XBee®/XBee-PRO® ZB RF Modules

ZigBee RF Modules by Digi International

Firmware Versions:

- 20xx - Coordinator - AT/Transparent Operation
- 21xx - Coordinator - API Operation
- 22xx - Router - AT/Transparent Operation
- 23xx - Router - API Operation
- 28xx - End Device - AT/Transparent Operation
- 29xx - End Device - API Operation

Digi International Inc.
11001 Bren Road East
Minnetonka, MN 55343
877 912-3444 or 952 912-3444
http://www.digi.com

90000976_C
3/3/2009

Technical Support: Phone: (801) 765-9885 Worldwide
(866) 765-9885 toll-free U.S.A. & Canada

Live Chat: www.digi.com

Contents

Contents

Contents

1. Overview

The XBee/XBee-PRO ZB RF Modules are designed to operate within the ZigBee protocol and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices.

The modules operate within the ISM 2.4 GHz frequency band and are compatible with the following:

- XBee RS-232 Adapter
- XBee RS-232 PH (Power Harvester) Adapter
- XBee RS-485 Adapter
- XBee Analog I/O Adapter
- XBee Digital I/O Adapter
- XBee Sensor Adapter
- XBee USB Adapter
- XStick
- ConnectPort X Gateways
- XBee Wall Router.

The XBee/XBee-PRO ZB firmware release can be installed on XBee ZNet or ZB modules. The XBee ZB firmware is based on the EmberZNet 3.x ZigBee PRO Feature Set mesh networking stack, while the XBee ZNet 2.5 firmware is based on Ember's proprietary "designed for ZigBee" mesh stack (EmberZNet 2.5.x). ZB and ZNet 2.5 firmware are similar in nature, but not over-the-air compatible. Devices running ZNet 2.5 firmware cannot talk to devices running the ZB firmware.

Key Features

High Performance, Low Cost

XBee

- Indoor/Urban: up to 133’ (40 m)
- Outdoor line-of-sight: up to 400’ (120 m)
- Transmit Power: 2 mW (3 dBm)
- Receiver Sensitivity: -96 dBm

XBee-PRO

- Indoor/Urban: up to 300’ (90 m), 200' (60 m) for International variant
- Outdoor line-of-sight: up to 1 mile (1600 m), 2500' (750 m) for International variant
- Transmit Power: 50mW (17dBm), 10mW (10dBm) for International variant
- Receiver Sensitivity: -102 dBm

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channel has over 65,000 unique network addresses available

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Self-routing, self-healing and fault-tolerant mesh networking

Low Power

XBee

- TX Peak Current: 40 mA (@3.3 V)
- RX Current: 40 mA (@3.3 V)
- Power-down Current: < 1 uA

XBee-PRO

- TX Peak Current: 295mA (170mA for international variant)
- RX Current: 45 mA (@3.3 V)
- Power-down Current: < 10 uA

Easy-to-Use

No configuration necessary for out-of box RF communications

AT and API Command Modes for configuring module parameters

Small form factor

Extensive command set

Free X-CTU Software
(Testing and configuration software)

Free & Unlimited Technical Support

Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A for FCC Requirements.
Systems that contain XBee®/XBee-PRO® ZB RF Modules inherit Digi Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee®/XBee-PRO® ZB RF Modules are optimized for use in **US**, **Canada**, **Australia, Israel and Europe** (contact MaxStream for complete list of agency approvals).

Specifications

Table 1-01. Specifications of the XBee®/XBee-PRO® ZB OEM RF Module

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 133 ft. (40 m)	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)	Up to 1 mile (1600 m), up to 2500 ft (750 m) international variant
Transmit Power Output	2mW (+3dBm), boost mode enabled 1.25mW (+1dBm), boost mode disabled	50mW (+17 dBm) 10mW (+10 dBm) for International variant
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 230400 bps (non-standard baud rates also supported)	1200 - 230400 bps (non-standard baud rates also supported)
Receiver Sensitivity	-96 dBm, boost mode enabled -95 dBm, boost mode disabled	-102 dBm

Table 1-01. Specifications of the XBee®/XBee-PRO® ZB OEM RF Module

Specification	XBee	XBee-PRO
Power Requirements		
Supply Voltage	2.1 - 3.6 V	3.0 - 3.4 V
Operating Current (Transmit, max output power)	40mA (@ 3.3 V, boost mode enabled) 35mA (@ 3.3 V, boost mode disabled)	295mA (@3.3 V), 170mA (@3.3 V) international variant
Operating Current (Receive))	40mA (@ 3.3 V, boost mode enabled) 38mA (@ 3.3 V, boost mode disabled)	45 mA (@3.3 V)
Idle Current (Receiver off)	15mA	15mA
Power-down Current	< 1 uA @ 25^{o}C	< 10 uA @ 25^{o}C
General		
Operating Frequency Band	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960 x 1.297 (2.438cm x 3.294cm)
Operating Temperature	-40 to 85º C (industrial)	-40 to 85º C (industrial)
Antenna Options	Integrated Whip, Chip, RPSMA, or U.FL Connector*	Integrated Whip, Chip, RPSMA, or U.FL Connector*
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh
Number of Channels	16 Direct Sequence Channels	14 Direct Sequence Channels
Addressing Options	PAN ID and Addresses, Cluster IDs and Endpoints (optional)	PAN ID and Addresses, Cluster IDs and Endpoints (optional)
Agency Approvals		
United States (FCC Part 15.247)	FCC ID: OUR-XBEE2	FCC ID: MCQ-XBEEPRO2
Industry Canada (IC)	IC: 4214A-XBEE2	IC: 1846A-XBEEPRO2
Europe (CE)	ETSI	ETSI
Australia	C-Tick	C-Tick
Japan	R201WW07215214	R201WW08215142
RoHS	Compliant	Compliant

Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee®/XBee-PRO® ZB OEM RF Modules (antenna options not shown)

.

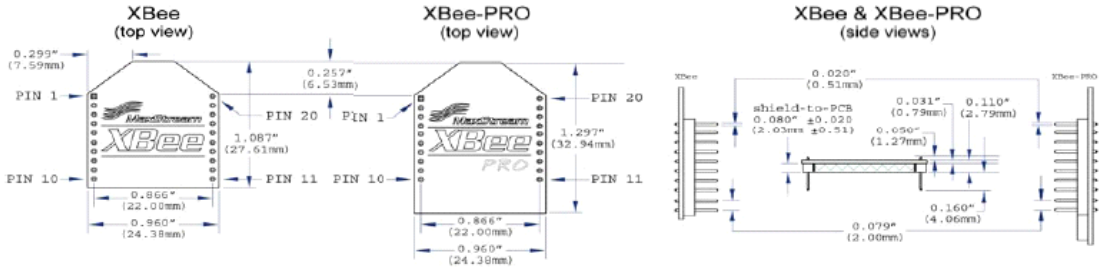
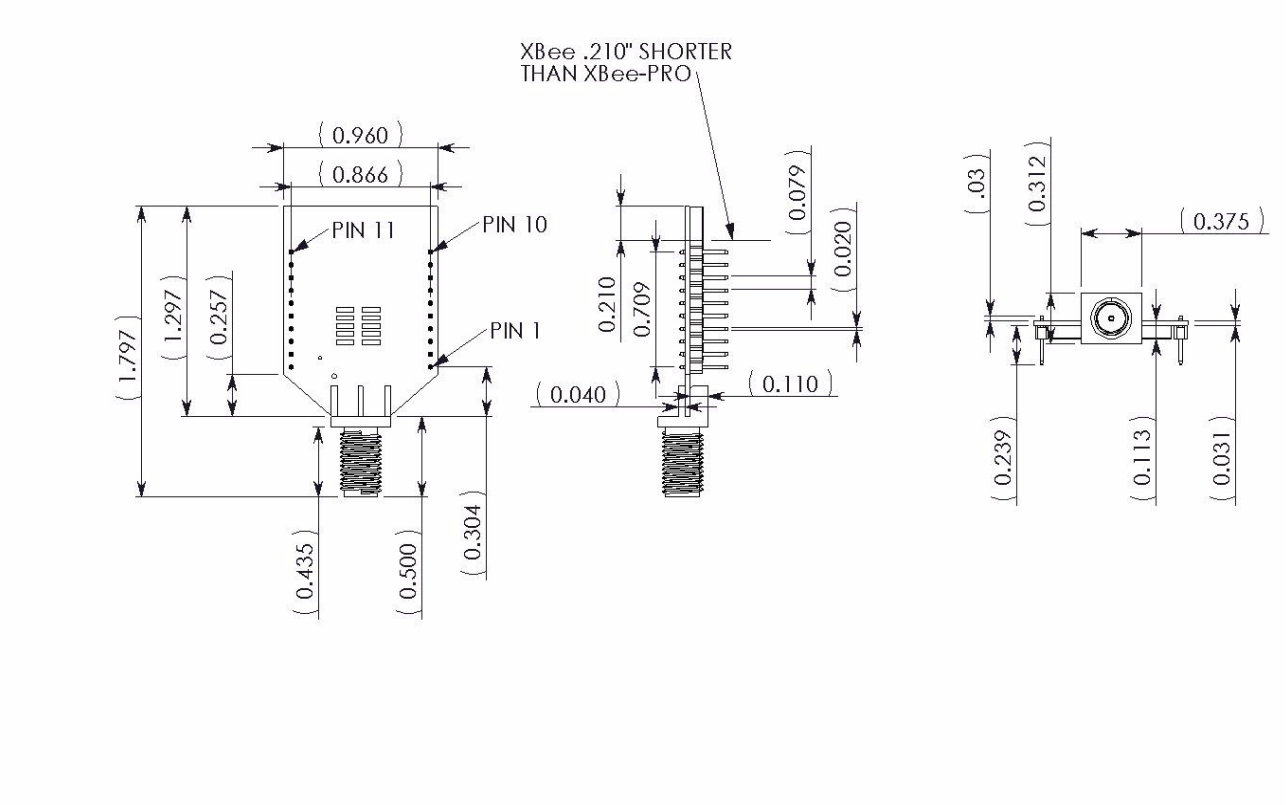
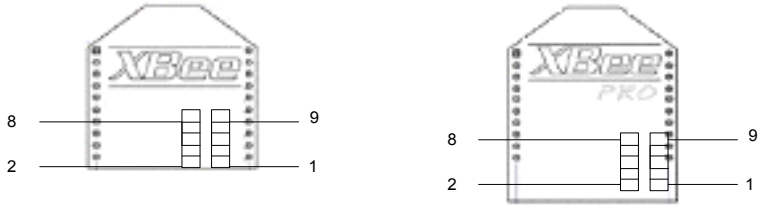


Figure 1-02. Mechanical Drawings for the RPSMA Variant



SIF Header Interface

The XBee/XBee-PRO ZB modules include a SIF programming header that can be used with Ember's programming tools to upload custom firmware images onto the XBee module. The SIF header orientation and pinout are shown below.



This figure shows the orientation of the insight port header .

Pin Number	Pin Name
1	VBRD
2	SIF-MISO
3	Ground
4	SIF-MOSI
5	Ground
6	SIF-CLOCK
7	SIF-LOAD
8	RESET
9	PTI-EN
10	PTI-DATA

A male header can be populated on the XBee that mates with Ember's 2x5 ribbon cable. The male header and ribbon cables are available from Samtec:

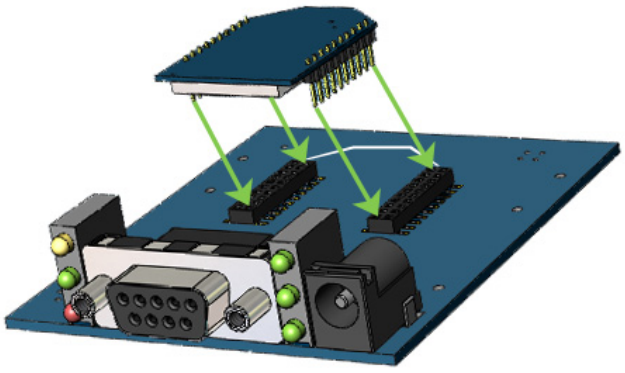
2x5 Male Header - FTSH-105-01-F-DV-K

2x5 Ribbon Cable - FFSD-05-D-12.00-01-N

Mounting Considerations

The XBee modules were designed to mount into a receptacle (socket) and therefore does not require any soldering when mounting it to a board. The XBee-PRO Development Kits contain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules.

Figure 1-03. XBee-PRO Module Mounting to an RS-232 Interface Board.



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

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

- Surface-mount single-row receptacles -
  Samtec P/N: SMM-110-02-SM-S

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

Pin Signals

Figure 1-04. XBee®/XBee-PRO® ZB RF Module Pin Number
(top sides shown - shields on bottom)

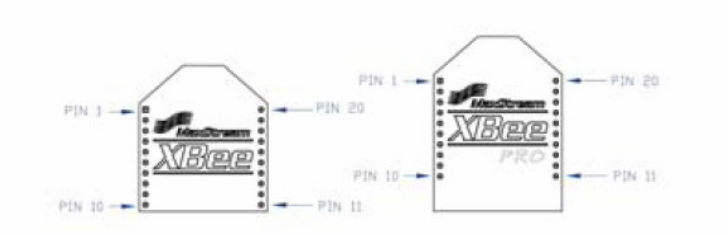


Table 1-02. Pin Assignments for the XBee-PRO Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DIO12	Either	Digital I/O 12
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI / DIO10	Either	PWM Output 0 / RX Signal Strength Indicator / Digital IO
7	DIO11	Either	Digital I/O 11
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ/ DIO8	Either	Pin Sleep Control Line or Digital IO 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7. CTS, if enabled, is an output.
13	ON / **SLEEP**	Output	Module Status Indicator or Digital I/O 9
14	VREF	Input	Not used on this module. For compatibility with other XBee modules, we recommend connecting this pin to a voltage reference if Analog sampling is desired. Otherwise, connect to GND.
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 5
16	**RTS** / DIO6	Either	Request-to-Send Flow Control, Digital I/O 6. RTS, if enabled, is an input.
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Either	Analog Input 0, Digital IO 0, or Commissioning Button

- Signal Direction is specified with respect to the module
- See Design Notes section below for details on pin connections.
- PWM functionality not currently supported.

EM250 Pin Mappings

The following table shows how the EM250 pins are used on the XBee.

EM250 Pin Number	XBee Pin Number	Other Usage
13 (Reset)	5	Connected to pin 8 on 2x5 SIF header.
19 (GPIO 11)	16	
20 (GPIO 12)	12	
21 (GPIO 0)	15	
22 (GPIO 1)		**XBee** Tied to ground (module identification) **XBee-PRO** Low-asserting shutdown line for output power compensation circuitry.
24 (GPIO 2)		**XBee** Not connected. Configured as output low. **XBee-PRO** Powers the output power compensation circuitry.
25 (GPIO 3)	13	
26 (GPIO 4 / ADC 0)	20	Connected to pin 9 on 2x5 SIF header.
27 (GPIO 5 / ADC 1)	19	Connected to pin 10 on 2x5 SIF header.
29 (GPIO 6 /ADC 2)	18	
30 (GPIO 7 / ADC 3	17	
31 (GPIO 8)	4	
32 (GPIO 9)	2	
33 (GPIO 10)	3	
34 (SIF_CLK)		Connected to pin 6 on 2x5 SIF header.
35 (SIF_MISO)		Connected to pin 2 on 2x5 SIF header.
36 (SIF_MOSI)		Connected to pin 4 on 2x5 SIF header.
37 (SIF_LOAD)		Connected to pin 7 on 2x5 SIF header.
40 (GPIO 16)	7	
41 (GPIO 15)	6	
42 (GPIO 14)	9	
43 (GPIO 13)	11	

Design Notes

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

Power Supply Design

Poor power supply can lead to poor radio performance especially if the supply voltage is not kept within tolerance or is excessively noisy. To help reduce noise a 1.uF and 8.2pF capacitor are recommended to be placed as near to pin1 on the PCB as possible. If using a switching regulator for your power supply, switching frequencies above 500kHz are preferred. Power supply ripple should be limited to a maximum 250mV peak to peak.

Recommended Pin Connections

The only required pin connections are VCC, GND, DOUT and DIN. To support serial firmware updates, VCC, GND, DOUT, DIN, RTS, and DTR should be connected.

All unused pins should be left disconnected. All inputs on the radio can be pulled high with 30k internal pull-up resistors using the PR software command. No specific treatment is needed for unused outputs.

Other pins may be connected to external circuitry for convenience of operation including the Associate LED pin (pin 15) and the Commissioning pin (pin 20). The Associate LED pin will flash differently depending on the state of the module to the network, and a pushbutton attached to pin 20 can enable various join functions without having to send UART commands. Please see the commissioning pushbutton and associate LED section in chapter 7 for more details. The source and sink capabilities are limited to 4mA for all pins on the module.

The VRef pin (pin 14) is not used on this module. For compatibility with other XBee modules, we recommend connecting this pin to a voltage reference if analog sampling is desired. Otherwise, connect to GND.

Board Layout

XBee modules do not have any specific sensitivity to nearby processors, crystals or other PCB components. Other than mechanical considerations, no special PCB placement is required for integrating XBee radios. In general, Power and GND traces should be thicker than signal traces and be able to comfortably support the maximum currents.

The radios are also designed to be self sufficient and work with the integrated and external antennas without the need for additional ground planes on the host PCB. Large ground planes on a host PCB should not adversely affect maximum range, but they may affect radiation patterns of onboard XBee antennas.

Electrical Characteristics

Table 1-03. DC Characteristics of the XBee-PRO (VCC = 3.0 - 3.4 VDC).

Symbol	Parameter	Condition	Min	Typical	Max	Units
VIL	Input Low Voltage	All Digital Inputs	-	-	0.2 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.8 * VCC	-	-	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-	0.18*VCC	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	0.82*VCC	-	-	V
II_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	-	0.5uA	uA

2. RF Module Operation

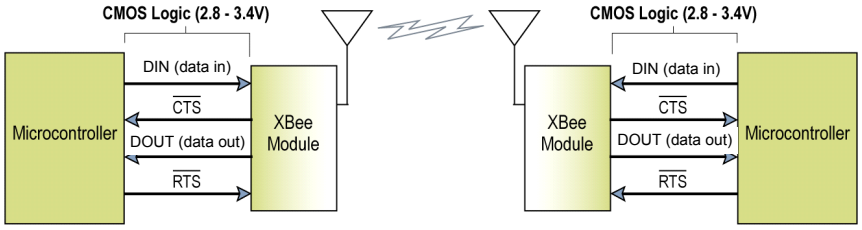
Serial Communications

The XBee OEM RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (For example: Through a Digi proprietary RS-232 or USB interface board).

UART Data Flow

Devices that have a UART interface can connect directly to the pins of the RF module as shown in the figure below.

Figure 2-01. System Data Flow Diagram in a UART-interfaced environment
(Low-asserted signals distinguished with horizontal line over signal name.)

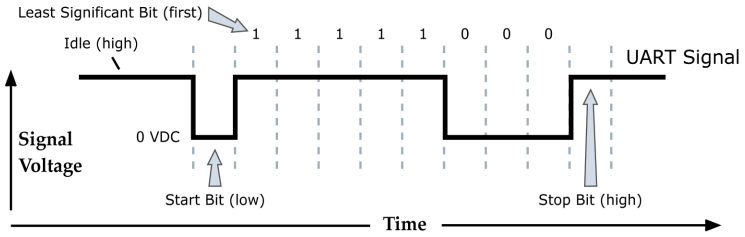


Serial Data

Data enters the module UART through the DIN (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

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

Figure 2-02. UART data packet 0x1F (decimal number "31") as transmitted through the RF module
Example Data Format is 8-N-1 (bits - parity - # of stop bits)

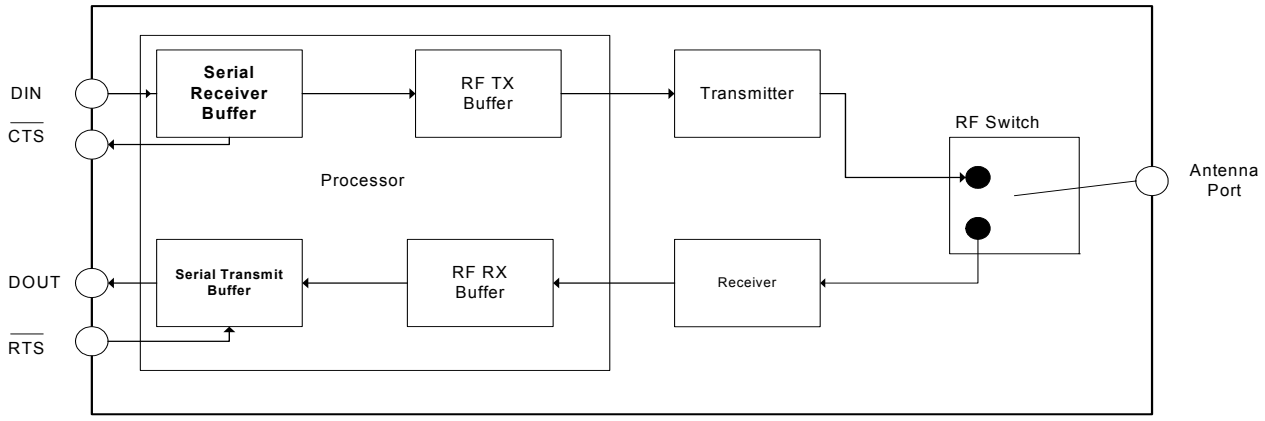


The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).

Serial Buffers

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

Figure 2-03. Internal Data Flow Diagram



Serial Receive Buffer

When serial data enters the RF module through the DIN Pin (pin 3), the data is stored in the serial receive buffer until it can be processed. Under certain conditions, the module may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the module, $\overline{CTS}$ flow control may be required to avoid overflowing the serial receive buffer.

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

1.If the module is receiving a continuous stream of RF data, the data in the serial receive buffer will not be transmitted until the module is no longer receiving RF data.

2.If the module is transmitting an RF data packet, the module may need to discover the destination address or establish a route to the destination. After transmitting the data, the module may need to retransmit the data if an acknowledgment is not received, or if the transmission is a broadcast. These issues could delay the processing of data in the serial receive buffer.

Serial Transmit Buffer

When RF data is received, the data is moved into the serial transmit buffer and sent out the UART. If the serial transmit buffer becomes full enough such that all data in a received RF packet won't fit in the serial transmit buffer, the entire RF data packet is dropped.

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

1. If the RF data rate is set higher than the interface data rate of the module, the module could receive data faster than it can send the data to the host.

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

Serial Flow Control

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

$\overline{CTS}$ Flow Control

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

$\overline{\text{RTS}}$ Flow Control

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

Serial Interface Protocols

The XBee modules support both transparent and API (Application Programming Interface) serial interfaces.

Transparent Operation

When operating in transparent mode, the modules act as a serial line replacement. All UART data received through the DIN pin is queued up for RF transmission. When RF data is received, the data is sent out through the DOUT pin. The module configuration parameters are configured using the AT command mode interface.

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

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

RF modules that contain the following firmware versions will support Transparent Mode: 20xx (AT coordinator), 22xx (AT router), and 28xx (AT end device).

API Operation

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

Transmit Data Frames (received through the DIN pin (pin 3)) include:

- RF Transmit Data Frame
- Command Frame (equivalent to AT commands)

Receive Data Frames (sent out the DOUT pin (pin 2)) include:

- RF-received data frame
- Command response
- Event notifications such as reset, associate, disassociate, etc.

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

The API operation option facilitates many operations such as the examples cited below:

-> Transmitting data to multiple destinations without entering Command Mode

-> Receive success/failure status of each transmitted RF packet

-> Identify the source address of each received packet

RF modules that contain the following firmware versions will support API operation: 21xx (API coordinator), 23xx (API router), and 29xx (API end device).

A Comparison of Transparent and API Operation

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

Transparent Operation Features	
Simple Interface	All received serial data is transmitted unless the module is in command mode.
Easy to support	It is easier for an application to support transparent operation and command mode
API Operation Features	
Easy to manage data transmissions to multiple destinations	Transmitting RF data to multiple remotes only requires changing the address in the API frame. This process is much faster than in transparent operation where the application must enter AT command mode, change the address, exit command mode, and then transmit data. Each API transmission can return a transmit status frame indicating the success or reason for failure.
Received data frames indicate the sender's address	All received RF data API frames indicate the source address.
Advanced ZigBee addressing support	API transmit and receive frames can expose ZigBee addressing fields including source and destination endpoints, cluster ID and profile ID. This makes it easy to support ZDO commands and public profile traffic.
Advanced networking diagnostics	API frames can provide indication of IO 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.

