general purpose memory WRITE commands. Erase the entire GPM prior to beginning an upload of an .ebin file. The contents of the .ebin file should be stored in order in the appropriate GPM memory blocks. The number of bytes that are sent in an individual GPM WRITE frame is flexible and can be catered to the user application.

### *Example*

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:

| GPM_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 |

| GPM_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 |

## Verify the new application

For an uploaded application to function correctly, every single byte from the .ebin file must be properly transferred to the GPM. To guarantee that this is the case, GPM VERIFY functions exist to ensure that all bytes are properly in place. The FIRMWARE_VERIFY function reports whether or not the uploaded data is valid. The FIRMWARE_VERIFY_AND_INSTALL command reports if the uploaded data is invalid. If the data is valid, it begins installing the application. No installation takes place on invalid data.

## Install the application

When the entire .ebin file is uploaded to the GPM of the target node, you can issue a FIRMWARE_ VERIFY_AND_INSTALL command. Once the target receives the command it verifies the .ebin file loaded in the GPM. If it is valid, then the device installs the new firmware. This installation process can take up to eight seconds. During the installation the device is unresponsive to both serial and RF communication. To complete the installation, the target module resets. AT parameter settings which have not been written to flash using the **WR** command will be lost.

### *Important considerations*

The firmware upgrade process requires that the device resets itself. Because of this reset parameters which have not been written to flash are 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 device in either case.

# Configure the XBee Wi-Fi RF Module using Digi Remote Manager

Use Digi Remote Manager (https://remotemanager.digi.com/) to perform the operations in this section. Each operation requires that you enable Remote Manager 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 Digi Remote Manager.

**Note** Digi is consolidating our cloud services, Digi Device Cloud and Digi Remote Manager®, under the Remote Manager name. This phased process does not affect device functionality or the functionality of the web services and other features. However, customers will find that some user interface and firmware functionality mention both Device Cloud and Digi Remote Manager.

## Use XCTU to enable Remote Manager

To enable an XBee Wi-Fi RF Module connected to XCTU for Remote Manager:

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

## Configure the device

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

1. Log in to Remote Manager.
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, Remote Manager 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 Remote Manager.
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 Remote Manager.
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 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 device 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 I/O 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 | I/O pin |
|---|---|
| 0 | DIO0 |
| 1 | DIO1 |

| Bit | I/O pin |
|---|---|
| 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 Remote Manager

To send I/O samples to Remote Manager:

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

If Remote Manager is not enabled, then I/O samples go to the address specified by the **DL** command. See Periodic I/O sampling.

## View I/O samples in Remote Manager

1. In your Remote Manager 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 Remote Manager 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 Remote Manager 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 Remote Manager. To send a request, a host connected to the serial port of the device sends a Send Data Request - 0x28 API frame.

## Enable messages to the host

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

1. In Remote Manager, 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 - 0x2A, the XBee Wi-Fi RF Module sends a timeout response to Remote Manager.
4. After the host sends a Device Response frame (0x2A), the XBee Wi-Fi RF Module sends a Device Response Status frame - 0xBA to the host.