As a general rule of thumb, API firmware is recommended 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
- must support multiple ZigBee endpoints, cluster IDs, and/or profile IDs
- uses the ZigBee Device Profile services.

If the above conditions do not apply (i.e. a sensor node, router, or a simple application), then AT firmware might be suitable. It is acceptable to use a mixture of devices running API and AT firmware in a network.

Modes of Operation

Idle Mode

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

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

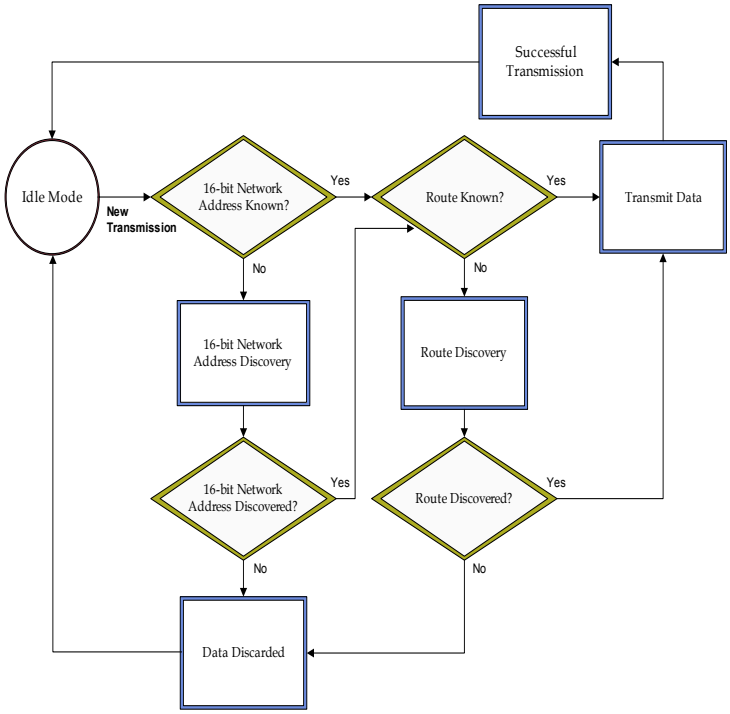
Transmit Mode

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

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

If the destination 16-bit network address is not known, network address discovery will take place. If a route is not known, route discovery will take place for the purpose of establishing a route to the destination node. If a module with a matching network address is not discovered, the packet is discarded. The data will be transmitted once a route is established. If route discovery fails to establish a route, the packet will be discarded.

Figure 2-04. Transmit Mode Sequence



When data is transmitted from one node to another, a network-level acknowledgement is transmitted back across the established route to the source node. This acknowledgement packet indicates to the source node that the data packet was received by the destination node. If a network acknowledgement is not received, the source node will re-transmit the data.

It is possible in rare circumstances for the destination to receive a data packet, but for the source to not receive the network acknowledgment. In this case, the source will retransmit the data, which could cause the destination to receive the same data packet multiple times. The XBee modules do not filter out duplicate packets. The application should include provisions to address this potential issue

See Data Transmission and Routing in chapter 4 for more information.

Receive Mode

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

Command Mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming serial characters are interpreted as commands. Refer to the API Mode section in Chapter 9 for an alternate means of configuring modules.

AT Command Mode

To Enter AT Command Mode:
Send the 3-character command sequence "+++" and observe guard times before and after the command characters. [Refer to the "Default AT Command Mode Sequence" below.]

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

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

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

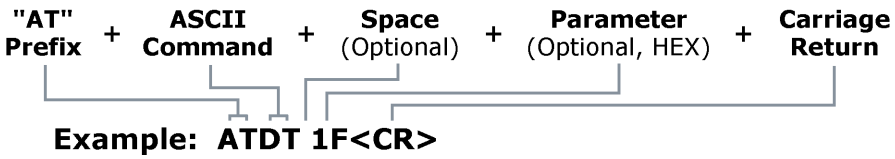
When command mode has been entered, the command mode timer is started (CT command), and the module is able to receive AT commands on the DIN pin.

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

NOTE: Failure to enter AT Command Mode is most commonly due to baud rate mismatch. By default, the BD (Baud Rate) parameter = 3 (9600 bps).

To Send AT Commands:
Send AT commands and parameters using the syntax shown below.

Figure 2-05. Syntax for sending AT Commands



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

The preceding example would change the RF module Destination Address (Low) to "0x1F". To store the new value to non-volatile (long term) memory, subsequently send the WR (Write) command.

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

Command Response

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

Applying Command Changes

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

- The AC (Apply Changes) command is issued.
- AT command mode is exited.

To Exit AT Command Mode:

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

   [OR]

2. If no valid AT Commands are received within the time specified by CT (Command Mode Timeout) Command, the RF module automatically returns to Idle Mode.

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, refer to the "Examples" and "XBee Command Reference Tables" chapters.

Sleep Mode

Sleep modes allow the RF module to enter states of low power consumption when not in use. The XBee RF modules support both pin sleep (sleep mode entered on pin transition) and cyclic sleep (module sleeps for a fixed time). XBee sleep modes are discussed in detail in section 6.

3. XBee ZigBee Networks

Introduction to ZigBee

ZigBee is an open global standard built on the IEEE 802.15.4 Mac/Phy. ZigBee defines a network layer above the 802.15.4 layers to support advanced mesh routing capabilities. The ZigBee specification is developed by a growing consortium of companies that make up the ZigBee Alliance. The Alliance is made up of over 300 members, including semiconductor, module, stack, and software developers.

ZigBee Stack Layers

The ZigBee stack consists of several layers including the PHY, MAC, Network, Application Support Sublayer (APS), and ZigBee Device Objects (ZDO) layers. Technically, an Application Framework (AF) layer also exists, but will be grouped with the APS layer in remaining discussions. The ZigBee layers are shown in the figure below.

A description of each layer appears in the following table:

ZigBee Layer	Description
PHY	Defines the physical operation of the ZigBee device including receive sensitivity, channel rejection, output power, number of channels, chip modulation, and transmission rate specifications. Most ZigBee applications operate on the 2.4 GHz ISM band at a 250kbps data rate. See the IEEE 802.15.4 specification for details.
MAC	Manages RF data transactions between neighboring devices (point to point). The MAC includes services such as transmission retry and acknowledgment management, and collision avoidance techniques (CSMA-CA).
Network	Adds routing capabilities that allows RF data packets to traverse multiple devices (multiple "hops") to route data from source to destination (peer to peer).
APS (AF)	Application layer that defines various addressing objects including profiles, clusters, and endpoints.
ZDO	Application layer that provides device and service discovery features and advanced network management capabilities.

Networking Concepts

Device Types

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

Node Types / Sample of a Basic ZigBee Network Topology

A **coordinator** has the following characteristics: it

- Selects a channel and PAN ID (both 64-bit and 16-bit) to start the network
- Can allow routers and end devices to join the network
- Can assist in routing data
- Cannot sleep--should be mains powered.

A **router** has the following characteristics: it

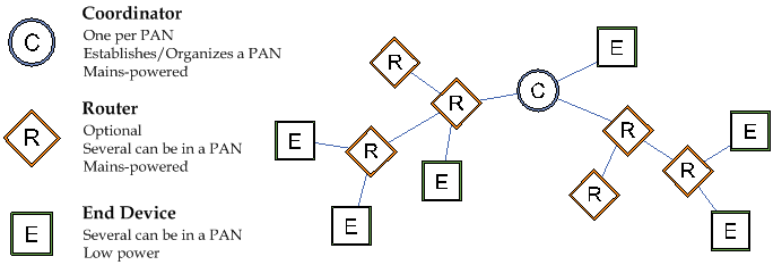
- Must join a ZigBee PAN before it can transmit, receive, or route data

- After joining, can allow routers and end devices to join the network
- After joining, can assist in routing data
- Cannot sleep--should be mains powered.

A **end device** has the following characteristics: it

- Must join a ZigBee PAN before it can transmit or receive data
- Cannot allow devices to join the network
- Must always transmit and receive RF data through its parent. Cannot route data.
- Can enter low power modes to conserve power and can be battery-powered.

An example of such a network is shown below:



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

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

PAN ID

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

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

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

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

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

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

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

Operating Channel

ZigBee utilizes direct-sequence spread spectrum modulation and operates on a fixed channel. The 802.15.4 PHY defines 16 operating channels in the 2.4 GHz frequency band. XBee modules support all 16 channels and XBee-PRO modules support 14 of the 16 channel

ZigBee Application Layers: In Depth

This section provides a more in-depth look at the ZigBee application stack layers (APS, ZDO) including a discussion on ZigBee endpoints, clusters, and profiles. Much of the material in this section can introduce unnecessary details of the ZigBee stack that are not required in many cases.

Skip this section if

- The XBee does not need to interoperate or talk to non-Digi ZigBee devices
- The XBee simply needs to send data between devices.

Read this section if

- The XBee may talk to non-Digi ZigBee devices
- The XBee requires network management and discovery capabilities of the ZDO layer
- The XBee is designed to operate in a public application profile (smart energy, home automation, etc.)

Application Support Sublayer (APS)

The APS layer in ZigBee adds support for endpoints, cluster IDs and application profiles. A brief discussion of each follows.

Application Profiles

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

Examples of public profiles include:

- Home Automation
- Smart Energy
- Commercial Building Automation

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

Digi XBee ZB firmware operates on a private profile called the Digi Drop-In Networking profile. However, the API firmware in the module can be used in many cases to talk to devices in public profiles or non-Digi private profiles. See the API chapter for details.

Clusters

A cluster is an application message type defined within a profile. Clusters are used to specify a unique function, service, or action. For example, the following are some clusters defined in the home automation profile:

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

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

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

XBee modules implement various clusters in the Digi private profile. In addition, the API can be used to send or receive messages on any cluster ID (and profile ID or endpoint). See the Explicit Addressing ZigBee Command API frame in chapter 3 for details.

Endpoints

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

ZigBee Device Profile

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

ZigBee Device Objects (ZDO)

The ZDO (endpoint 0) supports the discovery and management capabilities of the ZigBee Device Profile. A complete listing of all ZDP services is included in the ZigBee specification. Each service has an associated cluster ID.

The XBee ZB firmware allows applications to easily send ZDO messages to devices in the network using the API. See the "Management Transmissions" section in chapter 4 for details.

Coordinator Operation

Forming a Network

The coordinator is responsible for selecting the channel, PAN ID (16-bit and 64-bit), security policy, and stack profile for a network. Since a coordinator is the only device type that can start a network, each ZigBee network must have one coordinator. After the coordinator has started a network, it can allow new devices to join the network. It can also route data packets and communicate with other devices on the network.

To ensure the coordinator starts on a good channel and unused PAN ID, the coordinator performs a series of scans to discover any RF activity on different channels (energy scan) and to discover

any nearby operating PANs (PAN scan). The process for selecting the channel and PAN ID are described in the following sections.

Channel Selection

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

PAN ID Selection

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

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

Security Policy

The security policy determines which devices are allowed to join the network, and which device(s) can authenticate joining devices. See Chapter 5 for a detailed discussion of various security policies.

Persistent Data

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

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

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

XBee ZB Coordinator Startup

The following commands control the coordinator network formation process.

Network formation commands used by the coordinator to form a network.

Command	Description
ID	Used to determine the 64-bit PAN ID. If set to 0 (default), a random 64-bit PAN ID will be selected.
SC	Determines the scan channels bitmask (up to 16 channels) used by the coordinator when forming a network. The coordinator will perform an energy scan on all enabled SC channels. It will then perform a PAN ID scan and then form the network on one of the SC channels.
SD	Set the scan duration period. This value determines how long the coordinator performs an energy scan or PAN ID scan on a given channel.
ZS	Set the ZigBee stack profile for the network.
EE	Enable or disable security in the network.

NK	Set the network security key for the network. If set to 0 (default), a random network security key will be used.
KY	Set the trust center link key for the network. If set to 0 (default), a random link key will be used.
EO	Set the security policy for the network.

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

When the coordinator has successfully started a network, it

- Allows other devices to join the network for a time. (see NJ command.)
- Sets AI=0
- Starts blinking the Associate LED
- Sends an API modem status frame ("coordinator started") out the UART (API firmware only).

These behaviors are configurable using the following commands:

Command	Description
NJ	Sets the permit-join time on the coordinator, measured in seconds.
D5	Enables the Associate LED functionality.
LT	Sets the Associate LED blink time when joined. Default is 1 blink per second.

If any of the command values innthe network formation commands table changes, the coordinator will leave its current network and start a new network, possibly on a different channel. Note that command changes must be applied (AC or C

N command) before taking effect.

Permit Joining

The permit joining attribute on the coordinator is configurable with the NJ command. NJ can be configured to always allow joining, or to allow joining for a short time.

Joining Always Enabled

If NJ=0xFF (default), joining is permanently enabled. This mode should be used carefully. Once a network has been deployed, the application should strongly consider disabling joining to prevent unwanted joins from occurring.

Joining Temporarily Enabled

If NJ < 0xFF, joining will be enabled only for a number of seconds, based on the NJ parameter. The timer is started once the XBee joins a network. Joining will not be re-enabled if the module is power cycled or reset. The following mechanisms can restart the permit-joining timer:

- Changing NJ to a different value (and applying changes with the AC or CN commands)
- Pressing the commissioning button twice (enables joining for 1 minute)
- Issuing the CB command with a parameter of 2 (software emulation of a 2 button press - enables joining for 1 minute).

Resetting the Coordinator

When the coordinator is reset or power cycled, it checks its PAN ID, operating channel and stack profile against the network configuration settings (ID, SC, ZS). It also verifies the saved security policy against the security configuration commands (EE, NK, KY). If the coordinator's PAN ID, operating channel, stack profile, or security policy is not valid based on its network formation commands (table xyz), then the coordinator will leave the network and attempt to form a new network based on its network formation command values.

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

Leaving a Network

There are a couple of mechanisms that will cause the coordinator to leave its current PAN and start a new network based on its network formation parameter values. These include the following:

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

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

Replacing a Coordinator (Security Disabled Only)

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

NOTE: Having two coordinators on the same channel, stack profile, and PAN ID (16-bit and 64-bit) can cause problems in the network and should be avoided. When replacing a coordinator, the old coordinator should be turned off before starting the new coordinator.

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

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

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

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

ZS	Set the stack profile to match the read ZS value.

Note: II is the initial 16-bit PAN ID. Under certain conditions, the ZigBee stack can change the 16-bit PAN ID of the network. For this reason, the II command cannot be saved using the WR command. Once II is set, the coordinator leaves the network and starts on the 16-bit PAN ID specified by II.

Example: Starting a Coordinator

1. Set SC and ID to the desired scan channels and PAN ID values. (The defaults should suffice.)

2. If SC or ID is changed from the default, issue the WR command to save the changes.

3. If SC or ID is changed from the default, apply changes (make SC and ID changes take effect) either by sending the AC command or by exiting AT command mode.

4. The Associate LED will start blinking once the coordinator has selected a channel and PAN ID.

5. The API Modem Status frame ("Coordinator Started") is sent out the UART (API firmware only).

6. Reading the AI command (association status) will return a value of 0, indicating a successful startup.

7. Reading the MY command (16-bit address) will return a value of 0, the ZigBee-defined 16-bit address of the coordinator

After startup, the coordinator will allow joining based on its NJ value.

Example: Replacing a Coordinator (security disabled)

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

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

3. Turn off the running coordinator.

4. Set the II parameter on the new coordinator to match the read OI value on the old coordinator.

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

Router Operation

Joining a Network

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

Discovering ZigBee Networks

To discover nearby ZigBee networks, the router performs a PAN (or active) scan, just like the coordinator does when it starts a network. During the PAN scan, the router sends a beacon request (broadcast) transmission on the first channel in its scan channels list. All nearby coordinators and routers operating on that channel (that are already part of a ZigBee network) respond to the beacon request by sending a beacon back to the router. The beacon contains information about the PAN the nearby device is on, including the PAN identifier (PAN ID), and whether or not joining is allowed. The router evaluates each beacon received on the channel to determine if a valid PAN is found. A router considers a PAN to be valid if the PAN:

- Has a valid 64-bit PAN ID (PAN ID matches ID if ID > 0)
- Has the correct stack profile (ZS command)
- Is allowing joining.

If a valid PAN is not found, the router performs the PAN scan on the next channel in its scan channels list and continues scanning until a valid network is found, or until all channels have been scanned. If all channels have been scanned and a valid PAN was not discovered, all channels will be scanned again.

The ZigBee Alliance requires certified solutions not send beacon request messages to frequently. To meet certification requirements, the XBee firmware attempts 9 scans per minute for the first 5 minutes, and 3 scans per minute thereafter. If a valid PAN is within range of a joining router, it should typically be discovered within a few seconds.

Joining a Network

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

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

Authentication

In a network where security is enabled, the router must then go through an authentication process. See the Security chapter for a discussion on security and authentication.

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

Persistent Data

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

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

The router will retain this information indefinitely until it leaves the network. When the router leaves a network, the previous PAN ID, operating channel, and child table data are lost.

XBee ZB Router Joining

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

Note: The DJ command can be set to 1 to disable joining. The DJ parameter cannot be written with WR, so a power cycle always clears the DJ setting.

The following commands control the router joining process.

Command	Description
ID	Sets the 64-bit PAN ID to join. Setting ID=0 allows the router to join any 64-bit PAN ID.
SC	Set the scan channels bitmask that determines which channels a router will scan to find a valid network. SC on the router should be set to match SC on the coordinator. For example, setting SC to 0x281 enables scanning on channels 0x0B, 0x12, and 0x14, in that order.
SD	Set the scan duration, or time that the router will listen for beacons on each channel.
ZS	Set the stack profile on the device.
EE	Enable or disable security in the network. This must be set to match the EE value (security policy) of the coordinator.
KY	Set the trust center link key. If set to 0 (default), the link key is expected to be obtained (unencrypted) during joining.

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

If any of the above command values change, when command register changes are applied (AC or CN commands), the router will leave its current network and attempt to discover and join a new valid network.

When a ZB router has successfully joined a network, it:

- Allows other devices to join the network for a time
- Sets AI=0
- Starts blinking the Associate LED
- Sends an API modem status frame ("associated") out the UART (API firmware only).

These behaviors are configurable using the following commands:

Command	Description
NJ	Sets the permit-join time on the router, or the time that it will allow new devices to join the network, measured in seconds. If NJ=0xFF, permit joining will always be enabled.
D5	Enables the Associate LED functionality.
LT	Sets the Associate LED blink time when joined. Default is 2 blinks per second (router).

Permit Joining

The permit joining attribute on the router is configurable with the NJ command. NJ can be configured to always allow joining, or to allow joining for a short time.

Joining Always Enabled

If NJ=0xFF (default), joining is permanently enabled. This mode should be used carefully. Once a network has been deployed, the application should strongly consider disabling joining to prevent unwanted joins from occurring.

Joining Temporarily Enabled

If NJ < 0xFF, joining will be enabled only for a number of seconds, based on the NJ parameter. The timer is started once the XBee joins a network. Joining will not be re-enabled if the module is power cycled or reset. The following mechanisms can restart the permit-joining timer:

- Changing NJ to a different value (and applying changes with the AC or CN commands)
- Pressing the commissioning button twice (enables joining for 1 minute)
- Issuing the CB command with a parameter of 2 (software emulation of a 2 button press - enables joining for 1 minute).
- Causing the router to leave and rejoin the network.

Resetting the Router

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

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

Leaving a Network

There are a couple of mechanisms that will cause the router to leave its current PAN and attempt to discover and join a new network based on its network joining parameter values (see table xyz). These include the following:

- Change the ID command such that the current 64-bit PAN ID is invalid.
- Change the SC command such that the current channel (CH) is not included in the channel mask.

- Change the ZS or any of the security command values.
- Issue the NR0 command to cause the router to leave.
- Issue the NR1 command to send a broadcast transmission, causing all devices in the network to leave and migrate to a different channel.
- Press the commissioning button 4 times or issue the CB command with a parameter of 4.

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

Example: Joining a Network

After starting a coordinator (that is allowing joins), the following steps will cause an XBee router to join the network:

1. Set ID to the desired 64-bit PAN ID, or to 0 to join any PAN.
2. Set SC to the list of channels to scan to find a valid network.
3. If SC or ID is changed from the default, apply changes (make SC and ID changes take effect) by issuing the AC or CN command.
4. The Associate LED will start blinking once the router has joined a PAN.
5. If the Associate LED is not blinking, the AI command can be read to determine the cause of join failure.
6. Once the router has joined, the OP and CH commands will indicate the operating 64-bit PAN ID and channel the router joined.
7. The MY command will reflect the 16-bit address the router received when it joined.
8. The API Modem Status frame ("Associated") is sent out the UART (API firmware only).
9. The joined router will allow other devices to join for a time based on its NJ setting.

End Device Operation

Joining a Network

Similar to routers, end devices must also discover and join a valid ZigBee network before they can participate in a network. After an end device has joined a network, it can communicate with other devices on the network. Since end devices are intended to be battery powered and therefore support low power (sleep) modes, end devices cannot allow other devices to join, nor can they route data packets.

Discovering ZigBee Networks

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

- Has a valid 64-bit PAN ID (PAN ID matches ID if ID > 0)
- Has the correct stack profile (ZS command)
- Is allowing joining
- Has capacity for additional end devices (see End Device Capacity section below).

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

If scanning all SC channels fails to discover a valid PAN, XBee ZB modules will attempt to enter a low power state and will retry scanning all SC channels after the module wakes from sleeping. If the module cannot enter a low power state, it will retry scanning all channels, similar to the router.

To meet ZigBee Alliance requirements, the end device will attempt up to 9 scans per minute for the first 5 minutes, and 3 scans per minute thereafter.

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

Joining a Network

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

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

Parent Child Relationship

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

End Device Capacity

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

In ZB firmware, the NC command (number of remaining end device children) can be used to determine how many additional end devices can join a router or coordinator. If NC returns 0, then the router or coordinator device has no more end device capacity. (Its child table is full.)

Also of note, since routers cannot sleep, there is no equivalent need for routers or coordinators to track joined routers. Therefore, there is no limit to the number of routers that can join a given router or coordinator device. (There is no "router capacity" metric.)

Authentication

In a network where security is enabled, the end device must then go through an authentication process. See chapter 5 for a discussion on security and authentication.

Persistent Data

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

Orphan Scans

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

If the orphaned end device receives a coordinator realignment command, it is considered joined to the network. Otherwise, it will attempt to discover and join a valid network.

XBee: ZB End Device Joining

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

Note: The DJ command can be set to 1 to disable joining. The DJ parameter cannot be written with WR, so a power cycle always clears the DJ setting.

Similar to a router, the following commands control the end device joining process.

Network joining commands used by an end device to join a network.

Command	Description
ID	Sets the 64-bit PAN ID to join. Setting ID=0 allows the router to join any 64-bit PAN ID.
SC	Set the scan channels bitmask that determines which channels an end device will scan to find a valid network. SC on the end device should be set to match SC on the coordinator and routers in the desired network. For example, setting SC to 0x281 enables scanning on channels 0x0B, 0x12, and 0x14, in that order.
SD	Set the scan duration, or time that the end device will listen for beacons on each channel.
ZS	Set the stack profile on the device.
EE	Enable or disable security in the network. This must be set to match the EE value (security policy) of the coordinator.
KY	Set the trust center link key. If set to 0 (default), the link key is expected to be obtained (unencrypted) during joining.

Once the end device joins a network, the network configuration settings can persist through power cycles as mentioned in the "Persistent Data" section previously. If joining fails, the status of the last join attempt can be read in the AI command register.

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

When a ZB end device has successfully started a network, it

- Sets AI=0.
- Starts blinking the Associate LED
- Sends an API modem status frame ("associated") out the UART (API firmware only)
- Attempts to enter low power modes.

These behaviors are configurable using the following commands:

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

Parent Connectivity

The XBee ZB end device sends regular poll transmissions to its parent when it is awake. These poll transmissions query the parent for any new received data packets. The parent always sends a MAC layer acknowledgment back to the end device. The acknowledgment indicates if the parent has data for the end device or not.

If the end device does not receive an acknowledgment for 3 consecutive poll requests, it considers itself disconnected from its parent and will attempt to discover and join a valid ZigBee network. See "Managing End Devices" chapter for details.

Resetting the End Device

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

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

Leaving a Network

There are a couple of mechanisms that will cause the router to leave its current PAN and attempt to discover and join a new network based on its network joining parameter values. These include the following:

- The ID command changes such that the current 64-bit PAN ID is invalid.
- The SC command changes such that the current operating channel (CH) is not included in the channel mask.
- The ZS or any of the security command values change.
- The NR0 command is issued to cause the end device to leave.
- The NR1 command is issued to send a broadcast transmission, causing all devices in the network to leave and migrate to a different channel.
- The commissioning button is pressed 4 times or the CB command is issued with a parameter of 4.
- The end device's parent is powered down or the end device is moved out of range of the parent such that the end device fails to receive poll acknowledgment messages.

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

Example: Joining a Network

After starting a coordinator (that is allowing joins), the following steps will cause an XBee end device to join the network

1. Set ID to the desired 64-bit PAN ID, or to 0 to join any PAN.

2. Set SC to the list of channels to scan to find a valid network.

3. If SC or ID is changed from the default, apply changes (make SC and ID changes take effect) by issuing the AC or CN command.

4. The Associate LED will start blinking once the end device has joined a PAN.

5. If the Associate LED is not blinking, the AI command can be read to determine the cause of join failure.

6. Once the end device has joined, the OP and CH commands will indicate the operating 64-bit PAN ID and channel the end device joined.

7. The MY command will reflect the 16-bit address the router received when it joined.

8. The API Modem Status frame ("Associated") is sent out the UART (API firmware only).

9. The joined end device will attempt to enter low power sleep modes based on its sleep configuration commands (SM, SP, SN, ST, SO).

Channel Scanning

As mentioned previously, routers and end devices must scan one or more channels to discover a valid network to join. When a join attempt begins, the XBee sends a beacon request transmission on the lowest channel specified in the SC (scan channels) command bitmask. If a valid PAN is found on the channel, the XBee will attempt to join the PAN on that channel. Otherwise, if a valid PAN is not found on the channel, it will attempt scanning on the next higher channel in the SC command bitmask. The XBee will continue to scan each channel (from lowest to highest) in the SC bitmask until a valid PAN is found or all channels have been scanned. Once all channels have been scanned, the next join attempt will start scanning on the lowest channel specified in the SC command bitmask.

For example, if the SC command is set to 0x400F, the XBee would start scanning on channel 11 (0x0B) and scan until a valid beacon is found, or until channels 11, 12, 13, 14, and 25 have been scanned (in that order).

Once an XBee router or end device joins a network on a given channel, if the XBee is told to leave (see "Leaving a Network" section), it will leave the channel it joined on and continue scanning on the next higher channel in the SC bitmask.

For example, if the SC command is set to 0x400F, and the XBee joins a PAN on channel 12 (0x0C), if the XBee leaves the channel, it will start scanning on channel 13, followed by channels 14 and 25 if a valid network is not found. Once all channels have been scanned, the next join attempt will start scanning on the lowest channel specified in the SC command bitmask.

Managing Multiple ZigBee Networks

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

- PAN ID Filtering
- Preconfigured Security Keys
- Permit Joining
- Application Messaging

PAN ID Filtering

The XBee can be configured with a fixed PAN ID by setting the ID command to a non-zero value. If the PAN ID is set to a non-zero value, the XBee will only join a network with the same PAN ID.

Preconfigured Security Keys

Similar to PAN ID filtering, this method requires a known security key be installed on a router to ensure it will join a ZigBee network with the same security key. If the security key (KY command) is set to a non-zero value, and if security is enabled (EE command), an XBee router or end device will only join a network with the same security key.

Permit Joining

The Permit Joining parameter can be disabled in a network to prevent unwanted devices from joining. When a new device must be added to a network, permit-joining can be enabled for a short time on the desired network. In the XBee firmware, joining is disabled by setting the NJ command to a value less than 0xFF on all routers and coordinator devices. Joining can be enabled for a short time using the commissioning push-button (see Network Commissioning chapter for details) or the CB command.

Application Messaging

If the above mechanisms are not feasible, the application could build in a messaging framework between the coordinator and devices that join its network. For example, the application code in joining devices could send a transmission to the coordinator after joining a network, and wait to receive a defined reply message. If the application does not receive the expected response message after joining, the application could force the XBee to leave and continue scanning. Forcing the XBee to leave will cause the XBee to continue scanning on the next higher channel in the SC bitmask.

4. Data Transmission, Addressing, and Routing

Addressing

All ZigBee devices have two different addresses, a 64-bit and a 16-bit address. The characteristics of each are described below.

64-bit Device Addresses

The 64-bit address is a unique device address assigned during manufacturing. This address is unique to each physical device. The 64-bit address includes a 3-byte Organizationally Unique Identifier (OUI) assigned by the IEEE. The 64-bit address is also called the extended address.

16-bit Device Addresses

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

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

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

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

Application Layer Addressing

ZigBee devices can support multiple application profiles, cluster IDs, and endpoints. (See "ZigBee Application Layers - In Depth" in chapter 3.) Application layer addressing allows data transmissions to be addressed to specific profile IDs, cluster IDs, and endpoints. Application layer addressing is useful if an application must

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

The API firmware provides a simple yet powerful interface that can easily send data to any profile ID, endpoint, and cluster ID combination on any device in a ZigBee network.

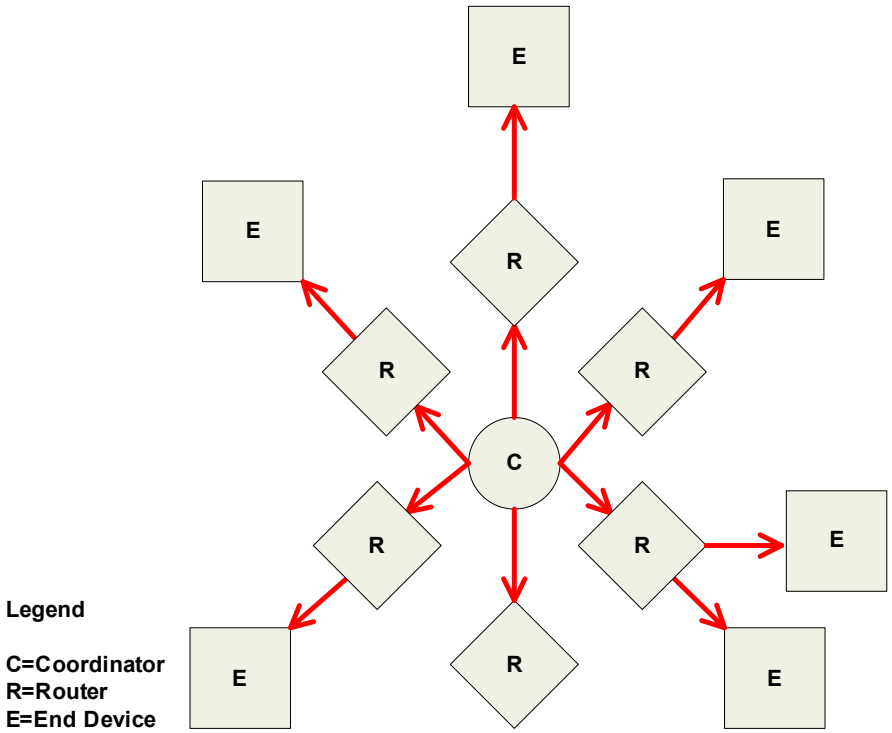
Data Transmission

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

Broadcast Transmissions

Broadcast transmissions within the ZigBee protocol are intended to be propagated throughout the entire network such that all nodes receive the transmission. To accomplish this, all devices that receive a broadcast transmission will retransmit the packet 3 times.

eBroadcast Data Transmission



Each node that transmits the broadcast will also create an entry in a local broadcast transmission table. This entry is used to keep track of each received broadcast packet to ensure the packets are not endlessly transmitted. Each entry persists for 8 seconds. The broadcast transmission table holds 8 entries.

For each broadcast transmission, the ZigBee stack must reserve buffer space for a copy of the data packet. This copy is used to retransmit the packet as needed.Large broadcast packets will require more buffer space.

Since broadcast transmissions are retransmitted by each device in the network, broadcast messages should be used sparingly.

Unicast Transmissions

Unicast transmissions are sent from one source device to another destination device. The destination device could be an immediate neighbor of the source, or it could be several hops away.

As mentioned previously, each device in a ZigBee network has both a 16-bit (network) address and a 64-bit (extended) address. Unicast transmissions are always addressed and routed to the 16-bit address of the destination. However, to ensure data is received by the correct device, the destination 64-bit address is often included in the RF transmission. If a receiving device has a matching 16-bit address, but not a matching 64-bit address, it will drop the packet and obtain a new 16-bit address.

XBee ZB firmware requires that data be sent to the 64-bit address of the destination device. However, since the actual RF transmission requires 16-bit addressing, the 16-bit address will be discovered by the XBee if unknown.

Address Table

Each ZigBee device maintains an address table that maps a 64-bit address to a 16-bit address. When a transmission is addressed to a 64-bit address, the ZigBee stack searches the address table for an entry with a matching 64-bit address, in hopes of determining the destination's 16-bit address. If a known 16-bit address is not found, the ZigBee stack will perform address discovery to discover the device's current 16-bit address.

Sample Address Table

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

Address Discovery

Address discovery is a service provided in the ZigBee Device Profile that can discover the 16-bit address of a given device, based on its 64-bit address. To discover a 16-bit address of a remote, the device initiating the discovery sends a broadcast address discovery transmission. The address discovery includes the 64-bit address of the remote device whose 16-bit address is being requested.

All nodes that receive this transmission check the 64-bit address in the payload and compare it to their own 64-bit address. If the addresses match, the device sends a response packet back to the initiator. This response includes the remote's 16-bit address.

When the discovery response is received, the initiator will then transmit the data.

Data Transmission Examples

AT Firmware

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

Example 1: Send a transmission to the coordinator.

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

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

1. Enter command mode ('+++')
2. After receiving an OK\r, issue the following commands:
   a. ATDH0\r
   b. ATDL0\r
   c. ATCN\r
3. Verify that each of the 3 commands returned an OK\r response.

4. After setting these command values, all serial characters will be sent as a unicast transmission to the coordinator.

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

1. Enter command mode ('+++')

2. After receiving an OK\r, issue the following commands:

   a. ATDH13A200\r

   b. ATDL404A2244\

   c. ATCN\r

3. Verify that each of the 3 commands returned an OK\r response.

4. After setting these command values, all serial characters will be sent as a unicast transmission to the coordinator.

API Firmware

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

To send an ascii "1" to the coordinator's 0x00 address, the following API frame can be used:

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

If the explicit transmit frame is used, the cluster ID should be set to 0x0011, the profile ID to 0xC105, and the source and destination endpoints to 0xE8 (recommended defaults for data transmissions in the Digi profile.) The same transmission could be sent using the following explicit transmit frame:

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

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

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

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

Example 2: Send a broadcast transmission.

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

Perform the following steps to configure a broadcast transmission:

1. Enter command mode ('+++')

2. After receiving an OK\r, issue the following commands:

   a. ATDH0\r

   b. ATDLffff\r

   c. ATCN\r

3. Verify that each of the 3 commands returned an OK\r response

4. After setting these command values, all serial characters will be sent as a broadcast transmission.

API Firmware

This example will use the transmit request API frame (0x10) to send an ASCII "1" in a broadcast transmission.

To send an ascii "1" as a broadcast transmission, the following API frame can be used:

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

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

RF Packet Routing

Unicast transmissions may require some type of routing. ZigBee includes several different ways to route data, each with its own advantages and disadvantages. These are summarized in the table below.

Routing Approach	Description	When to Use
Ad hoc On-demand Distance Vector (AODV) Mesh Routing	Routing paths are created between source and destination, possibly traversing multiple nodes ("hops"). Each device knows who to send data to next to eventually reach the destination	Use in networks where data does not need to be routed to many different destinations. The routing table is a finite size and each destination address requires one entry
Many-to-One Routing	A single broadcast transmission configured reverse routes on all devices into the device that sends the broadcast	Useful when many remote devices must send data to a single gateway or collector device.
Source Routing	Data packets include the entire route the packet should traverse to get from source to destination	Improves routing efficiency in large networks (over 40 remote devices)

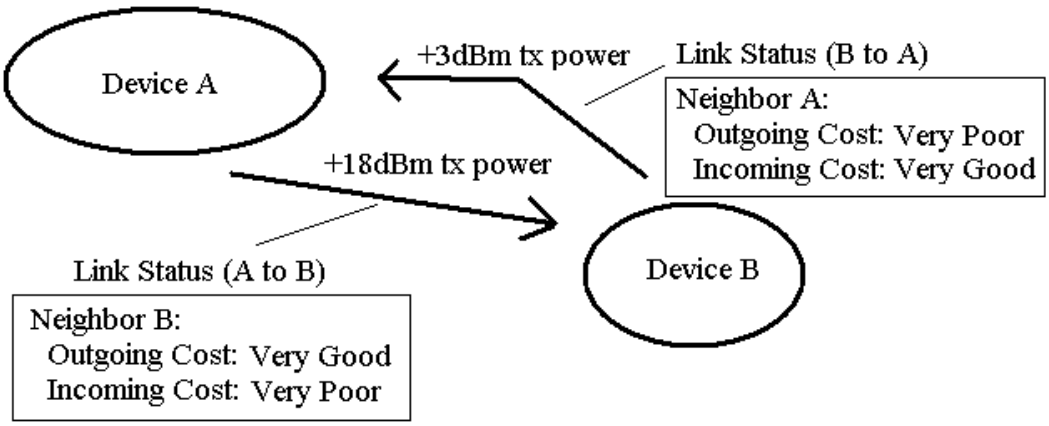
Note – End devices do not make use of these routing protocols. Rather, an end device sends a unicast transmission to its parent and allows the parent to route the data packet in its behalf.

Link Status Transmission

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

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

For example, consider a network of two neighboring devices that send periodic link status messages. Suppose that the output power of device A is +18dBm, and the output power of device B is +3dBm (considerably less than the output power of device A). The link status messages might indicate the following:

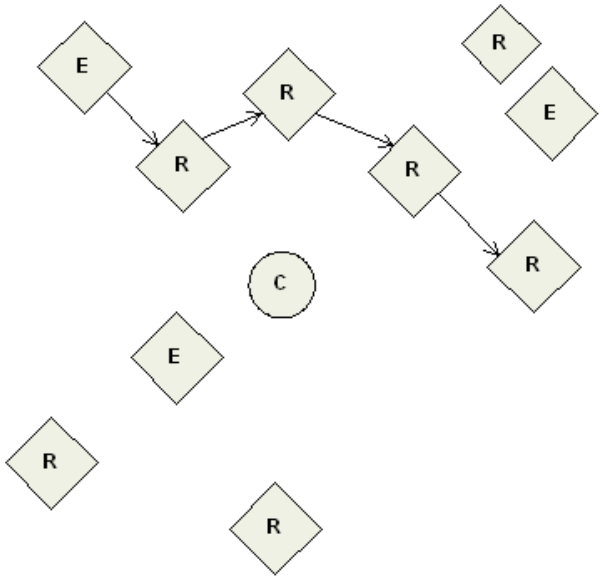


This mechanism enables devices A and B to recognize that the link is not reliable in both directions and select a different neighbor when establishing routes. (Such links are called asymmetric links, meaning the link quality is not similar in both directions.)

AODV Mesh Routing

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

Figure 4-06. Sample Transmission Through a Mesh Network



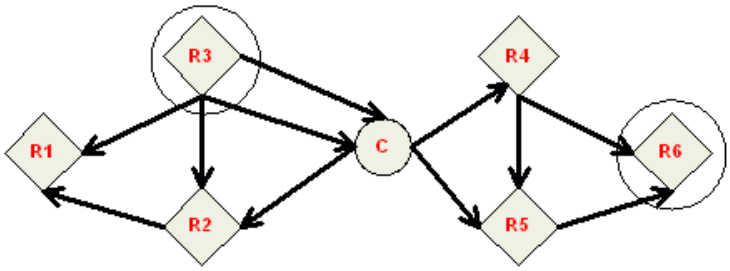
AODV (Ad-hoc On-demand Distance Vector) Routing Algorithm

Routing under the AODV protocol is accomplished using tables in each node that store the next hop (intermediary node between source and destination nodes) for a destination node. If a next hop is not known, route discovery must take place in order to find a path. Since only a limited number of routes can be stored on a Router, route discovery will take place more often on a large network with communication between many different nodes.

Node	Destination Address	Next Hop Address
R3	Router 6	Coordinator
C	Router 6	Router 5
R5	Router 6	Router 6

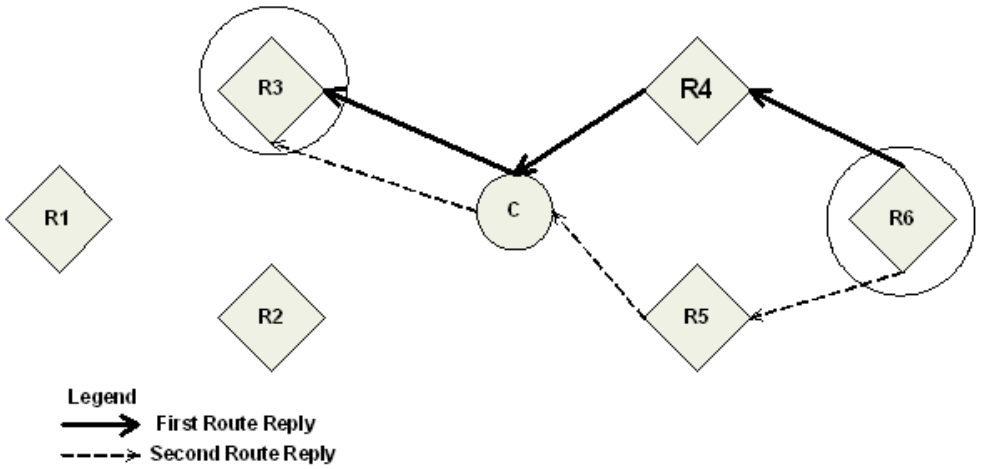
When a source node must discover a route to a destination node, it sends a broadcast route request command. The route request command contains the source network address, the destination network address and a path cost field (a metric for measuring route quality). As the route request command is propagated through the network (refer to the Broadcast Transmission), each node that re-broadcasts the message updates the path cost field and creates a temporary entry in its route discovery table.

Figure 4-07. Sample Route Request (Broadcast) Transmission Where R3 is Trying to Discover a Route to R6t



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

Figure 4-08. Route Reply Sample Route Reply (Unicast) Where R6 Sends a Route Reply to R3.



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

Retries and Acknowledgments

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

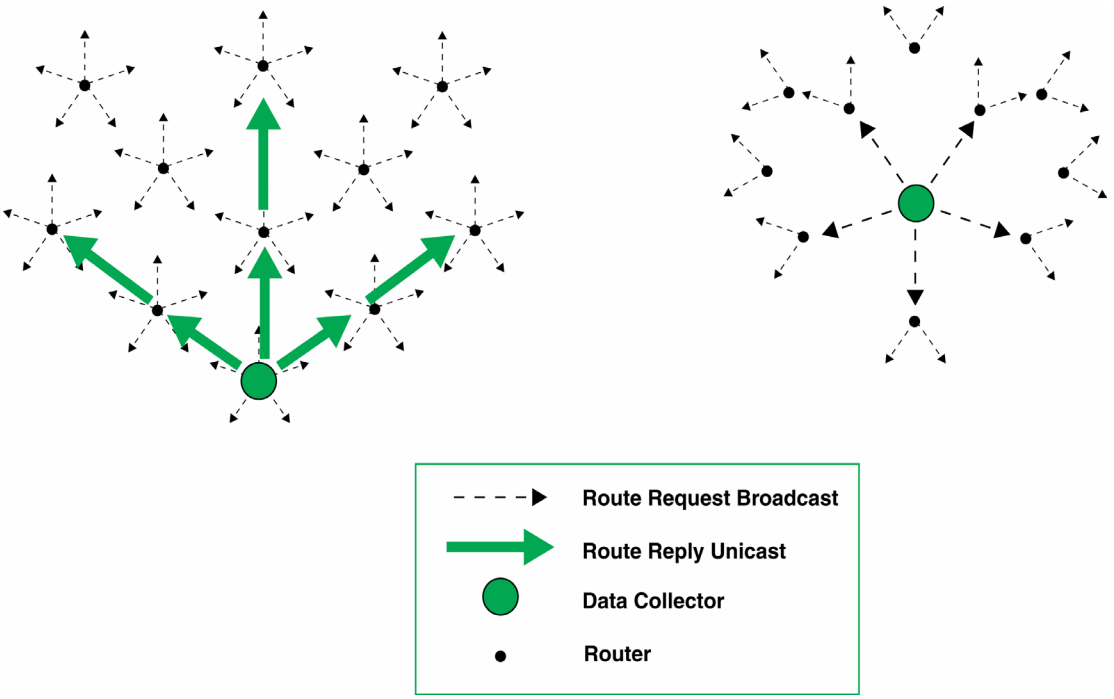
In addition, the device that originated the transmission expects to receive an acknowledgment packet (Ack) from the destination device. This Ack will traverse the same path that the data traversed, but in the opposite direction. If the originator fails to receive this Ack, it will retransmit the data, up to 2 times until an Ack is received. This Ack is called the ZigBee APS layer acknowledgment.

Refer to the ZigBee specification for more details.

Many-to-One Routing

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

Many-to-one routing is an optimization for these kinds of networks. Rather than require each device to do its own route discovery, a single many-to-one broadcast transmission is sent from the data collector to establish reverse routes on all devices. This is shown in the figure below. The left side shows the many broadcasts the devices can send when they create their own routes. The right side shows the benefits of many-to-one routing where a single broadcast creates reverse routes on all routers.



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

When a device sends data to a data collector, it finds a many-to-one routing table entry and transmits the data - no route discovery is required on the remotes. The many-to-one route request should be sent periodically to update and refresh the reverse routes in the network.

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

In ZB firmware, the AR command is used to enable many-to-one broadcasting on a device. The AR command sets a time interval (measured in 10 second units) for sending the many to one broadcast transmission. (See the command table for details.)

Source Routing

In applications where a device must transmit data to many remotes, AODV routing would require performing one route discovery for each destination device to establish a route. Also, if there are more destination devices than there are routing table entries, established AODV routes would be overwritten with new routes, causing route discoveries to occur more regularly. This could result in larger packet delays and poor network performance.

ZigBee source routing helps solve these problems. In contrast to many-to-one routing that establishes routing paths from many devices to one data collector, source routing allows the collector to store and specify routes for many remotes.

Acquiring Source Routes

Source routing should be used in conjunction with many-to-one routing to obtain optimal routes. (See "Many-to-One Routing" section.)

When a remote device sends a transmission using a many-to-one route, it first sends a ZigBee Route Record command frame. The Route Record is sent along the entire many-to-one route until it reaches the data collector. As the Route Record traverses multiple hops, it appends the 16-bit address of each hop into the payload. When the Route Record reaches the data collector, it

contains the destination address, and all devices that packet was routed through to reach the collector. The application can save these routes into a source routing table and use the route later to send data to the remote.

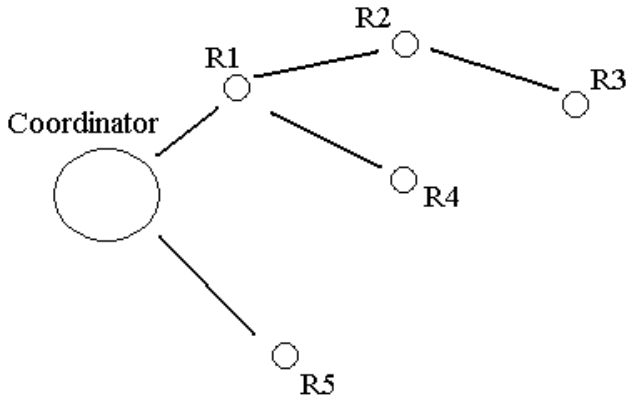
By maintaining source routes to all devices in a network, a device can eliminate the need to perform broadcast route discoveries to establish a route with each device in the network. This feature greatly expands the scalability of ZigBee networks.

To implement source routing in ZB firmware, the AR command must be set less than 0xFF on a data collector device (the device implementing source routing) to send periodic many-to-one route request transmissions. Upon receipt of the many-to-one route request transmissions, remote devices will transmit a route record command before each unicast transmission to the data collector. Upon receipt of a route record command, the Route Record Indicator API frame (0xA1) is sent out the UART of the data collector, indicating a new source route was received.

Note: API firmware should be used to support source routing.

To send a source routed transmission, the application should send a Create Source Route API frame (0x21) to the XBee to create a source route in its internal source route table. After sending the Create Source Route API frame, the application can send data transmission or remote command request frames as needed to the same destination, or any destination in the source route. Once data must be sent to a new destination (a destination not included in the last source route), the application should first send a new Create Source Route API frame. The XBee can buffer one source route that includes up to 10 hops (excluding source and destination).