# 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 device 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)

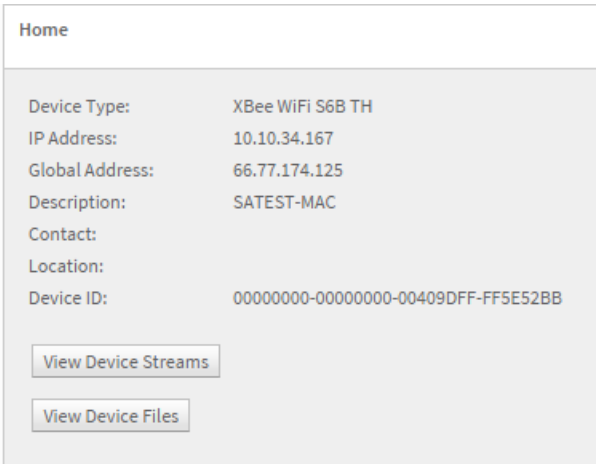
1. In Remote Manager, click **Documentation** > **API Explorer**.
2. Click **Examples** > **SCI** > **Data Service** > **Send request**. A sample XML file that you can modify appears as seen in the following code sample:

```
<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 Wi-Fi RF 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 Remote Manager, 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 device 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 device 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 Remote Manager transmissions and receptions in Transparent mode. Transparent data is sent to Remote Manager using the Send Data interface and Transparent data is received from Remote Manager using the Device Request interface. Some parts of those interfaces are lost due to not using the API interface.

## Send data to Remote Manager

The device can send serial data to Remote Manager as files or as binary data points. To make this selection, use bit 4 of DO command. If you set **DO** bit 4, then the serially-connected host sends Transparent data to Remote Manager 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.

## AT command settings to put serial data in Remote Manager

API Enable command:

0: Transparent Mode

1: API Mode without escapes

2: API Mode with escapes

EQ (Device Cloud FQDN) command:

The URL for Remote Manager must be my.devicecloud.com or devicecloud.digi.com

DO command command:

Bits 0-7: Encoded in Hex.

0: Enable Remote Manager (1), Disable (0).

1: Soft AP when **ID** = None. This has nothing to do with Remote Manager.

2: Send Transparent Data to Remote Manager (1), send to **DL** (0).

3: Send **IO** both to Remote Manager and **DL** (1), Remote Manager 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 Remote Manager at all.

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

/ws/FileData/~/00000000-00000000-00409DFF-FFxxxxxx/serial/0

25: Send transparent data to Remote Manager as a file which is overwritten (00100101)

/ws/FileData/~/00000000-00000000-00409DFF-FFxxxxxx/serial/0

15: Send transparent data to Remote Manager 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 Remote Manager, 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 Remote Manager is named **serial/0**. The file type is **text/plain**. DO command 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 isbroken up and only the last part shows 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 Remote Manager in Transparent mode: set DO command to 0x15 or 0x17, which means:

- Set **DO** to bit 4.
- Set **DO** bit 0 to enable Remote Manager.
- Set **DO** bit 2 so Transparent data goes to Remote Manager.

To view the files:

1. Double click the device.
2. Click **Home**.
3. 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.

## Receive data from Remote Manager

Transparent data is received from Remote Manager using the Device Request interface if the device is operating in Transparent mode (**AP** = 0) and Remote Manager is enabled with **DO** bit 0. Only the raw data is seen on the serial interface and the target string to which Remote Manager sent the data is not available.

You must use API mode to see the target string. In Transparent mode, Remote Manager should not request a response because none is given.

# API operation

# API mode overview

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

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

## API frame specifications

The firmware supports two API operating modes: without escaped characters and with escaped characters. Use the **AP** command to enable either mode. To configure a device to one of these modes, set the following **AP** parameter values:

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

# API operation (AP parameter = 1)

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

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

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

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

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

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

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

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

## Escaped characters in API frames

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

The following data bytes need to be escaped:

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

To escape a character:

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

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

## Example: escape an API frame

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

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

You must escape the 0x13 byte:

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

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

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

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

## Start delimiter field

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

## Length field

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

## Frame data

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

| | Length | | Frame data | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Data | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | n | n+1 |
| 0x7E | MSB | LSB | | Data | | | | | | | |

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

Multi-byte values are sent big-endian.

## Checksum field

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 exclude 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 the device's 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 device 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:

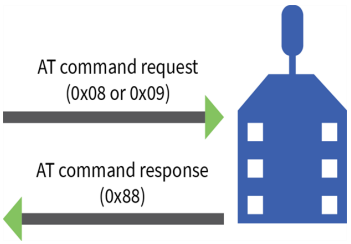
- 0x000E = length (14 bytes excluding checksum)
- 0x07 = Remote Command API frame type
- 0x01 = Frame ID
- 0x00000000 C0A80067 = Remote address (pad first 4 bytes with 00)
- 0x02 = Apply Changes (Remote Command Options)
- 0x4431 = AT command (**D1**)
- 0xB0 = Checksum

# API UART and SPI exchanges

You can use the Frame ID field to assign an identifier to each outgoing API frame. This Frame ID, if non-zero, can correlate between the outgoing frames and the associated responses.

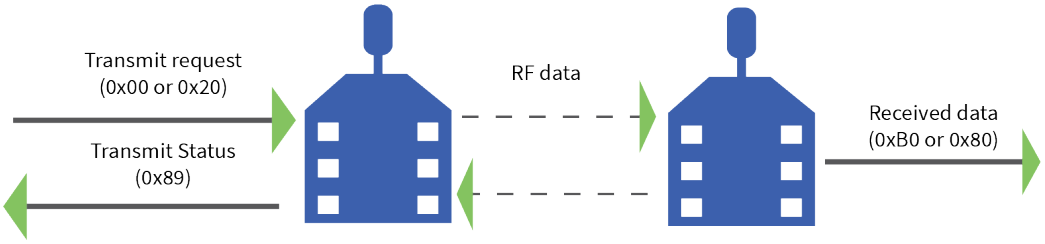
## AT command frames

The following image shows the API frame exchange that takes place at the UART or SPI interface when sending an AT command request to read or set an XBee parameter. To disable the response, set the frame ID to 0 in the request.



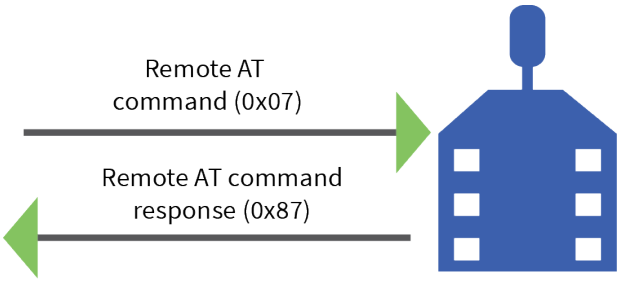

## Transmit and receive RF data

The following image shows the API exchanges that take place at the serial interface when sending RF data to another device. The transmit status frame is always sent at the end of a data transmission unless the frame ID is set to 0 in the TX request. If the packet cannot be delivered to the destination, the transmit status frame indicates the cause of failure. The received data frame type (standard 0x90, or explicit 0x91) is set by the **AP** command.




## Remote AT commands

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



# Frame descriptions

The following sections describe the API frames.

## TX (Transmit) Request: 64-bit - 0x00

### *Description*

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

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| API frame identifier | 3 | 0x00 |
| Frame ID | 4 | |
| 64-bit destination address | 5-12 | 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**.<br>A MAC address may also be in the lower 6 bytes of this field. But if the MAC address does not match the device's own MAC address, then this field is interpreted as an IP address, as described above. |
| TX options | 13 | 0x01 – Disable ACK.<br>Set all other bits to 0. |
| Data | 14 | Max is 1392 bytes. Data is sent to the XBee application service port. |

### *Example*

| Frame data fields | Offset | Description |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x0D |
| API frame identifier | 3 | 0x00 |
| Frame ID | 4 | 0x01 |

| Frame data fields | Offset | Description |
|---|---|---|
| 64-bit destination address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x00 |
| | 12 | 0x64 |
| TX options | 13 | 0x00 |
| Data | 14 | 0x1516 |
| Checksum | | 0x07 |

## Remote AT Command Request - 0x07

### *Description*

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.

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| API frame identifier | 3 | |
| Frame ID | 4 | |
| 64-bit destination address | 5 | Align IP address to low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex.<br><br>A MAC address may also be in the lower 6 bytes of this field. But if the MAC address doesnot match the device's own MAC address, then this field is interpreted as an IP address, as described above. |
| | 6 | |
| | 7 | |
| | 8 | |
| | 9 | |
| | 10 | |
| | 11 | |
| | 12 | |
| Command options | 13 | **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 | Command name: two ASCII characters that identify the AT command. |
| | LSB 15 | |
| Parameter value | - | If present, indicates the requested parameter value to set the given register. If no characters are present, the register is queried. |

### *Example*

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

| Frame data fields | Offset | Description |
|---|---|---|
| Start delimiter | 0 | 0x7E |

| Frame data fields | Offset | Description |
|---|---|---|
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x0D |
| API frame identifier | 3 | 0x07 |
| Frame ID | 4 | 0x01 |
| 64-bit destination address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x00 |
| | 12 | 0x64 |
| Command options | 13 | 0x02 |
| AT command | MSB 14 | 0x44 (D) |
| | LSB 15 | 0x4C (L) |
| Parameter value | -- | -- |
| Checksum | 16 | 0x99 |

## AT Command Frame - 0x08

### *Description*

Use this frame to query or set device parameters on the local device. This API command applies changes after running the command. You can query parameter values by sending the 0x08 AT Command frame with no parameter value field (the two-byte AT command is immediately followed by the frame checksum).

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x08 |
| Frame ID | 4 | |
| AT command | 5-6 | Command name: two ASCII characters that identify the AT command. |
| Parameter value | 7-n | If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the register. |

### *Example*

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x05 |
| Frame type | 3 | 0x08 |
| Frame ID | 4 | 0x01 |
| AT command | 5 | 0x4E (N) |
| | 6 | 0x49 (I) |
| Parameter value | 7 | - |
| Checksum | 8 | 0x5E |

# ZigBee Transmit Packet - 0x10

## Description

This frame type is only provided for software compatibility with other XBee devices. We recommend Transmit (TX) Request: IPv4 - 0x20 for data transmissions from this device.