For example, suppose a network exists with a coordinator and 5 routers (R1, R2, R3, R4, R5) with known source routes as shown below.



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

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

Then the Create Source Route API frame would be:

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

Where:

0x0012 - length

0x21 - API ID (create source route)

0x00 - frame ID (set to 0 always)

0x0013A200 404A1234 - 64-bit address of R3 (destination)

0xEEFF - 16-bit address of R3 (destination)

0x00 - Route options (set to 0)

0x02 - Number of intermediate devices in the source route

0xCCDD - Address of furthest device (1-hop from target)

0xAABB - Address of next-closer device

0x5C - Checksum (0xFF - SUM(all bytes after length))

After sending the Create Source Route frame, data can be sent to R1, R2, or R3 and the same source route will be used. However, if data must be sent to R4 or R5, a new Create Source Route frame should be sent to the XBee prior to the transmissions.

Retries and Acknowledgments

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

In addition, the device that originated the transmission expects to receive an acknowledgment packet (Ack) from the destination device. This Ack will traverse the same path that the data traversed, but in the opposite direction. If the originator fails to receive this Ack, it will retransmit the data, up to 2 times until an Ack is received. This Ack is called the ZigBee APS layer acknowledgment.

Refer to the ZigBee specification for more details.

Encrypted Transmissions

Encrypted transmissions are routed similar to non-encrypted transmissions with one exception. As an encrypted packet propagates from one device to another, each device decrypts the packet using the network key, and authenticates the packet by verifying packet integrity. It then re-encrypts the packet with its own source address and frame counter values, and sends the message to the next hop. This process adds some overhead latency to unicast transmissions, but it helps prevent replay attacks. See chapter 5 for details.

Improving Routing Efficiency with the API

If a network has one or more devices that need to transmit to more than 10 different devices, these applications can benefit by implementing an address table or source routing table and interacting with the API firmware. The benefits of these features are described below.

Address Table

Devices that transmit data to more that 10 remotes can see significant routing improvements by supporting an address table that maps a 64-bit address to a 16-bit address, and by using the API to send data. An example address table was shown previously in table xyz. Maintaining an address table significantly reduces the number of 16-bit address discoveries that must take place to route data to the appropriate 16-bit address.

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

When sending a transmission to a destination 64-bit address, the application should search the address table for a matching 64-bit address. If a match is found, the 16-bit address should be populated into the 16-bit address field of the API frame. If a match is not found, the 16-bit address should be set to 0xFFFE (unknown) in the API transmit frame.

The API provides indication of a remote device's 16-bit address in the following frames:

- All receive data frames
  Rx Data (0x90)
  Rx Explicit Data (0x91)
  IO Sample Data (0x92)
  Node Identification Indicator (0x95)
  Etc.
- Transmit status frame (0x8B)

The application should always update the 16-bit address in the address table when one of these frames is received to ensure the table has the most recently known 16-bit address. If a transmission failure occurs, the application should set the 16-bit address in the table to 0xFFFE (unknown).

Maximum RF Payload Size

XBee ZB firmware includes a command (ATNP) that returns the maximum number of RF payload bytes that can be sent in a unicast transmission. Querying the NP command, like most other commands, returns a HEXADECIMAL value. This number will change based on whether security is enabled or not. If security is enabled (EE command), the maximum number of RF payload bytes decreases since security requires added overhead.

Broadcast transmissions can support 8 bytes more than unicast transmissions.

If source routing is used, the addresses in the source route must fit into the RF payload space. For example, if NP returns 84 bytes, and a source route must traverse 3 intermediate hops (3 16-bit addresses), the total number of bytes that can be sent in one RF packet is 78.

Management Transmissions

ZigBee defines a ZigBee Device Objects layer (ZDO) that can provide device and service discovery and network management capabilities. This layer is described below.

ZigBee Device Objects (ZDO)

The ZigBee Device Objects (ZDO) is supported to some extent on all ZigBee devices. The ZDO is an endpoint that implements services described in the ZigBee Device Profile in the ZigBee specification. Each service has an assigned cluster ID, and most service requests have an associated response. The following table describes some common ZDO services.

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

Refer to the ZigBee specification for a detailed description of all ZigBee Device Profile services.

Sending a ZDO Command

To send a ZDO command, an explicit transmit API frame must be used and formatted correctly. The source and destination endpoints must be set to 0, and the profile ID must be set to 0. The cluster ID must be set to match the cluster ID of the appropriate service. For example, to send an active endpoints request, the cluster ID must be set to 0x0005.

The first byte of payload in the API frame is an application sequence number (transaction sequence number) that can be set to any single byte value. This same value will be used in the first byte of the ZDO response. All remaining payload bytes must be set as required by the ZDO. All multi-byte values must be sent in little endian byte order.

Receiving ZDO Commands and Responses

In XBee ZB firmware, ZDO commands can easily be sent using the API. In order to receive incoming ZDO commands, receiver application addressing must be enabled with the AO command. (See examples later in this section.) Not all incoming ZDO commands are passed up to the application.

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

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

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

API Frame:

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

0x0016 - length

0x11 - Explicit transmit request

0x01 - frame ID (set to a non-zero value to enable the transmit status message, or set to 0 to disable)

0x0013A200 40401234 - 64-bit address of the remote

0xFFFE - 16-bit address of the remote (0xFFFE = unknown). Optionally, set to the 16-bit address of the destination if known.

0x00 - Source endpoint

0x00 - Destination endpoint

0x0031 - Cluster ID (LQI Request, or Neighbor table request)

0x0000 - Profile ID (ZigBee Device Profile)

0x00 - Broadcast radius

0x00 - Tx Options

0x76 - Transaction sequence number

0x00 - Required payload for LQI request command

0xCE - Checksum (0xFF - SUM(all bytes after length))

Description:

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

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

Looking at the ZigBee specification, the cluster ID for an Network Address Request is 0x0000, and the payload only requires the following:

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

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

API Frame:

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

0x001F - length

0x11 - Explicit transmit request

0x01 - frame ID (set to a non-zero value to enable the transmit status message, or set to 0 to disable)

0x00000000 0000FFFF - 64-bit address for a broadcast transmission

0xFFFE - Set to this value for a broadcast transmission.

0x00 - Source endpoint

0x00 - Destination endpoint

0x0000 - Cluster ID (Network Address Request)

0x0000 - Profile ID (ZigBee Device Profile)

0x00 - Broadcast radius

0x00 - Tx Options

0x44 - Transaction sequence number

0x34124040 00A21300 00 00 - Required payload for Network Address Request command

0x33 - Checksum (0xFF - SUM(all bytes after length))

Description:

This API frame sends a broadcast ZDO Network Address Request to obtain the 16-bit address of a device with a 64-bit address of 0x0013A200 40401234. Note the bytes for the 64-bit address were inserted in little endian byte order. All multi-byte fields in the API payload of a ZDO command must have their data inserted in little endian byte order. Also recall that the AO command must be set correctly on an API device to enable the explicit API receive frames in order to receive the ZDO response.

Transmission Timeouts

The transmission timeout varies depending on the nature of the transmission. Timeouts are provided for the following transmission types:

- Transmitting to remote router or coordinator
- Transmitting to child end device
- Transmitting to remote end device.

Note: The timeouts in this section are theoretical timeouts and not precisely accurate. The application should pad the calculated maximum timeouts by a few hundred milliseconds.

Transmitting to Remote Router or Coordinator

The timeout when transmitting to a remote router or coordinator is settable with the NH command. The actual unicast timeout is computed as ((50 * NH) + 100). The default NH value is 30 which equates to a 1.6 second timeout.

The worst case transmission timeout to a remote router or coordinator includes 3 transmission attempts (1 attempt and 2 retries). The maximum total timeout to a remote router or coordinator is about:

3 * ((50 * NH) + 100).

For example, if NH=30 (0x1E), the total transmission to a remote router or coordinator is about

3 * ((50 * 30) + 100), or

3 * (1500 + 100), or

3 * (1600), or

4800 ms, or

4.8 seconds

Transmitting to Child End Device

When sending data to a child end device, a router or coordinator buffers data for a time based on the SP setting. The XBee ZB firmware buffers data for (1.2 * SP) to ensure data is buffered slightly longer than the SP time on an end device. (See "Managing End Devices" chapter for details.) The minimum packet buffer time is 1.2 seconds.

The worst case timeout would occur if the parent believes the end device is a child, but the end device is powered off or otherwise no longer within range of the parent. In this case, the worst case transmission timeout is about the same as when transmitting to a remote end device. (See section below.)

It is also worth noting that a number of provisions exist to help a parent router or coordinator remove stale children from its child table. In ZB firmware, the parent has a poll timeout that will expire an end device child from its child table if it has not received a poll request from the end device child within a certain timeout. Alternatively, if the end device joins a new parent, it will send a broadcast transmission to alert all devices that it has joined a new parent. This mechanism also allows the former parent to remove the end device from its child table.

Transmitting to Remote End Device

The worst case transmission to a remote end device occurs when a previously known end device has left or moved to a different parent, without the sender knowing about it. In this case, the transmission timeout would is defined below.

A transmission to a remote end device is based on the XBee SP parameter. (SP determines the sleep period on XBee end devices. See "Managing End Devices" chapter for details.) A device sending data to a remote end device waits for the unicast timeout, plus the sleep period, before failing the transmission attempt. This is calculated as (50 * NH) + SP. Similar to the remote router or coordinator case, two transmission retries are also expected. Taking retries into account, the worst case transmission timeout is about 3 * ((50 * NH) + (1.2 * SP)).

For example, suppose a router is configured with NH=30 (0x1E) and SP=1000 (0x3E8), and that it is trying to communicate with a remote end device that sleeps for 10 seconds. (Since SP is measured in 10ms units, a value of 1000 is 10 seconds.) The total transmission timeout to the remote end device would be:

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

3 * (1500 + (1.2 * SP)), or

3 * (1500 + 12,000)

3 * (13500), or

40,500ms, or

40.5 seconds.

Transmission Examples

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

API Frame:

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

Field Composition:

0x0014 - length

0x10 - API ID (tx data)

0x01 - frame ID (set greater than 0 to enable the tx-status response)

0x00000000 00000000 - 64-bit address of coordinator (ZB definition)

0xFFFE - Required 16-bit address if sending data to 64-bit address of 0.

0x00 - Broadcast radius (0 = max hops)

0x00 - Tx options

0x54 78 44 61 74 61 - ASCII representation of "TxData" string

0xAB - Checksum (0xFF - SUM(all bytes after length))

Description:

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

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

API Frame:

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

Field Composition:

0x0014 - length

0x10 - API ID (tx data)

0x01 - frame ID (set to a non-zero value to enable the tx-status response)

0x00000000 0000FFFF - Broadcast definition (including sleeping end devices

0xFFFE - Required 16-bit address to send broadcast transmission.

0x00 - Broadcast radius (0 = max hops)

0x00 - Tx options

0x54 78 44 61 74 61 - ASCII representation of "TxData" string

0xAD - Checksum (0xFF - SUM(all bytes after length))

Description:

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

If receiver application addressing is enabled, the XBee will report all received data frames in the explicit format (0x91) to indicate the source and destination endpoints, cluster ID, and profile ID that each packet was received on. (Status messages like modem status and route record indicators are not affected.)

To enable receiver application addressing, set the AO command to 1 using the AT command frame (0x08).

API Frame:

7E 0005 08 01 414F 01 65

Field Composition:

0x0005 - length

0x08 - API ID (at command)

0x01 - frame ID (set to a non-zero value to enable AT command response frames)

0x414F - ASCII representation of 'A','O' (the command being issued)

0x01 - Parameter value

0x65 - Checksum (0xFF - SUM(all bytes after length))

Description:

Setting AO=1 is required for the XBee to use the explicit receive API frame (0x91) when RF data packets are received. This is required if the application needs indication of source or destination endpoint, cluster ID, and/or profile ID values used in received ZigBee data packets. ZDO messages can only be received if AO=1.

5. Security

ZigBee supports various levels of security that can be configured depending on the needs of the application. Security provisions include:

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

The first half of this chapter describes various security features defined in the ZigBee-PRO specification, while the last half illustrates how the XBee and XBee-PRO modules can be configured to support these features

Security Modes

The ZigBee standard supports three security modes – residential, standard, and high security. Residential security was first supported in the ZigBee 2006 standard. This level of security requires a network key be shared among devices. Standard security adds a number of optional security enhancements over residential security, including an APS layer link key. High security adds entity authentication, and a number of other features not widely supported.

XBee ZB modules primarily support standard security, although end devices that support residential security can join and interoperate with standard security devices. The remainder of this chapter focuses on material that is relevant to standard security.

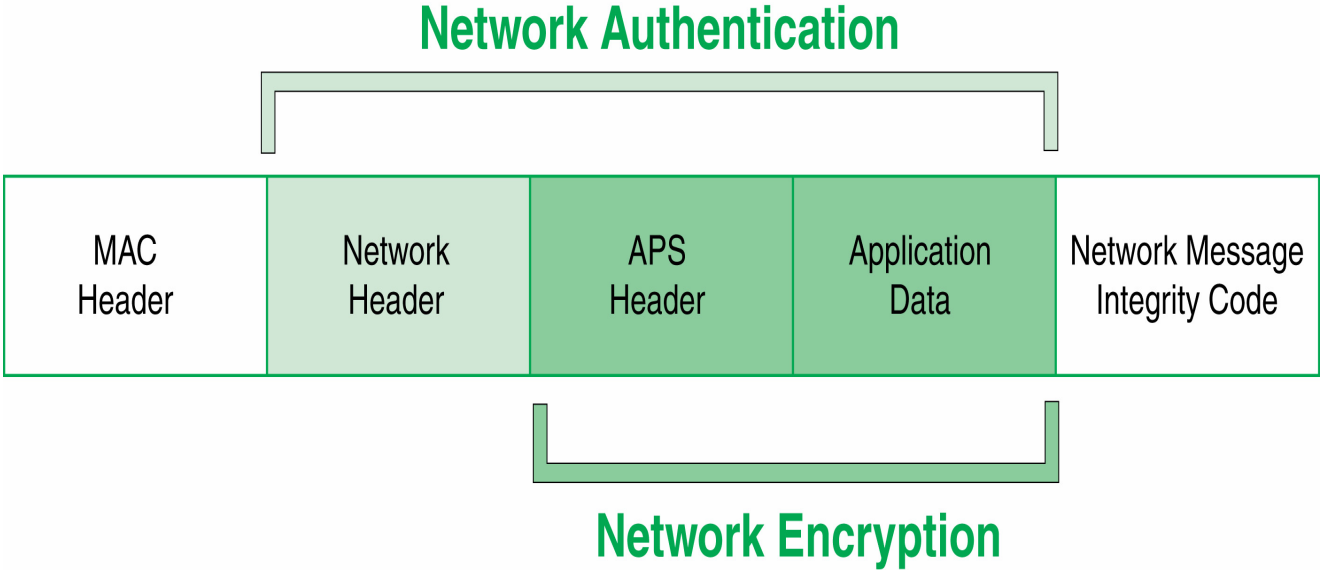
ZigBee Security Model

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

Network Layer Security

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

Packets are encrypted and authenticated using 128-bit AES. This is shown in the figure below.



Frame Counter

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

If the frame counter reaches a maximum value of 0xFFFFFFFF, it does not wrap to 0 and no more transmissions can be sent. Due to the size of the frame counters, reaching the maximum value is a very unlikely event for most applications. The following table shows the required time under different conditions, for the frame counter to reach its maximum value.

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

To clear the frame counters without compromising security, the network key can be changed in the network. When the network key is updated, the frame counters on all devices reset to 0. (See the Network Key Updates section for details.)

Message Integrity Code

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

Network Layer Encryption and Decryption

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

Since network encryption is performed at each hop, packet latency is slightly longer in an encrypted network than in a non-encrypted network. Also, security requires 18 bytes of overhead to include a 32-bit frame counter, an 8-byte source address, 4-byte MIC, and 2 other bytes. This reduces the number of payload bytes that can be sent in a data packet.

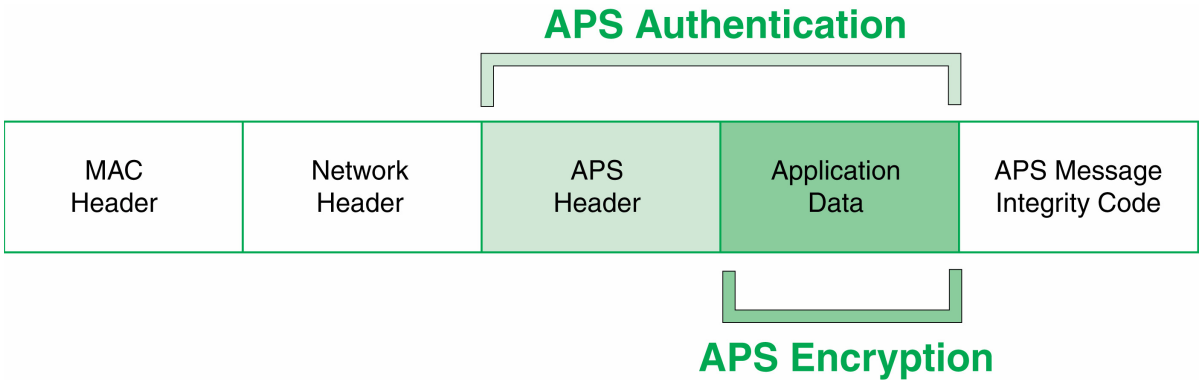
Network Key Updates

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

APS Layer Security

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

If APS security is enabled, packets are encrypted and authenticated using 128-bit AES. This is shown in the figure below:



Message integrity Code

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

APS Link Keys

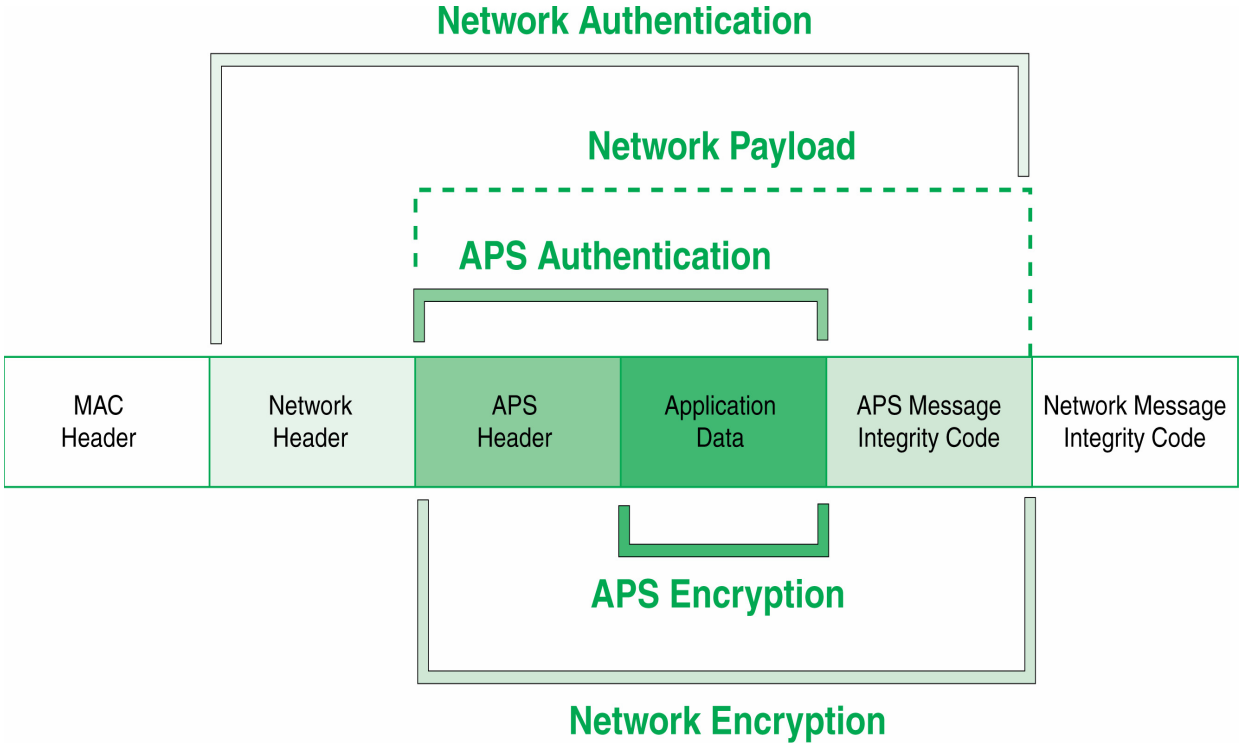
There are two kinds of APS link keys – trust center link keys and application link keys. A trust center link key is established between a device and the trust center, where an application link key is established between a device and another device in the network where neither device is the trust center.

APS Layer Encryption and Decryption

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

Network and APS Layer Encryption

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



Trust Center

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

Forming and Joining a Secure Network

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

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

that are not pre-configured with the link key. Sending the network key unencrypted is not recommended as it can open a security hole in the network. To maximize security, devices should be pre-configured with the correct link key.

Implementing Security on the XBee

If security is enabled in the XBee ZB firmware, devices acquire the network key when they join a network. Data transmissions are always encrypted with the network key, and can optionally be end-to-end encrypted with the APS link key. The following sections discuss the security settings and options in the XBee ZB firmware.

Enabling Security

To enable security on a device, the EE command must be set to 1. If the EE command value is changed and changes are applied (i.e. AC command), the XBee module will leave the network (PAN ID and channel) it was operating on, and attempt to form or join a new network.

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

Setting the Network Security Key

The coordinator must select the network security key for the network. The NK command (write-only) is used to set the network key. If NK=0 (default), a random network key will be selected. (This should suffice for most applications.) Otherwise, if NK is set to a non-zero value, the network security key will use the value specified by NK. NK is only supported on the coordinator.

Routers and end devices with security enabled (ATEE=1)acquire the network key when they join a network. They will receive the network key encrypted with the link key if they share a pre-configured link key with the coordinator. See the following section for details.

Setting the APS Trust Center Link Key

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

Note: Application link keys (sent between two devices where neither device is the coordinator) are not supported in ZB firmware at this time.

Random Trust Center Link Keys

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

Pre-configured Trust Center Link Keys

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

Using a Trust Center

The EO command can be used to define the coordinator as a trust center. If the coordinator is a trust center, it will be alerted to all new join attempts in the network. The trust center also has the ability to update or change the network key on the network.

In ZB firmware, a secure network can be established with or without a trust center. Network and APS layer encryption are supported if a trust center is used or not.

Updating the Network Key with a Trust Center

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

Updating the Network Key without a Trust Center

If the coordinator is not running as a trust center, the network reset command (NR1) can be used to force all devices in the network to leave the current network and rejoin the network on another channel. When devices leave and reform then network, the frame counters are reset to 0. This approach will cause the coordinator to form a new network that the remaining devices should join. Resetting the network in this manner will bring the coordinator and routers in the network down for about 10 seconds, and will likely cause the 16-bit PAN ID and 16-bit addresses of the devices to change.

XBee Security Examples

This section covers some sample XBee configurations to support different security modes. Several AT commands are listed with suggested parameter values. The notation in this section includes an '=' sign to indicate what each command register should be set to - for example, EE=1. This is not the correct notation for setting command values in the XBee. In AT command mode, each command is issued with a leading 'AT' and no '=' sign - for example ATEE1. In the API, the two byte command is used in the command field, and parameters are populated as binary values in the parameter field.

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

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

In this example, EE, ID, and KY are set the same on all devices. After successfully joining the secure network, all application data transmissions will be encrypted by the network key. Since NK was set to 0 on the coordinator, a random network key was selected. And since the link key (KY) was configured the same on all devices, to a non-zero value, the network key was sent encrypted by the pre-configured link key (KY) when the devices joined.

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

1. Start a coordinator with the following settings:
   a. ID=2235
   b. EE=1
   c. NK=0

d. KY=0

e. WR (save networking parameters to preserve them through power cycle)

2. Configure one or more routers or end devices with the following settings:

a. ID=2235

b. EE=1

c. KY=0

d. WR (save networking parameters to preserve them through power cycle)

3. Read the AI setting on the coordinator and joining devices until they return 0 (formed or joined a network).

In this example, EE, ID, and KY are set the same on all devices. Since NK was set to 0 on the coordinator, a random network key was selected. And since KY was set to 0 on all devices, the network key was sent unencrypted ("in the clear") when the devices joined. This approach introduces a security vulnerability into the network and is not recommended.

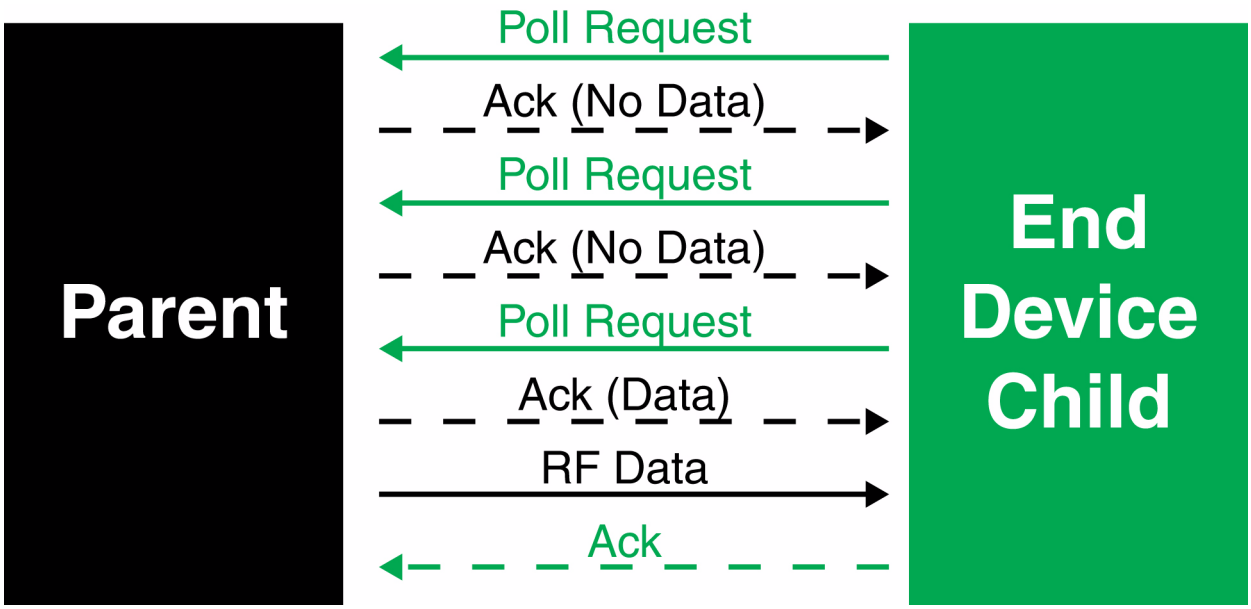
6. Managing End Devices

ZigBee end devices are intended to be battery-powered devices capable of sleeping for extended periods of time. Since end devices may not be awake to receive RF data at a given time, routers and coordinators are equipped with additional capabilities (including packet buffering and extended transmission timeouts) to ensure reliable data delivery to end devices.

End Device Operation

When an end device joins a ZigBee network, it must find a router or coordinator device that is allowing end devices to join. Once the end device joins a network, a parent-child relationship is formed between the end device and the router or coordinator that allowed it to join. See chapter 4 for details.

When the end device is awake, it sends poll request messages to its parent. When the parent receives a poll request, it checks a packet queue to see if it has any buffered messages for the end device. It then sends a MAC layer acknowledgment back to the end device that indicates if it has data to send to the end device or not.



If the end device receives the acknowledgment and finds that the parent has no data for it, the end device can return to idle mode or sleep. Otherwise, it will remain awake to receive the data. This polling mechanism allows the end device to enter idle mode and turn its receiver off when RF data is not expected in order to reduce current consumption and conserve battery life.

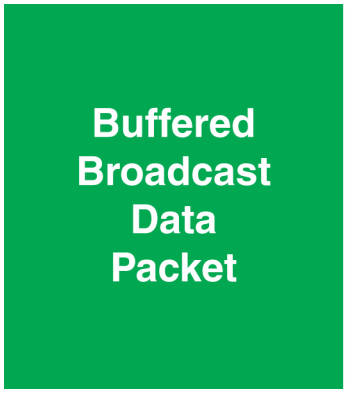
The end device can only send data directly to its parent. If an end device must send a broadcast or a unicast transmission to other devices in the network, it sends the message directly to its parent and the parent performs any necessary route or address discoveries to route the packet to the final destination.

Parent Operation

Each router or coordinator maintains a child table that contains the addresses of its end device children. A router or coordinator that has unused entries in its child table is said to have end device capacity, or the ability to allow new end devices to join. If the child table is completely filled (such that the number of its end device children matches the number of child table entries), the device cannot allow any more end devices to join to it.

Since the end device children are not guaranteed to be awake at a given time, the parent is responsible to manage incoming data packets in behalf of its end device children. If a parent receives an RF data transmission destined for one of its end device children, and if the parent has enough unused buffer space, it will buffer the packet. The data packet will remain buffered until a timeout expires, or until the end device sends a poll request to retrieve the data.

The parent can buffer one broadcast transmission for all of its end device children. When a broadcast transmission is received and buffered, the parent sets a flag in its child table when each child polls and retrieves the packet. Once all children have received the broadcast packet, the buffered broadcast packet is discarded. If all children have not received a buffered broadcast packet and a new broadcast is received, the old broadcast packet is discarded, the child table flags are cleared, and the new broadcast packet is buffered for the end device children. This is demonstrated in the figure below.



End Device Child Table

Address	Received Broadcast
Ox2120	T
OxF220	F
OxC100	F
Ox5750	T

When an end device sends data to its parent that is destined for a remote device in the network, the parent buffers the data packet until it can establish a route to the destination. The parent may perform a route or 16-bit address discovery in behalf of its end device children. Once a route is established, the parent sends the data transmission to the remote device.

End Device Poll Timeouts

To better support mobile end devices (end devices that can move around in a network), parent router and coordinator devices have a poll timeout for each end device child. If an end device does not send a poll request to its parent within the poll timeout, the parent will remove the end device from its child table. This allows the child table on a router or coordinator to better accommodate mobile end devices in the network.

Packet Buffer Usage

Packet buffer usage on a router or coordinator varies depending on the application. The following activities can require use of packet buffers for up to several seconds:

- Route and address discoveries
- Application broadcast transmissions
- Stack broadcasts (i.e. ZDO "Device Announce" messages when devices join a network)
- Unicast transmissions (buffered until acknowledgment is received from destination or retries exhausted)
- Unicast messages waiting for end device to wake.

Applications that use regular broadcasting or that require regular address or route discoveries will use up a significant number of buffers, reducing the buffer availability for managing packets for end device children. Applications should reduce the number of required application broadcasts, and consider implementing an external address table or many-to-one and source routing if necessary to improve routing efficiency.

Non-Parent Device Operation

Devices in the ZigBee network treat data transmissions to end devices differently than transmissions to other routers and coordinators. Recall that when a unicast transmission is sent, if a network acknowledgment is not received within a timeout, the device resends the transmission. When transmitting data to remote coordinator or router devices, the transmission timeout is relatively short since these devices are powered and responsive. However, since end devices may sleep for some time, unicast transmissions to end devices use an extended timeout mechanism in order to allow enough time for the end device to wake and receive the data transmission from its parent.

If a non-parent device does not know the destination is an end device, it will use the standard unicast timeout for the transmission. However, provisions exist in the Ember ZigBee stack for the parent to inform the message sender that the destination is an end device. Once the sender discovers the destination device is an end device, future transmissions will use the extended timeout. See the XBee Router / Coordinator Configuration section in this chapter for details.

XBee End Device Configuration

XBee end devices support two different sleep modes:

- Pin Sleep
- Cyclic Sleep.

Pin sleep allows an external microcontroller to determine when the XBee should sleep and when it should wake by controlling the Sleep_RQ pin. In contrast, cyclic sleep allows the sleep period and wake times to be configured through the use of AT commands. The sleep mode is configurable with the SM command.

In both pin and cyclic sleep modes, XBee end devices poll their parent every 100ms while they are awake to retrieve buffered data. When a poll request has been sent, the end device enables the receiver until an acknowledgment is received from the parent. (It generally takes about 10ms from the time the poll request is sent until the acknowledgment is received.) The acknowledgment indicates if the parent has buffered data for the end device child or not. If the acknowledgment indicates the parent has pending data, the end device will leave the receiver on to receive the data. Otherwise, the end device will turn off the receiver and enter idle mode (until the next poll request is sent) to reduce current consumption (and improve battery life).

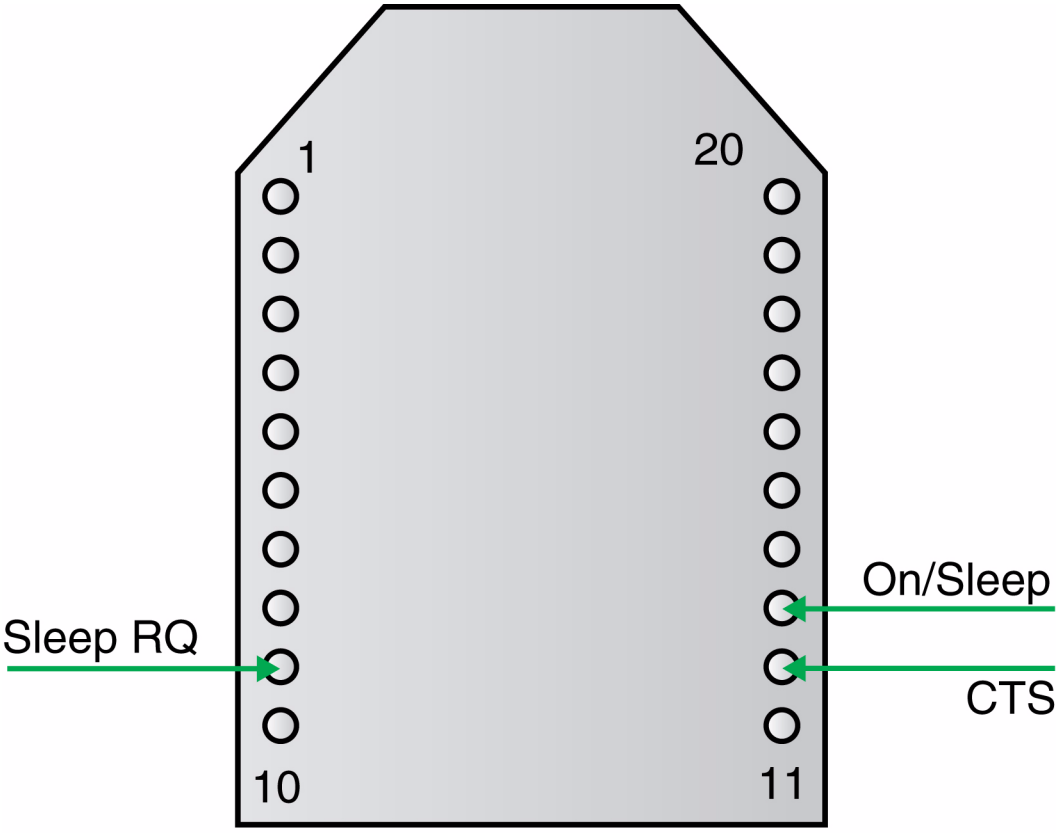
Once the module enters sleep mode, the On/Sleep pin (pin 13) is de-asserted (low) to indicate the module is entering sleep mode. If CTS hardware flow control is enabled (D7 command), the CTS pin (pin 12) is de-asserted (high) when entering sleep to indicate that serial data should not be sent to the module. The module will not respond to serial or RF data when it is sleeping. Applications that must communicate serially to sleeping end devices are encouraged to observe CTS flow control.

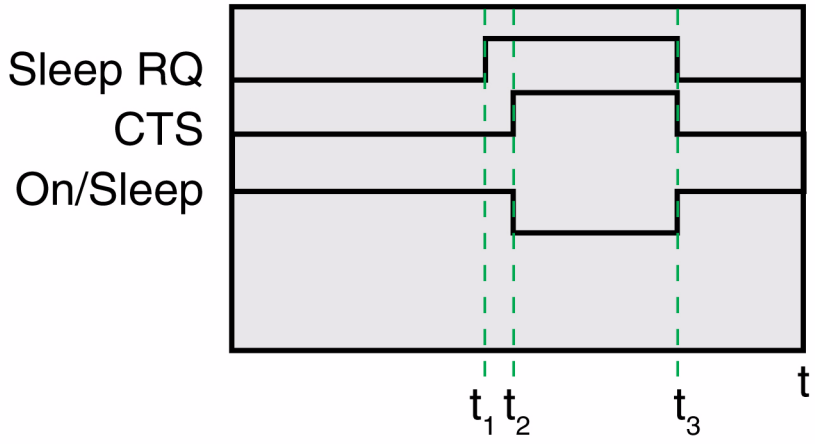
When the XBee wakes from sleep, the On/Sleep pin is asserted (high), and if flow control is enabled, the CTS pin is also asserted (low). If the module has not joined a network, it will scan all SC channels after waking to try and find a valid network to join.

Pin Sleep

Pin sleep allows the module to sleep and wake according to the state of the Sleep_RQ pin (pin 9). Pin sleep mode is enabled by setting the SM command to 1.

When Sleep_RQ is asserted (high), the module will finish any transmit or receive operations and enter a low power state. For example, if the module has not joined a network and Sleep_RQ is asserted (high), the module will sleep once the current join attempt completes (i.e. when scanning for a valid network completes). The module will wake from pin sleep when the Sleep_RQ pin is de-asserted (low).





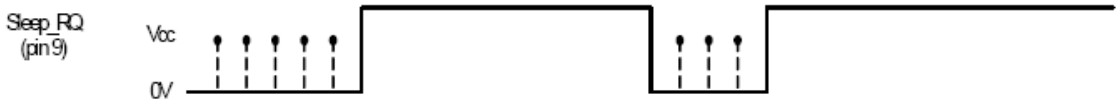
In the figure above, t1, t2, and t3 represent the following events:

- T1 - Time when Sleep_RQ is asserted (high)
- T2 - Time when the XBee enters sleep (CTS state change only if hardware flow control is enabled)
- T3 - Time when Sleep_RQ is de-asserted (low) and the module wakes.

The time between T1 and T2 varies depending on the state of the module. In the worst case scenario, if the end device is trying to join a network, or if it is waiting for an acknowledgment from a data transmission, the delay could be up to a few seconds.

When the XBee is awake and is joined to a network, it sends a poll request to its parent to see if the parent has any buffered data for it. The end device will continue to send poll requests every 100ms while it is awake.

Demonstration of Pin Sleep



Demonstration of a pin sleep end device that sends poll requests to its parent when awake

Legend

Sleep_RQ

Transmitting Poll Request

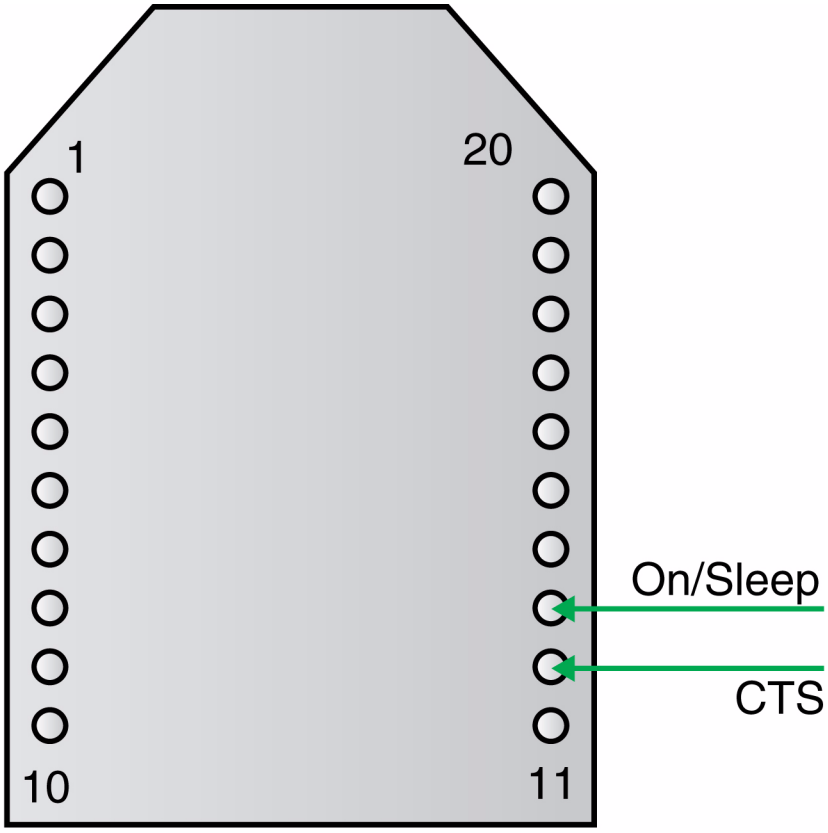
Parent and remote devices must be configured to buffer data correctly and to utilize adequate transmission timeouts. See the XBee Router / Coordinator Configuration section in this chapter for details.

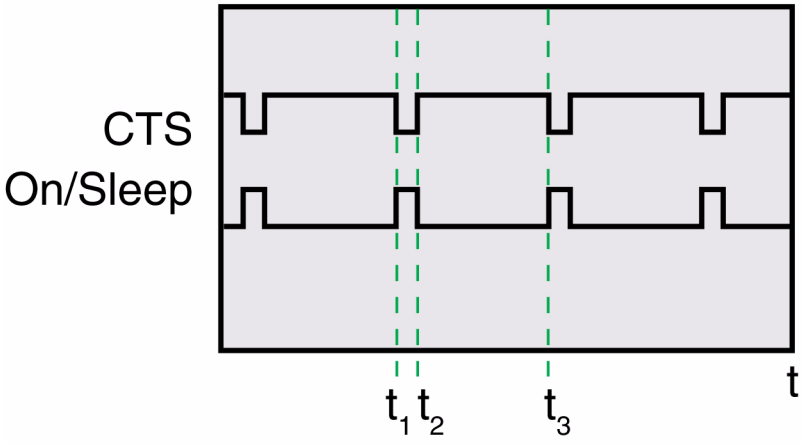
Cyclic Sleep

Cyclic sleep allows the module to sleep for a specified time and wake for a short time to poll its parent for any buffered data messages before returning to sleep again. Cyclic sleep mode is enabled by setting the SM command to 4 or 5. SM5 is a slight variation of SM4 that allows the module to be woken prematurely by asserting the Sleep_RQ pin (pin 9). In SM5, the XBee can wake after the sleep period expires, or if a high-to-low transition occurs on the Sleep_RQ pin. Setting SM to 4 disables the pin wake option.

In cyclic sleep, the module sleeps for a specified time, and then wakes and sends a poll request to its parent to discover if the parent has any pending data for the end device. If the parent has buffered data for the end device, or if serial data is received, the XBee will remain awake for a time. Otherwise, it will enter sleep mode immediately.

The On/Sleep line is asserted (high) when the module wakes, and is de-asserted (low) when the module sleeps. If hardware flow control is enabled (D7 command), the CTS pin will assert (low) when the module wakes and can receive serial data, and de-assert (high) when the module sleeps.





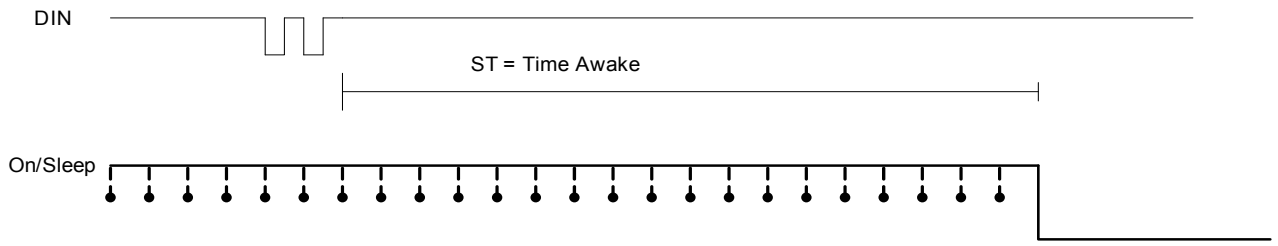
In the figure above, t1, t2, and t3 represent the following events:

- T1 - Time when the module wakes from cyclic sleep
- T2 - Time when the module returns to sleep
- T3 - Later time when the module wakes from cyclic sleep.

The wake time and sleep time are configurable with software commands as described in the sections below.

Wake Time (Until Sleep)

In cyclic sleep mode (SM=4 or 5), if serial or RF data is received, the module will start a sleep timer (time until sleep). Any data received serially or over the RF link will restart the timer. The sleep timer value is settable with the ST command. While the module is awake, it will send poll request transmissions every 100ms to check its parent for buffered data messages. The module returns to sleep when the sleep timer expires, or if the SI command is sent to it. The following image shows this behavior.



A cyclic sleep end device enters sleep mode when no serial or RF data is received for ST time .

Legend

On/Sleep

Transmitting Poll Request

Sleep Period

The sleep period is configured based on the SP, SN, and SO commands. The following table lists the behavior of these commands.

Com-mand	Range	Description
SP	0x20 - 0xAF0 (x 10 ms) (320 - 28,000 ms)	Configures the sleep period of the module.
SN	1 - 0xFFFF	Configures the number of sleep periods multiplier.
SO	0 - 0xFF	Defines options for sleep mode behavior. 0x02 - Always wake for full ST time 0x04 - Enable extended sleep (sleep for full (SP * SN) time)

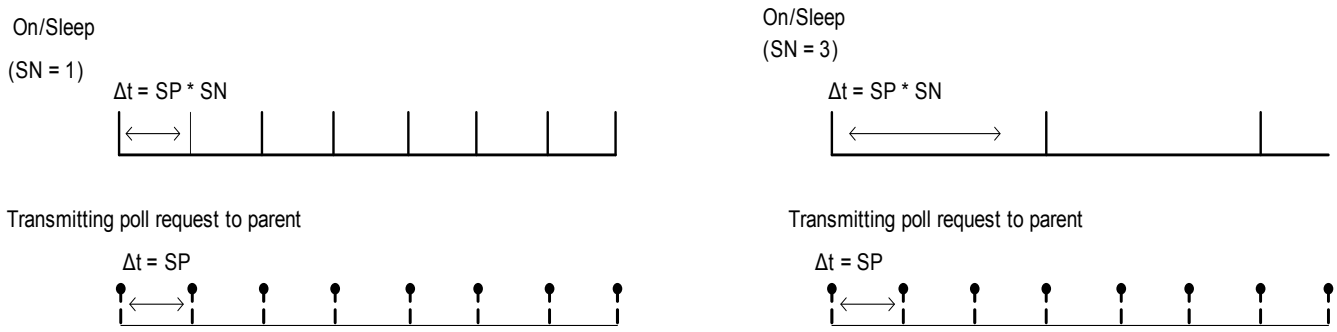
The XBee module supports both a short cyclic sleep and an extended cyclic sleep that make use of these commands.. These two modes allow the sleep period to be configured according to the application requirements.

Short Cyclic Sleep

In short cyclic sleep mode, the sleep behavior of the module is defined by the SP and SN commands, and the SO command must be set to 0x00 (default) or 0x02. In short cyclic sleep mode, the SP command defines the sleep period and is settable up to 28 seconds. When the XBee enters short cyclic sleep, it remains in a low power state until the SP time has expired.

After the sleep period expires, the XBee sends a poll request transmission to its parent to determine if its parent has any buffered data waiting for the end device. Since router and coordinator devices can buffer data for end device children up to 30 seconds, the SP range (up to 28 seconds) allows the end device to poll regularly enough to receive buffered data. If the parent has data for the end device, the end device will start its sleep timer (ST) and continue polling every 100ms to receive data. If the end device wakes and finds that its parent has no data for it, the end device can return to sleep immediately.

The SN command can be used to control when the On/Sleep line is asserted (high). If SN is set to 1 (default), the On/Sleep line will be set high each time the XBee wakes from sleep. Otherwise, if SN is greater than 1, the On/Sleep line will only be set high if RF data is received, or after SN wake cycles occur. This allows an external device to remain powered off until RF data is received, or until a number of sleep periods have expired (SN sleep periods). This mechanism allows the XBee to wake at regular intervals to poll its parent for data without waking an external device for an extended time (SP * SN time). This is shown in the figure below.



Setting SN > 1 allows the XBee to silently poll for data without asserting On /Sleep. If RF data is received when polling, On/Sleep will immediately assert .

Legend

Sleep_RQ

Transmitting Poll Request

NOTE: SP controls the packet buffer time on routers and coordinators. SP should be set on all router and coordinator devices to match the longest end device SP time. See the XBee Router / Coordinator Configuration section for details.