## Format

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x10 |
| Frame ID | 4 | Correlates request with a later TX STATUS frame (0x8B). If set to **0**, the device does not send a response. |
| 64-bit source address | 5-12 | Align the IP address to the low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex.<br>A MAC address may also be in the lower 6 bytes of this field. But if the MAC address does not match the device's own MAC address, then this field is interpreted as an IP address, as described above. |
| Reserved | 13-15 | Unused placeholders. |
| Options | 16 | 0x01 – Disable ACK.<br>Set all other bits to 0. |
| RF data | 17-n | Up to 1392 bytes of data. |

## Example

The example uses **192.168.0.130** for the 64-bit source address.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x13 |
| Frame type | 3 | 0x10 |
| Frame ID | 4 | 0x01 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x01 |
| | 12 | 0x82 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| | 15 | 0x00 |
| Options | 16 | 0x00 |
| RF data | 17 | 0x48 (H) |
| | 18 | 0x65 (e) |
| | 19 | 0x6C (l) |
| | 20 | 0x6C (l) |
| | 21 | 0x6F (o) |
| Checksum | 22 | 0x12 |

## ZigBee Explicit Transmit Packet - 0x11

### *Description*

This frame type is only provided for software compatibility with other XBee devices and for sending GPM requests. If neither of these is required, then we recommend Transmit (TX) Request: IPv4 - 0x20 for data transmissions from this device.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x11 |
| Frame ID | 4 | Correlates request with a later TX STATUS frame (0x8B). If set to **0**, the device does not send a TX STATUS frame. |
| 64-bit source address | 5 | Align the IP address to the low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex.<br>A MAC address may also be in the lower 6 bytes of this field. But if the MAC address does not match the device's own MAC address, then this field is interpreted as an IP address, as described above. |
| | 6 | |
| | 7 | |
| | 8 | |
| | 9 | |
| | 10 | |
| | 11 | |
| | 12 | |
| Reserved | 13 | Unused placeholders |
| | 14 | |
| | 15 | |
| Destination endpoint | 16 | Digi Device Object |
| Cluster ID | 17 | Memory Access Cluster ID |
| | 18 | |
| Reserved | 19 | Unused placeholders |
| | 20 | |
| | 21 | |
| Options | 22 | 0x01 – Disable ACK<br>Set all other bits to **0**. |

| Frame data fields | Offset | Description |
|---|---|---|
| RF data | 23 | Up to 1392 bytes of data. |
| | 24 | |
| | 25 | |
| | 26 | |
| | 27 | |
| | 28 | |
| | 29 | |
| | 30 | |

### *Example*

The example uses **192.168.1.130** for the 64-bit source address.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x1C |
| API frame identifier | 3 | 0x11 |
| Frame ID | 4 | 0x01 |
| 64-bit source address | 5 | 0x00 |
| | 6 | 0x00 |
| 64-bit source address | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x01 |
| | 12 | 0x82 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| | 15 | 0xE6 |
| Destination endpoint | 16 | 0xE6 |

| Frame data fields | Offset | Example |
|---|---|---|
| Cluster ID | 17 | 0x00 |
| | 18 | 0x23 |
| Reserved | 19 | 0xC1 |
| | 20 | 0x05 |
| | 21 | 0x00 |
| Options | 22 | 0x00 |
| RF data | 23 | 0x00 |
| | 24 | 0x00 |
| | 25 | 0x00 |
| | 26 | 0x00 |
| | 27 | 0x00 |
| | 28 | 0x00 |
| | 29 | 0x00 |
| | 30 | 0x00 |
| Checksum | 31 | 0x50 |

## ZigBee Remote AT Command - 0x17

### Description

This frame type is only provided for software compatibility with other XBee devices. We recommend Remote AT Command Request - 0x07 for sending remote commands from this device.

### Format

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x17 |
| Frame ID | 4 | |
| 64-bit source address | 5 | Align the IP address to the low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex. |
| | 6 | |
| | 7 | |
| | 8 | |
| | 9 | |
| | 10 | |
| | 11 | |
| | 12 | |
| Reserved | 13 | Unused placeholders |
| | 14 | |
| Command options | 15 | 0x02 – Apply changes on remote.<br>If not set, then you must send the **AC** command or the last remote command sent must set this option. |
| AT command | 16 | Two ASCII characters representing the command name (**DL** in the example below). |
| | 17 | |
| Parameter value | 18 | Sets **DL** to **192.168.1.140** in the example below.<br>The parameter value field does not exist on a query. |
| | 19 | |
| | 20 | |
| | 21 | |

### Example

The example uses **192.168.1.130** for the 64-bit source address.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x13 |
| API frame identifier | 3 | 0x17 |
| Frame ID | 4 | 0x01 |
| 64-bit source address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x01 |
| | 12 | 0x82 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| Command options | 15 | 0x02 |
| AT command | 16 | 0x44 (D) |
| | 17 | 0x4C (L) |
| Parameter value | 18 | 0xC0 |
| | 19 | 0xA8 |
| | 20 | 0x01 |
| | 21 | 0x8C |
| Checksum | 22 | 0x78 |

## Transmit (TX) Request: IPv4 - 0x20

### *Description*

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

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x20 |
| Frame ID | 4 | 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 | Use **0xFFFFFFFF** for broadcast when the protocol is UDP. |
| | 6 | |
| | 7 | |
| | 8 | |
| 16-bit destination port | MSB 9 | UDP or TCP port number |
| | LSB 10 | |
| 16-bit source port | MSB 11 | UDP or TCP port number<br>To send a UDP packet, this must match the port number of the listening port as specified by **C0**.<br>To send a TCP packet on a new connection, this must be **0**. |
| | LSB 12 | |
| Protocol | 13 | Protocol use for the transmitted data:<br>**0** = UDP<br>**1** = TCP |
| Transmit options bitfield | 14 | Bit fields are offset 0<br>Bit field 0 - 7. Bits 0, and 2-7 are reserved, bit 1 is not.<br>BIT 1 =<br>**1** - Terminate the socket after transmission is complete<br>**0** - Leave the socket open (use TCP timeout)<br>Ignore this bit for UDP packets.<br>All other bits are reserved and should be **0**. |
| RF data | 15 | Up to 1400 bytes of data. This is 8 bytes more than the maximum size reported by **NP** because it does not require an application header. |
| | 16 | |
| | 17 | |
| | 18 | |
| | 19 | |

### *Example*

The example uses **192.168.0.100** for the IPv4 32-bit destination address.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x11 |
| API frame identifier | 3 | 0x20 |
| Frame ID | 4 | 0x01 |
| IPv4 32-bit destination address | MSB 5 | 0xC0 |
| | 6 | 0xA8 |
| | 7 | 0x00 |
| | 8 | 0x64 |
| 16-bit destination port | MSB 9 | 0x26 |
| | LSB 10 | 0x16 |
| 16-bit source port | MSB 11 | 0x26 |
| | LSB 12 | 0x16 |
| Protocol | 13 | 0x00 |
| Transmit options bitfield | 14 | 0x00 |
| RF data | 15 | 0x48 (H) |
| | 16 | 0x65 (e) |
| | 17 | 0x6C (l) |
| | 18 | 0x6C (l) |
| | 19 | 0x6F (o) |
| Checksum | 20 | 0xA6 |

## Send Data Request - 0x28

### *Description*

Send a file of the given name and type to Remote Manager.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x28. |
| Frame ID | 4 | Identifies the frame for send data response. If **0**, then no send data response status will be received. |
| Path length | 5 | Length of path and file name. |
| Path | 6-13 | Path and file name. |
| Content type length | 14 | Length of target string (up to 16 bytes). |
| Content type | 15-24 | Indicates file type, such as text/plain, text/xml, or application/json. |
| Transport | 25 | Must be **0** to indicate TCP. |
| Options | 26 | 0—overwrite<br>1—archive<br>2—append<br>3—transient data (do not store) |
| Data | 27-53 | |

### *Example*

| Frame data fields | Offset | Example |
|---|---|---|
| Start | 0 | 0x7E |
| Length | 1-2 | 0x0033 |
| Frame type | 3 | 0x28 |
| Frame ID | 4 | 0x55 |
| Path length | 5 | 0x08 |
| Path | 6-13 | TestFile |
| Content type length | 14 | 0x0A |
| Content type | 15-24 | Text/plain |
| Transport | 25 | 0x00 |

| Frame data fields | Offset | Example |
|---|---|---|
| Options | 26 | 0x00 |
| Data | 27-53 | abcdefghij<br>klmnopqr<br>stuvwxya |
| Checksum | 54 | 0x49 |

## Device Response - 0x2A

### *Description*

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.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x2A |
| Frame ID | 4 | Identifies the frame for the device response status. If **0**, then no device response status is received. |
| Device request ID | 5 | This number should match the device request ID in the device request. Otherwise, an error occurs. 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 | Must be **0** for now. |
| Data | 7-11 | The particular data for the device response is application dependent. |

### *Example*

| Frame data fields | Offset | Example |
|---|---|---|
| Start Delimiter | 0 | 0x7E |
| Length | 1-2 | 0x0009 |
| Frame type | 3 | 0x2A |
| Frame ID | 4 | 0x01 |
| Device request ID | 5 | 0x00 |
| Reserved | 6 | 0x00 |
| Data | 7-11 | Hello |
| Checksum | 12 | 0xE0 |

## Rx (Receive) Packet: 64-bit - 0x80

### *Description*

This frame type is only provided for software compatibility with other XBee devices. We recommend for data transmissions from this device.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x80 |
| 64-bit source address | 4 | Align IP address to low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex. |
| | 5 | |
| | 6 | |
| | 7 | |
| | 8 | |
| | 9 | |
| | 10 | |
| | 11 | |
| RSSI | 12 | RSSI in terms of dB above sensitivity (link margin). |
| Options | 13 | None currently defined. |
| RF data | 14 | Up to 1392 bytes of data. |
| | 15 | |
| | 16 | |
| | 17 | |
| | 18 | |

### *Example*

The example uses address **192.168.0.103**.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| Frame type | 3 | 0x80 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source address | 4 | 0x00 |
| | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0xC0 |
| | 9 | 0xA8 |
| | 10 | 0x00 |
| | 11 | 0x67 |
| RSSI | 12 | 0x2E |
| Options | 13 | 0x00 |
| RF data | 14 | 0x48 (H) |
| | 15 | 0x65 (e) |
| | 16 | 0x6C (l) |
| | 17 | 0x6C (l) |
| | 18 | 0x6F (o) |
| Checksum | 19 | 0x8E |

## Remote Command Response - 0x87

### *Description*

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

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x87 |
| Frame ID | 4 | |
| 64-bit responder address | 5 | Align IP address to low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex. |
| | 6 | |
| | 7 | |
| | 8 | |
| | 9 | |
| | 10 | |
| | 11 | |
| | 12 | |
| AT command | MSB 13 | Command Name - two ASCII characters that identify the AT command. |
| | LSB 14 | |
| Status | 15 | 0 = OK<br>1 = ERROR<br>2 = Invalid command<br>3 = Invalid parameter<br>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. |

### *Example*

In this example the IP address is **192.168.0.103**.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |

| Frame data fields | Offset | Example |
|---|---|---|
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x0D |
| API frame identifier | 3 | 0x87 |
| Frame ID | 4 | 0x01 |
| 64-bit responder address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x00 |
| | 12 | 0x67 |
| AT command | MSB 13 | 0x44 (D) |
| | LSB 14 | 0x31 (1) |
| Status | 15 | 0x00 |
| Parameter value | | - |
| Checksum | 16 | 0x33 |