Extended Cyclic Sleep

In extended cyclic sleep operation, an end device can sleep for a multiple of SP time which can extend the sleep time up to several days. The sleep period is configured using the SP and SN commands. The total sleep period is equal to (SP * SN) where SP is measured in 10ms units. The SO command must be set correctly to enable extended sleep.

Since routers and coordinators can only buffer incoming RF data for their end device children for up to 30 seconds, if an end device sleeps longer than 30 seconds, devices in the network need some indication when an end device is awake before they can send data to it. End devices that use extended cyclic sleep should send a transmission (such as an IO sample) when they wake to inform other devices that they are awake and can receive data. It is recommended that extended sleep end devices set SO to wake for the full ST time in order to provide other devices with enough time to send messages to the end device.

Similar to short cyclic sleep, end devices running in this mode will return to sleep when the sleep timer expires, or when the SI command is received.

Transmitting RF Data

An end device may transmit data when it wakes from sleep and has joined a network. End devices transmit directly to their parent and then wait for an acknowledgment to be received. The parent will perform any required address and route discoveries to help ensure the packet reaches the intended destination before reporting the transmission status to the end device.

Receiving RF Data

After waking from sleep, an end device sends a poll request to its parent to determine if the parent has any buffered data for it. In pin sleep mode, the end device polls every 100ms while the Sleep_RQ pin is de-asserted (low). In cyclic sleep mode, the end device will only poll once before returning to sleep unless the sleep timer (ST) is started (serial or RF data is received). If the sleep timer is started, the end device will continue to poll every 100ms until the sleep timer expires.

IO Sampling

End devices can be configured to send one or more IO samples when they wake from sleep. To enable IO sampling on an end device, the IR command must be set to a non-zero value, and at least one analog or digital IO pin must be enabled for sampling (D0 - D9, P0-P2 commands). If IO sampling is enabled, an end device sends an IO sample when it wakes and starts the ST timer. It will continue sampling at the IR rate until the sleep timer (ST) has expired. See chapter 8 for details.

Waking End Devices with the Commissioning Pushbutton

If the commissioning pushbutton functionality is enabled (D0 command), a high-to-low transition on the AD0/DIO0 pin (pin 20) will cause an end device to wake for 30 seconds. See the Commissioning Pushbutton section in chapter 7 for details.

Parent Verification

Since an end device relies on its parent to maintain connectivity with other devices in the network, XBee end devices include provisions to verify its connection with its parent. End devices monitor their link with their parent when sending poll messages and after a power cycle or reset event as described below.

When an end device wakes from sleep, it sends a poll request to its parent. In cyclic sleep, if RF or serial data is not received and the sleep timer is not started, the end device polls one time and returns to sleep for another sleep period. Otherwise, the end device continues polling every 100ms. If the parent does not send an acknowledgment response to three consecutive poll request transmissions, the end device assumes the parent is out of range, and attempts to find a new parent.

After a power-up or reset event, the end device does an orphan scan to locate its parent. If the parent does not send a response to the orphan scan, the end device attempts to find a new parent.

Rejoining

Once all devices have joined a ZigBee network, the permit-joining attribute should be disabled such that new devices are no longer allowed to join the network. Permit-joining can be enabled later as needed for short times. This provides some protection in preventing other devices from joining a live network.

If an end device cannot communicate with its parent, the end device must be able to join a new parent to maintain network connectivity. However, if permit-joining is disabled in the network, the end device will not find a device that is allowing new joins.

To overcome this problem, ZigBee supports rejoining, where an end device can obtain a new parent in the same network even if joining is not enabled. When an end device joins using rejoining, it performs a PAN ID scan to discover nearby networks. If a network is discovered that has the same 64-bit PAN ID as the end device, it will join the network by sending a rejoin request to one of the discovered devices. The device that receives the rejoin request will send a rejoin response if it can allow the device to join the network (i.e. child table not full). The rejoin mechanism can be used to allow a device to join the same network even if permit-joining is disabled.

To enable rejoining, NJ should be set less than 0xFF on the device that will join. If NJ < 0xFF, the device assumes the network is not allowing joining and first tries to join a network using rejoining. If multiple rejoining attempts fail, or if NJ=0xFF, the device will attempt to join using association.

XBee Router/Coordinator Configuration

XBee routers and coordinators may require some configuration to ensure the following are set correctly:

- RF packet buffering timeout
- Child poll timeout
- Transmission timeout.

The value of these timeouts depends on the sleep time used by the end devices. Each of these timeouts are discussed below.

RF Packet Buffering Timeout

When a router or coordinator receives an RF data packet intended for one of its end device children, it buffers the packet until the end device wakes and polls for the data, or until a packet buffering timeout occurs. This timeout is settable using the SP command. The actual timeout is (1.2 * SP), with a minimum timeout of 1.2 seconds and a maximum of 30 seconds. Since the packet buffering timeout is set slightly larger than the SP setting, SP should be set the same on routers and coordinators as it is on cyclic sleep end devices. For pin sleep devices, SP should be set as long as the pin sleep device can sleep, up to 30 seconds.

Note: In pin sleep and extended cyclic sleep, end devices can sleep longer than 30 seconds. If end devices sleep longer than 30 seconds, parent and non-parent devices must know when the end device is awake in order to reliably send data. For applications that require sleeping longer than 30 seconds, end devices should transmit an IO sample or other data when they wake to alert other devices that they can send data to the end device.

Child Poll Timeout

Router and coordinator devices maintain a timestamp for each end device child indicating when the end device sent its last poll request to check for buffered data packets. If an end device does not send a poll request to its parent for a certain period of time, the parent will assume the end device has moved out of range and will remove the end device from its child table. This allows routers and coordinators to be responsive to changing network conditions. The NC command can be issued at any time to read the number of remaining (unused) child table entries on a router or coordinator.

The child poll timeout is settable with the SP and SN commands. SP and SN should be set such that SP * SN matches the longest expected sleep time of any end devices in the network. The actual timeout is calculated as (3 * SP * SN), with a minimum of 5 seconds. For networks consisting of pin sleep end devices, the SP and SN values on the coordinator and routers should be set such that SP * SN matches the longest expected sleep period of any pin sleep device. The 3 multiplier ensures the end device will not be removed unless 3 sleep cycles pass without receiving a poll request. The poll timeout is settable up to a couple months.

Transmission Timeout

As mentioned in chapter 4, when sending RF data to a remote router, since routers are always on, the timeout is based on the number of hops the transmission may traverse. This timeout it settable using the NH command. (See chapter 4 for details.)

Since end devices may sleep for lengthy periods of time, the transmission timeout to end devices also includes some allowance for the sleep period of the end device. When sending data to a remote end device, the transmission timeout is calculated using the SP and NH commands. If the timeout occurs and an acknowledgment has not been received, the source device will resend the transmission until an acknowledgment is received, up to two more times.

The transmission timeout per attempt is:

3 * ((unicast router timeout) + (end device sleep time)), or

3 * ((50 * NH) + (1.2 * SP)), where SP is measured in 10ms units.

For best results, SP should be set on routers and coordinator devices to match the SP setting on the end devices.

Note: The NH command is used to determine the timeout when transmitting to routers.

Putting it all Together

Short Sleep Periods

Pin and cyclic sleep devices that sleep less than 30 seconds can receive data transmissions at any time since their parent device(s) will be able to buffer data long enough for the end devices to wake and poll to receive the data. SP should be set the same on all devices in the network. If end devices in a network have more than one SP setting, SP on the routers and coordinators should be set to match the largest SP setting of any end device. This will ensure the RF packet buffering, poll timeout, and transmission timeouts are set correctly.

Extended Sleep Periods

Pin and cyclic sleep devices that might sleep longer than 30 seconds cannot receive data transmissions reliably unless certain design approaches are taken. Specifically, the end devices should use IO sampling or another mechanism to transmit data when they wake to inform the network they can receive data. SP and SN should be set on routers and coordinators such that (SP * SN) matches the longest expected sleep time. This configures the poll timeout so end devices are not expired from the child table unless a poll request is not received for 3 consecutive sleep periods.

As a general rule of thumb, SP and SN should be set the same on all devices in almost all cases.

Sleep Examples

This section covers some sample XBee configurations to support different sleep modes. Several AT commands are listed with suggested parameter values. The notation in this section includes an '=' sign to indicate what each command register should be set to - for example, SM=4. This is not the correct notation for setting command values in the XBee. In AT command mode, each command is issued with a leading 'AT' and no '=' sign - for example ATSM4. In the API, the two byte command is used in the command field, and parameters are populated as binary values in the parameter field.

Example 1

Configure a device to sleep for 20 seconds, but set SN such that the On/Sleep line will remain de-asserted for up to 1 minute.

The following settings should be configured on the end device.

SM = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake)

SP = 0x7D0 (2000 decimal). This causes the end device to sleep for 20 seconds since SP is measured in units of 10ms.

SN = 3. (With this setting, the On/Sleep pin will assert once every 3 sleep cycles, or when RF data is received)

SO = 0

All router and coordinator devices on the network should set SP to match SP on the end device. This ensures that RF packet buffering times and transmission timeouts will be set correctly.

Since the end device wakes after each sleep period (ATSP), the SN command can be set to 1 on all routers and the coordinator.

Example 2

Configure an end device to sleep for 20 seconds, send 4 IO samples in 2 seconds, and return to sleep.

Since SP is measured in 10ms units, and ST and IR are measured in 1ms units, configure an end device with the following settings:

SM = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake)

SP = 0x7D0 (2000 decimal). This causes the end device to sleep for 20 seconds.

SN = 1

SO = 0

ST = 0x7D0 (2000 decimal). This sets the sleep timer to 2 seconds.

IR = 0x258 (600 decimal). Set IR to a value greater than (2 seconds / 4) to get 4 samples in 2 seconds. The end device sends an IO sample at the IR rate until the sleep timer has expired.

At least one analog or digital IO line must be enabled for IO sampling to work. To enable pin 19 (AD1/DIO1) as a digital input line, the following must be set:

D1 = 3

All router and coordinator devices on the network should set SP to match SP on the end device. This ensures that RF packet buffering times and transmission timeouts will be set correctly.

Example 3

Configure a device for extended sleep: to sleep for 4 minutes.

SP and SN must be set such that SP * SN = 4 minutes. Since SP is measured in 10ms units, the following settings can be used to obtain 4 minute sleep.

SM = 4 (cyclic sleep) or 5 (cyclic sleep, pin wake)

SP = 0x7D0 (2000 decimal, or 20 seconds)

SN = 0x0B (12 decimal)

SO = 0x04 (enable extended sleep)

With these settings, the module will sleep for SP * SN time, or (20 seconds * 12) = 240 seconds = 4 minutes.

For best results, the end device should send a transmission when it wakes to inform the coordinator (or network) when it wakes. It should also remain awake for a short time to allow devices to send data to it. The following are recommended settings.

ST = 0x7D0 (2 second wake time)

SO = 0x06 (enable extended sleep and wake for ST time)

IR = 0x800 (send 1 IO sample after waking). At least one analog or digital IO sample should be enabled for IO sampling.

With these settings, the end device will wake after 4 minutes and send 1 IO sample. It will then remain awake for 2 seconds before returning to sleep.

SP and SN should be set to the same values on all routers and coordinators that could allow the end device to join. This will ensure the parent does not timeout the end device from its child table too quickly.

The SI command can optionally be sent to the end device to cause it to sleep before the sleep timer expires.

7. Network Commissioning and Diagnostics

Network commissioning is the process whereby devices in a mesh network are discovered and configured for operation. The XBee modules include several features to support device discovery and configuration. In addition to configuring devices, a strategy must be developed to place devices to ensure reliable routes.

To accommodate these requirements, the XBee modules include various features to aid in device placement, configuration, and network diagnostics.

Device Configuration

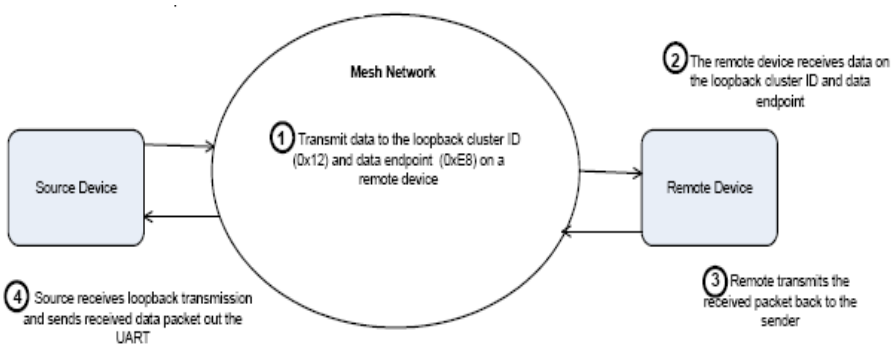
XBee/XBee-PRO ZB modules can be configured locally through serial commands (AT or API), or remotely through remote API commands. API devices can send configuration commands to set or read the configuration settings of any device in the network.

Device Placement

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

Link Testing

A good way to measure the performance of a mesh network is to send unicast data through the network from one device to another to determine the success rate of many transmissions. To simplify link testing, the modules support a loopback cluster ID (0x12) on the data endpoint (0xE8). Any data sent to this cluster ID on the data endpoint will be transmitted back to the sender. This is shown in the figure below:



Demonstration of how the loopback cluster ID and data endpoint can be used to measure the link quality in a mesh network

The configuration steps to send data to the loopback cluster ID depend on the firmware type.

AT Firmware

To send data to the loopback cluster ID on the data endpoint of a remote device, set the CI command value to 0x12. The SE and DE commands should be set to 0xE8 (default value). The

DH and DL commands should be set to the address of the remote (0 for the coordinator, or the 64-bit address of the remote). After exiting command mode, any received serial characters will be transmitted to the remote device, and returned to the sender.

API Firmware

Send an Explicit Addressing ZigBee Command API frame (0x11) using 0x12 as the cluster ID and 0xE8 as the source and destination endpoint. Data packets received by the remote will be echoed back to the sender.

RSSI Indicators

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

The DB value can be determined in hardware using the RSSI/PWM module pin (pin 6). If the RSSI PWM functionality is enabled (P0 command), when the module receives data, the RSSI PWM is set to a value based on the RSSI of the received packet. (Again, this value only indicates the quality of the last hop.) This pin could potentially be connected to an LED to indicate if the link is stable or not.

Device Discovery

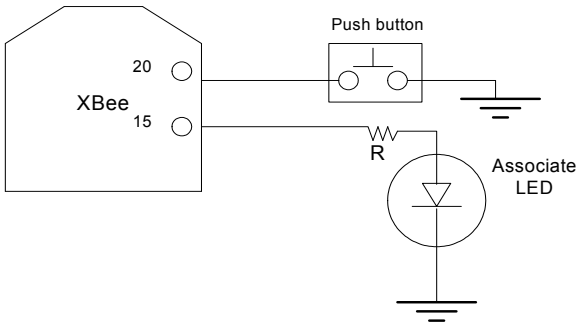
The node discovery command can be used to discover all modules that have joined a network. Issuing the ND command sends a broadcast node discovery command throughout the network. All devices that receive the command will send a response that includes the device's addressing information, node identifier string (see NI command), and other relevant information. This command is useful for generating a list of all module addresses in a network.

When a device receives the node discovery command, it waits a random time before sending its own response. The maximum time delay is set on the ND sender with the NT command. The ND originator includes its NT setting in the transmission to provide a delay window for all devices in the network. Large networks may need to increase NT to improve network discovery reliability. The default NT value is 0x3C (6 seconds).

Commissioning Pushbutton and Associate LED

The XBee modules support a set of commissioning and LED behaviors to aid in device deployment and commissioning. These include the commissioning push button definitions and associate LED behaviors. These features can be supported in hardware as shown below.

Figure 7-09. Commissioning Pushbutton and Associate LED Functionalities



A pushbutton and an LED can be connected to module pins 20 and 15 respectively to support the commissioning pushbutton and associate LED functionalities.

Commissioning Pushbutton

The commissioning pushbutton definitions provide a variety of simple functions to aid in deploying devices in a network. The commissioning button functionality on pin 20 is enabled by setting the D0 command to 1 (enabled by default)..

Table 7-01.

Button Presses	If module is joined to a network	If module is not joined to a network
1	• Wakes an end device for 30 seconds • Sends a node identification broadcast transmission	• Wakes an end device for 30 seconds • Blinks a numeric error code on the Associate pin indicating the cause of join failure (see section 6.4.2).
2	• Sends a broadcast transmission to enable joining on the coordinator and all devices in the network for 1 minute. (If joining is permanently enabled on a device (NJ = 0xFF), this action has no effect on that device.)	• N/A
4	• Causes the device to leave the PAN. • Issues ATRE to restore module parameters to default values, including ID and SC. • The device attempts to join a network based on its ID and SC settings.	• Issues ATRE to restore module parameters to default values, including ID and SC. • The device attempts to join a network based on its ID and SC settings.

Button presses may be simulated in software using the ATCB command. ATCB should be issued with a parameter set to the number of button presses to execute. (i.e. sending ATCB1 will execute the action(s) associated with a single button press.)

The node identification frame is similar to the node discovery response frame – it contains the device's address, node identifier string (NI command), and other relevant data. All API devices that receive the node identification frame send it out their Uart as an API Node Identification Indicator frame (0x95).

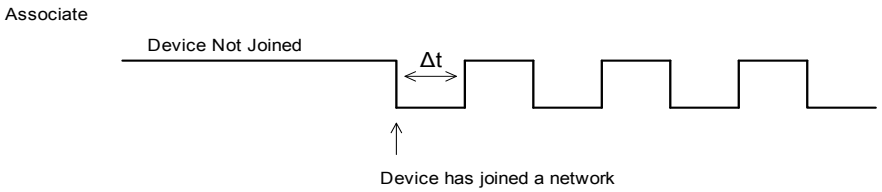
Associate LED

The Associate pin (pin 15) can provide indication of the device's network status and diagnostics information. To take advantage of these indications, an LED can be connected to the Associate pin as shown in the figure above. The Associate LED functionality is enabled by setting the D5 command to 1 (enabled by default). If enabled, the Associate pin is configured as an output and will behave as described in the following sections.

Joined Indication

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

Figure 7-010. Joined Status of a Device



The associate pin can indicate the joined status of a device . Once the device has joined a network, the associate pin toggles state at a regular interval (Δt). The time can be set by using the LT command.

The LT command defines the blink time of the Associate pin. If set to 0, the device uses the default blink time (500ms for coordinator, 250ms for routers and end devices).

Diagnostics Support

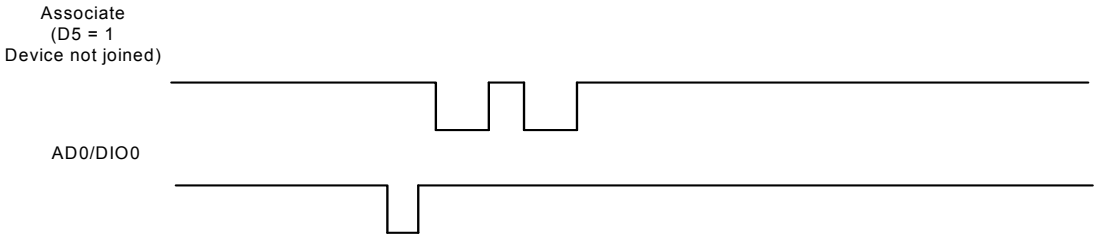
The Associate pin works with the commissioning pushbutton to provide additional diagnostics behaviors to aid in deploying and testing a network. If the commissioning push button is pressed once, and the device has not joined a network, the Associate pin blinks a numeric error code to indicate the cause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

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

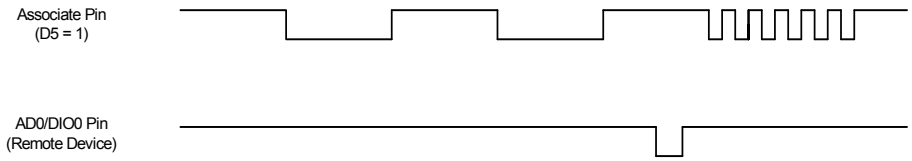
The following figures demonstrate these behaviors.

Figure 7-011. AI = 0x22

Figure 7-012.s Broadcast Node Identification Transmission



A single commissioning button press when the device has not joined a network tha causes the associate pin to blink to indicate the AI Code where: AI = # blinks + 0x2 In this example, AI = 0x22.



A single button press on a remote device causes a broadcast node identification transmission to be sent. All devices that receive this transmission blink their associate pin rapidly for one second if the associate LED functionality is enabled. (D5 = 1)

8. XBee Analog and Digital IO Lines

XBee ZB firmware supports a number of analog and digital IO pins that are configured through software commands. Analog and digital IO lines can be set or queried. The following table lists the configurable IO pins and the corresponding configuration commands.

Table 8-02.

Module Pin Names	Module Pin Numbers	Configuration Command
CD/DIO12	4	P2
PWM0/RSSIM/DIO10	6	P0
PWM/DIO11	7	P1
DIO4	11	D4
CTS/DIO7	12	D7
ASSOC/DIO5	15	D5
RTS/DIO6	16	D6
AD3/DIO3	17	D3
AD2/DIO2	18	D2
AD1/DIO1	19	DI
AD0/DIO0	20	D0

IO Configuration

To enable an analog or digital IO function on one or more XBee module pin(s), the appropriate configuration command must be issued with the correct parameter. After issuing the configuration command, changes must be applied on the module for the IO settings to take effect.

Table 8-03.

Pin Command Parameter	Description
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionalities
2	Analog input, single ended (A/D pins only)
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high
6-9	Alternate functionalities, where applicable

Pull-up resistors can be set for each digital input line using the PR command. The PR value updates the state of all pull-up resistors.

IO Sampling

The XBee ZB modules have the ability to monitor and sample the analog and digital IO lines. IO samples can be read locally or transmitted to a remote device to provide indication of the current IO line states. (Only API firmware devices can send remote IO sample data out their UART.)

There are three ways to obtain IO samples, either locally or remotely:

- Queried Sampling
- Periodic Sampling
- Change Detection Sampling.

IO sample data is formatted as shown in the table below

Table 8-04.

Bytes	Name	Description
1	Sample Sets	Number of sample sets in the packet. (Always set to 1.)
2	Digital Channel Mask	Indicates which digital IO lines have sampling enabled. Each bit corresponds to one digital IO line on the module. • bit 0 = AD0/DIO0 • bit 1 = AD1/DIO1 • bit 2 = AD2/DIO2 • bit 3 = AD3/DIO3 • bit 4 = DIO4 • bit 5 = ASSOC/DIO5 • bit 6 = RTS/DIO6 • bit 7 = CTS/GPIO7 • bit 8 = N/A • bit 9 = N/A • bit 10 = RSSI/DIO10 • bit 11 = PWM/DIO11 • bit 12 = CD/DIO12 For example, a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital IO.
1	Analog Channel Mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. • bit 0 = AD0/DIO0 • bit 1 = AD1/DIO1 • bit 2 = AD2/DIO2 • bit 3 = AD3/DIO3 • bit 7 = Supply Voltage
Variable	Sampled Data Set	A sample set consisting of 1 sample for each enabled ADC and/or DIO channel, which has voltage inputs of 1143.75 and 342.1875mV. If any digital IO lines are enabled, the first two bytes of the data set indicate the state of all enabled digital IO. Only digital channels that are enabled in the Digital Channel Mask bytes have any meaning in the sample set. If no digital IO are enabled on the device, these 2 bytes will be omitted. Following the digital IO data (if any), each enabled analog channel will return 2 bytes. The data starts with AIN0 and continues sequentially for each enabled analog input channel up to AIN3, and the supply voltage (if enabled) at the end.

The sampled data set will include 2 bytes of digital IO data only if one or more IO lines on the device are configured as digital IO. If no pins are configured as digital IO, these 2 bytes will be omitted.

The digital IO data is only relevant if the same bit is enabled in the digital IO mask as shown in the following figure:

Analog samples are returned as 10-bit values. The analog reading is scaled such that 0x0000 represents 0V, and 0x3FF = 1.2V. (The analog inputs on the module cannot read more than 1.2V.) Analog samples are returned in order starting with AIN0 and finishing with AIN3, and the supply voltage. Only enabled analog input channels return data as shown in the figure below.

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

AD(mV) = (A/D reading * 1200mV) / 1024

TThe reading in the sample frame represents voltage inputs of 1143.75 and 342.1875mV for AD0 and AD1 respectively.

Queried Sampling

The IS command can be sent to a device locally, or to a remote device using the API remote command frame (see Chapter 8 for details). When the IS command is sent, the receiving device samples all enabled digital IO and analog input channels and returns an IO sample. If IS is sent locally, the IO sample is sent out the uart. If the IS command was received as a remote command, the IO sample is sent over-the-air to the device that sent the IS command.

If the IS command is issued in AT firmware, the module returns a carriage return-delimited list containing the above-listed fields. The API firmware returns an AT command response packet with the IO data included in the command data portion of the response frame.

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

.

Table 8-05.

Example	Sample AT Response
0x01	[1 sample set]
0x0C0C	[Digital Inputs: DIO 2, 3, 10, 11 low]
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 IO Sampling

Periodic sampling allows an XBee / XBee-PRO module to take an IO sample and transmit it to a remote device at a periodic rate. The periodic sample rate is set by the IR command. If IR is set to 0, periodic sampling is disabled. For all other values of IR, data will be sampled after IR milliseconds have elapsed and transmitted to a remote device. The DH and DL commands determine the destination address of the IO samples. DH and DL can be set to 0 to transmit to the coordinator, or to the 64-bit address of the remote device (SH and SL). Only devices running API firmware can send IO data samples out their Uart. Devices running AT firmware will discard received IO data samples.

A sleepy end device will transmit periodic IO samples at the IR rate until the ST timer expires and the device can resume sleeping.

Change Detection Sampling

Modules can be configured to transmit a data sample immediately whenever a monitored digital IO pin changes state. The IC command is a bitmask that can be used to set which digital IO lines should be monitored for a state change. If one or more bits in IC is set, an IO sample will be transmitted as soon as a state change is observed in one of the monitored digital IO lines. Change detection samples are transmitted to the 64-bit address specified by DH and DL.

RSSI PWM

The XBee module features an RSSI/PWM pin (pin 6) that, if enabled, will adjust the PWM output to indicate the signal strength of the last received packet. The P0 (P-zero) command is used to enable the RSSI pulse width modulation (PWM) output on the pin. If P0 is set to 1, the RSSI/PWM pin will output a pulse width modulated signal where the frequency is adjusted based on the received signal strength of the last packet. Otherwise, for all other P0 settings, the pin can be used for general purpose IO.

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 RSSI of the last packet. The RSSI/PWM output will be enabled for a time based on the RP command. Each time an RF packet is received, the RSSI/PWM output is adjusted based on the RSSI of the new packet, and the RSSI timer is reset. If the RSSI timer expires, the RSSI/PWM pin is driven low. RP is measured in 100ms units and defaults to a value of 40 (4 seconds).

The RSSI PWM runs at 12MHz and has 2400 total counts (200us period).

RSSI (in dBm) is converted to PWM counts using the following equation:

PWM counts = (41 * RSSI_Unsigned) - 5928

IO Examples

Example 1: Configure the following IO settings on the XBee.

Configure AD1/DIO1 as a digital input with pullup resistor enabled

Configure AD2/DIO2 as an analog input

Configure DIO4 as a digital output, driving high.

To configure AD1/DIO1 as an input, issue the ATD1 command with a parameter of 3 ("ATD13"). To enable pull-up resistors on the same pin, the PR command should be issued with bit 3 set (i.e. ATPR8, ATPR1FFF, etc).

The ATD2 command should be issued with a parameter of 2 to enable the analog input ("ATD22"). Finally, DIO4 can be set as an output, driving high by issuing the ATD4 command with a parameter value of 5 ("ATD45").

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

Example 2: Calculate the PWM counts for a packet received with an RSSI of -84dBm.

RSSI = -84 = 0xAC = 172 decimal (unsigned)

PWM counts = (41 * 172) - 5928

PWM counts = 1124

With a total of 2400 counts, this yields an ON time of (1124 / 2400) = 46.8%

Example 3: Configure the RSSI/PWM pin to operate for 2 seconds after each received RF packet.

First, ensure the RSSI/PWM functionality is enabled by reading the P0 (P-zero) command. It should be set to 1 (default).

To configure the duration of the RSSI/PWM output, set the RP command. To achieve a 2 second PWM output, set RP to 0x14 (20 decimal, or 2 seconds) and apply changes (AC command).

After applying changes, all received RF data packets should set the RSSI timer for 2 seconds.

9. API Operation

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

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

API Frame Specifications

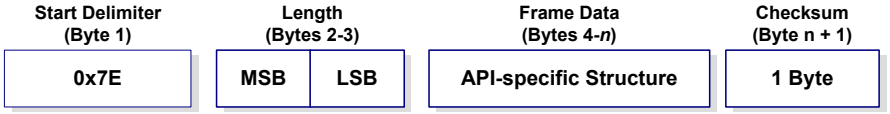
Two API modes are supported and both can be enabled using the AP (API Enable) command. Use the following AP parameter values to configure the module to operate in a particular mode:

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

API Operation (AP parameter = 1)

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

Figure 9-01. UART Data Frame Structure:



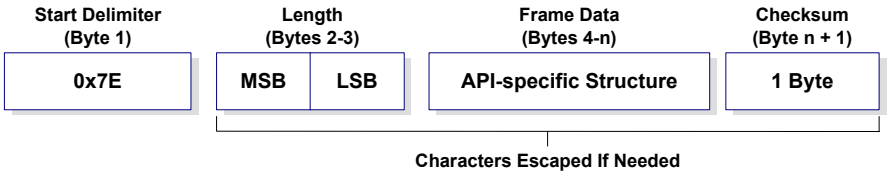
MSB = Most Significant Byte, LSB = Least Significant Byte

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

API Operation - with Escape Characters (AP parameter = 2)

When this API mode is enabled (AP = 2), the UART data frame structure is defined as follows:

Figure 9-02. UART Data Frame Structure - with escape control characters:



MSB = Most Significant Byte, LSB = Least Significant Byte

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

Data bytes that need to be escaped:

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

Example - Raw UART Data Frame (before escaping interfering bytes):
0x7E 0x00 0x02 0x23 0x11 0xCB

0x11 needs to be escaped which results in the following frame:
0x7E 0x00 0x02 0x23 0x7D 0x31 0xCB

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

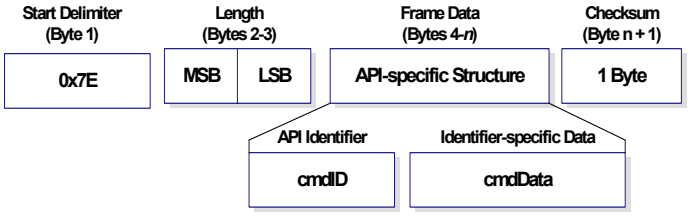
Length

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

Frame Data

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

Figure 9-03. UART Data Frame & API-specific Structure:



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

Table 9-06. API Frame Names and Values

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1

Checksum

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

To calculate: Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract the result from 0xFF.

To verify: Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

API Examples

Example: Create an API AT command frame to configure an XBee to allow joining (set NJ to 0xFF). The frame should look like:

0x7E 0x00 0x05 0x08 0x01 0x4E 0x4A 0xFF 5F

Where 0x0005 = length

0x08 = AT Command API frame type

0x01 = Frame ID (set to non-zero value)

0x4E4A = AT Command ('NJ')

0xFF = value to set command to

0x5F = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x4A + 0xFF)]

Example: Send an ND command to discover the devices in the PAN. The frame should look like:

0x7E 0x00 0x04 0x08 0x01 0x4E 0x44 0x64

Where 0x0004 = length

0x08 = AT Command API frame type

0x01 = Frame ID (set to non-zero value)

0x4E44 = AT command ('ND')

0x64 = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x44)]

Example: Send a remote command to the coordinator to set AD1/DIO1 as a digital input (D1=3) and apply changes to force the IO update. The API remote command frame should look like:

0x7E 0x00 0x10 0x17 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFE 0x02 0x44 0x31 0x03 0x70

Where

0x10 = length (16 bytes excluding checksum)

0x17 = Remote Command API frame type

0x01 = Frame ID

0x0000000000000000 = Coordinator's address (can be replaced with coordinator's actual 64-bit address if known)

0xFFFE = 16- bit Destination Address

0x02 = Apply Changes (Remote Command Options)

0x4431 = AT command ('D1')

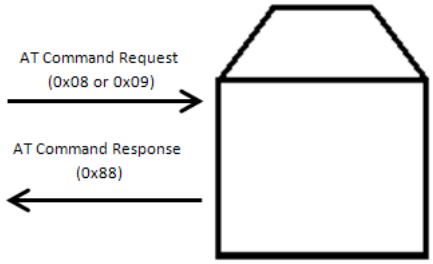
0x03 = Command Parameter (the parameter could also be sent as 0x0003 or 0x00000003)

0x70 = Checksum

API UART Exchanges

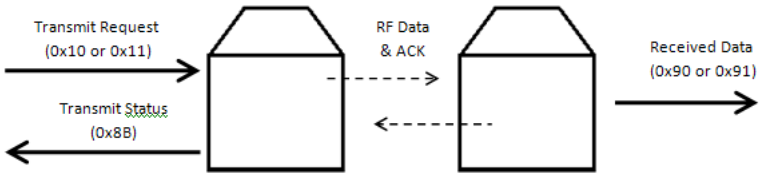
AT Commands

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



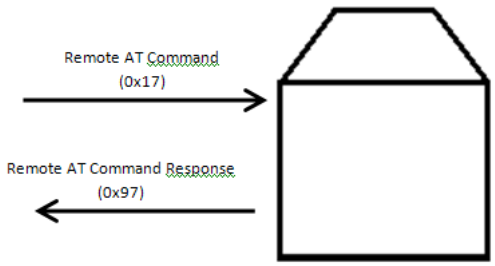
Transmitting and Receiving RF Data

The following image shows the API exchanges that take place at the UART 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 transmit request. If the packet cannot be delivered to the destination, the transmit status frame will indicate the cause of failure. The received data frame (0x90 or 0x91) is set by the AP command.



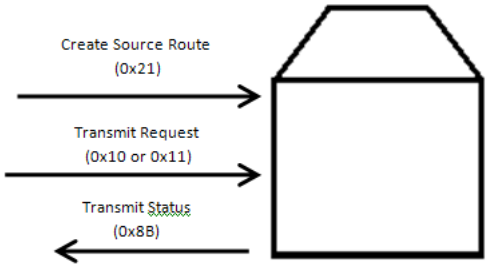
Remote AT Commands

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



Source Routing

The following image shows the API frame exchanges that take place at the UART when sending a source routed transmission.



Supporting the API

Applications that support the API should make provisions to deal with new API frames that may be introduced in future releases. For example, a section of code on a host microprocessor that handles received serial API frames (sent out the module's DOUT pin) might look like this:

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

    case RX_IO_SAMPLE_FRAME:
      //process IO sample frame
      break;

    case NODE_IDENTIFICATION_FRAME:
      //process node identification frame
      break;

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

API Frames

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

AT Command

Frame Type: 0x08
Used to query or set module parameters on the local device. This API command applies changes after executing the command. (Changes made to module parameters take effect once changes are applied.) The API example below illustrates an API frame when modifying the NJ parameter value of the module

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x04	
	Frame-specific Data	Frame Type	3	0x08	
		Frame ID	4	0x52 (R)	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		AT Command	5	0x4E (N)	Command Name - Two ASCII characters that identify the AT Command.
			6	0x4A (J)	
		Parameter Value (optional)			If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
	Checksum		9	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

The above example illustrates an AT command when querying an NJ value.

AT Command - Queue Parameter Value

Frame Type: 0x09
This API type allows module parameters to be queried or set. In contrast to the "AT Command" API type, new parameter values are queued and not applied until either the "AT Command" (0x08) API type or the AC (Apply Changes) command is issued. Register queries (reading parameter values) are returned immediately.

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x05	
	Frame-specific Data	Frame Type	3	0x09	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		AT Command	5	0x42 (B)	Command Name - Two ASCII characters that identify the AT Command.
			6	0x44 (D)	
		Parameter Value (ATBD7 = 115200 baud)		0x07	If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
	Checksum		9	0x68	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

ZigBee Transmit Request

Frame Type: 0x10

A Transmit Request API frame causes the module to send data as an RF packet to the specified destination.

The 64-bit destination address should be set to 0x000000000000FFFF for a broadcast transmission (to all devices). The coordinator can be addressed by either setting the 64-bit address to all 0x00s and the 16-bit address to 0xFFFE, OR by setting the 64-bit address to the coordinator's 64-bit address and the 16-bit address to 0x0000. For all other transmissions, setting the 16-bit address to the correct 16-bit address can help improve performance when

transmitting to multiple destinations. If a 16-bit address is not known, this field should be set to 0xFFFE (unknown). The Transmit Status frame (0x8B) will indicate the discovered 16-bit address, if successful.

The broadcast radius can be set from 0 up to NH. If set to 0, the value of NH specifies the broadcast radius (recommended). This parameter is only used for broadcast transmissions.

The maximum number of payload bytes can be read with the NP command.

Note: if source routing is used, the RF payload will be reduced by two bytes per intermediate hop in the source route. This exampe shows if escaping is disabled (AP=1).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Options	16	0x00	All other bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: The example above shows how to send a transmission to a module where escaping is disabled (AP=1) with destination address 0x0013A200 40014011, payload "TxData1B". If escaping is enabled (AP=2), the frame should look like:

0x7E 0x00 0x16 0x10 0x01 0x00 0x7D 0x33 0xA2 0x00 0x40 0x0A 0x01 0x27

0xFF 0xFE 0x00 0x00 0x54 0x78 0x44 0x61 0x74 0x61 0x30 0x41 0x7D 0x33

The checksum is calculated (on all non-escaped bytes) as [0xFF - (sum of all bytes from API frame type through data payload)].

Example: Send a transmission to the coordinator without specifying the coordinator's 64-bit address. The API transmit request frame should look like:

0x7E 0x00 0x16 0x10 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFE 0x00 0x00 0x54 0x78 032 0x43 0x6F 0x6F 0x72 0x64 0xFC

Where 0x16 = length (22 bytes excluding checksum)

0x10 = ZigBee Transmit Request API frame type

0x01 = Frame ID (set to non-zero value)

0x0000000000000000 = Coordinator's address (can be replaced with coordinator's actual 64-bit address if known

0xFFFE = 16-bit Destination Address

0x00 = Broadcast radius

0x00 = Options

0x547832436F6F7264 = Data payload ("Tx2Coord")

0xFC = Checksum

Explicit Addressing ZigBee Command Frame

Frame Type: 0x11

Allows ZigBee application layer fields (endpoint and cluster ID) to be specified for a data transmission.
Similar to the ZigBee Transmit Request, but also requires ZigBee application layer addressing fields to be specified (endpoints, cluster ID, profile ID). An Explicit Addressing Request API frame causes the module to send data as an RF packet to the specified destination, using the specified source and destination endpoints, cluster ID, and profile ID.

The 64-bit destination address should be set to 0x000000000000FFFF for a broadcast transmission (to all devices). The coordinator can be addressed by either setting the 64-bit address to all 0x00s and the 16-bit address to 0xFFFE, OR by setting the 64-bit address to the coordinator's 64-bit address and the 16-bit address to 0x0000. For all other transmissions, setting the 16-bit address to the correct 16-bit address can help improve performance when transmitting to multiple destinations. If a 16-bit address is not known, this field should be set to 0xFFFE (unknown). The Transmit Status frame (0x8B) will indicate the discovered 16-bit address, if successful.

The broadcast radius can be set from 0 up to NH. If set to 0, the value of NH specifies the broadcast radius (recommended). This parameter is only used for broadcast transmissions.

The maximum number of payload bytes can be read with the NP command. Note: if source routing is used, the RF payload will be reduced by two bytes per intermediate hop in the source route.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x1A	
	Frame-specific Data	Frame Type	3	0x11	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x00	
			7	0x00	
			8	0x00	
			9	0x00	
			10	0x00	
			11	0x00	
			12	0x00	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Source Endpoint	15	0xA0	Source endpoint for the transmission.
		Destination Endpoint	16	0xA1	Destination endpoint for the transmission.
		Cluster ID	17	0x15	Cluster ID used in the transmission
			18	0x54	
		Profile ID	19	0xC1	Profile ID used in the transmission
			20	0x05	
		Broadcast Radius	21	0x00	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value.
		Transmit Options	22	0x00	All bits must be set to 0.
		Data Payload	23	0x54	
			24	0x78	
			25	0x44	
			26	0x61	
			27	0x74	
			28	0x61	
	Checksum		29	0x3A	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Send a data transmission to the coordinator (64-bit address of 0x00s) using a source endpoint of 0xA0, destination endpoint 0xA1, cluster ID =0x1554, and profile ID 0xC105. Payload will be "TxData".

Remote AT Command Request

Frame Type: 0x17

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x10	
	Frame-specific Data	Frame Type	3	0x17	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x40	
			11	0x11	
			LSB 12	0x22	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Remote Command Options	15	0x02 (apply changes)	0x02 - Apply changes on remote. (If not set, AC command must be sent before changes will take effect.) All other bits must be set to 0.
		AT Command	16	0x42 (B)	Name of the command
			17	0x48 (H)	
		Command Parameter	18	0x01	If present, indicates the requested parameter value to set the given register. If no characters present, the register is queried.
	Checksum		18	0xF5	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Send a remote command to change the broadcast hops register on a remote device to 1 (broadcasts go to 1-hop neighbors only), and apply changes so the new configuration value immediately takes effect. In this example, the 64-bit address of the remote is 0x0013A200 40401122, and the destination 16-bit address is unknown.

Create Source Route

Frame Type: 0x21

This frame creates a source route in the module. A source route specifies the complete route a packet should traverse to get from source to destination. Source routing should be used with many-to-one routing for best results.

Note: Both the 64-bit and 16-bit destination addresses are required when creating a source route. These are obtained when a Route Record Indicator (0xA1) frame is received.

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x14	
Frame-specific Data	Frame Type	3	0x21	
	Frame ID	4	0x00	The Frame ID should always be set to 0.
	64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
		6	0x13	
		7	0xA2	
		8	0x00	
		9	0x40	
		10	0x40	
		11	0x11	
		LSB 12	0x22	
	16-bit Destination Network Address	MSB 13	0x33	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
		LSB 14	0x44	
	Route Command Options	15	0x00	Set to 0.
	Number of Addresses	16	0x03	The number of addresses in the source route (excluding source and destination).
	Address 1	17	0xEE	(neighbor of destination)
		18	0xFF	
	Address 2 (closer hop	19	0xCC	Address of intermediate hop
		20	0xDD	
	Address 3	21	0xAA	(neighbor of source)
		22	0xBB	
Checksum		23	0x01	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Intermediate hop addresses must be ordered starting with the neighbor of the destination, and working closer to the source. For example, suppose a route is found between A and E as shown below.

A ' B ' C ' D ' E

If device E has the 64-bit and 16-bit addresses of 0x0013A200 40401122 and 0x3344, and if devices B, C, and D have the following 16-bit addresses:

B = 0xAABB

C = 0xCCDD

D = 0xEEFF

The example above shows how to send the Create Source Route frame to establish a source route between A and E.

AT Command Response

Frame Type: 0x88
In response to an AT Command message, the module will send an AT Command Response message. Some commands will send back multiple frames (for example, the ND (Node Discover) command).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x05	
	Frame-specific Data	Frame Type	3	0x88	
		Frame ID	4	0x01	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
		AT Command	5	'B' = 0x42	Command Name - Two ASCII characters that identify the AT Command.
			6	'D' = 0x44	
		Command Status	7	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
		Command Data			Register data in binary format. If the register was set, then this field is not returned, as in this example.
	Checksum		8	0xF0	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Modem Status

Frame Type: (0x8A)
RF module status messages are sent from the module in response to specific conditions.

Example: The following API frame is returned when an API coordinator forms a network.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x02	
	Frame-specific Data	Frame Type	3	0x8A	
		Status	4	0x06	0 = Hardware reset 1 = Watchdog timer reset 2 =Joined network (routers and end devices) 3 =Disassociated 6 =Coordinator started
	Checksum		5	0x6F	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

ZigBee Transmit Status

Frame Type: 0x8B
When a TX Request is completed, the module sends a TX Status message. This message will indicate if the packet was transmitted successfully or if there was a failure.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x07	
	Frame-specific Data	Frame Type	3	0x8B	
		Frame ID	4	0x01	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
		16-bit address of destination0x7D	5	0x7D	16-bit Network Address the packet was delivered to (if success). If not success, this address matches the Destination Network Address that was provided in the Transmit Request Frame.
			6	0x84	
		Transmit Retry Count	7	0x00	The number of application transmission retries that took place.
		Delivery Status	8	0x00	0x00 = Success 0x02 = CCA Failure 0x15 = Invalid destination endpoint 0x21 = Network ACK Failure 0x22 = Not Joined to Network 0x23 = Self-addressed 0x24 = Address Not Found 0x25 = Route Not Found 0x74 = Data payload too large
		Discovery Status	9	0x01	0x00 = No Discovery Overhead 0x01 = Address Discovery 0x02 = Route Discovery 0x03 = Address and Route Discovery
	Checksum		10	0x71	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Suppose a unicast data transmission was sent to a destination device with a 16-bit address of 0x7D84. (The transmission could have been sent with the 16-bit address set to 0x7D84 or 0xFFFE.)

ZigBee Receive Packet

Frame Type: (0x90)

When the module receives an RF packet, it is sent out the UART using this message type.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
	Frame-specific Data		LSB 2	0x12	
		Frame Type	3	0x90	
		Frame ID	4	0x00	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Source Address	MSB 5	0x13	64-bit address of sender
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender
			LSB 13	0x84	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet
		Received Data	15	0x52	Received RF data
			16	0x78	
			17	0x44	
			18	0x61	
			19	0x74	
			20	0x61	
	Checksum		21	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Suppose a device with a 64-bit address of 0x0013A200 40522BAA, and 16-bit address 0x7D84 sends a unicast data transmission to a remote device with payload "RxData". If AO=0 on the receiving device, it would send the following frame out its UART.

ZigBee Explicit Rx Indicator

Frame Type:0x91

When the modem receives a ZigBee RF packet it is sent out the UART using this message type (when AO=1).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
	Frame-specific Data		LSB 2	0x18	
		Frame Type	3	0x91	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender.
			LSB 13	0x84	
		Source Endpoint	14	0xE0	Endpoint of the source that initiated the transmission
		Destination Endpoint	15	0xE0	Endpoint of the destination the message is addressed to.
		Cluster ID	16	0x22	Cluster ID the packet was addressed to.
			17	0x11	
		Profile ID	18	0xC1	Profile ID the packet was addressed to.
			19	0x05	
		Receive Options	20	0x02	0x01 – Packet Acknowledged 0x02 – Packet was a broadcast packet
		Received Data	21	0x52	Received RF data
			22	0x78	
			23	0x44	
			24	0x61	
			25	0x74	
			26	0x61	
	Checksum		27	0x52	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Suppose a device with a 64-bit address of 0x0013A200 40522BAA, and 16-bit address 0x7D84 sends a broadcast data transmission to a remote device with payload "RxData". Suppose the transmission was sent with source and destination endpoints of 0xE0, cluster ID=0x2211, and profile ID=0xC105. If AO=1 on the receiving device, it would send the following frame out its UART.

ZigBee IO Data Sample Rx Indicator

Frame Type: 0x92

When the module receives an IO sample frame from a remote device, it sends the sample out the UART using this frame type (when AO=0). Only modules running API firmware will send IO samples out the UART.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
	Frame-specific Data		LSB 2	0x14	
		Frame Type	3	0x92	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender.
			LSB 13	0x84	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was abroadcast packet
		Number of Samples	15	0x01	Number of sample sets included in the payload. (Always set to 1)
		Digital Channel Mask*	16	0x00	Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any).
			17	0x1C	
		Analog Channel Mask**	18	0x02	Bitmask field that indicates which analog IO lines on the remote have sampling enabled (if any).
		Digital Samples (if included)	19	0x00	If the sample set includes any digital IO lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital IO lines. DIO lines that do not have sampling enabled return 0. Bits in these 2 bytes map the same as they do in the Digital Channels Mask field.
			20	0x14	
		Analog Sample	21	0x02	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 AD0/DIO0 to AD3/DIO3, to the supply voltage.
			22	0x25	
	Checksum		23	0xF5	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Example: Suppose an IO sample is received with analog and digital IO, from a remote with a 64-bit address of 0x0013A200 40522BAA and a 16-bit address of 0x7D84. If pin AD1/DIO1 is enabled as an analog input, AD2/DIO2 and DIO4 are enabled as a digital inputs (currently high), and AD3/DIO3 is enabled as a digital output (low) the IO sample would look like:

XBee Sensor Read Indicator

Frame Type: 0x94

When the module receives a sensor sample (from a Digi 1-wire sensor adapter), it is sent out the UART using this message type (when AO=0).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
	Frame-specific Data		LSB 2	0x17	
		Frame Type	3	0x94	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0xDD	16-bit address of sender.
			LSB 13	0x6C	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was abroadcast packet
		1-Wire Sensors	15	0x03	0x01 = A/D Sensor Read 0x02 = Temperature Sensor Read 0x60 = Water present (module CD pin low)
		A/D Values	16	0x00	Indicates a two-byte value for each of four A/D sensors (A, B, C, D) Set to 0xFFFFFFFFFFFFFFFF if no A/Ds are found.
			17	0x02	
			18	0x00	
			19	0xCE	
			20	0x00	
			21	0xEA	
			22	0x00	
			23	0x52	
		Temperature Read	24	0x01	Indicates the two-byte value read from a digital thermometer if present. Set to 0xFFFF if not found.
			25	0x6A	
	Checksum		26	0x8B	0xFF - the 0x8 bit sum of bytes from offset 3 to this byte.