## AT Command Response frame - 0x88

### *Description*

A device sends this frame in response to an AT Command message. Some commands send back multiple frames (for example, the **AS** (Active Scan) command).

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x88 |
| Frame ID | 4 | |
| AT command | 5-6 | Command name: two ASCII characters that identify the command. |
| Command status | | 0 = OK<br>1 = ERROR<br>2 = Invalid command<br>3 = Invalid parameter |
| Parameter value | 7 | The register data in binary format. If the host sets the register, the device does not return this field. |

### *Example*

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x05 |
| Frame type | 3 | 0x88 |
| Frame ID | 4 | 0x01 |
| AT command | 5 | 0x42 (B) |
| | 6 | 0x44 (D) |
| Command status | 7 | 0x00 |
| Command data | | |
| Checksum | 8 | 0xF0 |

## Transmission Status frame - 0x89

### *Description*

The XBee Wi-Fi RF Module sends RF transmission status messages in response to transmission attempts.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x89 |
| Frame ID | 4 | 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 is not generated. |
| Status | 5 | 0x00 = Success.<br>0x03 = Transmission was purged because it was attempted before stack was completely up.<br>0x04 = Physical error occurred on the interface with the Wi-Fi transceiver.<br>0x21 = TX64 transmission timed out awaiting an acknowledgment from the remote device.<br>0x32 = Resource Error. Either buffers or sockets were depleted, preventing a transmission from occurring.<br>0x74 = Message not sent because it was too long.<br>0x76 = Attempt to create a client socket failed.<br>0x77 = TCP connection to given IP address and port does not exist. Source port is non-zero so that a new connection is not attempted.<br>0x78 = Source port on a UDP transmission does not match a listening port on the transmitting device. |

### *Example*

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x03 |
| Frame type | 3 | 0x89 |
| Frame ID | 4 | 0x01 |
| Status | 5 | 0x00 |
| Checksum | 6 | 0x75 |

## Modem Status frame - 0x8A

### *Description*

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

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8A |
| Status | 4 | 0 = Hardware reset or power up<br>1 = Watchdog timer reset<br>2 = Joined<br>3 = No longer joined to access point<br>0x0E = Remote Manager connected<br>0x0F = Remote Manager disconnected |

### *Example*

The XBee Wi-Fi RF Module returns the following API frame when it is powered on in API mode.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| LSB 2 | LSB 2 | 0x02 |
| Frame type | 3 | 0x8A |
| Status | 4 | 0x00 |
| Checksum | 5 | 0x75 |

## ZigBee TX Status frame - 0x8B

### *Description*

This frame type is only provided for software compatibility with other XBee devices. 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.

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8B |
| Frame ID | 4 | Frame ID of the correlating transmission. |
| Reserved | 5 - 7 | Hard coded values can be ignored. |
| Delivery status | 8 | 0x00 = Success.<br>0x03 = Transmission was purged because it was attempted before stack was completely up.<br>0x04 = Physical error occurred on the interface with the Wi-Fi transceiver.<br>0x21 = Transmission timed out awaiting an acknowledgment from the remote device.<br>0x32 = Resource Error; Either buffers or sockets were depleted, preventing a transmission from occurring.<br>0x74 = Message not sent because it was too long.<br>0x76 = Attempt to create a client socket failed. |
| Reserved | 9 | Hard coded value can be ignored. |

### *Example*

| Frame Fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x07 |
| Frame type | 3 | 0x8B |
| Frame ID | 4 | 0x01 |
| Reserved | 5 | 0xFF |
| | 6 | 0xFE |
| | 7 | 0x00 |

| Frame Fields | Offset | Example |
|---|---|---|
| Status | 8 | 0x00 |
| Reserved | 9 | 0x00 |
| Checksum | 10 | 0x76 |

# I/O Data Sample RX Indicator frame - 0x8F

## *Description*

When the XBee Wi-Fi RF Module receives an I/O sample frame from a remote device, it sends the sample out the UART or SPI using this frame type. Only devices running API mode are able to receive I/O samples.

## *Format*

| Frame fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x8F |
| 64-bit source address | 4 - 11 | Align IP address to low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex. The example below uses address **192.168.0.103**. |
| RSSI in terms of link margin | 12 | |
| Receive options | 13 | None currently defined. |
| Number of samples | 14 | Number of sample sets included in the payload. Always set to **1**. |
| Digital Channel Mask (see below) | MSB 15 | Bitmask field that indicates which digital I/O lines on the remote have sampling enabled (if any). In the example below, DIO8 is active. |
| | LSB 16 | |
| Analog Channel Mask (see below) | 17 | Bitmask field that indicates which analog I/O lines on the remote have sampling enabled (if any). The most significant bit signals that the Vcc value are included in the frame. In the example below, Analog input 1 and Vcc are active. |
| Digital Samples (if included) | MSB 18 | If the sample set includes any digital I/O lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital I/O lines. DIO lines that do not have sampling enabled return **0**. The bits in these 2 bytes map the same as they do in the Digital Channels Mask field. In the example below, DIO8 has value **0**. |
| | LSB 19 | |
| Analog Sample | MSB 20 | 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 | |

## *Digital Channel bitmask*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

### Analog Channel bitmask

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Supply voltage | N/A | N/A | N/A | AD3 | AD2 | AD1 | AD0 |

### Example

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

| Frame fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x13 |
| API frame identifier | 3 | 0x8F |
| 64-bit source address | 4 | 0x00 |
| | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0xC0 |
| | 9 | 0xA8 |
| | 10 | 0x00 |
| | 11 | 0x67 |
| RSSI in terms of link margin | 12 | 0x2E |
| Receive options | 13 | 0x00 |
| Number of samples | 14 | 0x01 |
| Digital Channel Mask | MSB 15 | 0x01 |
| | LSB 16 | 0x00 |
| Analog Channel Mask | 17 | 0x81 |
| Digital Samples (if included) | MSB 18 | 0x00 |
| | LSB 19 | 0x00 |
| Analog Sample | MSB 20 | 0x03 |
| | LSB 21 | 0xB5 |
| Checksum | 22 | 0x38 |

## ZigBee Receive Packet frame 0x90

### *Description*

The XBee Wi-Fi RF Module uses this frame when it receives RF data using the XBee application service and **AO** = 0. It is not generally used, but it allows for software compatibility with other XBee devices if desired.

### *Format*

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x90 |
| 64-bit source address | 4-11 | Align the IP address to low 32-bits of the field. Set the other bytes to **0**. The IP address is in hex. |
| Reserved | 12-13 | Unused placeholder. |
| Options | 14 | Bit 1: Broadcast packet. |
| RF data | 15-n | Up to 1392 bytes of data. |

### *Example*

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

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x11 |
| Frame type | 3 | 0x90 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source address | 4 | 0x00 |
| | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0xC0 |
| | 9 | 0xA8 |
| | 10 | 0x00 |
| | 11 | 0x67 |
| 16-bit source network address | 12 | 0xFF |
| | 13 | 0xFE |
| Options | 14 | 0x00 |
| RF data | 15 | 0x48 (H) |
| | 16 | 0x65 (e) |
| | 17 | 0x6C (l) |
| | 18 | 0x6C (l) |
| | 19 | 0x6F (o) |
| Checksum | 20 | 0xAF |

## Description

The XBee Wi-Fi RF Module uses this frame type when it receives RF data using the XBee application service and **AO** = 0. Even when **AO** is not **1**, this frame is also used for GPM response frames, see Sleep modes.

## Format

| Frame data fields | Offset | Example |
|---|---|---|
| Frame type | 3 | 0x91 |
| 64-bit source address | 4-11 | Align the IP address to low 32-bits of the field. The other bytes are set to **0**. The IP address is in hex. The example below uses address **192.168.0.103**. |
| Reserved | 12-13 | Unused placeholders. |
| Source endpoint | 14 | Digi device object endpoint. |
| Destination endpoint | 15 | Digi device object endpoint. |
| Cluster ID | 16 | Memory access cluster ID. |
| | 17 | |
| Profile ID | 18 | Digi profile ID. |
| | 19 | |
| Option | 20 | |
| RF data | 21 | Response to platform info request, indicating 160 GPM blocks available.<br>Each block size is 4096 bytes. |
| | 22 | |
| | 23 | |
| | 24 | |
| | 25 | |
| | 26 | |
| | 27 | |

## Example

This example uses address **192.168.0.103**.

| Frame data fields | Offset | Description |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x19 |
| Frame type | 3 | 0x91 |
| 64-bit source address | 4 | 0x00 |
| | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0xC0 |
| | 9 | 0xA8 |
| | 10 | 0x00 |
| | 11 | 0x67 |
| Reserved | 12 | 0xFF |
| | 13 | 0xFE |
| Source endpoint | 14 | 0xE6 |
| Destination endpoint | 15 | 0xE6 |
| Cluster ID | 16 | 0x00 |
| | 17 | 0x23 |
| Profile ID | 18 | 0xC1 |
| | 19 | 0x05 |
| Option | 20 | 0x00 |
| RF data | 21 | 0x80 |
| | 22 | 0x00 |
| | 23 | 0xA0 |
| | 24 | 0x10 |
| | 25 | 0x00 |
| | 26 | 0x00 |
| | 27 | 0x00 |
| Checksum | 28 | 0xBD |

## ZigBee Remote AT Command Response frame - 0x97

### Description

This frame type is only provided for software compatibility with other XBee devices. It generates a response to the ZigBee Remote AT Command - 0x17 frame. Normally, customer use Remote AT Command Request - 0x07 instead with Remote Command Response - 0x87.

### Format

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0x97 |
| Frame ID | 4 | |
| 64-bit source (remote) address | 5-12 | The address of the remote device returning this response. Align the IP address to the low 32-bits of the field. Set the other bytes to **0**.<br>The Value is in hex. In the example below, the IP address is **192.168.0.103**. |
| Reserved | 13-14 | Unused placeholder. |
| AT command | 15-16 | The name of the command. |
| Command status | 17 | 0 = OK<br>1 = ERROR<br>2 = Invalid command<br>3 = Invalid parameter |
| Command data | 18-n | If present, indicates the value of the requested parameter value.<br>If not present, this is not a response to a query command. |

### Example

In the example below, the IP address is **192.168.0.103**.

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x0F |
| Frame type | 3 | 0x97 |
| Frame ID | 4 | 0x01 |

| Frame data fields | Offset | Example |
|---|---|---|
| 64-bit source (remote) address | 5 | 0x00 |
| | 6 | 0x00 |
| | 7 | 0x00 |
| | 8 | 0x00 |
| | 9 | 0xC0 |
| | 10 | 0xA8 |
| | 11 | 0x00 |
| | 12 | 0x67 |
| Reserved | 13 | 0xFF |
| | 14 | 0xFE |
| AT commands | 15 | 0x44 (D) |
| | 16 | 0x4C (L) |
| Command status | 17 | 0x00 |
| Checksum | 18 | 0x0B |

## RX (Receive) Packet: IPv4 - 0xB0

### Description

This frame is used by the XBee Wi-Fi RF Module it receives RF data using the Serial Data service on the port defined by the **C0** command.

### Format

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

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0xB0 |
| IPv4 32-bit source address | MSB 4 | The address in the example below is for a source address of **192.168.0.104**. |
| | 5 | |
| | 6 | |
| | 7 | |
| 16-bit destination port | MSB 8 | Same value as the **C0** command. |
| | LSB 9 | |
| 16-bit source port | MSB 10 | |
| | LSB 11 | |
| Protocol | MSB 12 | 0 = UDP<br>1 = TCP - Protocol use for the transmitted data |
| Status | 13 | Reserved |
| RF data | 14 | Up to 1400 bytes of data.<br>This is 8 bytes more than the max size reported by the **NP** command because no application header is needed. |
| | 15 | |
| | 16 | |
| | 17 | |
| | 18 | |

### Example

When a device in API mode receives an IPv4 transmission, it produces an RX notification (0xB0) and sends it out the UART or SPI port. 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 data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| Frame type | 3 | 0xB0 |
| IPv4 32-bit source address | MSB 4 | 0xC0 |
| | 5 | 0xA8 |
| | 6 | 0x00 |
| | 7 | 0x68 |
| 16-bit destination port | MSB 8 | 0x26 |
| | LSB 9 | 0x16 |
| 16-bit source port | MSB 10 | 0x26 |
| | LSB 11 | 0x16 |
| Protocol | MSB 12 | 0x00 |
| Status | 13 | 0x00 |
| RF data | 14 | 0x48 (H) |
| | 15 | 0x65 (e) |
| | 16 | 0x6C (l) |
| | 17 | 0x6C (l) |
| | 18 | 0x6F (o) |
| Checksum | 19 | 0x13 |

## Send Data Response frame - 0xB8

### *Description*

This frame type is sent out the serial port in response to Send Data Request - 0x28, providing its frame ID is non-zero.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0xB8 |
| Frame ID | 4 | Identifies the frame ID of the corresponding send data request. |
| Status | 5 | 0x00 = Success<br>0x01 = Bad request<br>0x02 = Response unavailable<br>0x03 = Remote Manager Error<br>0x40 = Unknown error |

### *Example*

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | 1-2 | 0x0003 |
| Frame type | 3 | 0xB8 |
| Frame ID | 4 | 0x55 |
| Status | 5 | 0x00 |
| Checksum | 6 | 0xF2 |

## Device Request frame - 0xB9

### *Description*

This frame type is sent out the serial port when the XBee Wi-Fi RF Module receives a valid device request from Remote Manager.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0xB9 |
| Device request ID | 4 | Identifies the device request. (If 0, then no response is required. |
| Transport | 5 | Placeholders. Values can be ignored. |
| Flags | 6 | |
| Target length | 7 | Length of target string |
| Target string | 8-15 | String required by the host side, e.g. a file name. |
| Data | 16-40 | |

### *Example*

| Frame data fields | Offset | Example |
|---|---|---|
| Start | 0 | 0x7E |
| Length | 1-2 | 0x0026 |
| Frame type | 3 | 0xB9 |
| Device request ID | 4 | 0x01 |
| Transport | 5 | 0x00 |
| Flags | 6 | 0x00 |
| Target length | 7 | 0x08 |
| Target string | 8-15 | myTarget |
| Data | 16-40 | A message for serial host |
| Checksum | 41 | 0xC6 |

## Device Response Status frame - 0xBA

### *Description*

This frame type is sent to the serial port after the serial port sends a Device Response - 0x2A.

### *Format*

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0xBA |
| Frame ID | 4 | Identifies the frame for which status is being reported. Corresponds to the frame ID in the device response. |
| Status | 5 | 0x00 = Success<br>0x20 = Device Request canceled by user<br>0x21 = Session timed out<br>0x40 = Unknown Error |

### *Example*

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | 1-2 | 0x0003 |
| Frame type | 3 | 0xBA |
| Frame ID | 4 | 0x01 |
| Status | 5 | 0x00 |
| Checksum | 6 | 0x44 |

# Frame Error - 0xFE

## Description

This frame 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 Device Response - 0x2A.

## Format

The following table provides the contents of the frame.

| Frame data fields | Offset | Description |
|---|---|---|
| Frame type | 3 | 0xFE |
| Status | 4 | 0x02 = Invalid frame type.<br>0x03 = Invalid frame length.<br>0x04 = Erroneous Checksum on last frame.<br>0x05 = payload of last API frame was too big to fit into a buffer.<br>0x06 = string entry was too big on last API frame sent.<br>0x07 = Wrong state to receive frame (for example, a device response was sent out without first receiving a device request).<br>0x08 = Device request ID of device response didn't match the number in the request. |

## Example

| Frame data fields | Offset | Example |
|---|---|---|
| Start delimiter | 0 | 0x7E |
| Length | MSB 1 | 0x00 |
| | LSB 2 | 0x02 |
| Frame type | 3 | 0xFE |
| Status | 4 | 0x07 |
| Checksum | 6 | 0xFA |

# AT commands

# Addressing commands

The following AT commands are addressing commands.

## EQ (Device Cloud FQDN)

Sets or display the fully qualified domain name of the Remote Manager server.

**Range**

Valid fully qualified domain name (FQDN).

From 0 through 63 ASCII characters.

**Default**

**devicecloud.digi.com**

## LA command

Performs a DNS lookup of the given fully qualified domain name (FQDN) and outputs its IP address.

When you issue the command in API mode, the IP address is formatted in binary four byte big-endian numeric value. In all other cases (for example, Command mode) the format is dotted decimal notation.

**Range**

Valid FQDN.

From 0 through 63 ASCII characters.

**Default**

-

## PG (Ping an IP Address)

Ping a given IP address and indicate the response time or an error indication on failure. Response is always a string.

**Parameter range**

Valid IPv4 address in either dotted decimal notation or binary format.

**Default**

-

## NS (DNS Address)

Sets or displays the address of the DNS server. When reading API mode, the format is binary. In all other cases (for example Command mode), the read format is dotted decimal notation.

**Range**

Valid IPv4 address in dotted decimal notation

**Default**

208.67.222.222 (address of openDNS)

## DL command

Sets or displays the 32 bits of the IPv4 destination address. When setting, the format may be either dotted decimal (for example **192.168.0.100**) or binary (for example **C0A80064**). When reading in API mode, the format is binary. In all other cases (for example Command mode), the read format is dotted decimal notation.

**Parameter range**

0.0.0.0 – 255.255.255.255

**Default**

255.255.255.255

## MY command

Read the 32-bit network address of the device when using DHCP. Sets or displays values when using static IP address. When setting, the format may be either dotted decimal (for example **192.168.0.100**) or binary (for example **C0A80064**). When reading in API mode, the format is binary. In all other cases (for example Command mode) the read format is dotted decimal notation.

**Parameter range**

0.0.0.0 – 255.255.255.255

**Default**

0.0.0.0

## MK command

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 (for example **255.255.255.0**) or binary (for example **FFFFFF00**). When reading in API mode, the format is binary. In all other cases (for example Command mode) the read format is dotted decimal notation.

**Range**

0.0.0.0 - 255.255.255.255.

**Default**

0.0.0.0

## GW command

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 (for example **192.168.0.1**) or binary (for example **C0A80001**). When reading in API mode, the format is binary. In all other cases (for example Command mode) the read format is dotted decimal notation.

**Range**

0.0.0.0 - 255.255.255.255.

**Default**

0.0.0.0

## SH (Serial Number High)

Reads the high 16 bits of the device's unique 48-bit address.

**Parameter range**

0 - 0xFFFFFFFF [read-only]

**Default**

Set in the factory

## SL (Serial Number Low)

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

Read the low 32 bits of the device's unique 48-bit address.

**Parameter range**

0 - 0xFFFFFFFF [read-only]

**Default**

Set in the factory

## NI (Node Identifier)

Stores a string identifier. The register only accepts printable ASCII data. In Command mode, a string cannot start with a space. A carriage return ends the command. The command automatically ends when maximum bytes for the string have been entered.

**Parameter range**

A string of case-sensitive ASCII printable characters from 0 to 20 bytes in length.

**Default**

One ASCII space character (0x20)

## DE command

Sets or displays the destination UDP/TCP port value.

**Parameter range**

0x0 - 0xFFFF

**Default**

0x2616

## KP (Device Description)

Sets or displays a user-defined description for the XBee displayed in Remote Manager.

**Range**

Up to 20 ASCII characters

**Default**

One ASCII space character (0x20)

## KC (Device Cloud Contact)

Sets or displays a user-defined contact for the XBee displayed in Remote Manager.

**Range**

Up to 20 ASCII characters

**Default**

One ASCII space character (0x20).

## KL (Device Location)

Sets or displays a user-defined physical location for the XBee displayed in Remote Manager.

**Range**

Up to 20 ASCII characters

**Default**

One ASCII space character (0x20).

## C0 (Serial Communication Service Port)

Set or get the port number used to provide the serial communication service. Data sent to this port comes from the serial port of the module. The **IP** command sets the protocol used when UART is in Transparent mode.

**Range**

0xFFFF

**Default**

0x2616 )

## DD (Device Type Identifier)

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

Digi reserves the range 0 - 0xFFFFFF.

**Parameter range**

0 - 0xFFFFFFFF [read-only]

**Default**

0x90000

## NP (Maximum RF Payload Bytes)

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

**Note** **NP** returns a hexadecimal value. For example, if **NP** returns 0x54, this is equivalent to 84 bytes.

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.

**Parameter range**

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

**Default**

[read-only]

# Network commands

The following commands are network commands.

## DO command

Sets or displays the device options.

**Bit field:**

| Bit | Description |
|---|---|
| 0 | Enable Remote Manager. |
| 1 | Enable SoftAP when ID is NULL. |
| 2 | Enable sending transparent data to Remote Manager. |
| 3 | Send I/O samples to both Remote Manager and to **DL** if Remote Manager is enabled. |
| 4 | Send transparent data as binary data points rather than to a file. |
| 5 | Replace a Remote Manager file (1) rather than append to a file (0). |
| 6, 7 | Reserved. This should be 0. |

**Note** In Transparent mode, if **DO** is 0x25 and over 1400 bytes are sent at once, the Remote Manager file is overwritten twice, losing the first 1400 bytes.

**Parameter range**

0x03F

**Default**

1

## ID (SSID)

Set or read the SSID of the access point, which may be up to 31 ASCII characters.

**Parameter range**

Up to 31 bytes of printable ASCII

**Default**

NULL

## AH (Network Type)

Set or read the network type. The Network types supported are Infrastructure (using an access point) and Ad hoc (IBSS).

**Parameter range**

0 - IBSS Joiner

1 - IBSS Creator

2 - Infrastructure

**Default**

2

## IP (IP Protocol)

Set or displays the protocol used for the serial communication service. This is the port used by the **C0** command.

**Parameter range**

0, 1

| Value | Description |
|---|---|
| 0x0 | UDP |
| 0x1 | TCP |

**Default**

0

## MA (IP Addressing Mode)

Sets or displays the IP addressing mode: DHCP or static.

**Range**

| Value | Description |
|---|---|
| 0 | DHCP addressing mode. |
| 1 | Static addressing mode. |

**Default**

0

### TM (Timeout)

Set or displays the timeout for connection on TCP client sockets. If 0, socket closes immediately after data sent.

**Parameter range**

0 - 0xFFFF [x 100 ms]

**Default**

0x64

### TS (TCP Server Socket Timeout)

Set or read the timeout for a connection on a TCP server socket. The connection for this socket was initiated at the other end.

The IP server connection timeout. If no activity for this timeout then the connection is closed. When set to **0** the connection is closed immediately after data is sent.]

**Parameter Range**

0x000A– 0xFFFF (* 100 ms)S

**Default**

0x0258 (1 minute)

### CE (Infrastructure Mode)

Selects AP mode (1) or STA mode (2). For more information, see Enable Soft AP mode.

**Parameter range**

1 - Soft AP Mode

2 - STA Mode

**Default**

2

## Security commands

The following AT commands are security commands.

### EE (Encryption Enable)

Set or read the encryption enable setting.

**Parameter range**

0 - 3

| Parameter | Description |
|---|---|
| 0 | No security |

| Parameter | Description |
|---|---|
| 1 | WPA |
| 2 | WPA2 |
| 3 | WEP |

**Default**

0

## PK command

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. 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.

**Parameter Range**

8-63 ASCII or 64 hexadigit characters for WPA and WPA2.

WEP keys can be either 40 bits or 104 bits. Enter 40-bit WEP keys with 5 ASCII characters or 10 hex characters.

Enter 104-bit WEP keys with 13 ASCII characters or 26 hex characters.

**Default**

N/A

# RF interfacing commands

The following AT commands are RF interfacing commands.

## PL (Power Level)

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

**Parameter range**

0 - 4

| Parameter range | Default |
|---|---|
| 0 | 0 dBm |
| 1 | 5 dBm |
| 2 | 10 dBm |
| 3 | 15 dBm |

| Parameter range | Default |
|---|---|
| 4 | Maximum power. See Electrical specifications. |

**Default**

4

## CH (Channel)

Read the channel number of the access point or 0xFF if is not associated. You can set the channel 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.

**Parameter range**

1 - 0xB

**Default**

[read-only]

# Serial interfacing commands

## API Enable

Enables API Mode.

**Parameter range**

| Parameter | Description |
|---|---|
| 0 | Transparent mode |
| 1 | API-enabled |
| 2 | API-enabled (with escaped control characters) |

**Default**

0

## AO (API Options)

Displays the type of frame to output when receiving data on the IP services port.

**Parameter range**

0 - 2

| Parameter | Description |
|---|---|
| 0 | ZigBee Rx |
| 1 | Explicit ZigBee Rx |
| 2 | RX64 |

**Default**

2

## BD (Baud Rate)

Sets or displays the serial interface baud rate for communication between the device's serial port and the host.

The device interprets any value above 0x0A as an actual baud rate. When a value above 0x0A is sent, the device stores the closest interface data rate represented by the number in the **BD** register.

**Parameter range**

Standard baud rates: 0x1 - 0x7

| Value | Description |
|---|---|
| 0x1 | 2,400 b/s |
| 0x2 | 4,800 b/s |
| 0x3 | 9,600 b/s |
| 0x4 | 19,200 b/s |
| 0x5 | 38,400 b/s |
| 0x6 | 57,600 b/s |
| 0x7 | 115,200 b/s |
| 0x8 | 230,400 b/s |
| 0x9 | 460,800 b/s |
| 0xA | 921,600 b/s |
| 0X5B9 - 0X5B8D80 | (non-standard rates up to 6 Mb/s) |

**Default**

0x03 (9600 b/s)

## NB (Serial Parity)

Set or read the serial parity settings on the device.

**Parameter range**

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

**Default**

0x00

## SB (Stop Bits)

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

**Parameter range**

0 - 1

| Value | Description |
|---|---|
| 0 | One (1) stop bit. |
| 1 | Two (2) stop bits. |

**Default**

0

## RO (Packetization Timeout)

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

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

Regardless of the **RO** size, the inter-character silence required to trigger a transmission of the data is 100 µs.

**Parameter range**

0 - 0xFF (x character times)

**Default**

3

## FT (Flow Control Threshold)

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

**Parameter range**

0x11 - 0x823

**Default**

0x7F3

### D7 (DIO7 Configuration)

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration (TH pin 12/SMT pin 25).

**Parameter range**

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | $\overline{\text{CTS}}$ flow control |
| 3 | Digital input |
| 4 | Digital output, low |
| 5 | Digital output, high |
| 6 | RS-485 Tx enable (low enable) |
| 7 | RS-485 Tx enable (high enable) |

**Default**

0x1

### D6 (DIO6 Configuration)

Sets or displays the DIO6/$\overline{\text{RTS}}$ configuration (TH pin 16/SMT pin 29).

**Parameter range**

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

**Default**

0

## I/O settings commands

The following AT commands are I/O settings commands.

## IS (Force Sample)

Forces a read of all enabled digital and analog input lines. If no lines are enabled for digital or analog input, the command returns and error.

**Parameter range**

N/A

**Default**

N/A

Set or read the I/O sample rate to enable periodic sampling.

To enable periodic sampling, set **IR** to a non-zero value, and enable the analog or digital I/O functionality of at least one device pin (see D0 (DIO0/AD0/ CB Configuration)-D8 (DIO8 Configuration), P0 (DIO10 Configuration)-P2 (DIO12 Configuration).

The sample rate is measured in milliseconds.

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

## Parameter range

0 - 0xFFFF (x 1 ms)

## Default

0

## IC (Digital Change Detection)

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

**IC** works with the individual pin configuration commands (**D0** - **D9**, **P0** - **P2**) . If you enable a pin as a digital I/O, you can use the **IC** command to force an immediate I/O sample transmission when the DIO state changes. IC is a bitmask that you can use to enable or disable edge detection on individual channels.

Set unused bits to 0.

**Parameter range**

0 - 0xFFFF

**Default**

0

## IF (Sample from Sleep Rate)

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

**Parameter range**

1 - 0xFF (1 gives you a sample every sleep cycle)

**Default**

1

## P0 (DIO10 Configuration)

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

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | PWM RSSI output |
| 2 | PWM0 output |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## P1 (DIO11 Configuration)

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

**Parameter range**

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 2 | PWM1 Output |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

0

## P2 (DIO12 Configuration)

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

**Parameter range**

0, 1, 3-6

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_MISO[1] |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |
| 6 | TCP connection indicator |

**Default**

0

## P3 (DOUT)

Enables or disables output on UART port.

**Parameter range**

0, 1

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

**Default**

1

## P4 DIN

Enables or disables input on UART port.

**Parameter range**

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

**Default**

1

[1]Indicates that the option is available on the TH module, but not the SMT module.

## P5 (DIO15 Configuration)

Sets or displays the DIO15 configuration (TH pin 4/SMT pin 17).
This only applies to surface-mount devices.

**Parameter range**

0, 1, 4, 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_MISO |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## P6 (DIO16 Configuration)

Sets or displays the DIO16/SPI_MOSI configuration (TH pin 11/SMT pin 16).
This only applies to surface-mount devices.

**Parameter range**

0, 1, 4, 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_MOSI |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## P7 (DIO17 Configuration)

Sets or displays the DIO17 configuration (TH pin 17/SMT pin 15).
This only applies to surface-mount devices.

**Parameter range**

0, 1, 4, 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_$\overline{\text{SSEL}}$ |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## P8 (DIO18 Configuration)

Sets or displays the DIO18/SPI_CLK configuration (TH pin 18/SMT pin 14). This only applies to surface-mount devices.

**Parameter range**

0, 1, 4, 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_CLK |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## P9 (DIO19 Configuration)

Sets or displays the DIO19/SPI_$\overline{\text{ATTN}}$ configuration (TH pin 13/SMT pin 12). This only applies to surface-mount devices.

**Parameter range**

0, 1, 4 - 6

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_$\overline{\text{ATTN}}$ |
| 4 | Digital output, default low |
| 5 | Digital output, default high |
| 6 | UART data present indicator |

**Default**

1

## D0 (DIO0/AD0/ CB Configuration)

Sets or displays the DIO0/AD0/CB configuration (TH pin 20/SMT pin 33).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Commissioning Button |
| 2 | Analog input |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## D1 (DIO1/AD1 Configuration)

Sets or displays the DIO1/AD1 configuration (TH pin 19/SMT pin 32).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_$\overline{\text{ATTN}}$ |
| 2 | Analog input |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## D2 (DIO2/AD2 Configuration)

Sets or displays the DIO2/AD2 configuration (TH pin 18/SMT pin ).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI_CLK[1] |
| 2 | Analog input |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

0

## D3 (DIO3/AD3 Configuration)

Sets or displays the DIO3/AD3 configuration (TH pin 17/SMT pin 30).

**Parameter range**

0 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI Slave Select[2] |
| 2 | Analog input |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

0

## D4 (DIO4/AD4 Configuration)

Sets or displays the DIO4/AD4 configuration (TH pin 11/SMT pin 24).

**Parameter range**

0 - 5

[1]Indicates that the option is available on the TH module, but not the SMT module.

[2]Indicates that the option is available on the TH module, but not the SMT module.

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SPI Slave Select[1] |
| 2 | Analog input |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

0

## D5 (DIO5 Configuration)

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

**Parameter range**

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | Associated LED |
| 3 | Digital input |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

## D8 (DIO8 Configuration)

Sets or displays the DIO8 configuration (TH pin 9/SMT pin 10).

**Parameter range**

0, 1, 3 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | SleepRq |

[1]Indicates that the option is available on the TH module, but not the SMT module.

| Parameter | Description |
|---|---|
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

**Default**

1

Sets or displays the DIO9 configuration (TH pin 13/SMT pin 26).

## Parameter range

0, 1, 3 - 5

| Parameter | Description |
|---|---|
| 0 | Disabled |
| 1 | On/Sleep indicator |
| 3 | Digital input, monitored |
| 4 | Digital output, default low |
| 5 | Digital output, default high |

## Default

1

## LT (Associate LED Blink Time)

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

If **LT** = **0**, the device uses the default blink rate of 250 ms.

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

**Parameter range**

0, 0x14 - 0xFF (200 - 2550 ms)

**Default**

0

## PR (Pull-up Resistor)

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

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

Sets or displays the bit field that configures the internal resistor status for the digital input lines. Internal pull-up/down resistors are not available for digital output pins, analog input pins, or for disabled pins.

Use the **PD** command to specify whether the resistor is pull-up or pull-down. If you set a **PR** bit to 1, it enables the pull-up resistor. If you set a **PR** bit to 0, it specifies no internal pull-up.

Pin numbers are listed with the TH module pin first, followed by the SMT device pin (for example, pin 11/24 indicates pin 11 on the through-hole device and pin 24 on the surface mount device).

| Bit | I/O line |
|---|---|
| 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/$\overline{\text{RTS}}$ (Pin 16/29) |
| 6 | DIO8/DTR/Sleep Request (Pin 9/10) |
| 7 | DIN/$\overline{\text{CONFIG}}$ (Pin 3/4) |
| 8 | DIO5/Associate (Pin 15/28) |
| 9 | DIO9/On/$\overline{\text{SLEEP}}$ (Pin 13/26) |
| 10 | DIO12 (Pin 4/5) |
| 11 | DIO10/PWM RSSI/PWM0 (Pin 6/7) |
| 12 | DIO11/PWM1 (Pin 7/8) |
| 13 | DIO7 / $\overline{\text{CTS}}$ (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) |

**Parameter range**

0 - 0x7FFF (TH)

0-0xFFFFF (SMT)

**Default**

0 - 0x7FFF (TH)

0-0xFFFFF (SMT)

## PD (Pull Direction)

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

If the bit is not set in **PR**, the device uses **PD**.

**Note** Resistors are not applied to disabled lines.

**Parameter range**

0x0 – 0x7FFF on TH

0x0 – 0xFFFFF on SMT

**Default**

0 – 0x7FFF on TH

0 – 0xFFFFF on SMT

## DS (Drive Strength)

Set or read the output drive strength (output amperes) for DIO lines. Bits are mapped the same as the **PR** and **PD** commands. If you set the bit, the drive strength is 6 mA; otherwise, it is 2 mA.

**Parameter range**

0 – 0x7FFF on TH

0 – 0xFFFFF on SMT

**Default**

0

## AV (Analog Voltage Reference)

Set or 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 the values in the following table.

**Parameter range**

0, 1

| Parameter | Description |
|---|---|
| 0 | 1.25 V reference |
| 1 | 2.5 V reference |

**Default**

1

## M0 (PWM0 Duty Cycle)

Sets the duty cycle of PWM0 for **P0** = **2**, where a value of 0x200 is a 50% duty cycle.

**Parameter range**

0 - 0x3FF

**Default**

0

### M1 (PWM1 Duty Cycle)

Sets the duty cycle of PWM1 for **P1** = **2**, where a value of 0x200 is a 50% duty cycle.

**Parameter range**

0 - 0x3FF

**Default**

0

## Output Control

The following AT commands are output control commands.

### IO command

Set output pins to the designated level. Bit 0 corresponds with DIO0 and bit 1 with DIO1, up to bit 19 that corresponds with DIO19. See Output control for a functional description.

**Parameter range**

0 to 0x7FFF on through-hole

0 to 0xFFFFF on surface-mount

**Default**

N/A

### OM (Output Mask)

Sets the output mask for the IO command. If you set a bit, the corresponding bit in the IO command is enabled. If it is clear, then that same bit has no effect in the IO command.

**Parameter range**

0 to 0x7FFF on through-hole

0 to 0xFFFFF on surface-mount

**Default**

0 to 0x7FFF on through-hole

0 to 0xFFFFF on surface-mount

### 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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

0

## 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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

0

## 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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

0

## Q4 (Set time to hold DIO14)

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

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 for a functional description.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Note** This option is available on the SMT module, but not the TH module.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Note** This option is available on the SMT module, but not the TH module.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Note** This option is available on the SMT module, but not the TH module.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

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 for a functional description.

**Note** This option is available on the SMT module, but not the TH module.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

0

### Q9 (Set time to hold DIO19)

Sets how long an output level programmed by bit 19 of IO command is held in the selected state before reverting to its configured level. See Output control for a functional description.

**Note** This option is available on the SMT device, but not the TH device.

**Parameter range**

0 - 0x1770 (x 100 ms)

**Default**

0

## Diagnostics interfacing

### VR (Firmware Version)

Reads the firmware version on a device.

The firmware version returns four 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.

**Parameter range**

0 - 0xFFFF [read-only]

**Default**

Set in the factory

### HV (Hardware Version)

Read the device's hardware version. Use this command to distinguish between different hardware platforms. The upper byte returns a value that is unique to each device type. The lower byte indicates the hardware revision.

**Note** XBee Wi-Fi modules return 0x1Fxx for the **HV** command.

**Parameter range**

0 - 0xFFFF [read-only]

**Default**

Set in firmware

## HS (Hardware Series)

Read the device's hardware series number.

The XBee Wi-Fi RF Module should return 0x601 for S6B.

**Parameter range**

N/A

**Default**

N/A

## AI (Association Indication)

Read information regarding last node join request.

| Status code | Meaning |
|---|---|
| 0x00 | Successfully joined an access point, established IP addresses and IP listening sockets. |
| 0x01 | Wi-Fi transceiver initialization in progress. |
| 0x02 | Wi-Fi transceiver initialized, but not yet scanning for access point. |
| 0x13 | Disconnecting from access point. |
| 0x23 | SSID not configured. |
| 0x24 | Encryption key invalid (either NULL or invalid length for WEP). |
| 0x27 | SSID was found, but join failed. 0x40- Waiting for WPA or WPA2 Authentication. |
| 0x41 | Device joined a network and is waiting for IP configuration to complete, which usually means it is waiting for a DHCP provided address. |
| 0x42 | Device is joined, IP is configured, and listening sockets are being set up. |
| 0xFF | Device is currently scanning for the configured SSID. |

**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.

**Parameter range**

0 - 0xFF [read-only]

**Default**

N/A

## DI (Device Cloud Indicator)

Displays the current Remote Manager status for the XBee Wi-Fi RF Module.

**Parameter range**

0 - 4

| Value | Description |
|---|---|
| **0** | Remote Manager is connected |
| **1** | Initial state |
| **2** | Attempting to connect to Remote Manager |
| **3** | Disconnecting from Remote Manager |
| **4** | Remote Manager not configured |

**Default**

N/A

## AS (Active Scan)

Scan for access points in the vicinity. Issue this command in command mode or API mode to return the following information:

02 - Indicates scan type of 802.11 in this format unique to S6B.

CH - Channel number in use by access point.

ST - Security type where: 00=open, 01=WPA, 02=WPA2, and 03=WEP.

LM - Link Margin (Signal strength in dB above sensitivity).

ID - SSID of access point found.

When you issue this command in command mode, this 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.

When you issue the command in API mode, each record (that is, each access point) outputs a separate AT Command Response frame - 0x88 with the above fields in binary format. The command terminates with a null AT Command Response packet.

In Command mode, the **AS** command terminates with an additional carriage return.

**Note** This command is not available as a remote command. Also, 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.

**Parameter range**

N/A

**Default**

N/A

### TP command

Displays the temperature of the XBee Wi-Fi RF Module in degrees Celsius. The temperature value is displayed in 8-bit two's compliment format. For example, **0x1A** = 26 ℃, and **0xF6** = -10 ℃.

**Parameter range**

-30 to 85 ℃

**Default**

N/A

### CK (Configuration Code)

Reads the configuration code associated with the current AT command configuration.

**Parameter range**

2 bytes

**Default**

N/A

### %V (Supply Voltage)

Displays the supply voltage of the device in mV units.

**Parameter range**

3.1 to 3.5 V

**Default**

N/A

### LM (Link Margin)

Reads the received signal strength (RSSI) in terms of dB units above sensitivity and reports **0xFF** until the first reception after connection to access point.

**Parameter range**

0 - 0xFF

**Default**

N/A

## Command mode options

The following commands are Command mode option commands.

### CT (Command Mode Timeout)

Sets or displays the period of inactivity (that is, no valid commands received) after which the RF module automatically exits Command Mode and returns to Idle Mode. This time period can be up to ten minutes.

**Parameter range**

2 - 0x1770 (x 100 ms)

**Default**

0x64 (10 seconds)

### CN (Exit Command mode)

Immediately exits Command Mode and applies pending changes.

**Note** Whether Command mode is exited using the **CN** command or by **CT** timing out, changes are applied upon exit.

**Parameter range**

N/A

**Default**

N/A

### GT (Guard Times)

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

**Parameter range**

0x2 - 0x0576 [x 1 ms] (max of 1.4 seconds)

**Default**

0x3E8 (one second)

### CC (Command Mode Character)

Sets or displays the command mode character used between guard times of the Command mode sequence (**GT** + **CC** + **CC** + **CC** + **GT**). This sequence allows the device to enter into Command mode.

**Parameter range**

0 - 0xFF

**Default**

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

## Sleep commands

The following AT commands are sleep commands.

### SM (Sleep Mode)

Sets or displays the sleep mode of the device.

The sleep mode determines how the device enters and exits a power saving sleep.

Sleep mode is also affected by the **SO** command, option bit 6. See Sleep modes for more information about sleep modes.

**Parameter range**

0, 1, 4, 5

| Parameter | Description |
|---|---|
| 0 | Normal. In this mode the device never sleeps. |
| 1 | Pin Sleep. In this mode<br>the device honors the SLEEP_RQ pin. Set D8 (DIO8 Configuration) to the sleep request function: **1**. |
| 4 | Cyclic Sleep. In this mode<br>the device repeatedly sleeps for the value specified by **SP** and spends **ST** time awake. |
| 5 | Cyclic Sleep with Pin Wake. In this mode the device acts as in Cyclic Sleep but does not sleep if the SLEEP_RQ pin is inactive, allowing the device to be kept awake or woken by the connected system. |

**Default**

0

## SP (Sleep Period)

This value determines how long the device sleeps, up to 24 hours or 86,400 seconds. This corresponds to 0x83d600 in 10 ms units.

**Parameter range**

1 - 0x83D600 (x 10 ms)

**Default**

0xC8 (2 seconds)

## SO (Sleep Options)

Set or read the sleep mode options.

| Bit | Option |
|---|---|
| 0x40 | Stay associated with AP during sleep. Draw more current during sleep with this option enabled, but also avoid data loss. |
| 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. |

**Parameter range**

0 - 0x01FF

**Default**

0x100

### WH (Wake Host)

Sets or displays the wake host timer value.

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

**Parameter range**

0 - 0xFFFF (x 1 ms)

**Default**

0

### ST (Wake Time)

Sets or displays the time to spend awake in cyclic sleep modes.

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.

**Parameter range**

0x1 - 0x36EE80 (x 1 ms)

**Default**

0x7D0 (3 seconds)

### SA command

Time to wait for association before entering deep sleep. Wakeup from deep sleep is much faster if association occurs before going to sleep.

**Parameter range**

0x1 – 0x36EE80 (x1 ms)

**Default**

0x2710 (10 seconds)

## Execution commands

The location where most AT commands set or query register values, execution commands execute an action on the device. Execution commands are executed immediately and do not require changes to be applied.

### AC (Apply Changes)

Applies changes to all command registers and applies queued command register values. For example, changing the serial interface rate with the **BD** command does not change the UART interface rate

until changes are applied with the **AC** command. The **CN** command and 0x08 API command frame also apply changes.

**Parameter range**

N/A

**Default**

N/A

## WR (Write)

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

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

**Parameter range**

N/A

**Default**

N/A

## RE (Restore Defaults)

Restore device parameters to factory defaults.

**Parameter range**

N/A

**Default**

N/A

## FR (Software Reset)

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

Resets the device. The device responds immediately with an **OK** and performs a software reset approximately 2 seconds later.

**Parameter range**

N/A

**Default**

N/A

## NR (Network Reset)

Resets the network layer. For Wi-Fi, this means disassociating 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**).

**Note** **NR** and **NR0** both perform the same function and may be used interchangeably.

**Parameter range**

0

**Default**

N/A

## CB (Commissioning Pushbutton)

Use **CB** to simulate commissioning pushbutton presses in software.

**Parameter range**

2, 4

| Parameter | Description |
|---|---|
| 2 | WPS push button configuration. |
| 4 | Force Provisioning in Soft AP mode by issuing an **NR** command. |

**Default**

N/A

# Regulatory information

Systems that contain XBee Wi-Fi RF Modules inherit Digi 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 RF 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 RF Module may only be used with antennas that have been tested and approved for use with this module (for details, see FCC-approved antennas (2.4 GHz)).

## 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 RF Module can be installed using antennas and cables constructed with non-standard connectors (RPSMA, RPTNC, and so forth). 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 20 cm (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.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

The following table shows antennas approved for use with the through-hole device.

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| **Integrated antennas** | | | | | | | |
| 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 |
| 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** | | | | | | | |
| A24-F2NF | Omni-Directional (Fiberglass base station) | 2.1 dBi | Fixed/Mobile | 20 cm | N/A | N/A | N/A |

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| 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 |
| 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** | | | | | | | |
| 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 |

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| 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** | | | | | | | |
| 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 |

The following table shows antennas approved for use with the surface-mount device.

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| **Integrated antennas** | | | | | | | |

| Part number | Type (description) | Gain (dBi) | 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** | | | | | | | |
| 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** | | | | | | | |
| 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 |

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| 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** | | | | | | | |
| 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** | | | | | | | |
| 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 |

| Part number | Type (description) | Gain (dBi) | Application | Min. Separation | Minimum cable loss/power reduction/attenuation required | | |
|---|---|---|---|---|---|---|---|
| | | | | | b mode | g mode | n mode |
| 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 |
| 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

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

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 radio modules are certified for use in several European countries. For a complete list, refer to www.digi.com/resources/certifications.

If the XBee Wi-Fi is incorporated into a product, the manufacturer must ensure compliance of the final product with articles 3.1a and 3.1b of the RE Directive (Radio Equipment Directive). A Declaration of Conformity must be issued for each of these standards and kept on file as described in the RE Directive (Radio Equipment Directive).

Furthermore, the manufacturer must maintain a copy of the XBee Wi-Fi 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.

## Maximum power and frequency specifications

For the XBee S6B through-hole device:

- Maximum RF output power: 95.06 mW (19.78 dBm) Equivalent Isotropically Radiated Power (EIRP).

The radio can transmit on the following 13 802.11 channels:

- 2412, 2417, 2422, 2427, 2432, 2437, 2442, 2447, 2452, 2457, 2462, 2467, 2472.

For the XBee S6B surface-mount device:

- Maximum RF output power: 81.47 mW (19.11 dBm) EIRP.

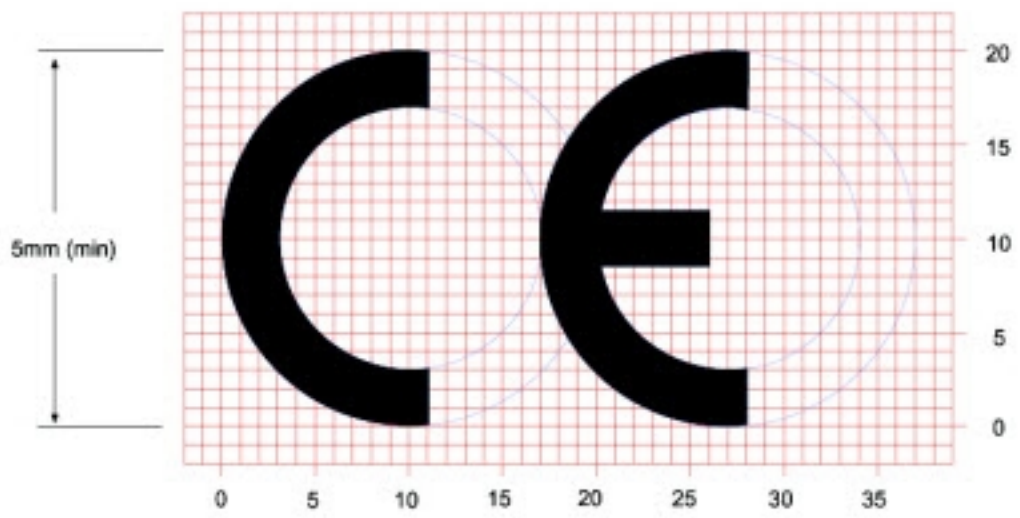
The radio can transmit on the following 13 802.11 channels:

- 2412, 2417, 2422, 2427, 2432, 2437, 2442, 2447, 2452, 2457, 2462, 2467, 2472.

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

## Declarations of conformity

Digi has issued Declarations of Conformity for the (product name) concerning emissions, EMC, and safety. For more information, see www.digi.com/resources/certifications.

### *Important note*

Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. Refer to the radio regulatory agency in the desired countries of operation for more information.

## Approved antennas

When integrating high-gain antennas, European regulations stipulate EIRP power maximums.

All antenna part numbers followed by an asterisk (*) are not available from Digi. Consult with an antenna manufacturer for an equivalent option.

The following antennas are approved for use with the XBee Wi-Fi RF 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 XBEES6B 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 (EIRP) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire àl'établissement d'une communication satisfaisante.

## 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.


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

The surface-mount XBee Wi-Fi RF Module is designed for surface-mount on the OEM PCB. It has castellated pads to allow for easy solder attach inspection. The pads are all located on the edge of the device so there are no hidden solder joints on these devices.

# Recommended solder reflow cycle

The following table lists the recommended solder reflow cycle. The chart shows the temperature setting and the time to reach the temperature.

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

The maximum temperature should not exceed 260 ℃.

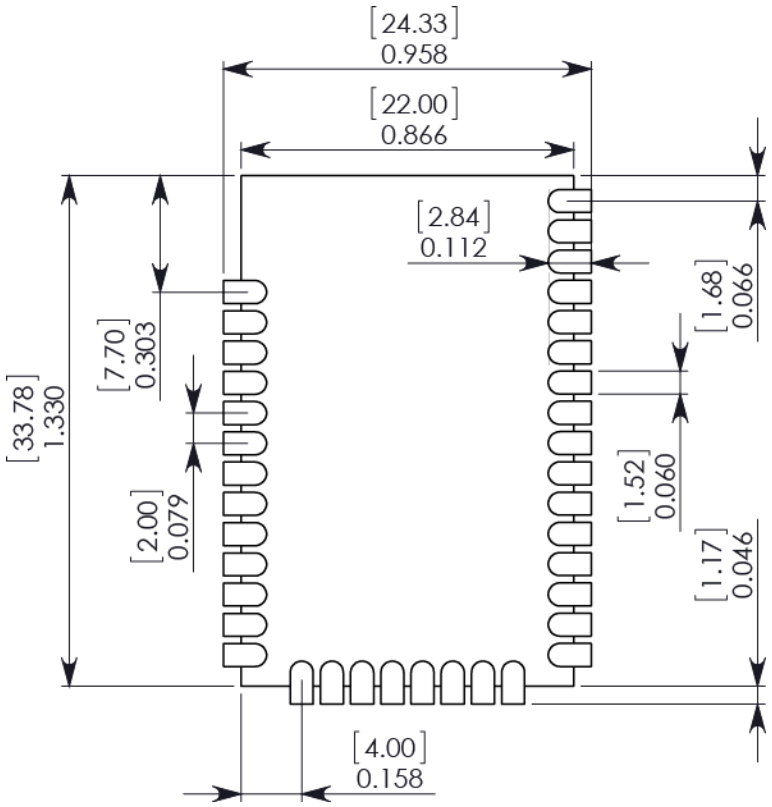
The device reflows during this cycle, and must not be reflowed upside down. Be careful not to jar the device while the solder is molten, as parts inside the device can be removed from their required locations.

Hand soldering is possible and should be done in accordance with approved standards.

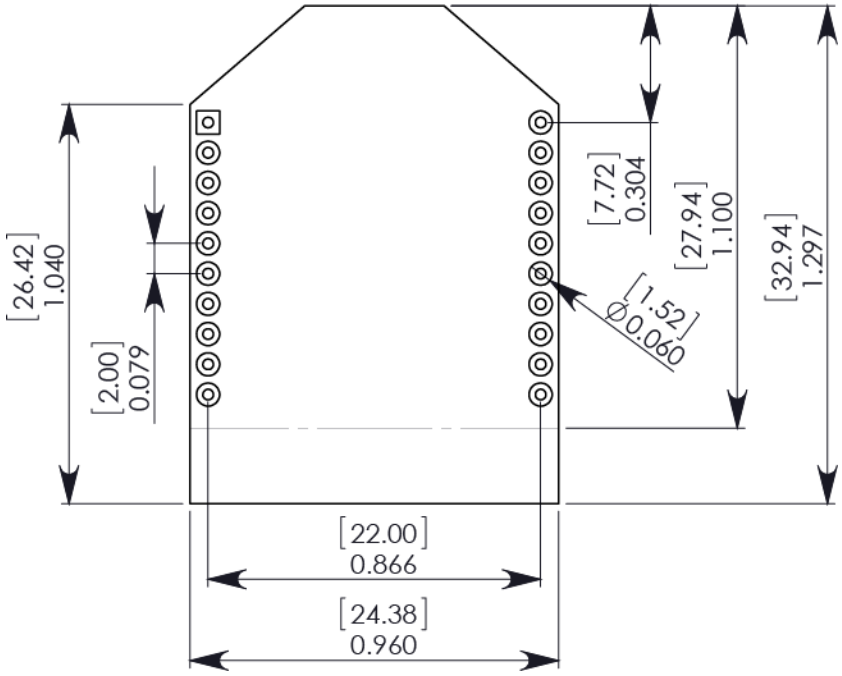
This device has a Moisture Sensitivity Level (MSL) of 3.

# 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.

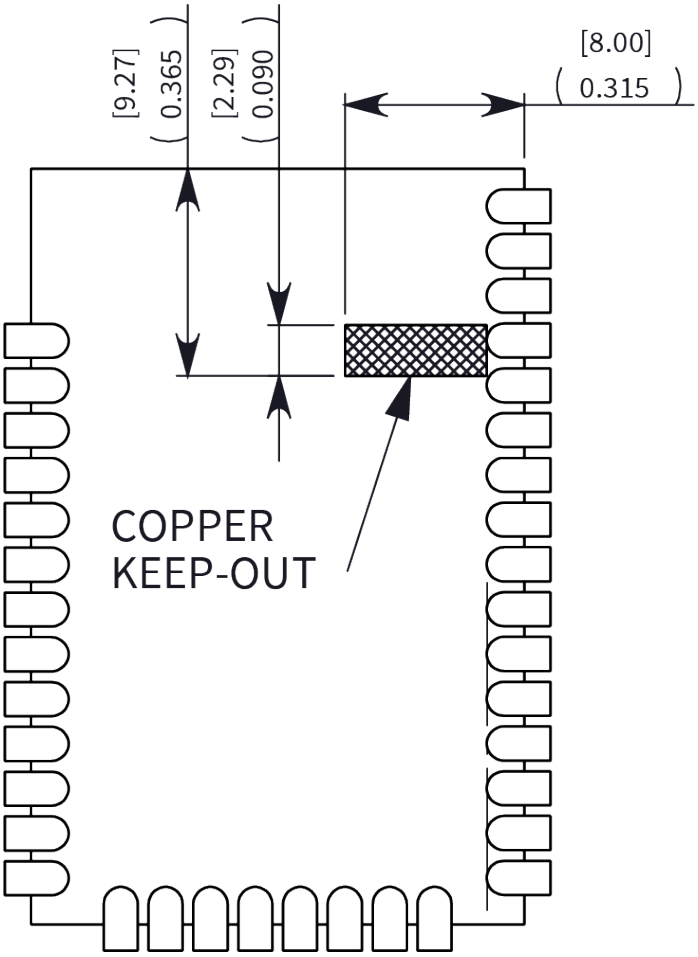



The following image shows the through-hole module.




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 device is mostly coated with solder resist, but we recommend that you leave the copper layer directly below the device open to avoid unintended contacts. Copper or vias must not interfere with the three exposed RF test points on

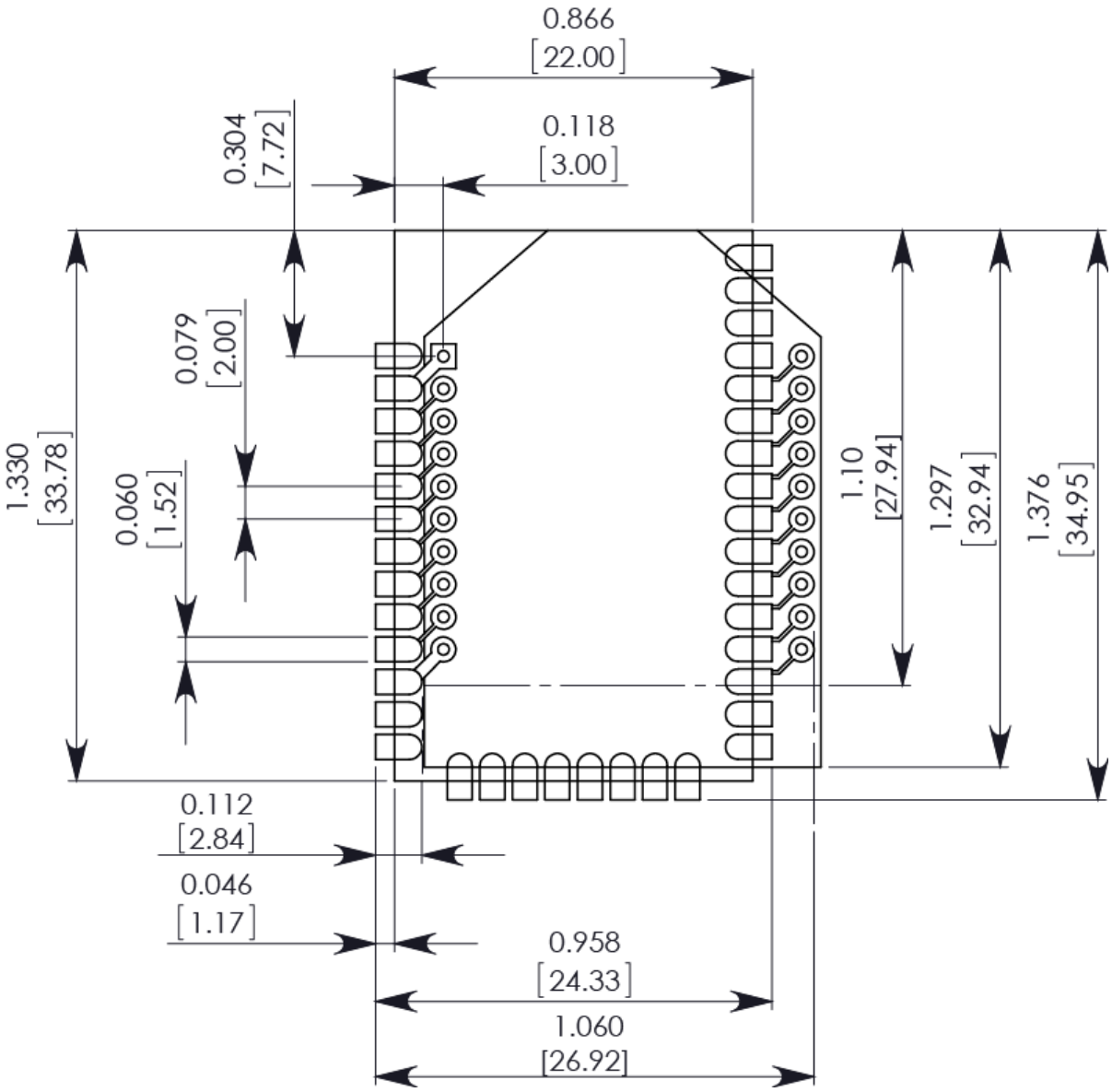
the bottom of the device (see below). These devices have a ground plane in the middle on the back side for shielding purposes, which can be affected by copper traces directly below the device.




## Mount the devices

One important difference between the SMT and TH devices is the way they mount to a printed circuit board (PCB). Each footprint requires different mounting techniques.

We designed a footprint that allows you to attach either device to a PCB. The following drawing shows the 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 devices.

## Flux and cleaning

We recommend that you use a “no clean” solder paste in assembling these devices. This eliminates the clean step and ensures that you do not leave unwanted residual flux under the device where it is difficult to remove. In addition:

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

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

# Rework

**CAUTION!** Any modification to the device voids the warranty coverage and certifications.

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.