Example: Suppose a 1-wire sensor sample is received from a device with a 64-bit address of 0x0013A200 40522BAA and a 16-bit address of 0xDD6C. If the sensor sample was taken from a 1-wire humidity sensor, the API frame could look like this (if AO=0):

For convenience, let's label the A/D and temperature readings as AD0, AD1, AD2, AD3, and T. Using the data in this example:

AD0 = 0x0002

AD1 = 0x00CE

AD2 = 0x00EA

AD3 = 0x0052

T = 0x016A

To convert these to temperature and humidity values, the following equations should be used.

Temperature (°C) = (T / 16), for T < 2048

= - (T & 0x7FF) / 16, for T >= 2048

Vsupply = (AD2 * 5.1) / 255

Voutput = (AD3 * 5.1) / 255

Relative Humidity = ((Voutput / Vsupply) - 0.16) / (0.0062)

True Humidity = Relative Humidity / (1.0546 - (0.00216 * Temperature (°C)))

Looking at the sample data, we have:

Vsupply = (234 * 5.1 / 255) = 4.68

Voutput = (82 * 5.1 / 255) = 1.64

Temperature = (362 / 16) = 22.625°C

Relative H = (161.2903 * ((1.64/4.68) - 0.16)) = 161.2903 * (0.19043) = 30.71%

True H = (30.71 / (1.0546 - (0.00216 * 22.625))) = (30.71 / 1.00573) = 30.54%

Node Identification Indicator

Frame Type: 0x95

This frame is received when a module transmits a node identification message to identify itself (when AO=0). The data portion of this frame is similar to a network discovery response frame (see ND command).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
	Frame-specific Data		LSB 2	0x20	
		Frame Type	3	0x95	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender.
			LSB 13	0x84	
		Receive Options	14	0x02	0x01 - Packet Acknowledged 0x02 - Packet was abroadcast packet
		Source 16-bit address	15	0x7D	Set to the 16-bit network address of the remote. Set to 0xFFFE if unknown.
			16	0x84	
		64-bit Network address	17	0x00	Indicates the 64-bit address of the remote module that transmitted the node identification frame.
			18	0x13	
			19	0xA2	
			20	0x00	
			21	0x40	
			22	0x52	
			23	0x2B	
			24	0xAA	
		NI String	25	0x20	Node identifier string on the remote device. The NI-String is terminated with a NULL byte (0x00).
			26	0x00	
		Parent 16-bit address	27	0xFF	Indicates the 16-bit address of the remote's parent or 0xFFFE if the remote has no parent.
			28	0xFE	
		Device Type	29	0x01	0 = Coordinator 1 = Router 2 = End Device
		Source Event	30	0x01	1 = Frame sent by node identification pushbutton event (see D0 command) 2 = Frame sent after joining event occurred (see JN command). 3 = Frame sent after power cycle event occurred (see JN command).
		Digi Profile ID	31	0xC1	Set to Digi's application profile ID.
			32	0x05	
		Manufacturer ID	33	0x10	Set to Digi's Manufacturer ID.
			34	0x1E	
	Checksum		35	0x1B	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: If the commissioning push button is pressed on a remote router device with 64-bit address 0x0013A200 40522BAA, 16-bit address 0x7D84, and default NI string, the following node identification indicator would be received.

Remote Command Response

Frame Type: 0x97

If a module receives a remote command response RF data frame in response to a Remote AT Command Request, the module will send a Remote AT Command Response message out the UART. Some commands may send back multiple frames--for example, Node Discover (ND) command.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x13	
	Frame-specific Data	Frame Type	3	0x97	
		Frame ID	4	0x55	This is the same value passed in to the request. .
		64-bit Source (remote) Address	MSB 5	0x00	The address of the remote radio returning this response.
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x52	
			11	0x2B	
			LSB 12	0xAA	
		16-bit Source (remote) Address	MSB 13	0x7D	Set to the 16-bit network address of the remote. Set to 0xFFFE if unknown.
			LSB 14	0x84	
		AT Commands	15	0x53	Name of the command
			16	0x4C	
		Command Status	17	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
		Command Data	18	0x40	Register data in binary format. If the register was set, then this field is not returned.
			19	0x52	
			20	0x2B	
			21	0xAA	
	Checksum		22	0xF0	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: If a remote command is sent to a remote device with 64-bit address 0x0013A200 40522BAA and 16-bit address 0x7D84 to query the SL command, and if the frame ID=0x55, the response would look like:

Over-the-Air Firmware Update Status

Frame Type: 0xA0

The Over-the-Air Firmware Update Status frame provides a status indication of a firmware update transmission attempt. If a query command (0x01 0x51) is sent to a target with a 64-bit address of 0x0013A200 40522BAA through an updater with 64-bit address 0x0013A200403E0750 and 16-bit address 0x0000, the following is the expected response.

API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0xA0	
		64-bit Source (remote) Address	MSB 4	0x00	The address of the remote radio returning this response.
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x3E	
			10	0x07	
			LSB 11	0x50	
		16-bit Destination Address	12	0x00	16-bit address of the updater device
			13	0x00	
		Receive Options	14	0x01	0x01 - Packet Acknowledged. 0x02 - Packet was a broadcast.
		Bootloader Message Type	15	0x52	0x06 - ACK 0x15 - NACK 0x40 - No Mac ACK 0x51 - Query (received if the bootloader is not active on the target) 0x52 - Query Response
		Block Number	16	0x00	Block number used in the update request. Set to 0 if not applicable.
		64-bit Target Address	17	0x00	64-bit Address of remote device that is being updated (target).
			18	0x13	
			19	0xA2	
			20	0x00	
			21	0x40	
			22	0x52	
			23	0x2B	
			24	0xAA	
	Checksum		25	0x66	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

If a query request returns a 0x15 (NACK) status, the target is likely waiting for a firmware update image. If no messages are sent to it for about 75 seconds, the target will timeout and accept new query messages.

If a query returns a 0x51 (QUERY) status, then the target's bootloader is not active and will not respond to query messages.

Route Record Indicator

Frame Type: 0xA1

The route record indicator is received whenever a device sends a ZigBee route record command. This is used with many-to-one routing to create source routes for devices in a network.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x13	
	Frame-specific Data	Frame Type	3	0xA1	
		64-bit Source Address	MSB 4	0x00	64-bit address of the device that initiated the route record.
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x40	
			10	0x11	
			LSB 11	0x22	
		Source (updater) 16-bit Address	12	0x33	16-bit address of the device that initiated the route record.
			13	0x44	
		Receive Options	14	0x01	0x01 - Packet Acknowledged. 0x02 - Packet was a broadcast.
		Number of Addresses	15	0x03	The number of addresses in the source route (excluding source and destination).
		Address 1	16	0xEE	(neighbor of destination)
			17	0xFF	
		Address 2 (closer hop	18	0xCC	Address of intermediate hop
			19	0xDD	
		Address n (neighbor of source)	20	0xAA	Two bytes per 16-bit address.
			21	0xBB	
	Checksum		22	0x80	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Suppose device E sends a route record that traverses multiple hops en route to data collector device A as shown below.

A B C D E

If device E has the 64-bit and 16-bit addresses of 0x0013A200 40401122 and 0x3344, and if devices B, C, and D have the following 16-bit addresses:

B = 0xAABB

C = 0xCCDD

D = 0xEEFF

The data collector will send the following API frame out its UART.

Sending ZigBee Device Objects (ZDO) Commands with the API

ZigBee Device Objects (ZDOs) are defined in the ZigBee Specification as part of the ZigBee Device Profile. These objects provide functionality to manage and map out the ZigBee network and to discover services on ZigBee devices. ZDOs are typically required when developing a ZigBee product that will interoperate in a public profile such as home automation or smart energy, or when communicating with ZigBee devices from other vendors. The ZDO can also be used to perform several management functions such as frequency agility (energy detect and channel changes - Mgmt Network Update Request), discovering routes (Mgmt Routing Request) and neighbors (Mgmt LQI Request), and managing device connectivity (Mgmt Leave and Permit Join Request).

The following table shows some of the more prominent ZDOs with their respective cluster identifier. Each ZDO command has a defined payload. See the "ZigBee Device Profile" section of the ZigBee Specification for details

ZDO Command	Cluster ID
Network Address Request	0x0000
IEEE Address Request	0x0001
Node Descriptor Request	0x0002
Simple Descriptor Request	0x0004
Active Endpoints Request	0x0005
Match Descriptor Request	0x0006
Mgmt LQI Request	0x0031
Mgmt Routing Request	0x0032
Mgmt Leave Request	0x0034
Mgmt Permit Joining Request	0x0036
Mgmt Network Update Request	0x0038

The Explicit Transmit API frame (0x11) is used to send ZigBee Device Objects commands to devices in the network. Sending ZDO commands with the Explicit Transmit API frame requires some formatting of the data payload field.

When sending a ZDO command with the API, all multiple byte values in the ZDO command (API payload) (i.e. u16, u32, 64-bit addresses) must be sent in little endian byte order for the command to be executed correctly on a remote device.

For an API XBee to receive ZDO responses, the AO command must be set to 1 to enable the explicit receive API frame.

The following table shows how the Explicit API frame can be used to send an "Active Endpoints" request to discover the active endpoints on a device with a 16-bit address of 0x1234.

	Frame Fields			Offset	Example	Description
API Packet	Start Delimiter			0	0x7E	
	Length			MSB 1	0x00	Number of bytes between the length and the checksum
				LSB 2	0x17	
	Frame-specific Data	Frame Type		3	0x11	
		Frame ID		4	0x01	Identifies the UART data frame for the host to correlate with a subsequent transmit status. If set to 0, no transmit status frame will be sent out the UART.
		64-bit Destination Address		MSB 5	0x00	64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
				6	0x00	
				7	0x00	
				8	0x00	
				9	0x00	
				10	0x00	
				11	0x00	
				12	0x00	
		16-bit Destination Network Address		MSB 13	0xFF	16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
				LSB 14	0xFE	
		Source Endpoint		15	0x00	Set to 0x00 for ZDO transmissions (endpoint 0 is the ZDO endpoint).
		Destination Endpoint		16	0x00	Set to 0x00 for ZDO transmissions (endpoint 0 is the ZDO endpoint).
		Cluster ID		MSB 17	0x00	Set to the cluster ID that corresponds to the ZDO command being sent. 0x0005 = Active Endpoints Request
				LSB 18	0x00	
		Profile ID		MSB 19	0x05	Set to 0x0000 for ZDO transmissions (Profile ID 0x0000 is the ZigBee Device Profile that supports ZDOs).
				LSB 20	0x00	
		Broadcast Radius		21	0x00	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value..
		Transmit Options		22	0x00	All bits must be set to 0.
		Data Payload	Transaction Sequence Number	23	0x01	The required payload for a ZDO command. All multi-byte ZDO parameter values (u16, u32, 64-bit address) must be sent in little endian byte order. The Active Endpoints Request includes the following payload: [16-bit NwkAddrOfInterest] Note the 16-bit address in the API example (0x1234) is sent in little endian byte order (0x3412).
			ZDO Payload	24	0x34	
				25	0x12	
	Checksum			26	0xA6	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

Sending ZigBee Cluster Library (ZCL) Commands with the API

The ZigBee Cluster Library defines a set of attributes and commands (clusters) that can be supported in multiple ZigBee profiles. The ZCL commands are typically required when developing a ZigBee product that will interoperate in a public profile such as home automation or smart energy, or when communicating with ZigBee devices from other vendors. Applications that are not designed for a public profile or for interoperability applications can skip this section.

The following table shows some prominent clusters with their respective attributes and commands.

Cluster (Cluster ID)	Attributes (Attribute ID)	Cluster ID
Basic (0x0000)	Application Version (0x0001) Hardware Version (0x0003) Model Identifier (0x0005)	-Reset to defaults (0x00)
Identify (0x0003)	Identify Time (0x0000)	Identify (0x00) Identify Query (0x01)
Time (0x000A)	Time (0x0000) Time Status (0x0001) Time Zone (0x0002)	
Thermostat (0x0201)	Local Temperature (0x0000) Occupancy (0x0002)	-Setpoint raise / lower (0x00)

The ZCL defines a number of profile-wide commands that can be supported on any profile, also known as general commands. These commands include the following.

Command (Command ID)	Description
Read Attributes (0x00)	Used to read one or more attributes on a remote device.
Read Attributes Response (0x01)	Generated in response to a read attributes command.
Write Attributes (0x02)	Used to change one or more attributes on a remote device.
Write Attributes Response (0x04)	Sent in response to a write attributes command.
Configure Reporting (0x06)	Used to configure a device to automatically report on the values of one or more of its attributes.
Report Attributes (0x0A)	Used to report attributes when report conditions have been satisfied.
Discover Attributes (0x0C)	Used to discover the attribute identifiers on a remote device.
Discover Attributes Response (0x0D)	Sent in response to a discover attributes command.

The Explicit Transmit API frame (0x11) is used to send ZCL commands to devices in the network. Sending ZCL commands with the Explicit Transmit API frame requires some formatting of the data payload field.

When sending a ZCL command with the API, all multiple byte values in the ZCL command (API Payload) (i.e. u16, u32, 64-bit addresses) must be sent in little endian byte order for the command to be executed correctly on a remote device.

Note: When sending ZCL commands, the AO command should be set to 1 to enable the explicit receive API frame. This will provide indication of the source 64- and 16-bit addresses, cluster ID, profile ID, and endpoint information for each received packet. This information is required to properly decode received data.

The following table shows how the Explicit API frame can be used to read the hardware version attribute from a device with a 64-bit address of 0x0013A200 40401234 (unknown 16-bit address). This example uses arbitrary source and destination endpoints. Recall the hardware version attribute (attribute ID 0x0003) is part of the basic cluster (cluster ID 0x0000). The Read Attribute general command ID is 0x00.

	Frame Fields				Offset	Example	Description
API Packet	Start Delimiter				0	0x7E	
	Length				MSB 1	0x00	Number of bytes between the length and the checksum
					LSB 2	0x19	
	Frame-specific Data	Frame Type			3	0x11	
		Frame ID			4	0x01	Identifies the UART data frame for the host to correlate with a subsequent transmit status. If set to 0, no transmit status frame will be sent out the UART.
		64-bit Destination Address			MSB 5	0x00	64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
					6	0x13	
					7	0xA2	
					8	0x00	
					9	0x40	
					10	0x40	
					11	0x12	
					12	0x34	
		16-bit Destination Network Address			MSB 13	0xFF	16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
					LSB 14	0xFE	
		Source Endpoint			15	0x41	Set to the source endpoint on the sending device. (0x41 arbitrarily selected).
		Destination Endpoint			16	0x42	Set to the destination endpoint on the remote device. (0x42 arbitrarily selected)
		Cluster ID			MSB 17	0x00	Set to the cluster ID that corresponds to the ZCL command being sent. 0x0000 = Basic Cluster
					LSB 18	0x00	
		Profile ID			MSB 19	0x00	Set to the profile ID supported on the device. (0xD123 arbitrarily selected).
					LSB 20	0xD1	
		Broadcast Radius			21	0x00	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value..
		Transmit Options			22	0x00	All bits must be set to 0.
		Data Payload		Frame Control	23	0x00	Bitfield that defines the command type and other relevant information in the ZCL command. See the ZCL specification for details.
			ZCL Frame Header	Transaction Sequence Number	24	0x01	A sequence number used to correlate a ZCL command with a ZCL response. (The hardware version response will include this byte as a sequence number in the response.) The value 0x01 was arbitrarily selected.
				Command ID	25	0x00	Since the frame control "frame type" bits are 00, this byte specifies a general command. Command ID 0x00 is a Read Attributes command.
			ZCL Payload	Attribute ID	26	0x03	The payload for a "Read Attributes" command is a list of Attribute Identifiers that are being read. Note the 16-bit Attribute ID (0x0003) is sent in little endian byte order (0x0300). All multi-byte ZCL header and payload values must be sent in little endian byte order.
					27	0x00	
	Checksum				28	0xFA	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

In the above example, the Frame Control was constructed as follows:

Name	Bits	Example Value Description
Frame Type	0-1	00 - Command acts across the entire profile
Manufacturer Specific	2	0 - The manufacturer code field is omitted from the ZCL Frame Header.
Direction	3	0 - The command is being sent from the client side to the server side.
Disable Default Response	4	0 - Default response not disabled
Resvered	5-7	Set to 0.

See the ZigBee Cluster Library specification for details.

Sending Public Profile Commands with the API

Commands in public profiles such as Smart Energy and Home Automation can be sent with the XBee API using the Explicit Transmit API frame (0x11). Sending public profile commands with the Explicit Transmit API frame requires some formatting of the data payload field. Most of the public profile commands fit into the ZigBee Cluster Library (ZCL) architecture as described in the previous section.

The following table shows how the Explicit API frame can be used to send a demand response and load control message (cluster ID 0x701) in the smart energy profile (profile ID 0x0109) in the revision 14 Smart Energy specification. The message will be a "Load Control Event" (command ID 0x00) and will be sent to a device with 64-bit address of 0x0013A200 40401234 with a 16-bit address of 0x5678. The event will start a load control event for water heaters and smart appliances, for a duration of 1 minute, starting immediately.

Note: When sending public profile commands, the AO command should be set to 1 to enable the explicit receive API frame. This will provide indication of the source 64- and 16-bit addresses, cluster ID, profile ID, and endpoint information for each received packet. This information is required to properly decode received data.

	Frame Fields				Offset	Example	Description
API Packet	Start Delimiter				0	0x7E	
	Length				MSB 1	0x00	Number of bytes between the length and the checksum
					LSB 2	0x19	
	Frame-specific Data	Frame Type			3	0x11	
		Frame ID			4	0x01	Identifies the UART data frame for the host to correlate with a subsequent transmit status. If set to 0, no transmit status frame will be sent out the UART.
		64-bit Destination Address			MSB 5	0x00	64-bit address of the destination device (big endian byte order). For unicast transmissions, set to the 64-bit address of the destination device, or to 0x0000000000000000 to send a unicast to the coordinator. Set to 0x000000000000FFFF for broadcast.
					6	0x13	
					7	0xA2	
					8	0x00	
					9	0x40	
					10	0x40	
					11	0x12	
					12	0x34	
		16-bit Destination Network Address			MSB 13	0x56	16-bit address of the destination device (big endian byte order). Set to 0xFFFE for broadcast, or if the 16-bit address is unknown.
					LSB 14	0x78	
		Source Endpoint			15	0x41	Set to the source endpoint on the sending device. (0x41 arbitrarily selected).
		Destination Endpoint			16	0x42	Set to the destination endpoint on the remote device. (0x42 arbitrarily selected)
		Cluster ID			MSB 17	0x07	Set to the cluster ID that corresponds to the ZCL command being sent. 0x0701 = Demand response and load control cluster ID
					LSB 18	0x01	
		Profile ID			MSB 19	0x01	Set to the profile ID supported on the device. 0x0109 = Smart Energy profile ID.
					LSB 20	0x09	
		Broadcast Radius			21	0x00	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value..
		Transmit Options			22	0x00	All bits must be set to 0.
		Data Payload	ZCL Frame Header	Frame Control	23	0x09	Bitfield that defines the command type and other relevant information in the ZCL command. See the ZCL specification for details.
				Transaction Sequence Number	24	0x01	A sequence number used to correlate a ZCL command with a ZCL response. (The hardware version response will include this byte as a sequence number in the response.) The value 0x01 was arbitrarily selected.
					25	0x00	Since the frame control "frame type" bits are 01, this byte specifies a cluster-specific command. Command ID 0x00 in the Demand Response and Load Control cluster is a Load Control Event command. (See Smart Energy specification.)

Frame Fields				Offset	Example	Description
			Issuer Event ID	26	0x78	4-byte unique identifier. Note the 4-byte ID is sent in little endian byte order (0x78563412). The event ID in this example (0x12345678) was arbitrarily selected.
				27	0x56	
				28	0x34	
				29	0x12	
			Device Class	30	0x14	to apply the load control event. A bit value of 0x0014 enables smart appliances and water heaters. Note the 2-byte bit field value is sent in little endian byte order.
				31	0x00	
			Utility Enrollment Group	32	0x00	Used to identify sub-groups of devices in the device-class. 0x00 addresses all groups.
			Start Time	33	0x00	UTC timestamp representing when the event should start. A value of 0x00000000 indicates "now".
				34	0x00	
				35	0x00	
				36	0x00	
			Duration in Minutes	37	0x01	This 2-byte value must be sent in little endian byte order.
				38	0x00	
		ZCL Payload - Load Control Event Data	Criticality Level	39	0x04	Indicates the criticality level of the event. In this example, the level is "voluntary".
			Cooling Temperature	40	0xFF	Requested offset to apply to the normal cooling set point. A value of 0xFF indicates the temperature offset value is not used.
			Heating Temperature Offset	41	0xFF	Requested offset to apply to the normal heating set point. A value of 0xFF indicates the temperature offset value is not used.
			Cooling Temperature Set Point	42	0x00	Requested cooling set point in 0.01 degrees Celsius. A value of 0x8000 means the set point field is not used in this event. Note the 0x80000 is sent in little endian byte order.
				43	0x80	
			Heating Temperature Set Point	44	0x00	Requested heating set point in 0.01 degrees Celsius. A value of 0x8000 means the set point field is not used in this event. Note the 0x80000 is sent in little endian byte order.
				45	0x80	
			Average Load Adjustment Percentage	46	0x80	Maximum energy usage limit. A value of 0x80 indicates the field is not used.
			Duty Cycle	47	0xFF	Defines the maximum "On" duty cycle. A value of 0xFF indicates the duty cycle is not used in this event.
			Duty Cycle Event Control	48	0x00	A bitmap describing event options.
Checksum				49	0x5B	0xFF minus the 8 bit sum of bytes from offset 3 to this byte.

In the above example, the Frame Control was constructed as follows:

Name	Bits	Example Value Description
Frame Type	0-1	01- Command acts across the entire profile
Manufacturer Specific	2	0 - The manufacturer code field is omitted from the ZCL Frame Header.

Name	Bits	Example Value Description
Direction	3	0 - The command is being sent from the client side to the server side.
Disable Default Response	4	0 - Default response not disabled
Resvered	5-7	Set to 0.

10. XBee Command Reference Tables

Addressing

Table 10-07. Addressing Commands)

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
DH	**Destination Address High**.Set/Get the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the 64-bit destination address for data transmission. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0
DL	**Destination Address Low**. Set/Get the lower 32 bits of the 64-bit destination address. When combined with DH, it defines the 64-bit destination address for data transmissions. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0xFFFF(Coordinator) 0 (Router/End Device)
MY	**16-bit Network Address**. Read the 16-bit network address of the module. A value of 0xFFFE means the module has not joined a ZigBee network	CRE	0 - 0xFFFE [read-only]	0xFFFE
MP	**16-bit Parent Network Address**. Read the 16-bit network address of the module's parent. A value of 0xFFFE means the module does not have a parent.	E	0 - 0xFFFE [read-only]	0xFFFE
NC	**Number of Remaining Children**. Read the number of end device children that can join the device. If NC returns 0, then the device cannot allow any more end device children to join.	CR	0 - MAX_CHILDREN (maximum varies)	read-only
SH	**Serial Number High**. Read the high 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
SL	**Serial Number Low**. Read the low 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
NI	**Node Identifier.** Stores a string identifier. The register only accepts printable ASCII data. In AT Command Mode, a string can not start with a space. A carriage return ends the command. Command will automatically end when maximum bytes for the string have been entered. This string is returned as part of the ND (Node Discover) command. This identifier is also used with the DN (Destination Node) command.	CRE	20-Byte printable ASCII string	ASCII space character (0x20)
DD	**Device Type Identifier**. Stores a device type value. This value can be used to differentiate different XBee-based devices. Digi reserves the range 0 - 0xFFFFFF.	CRE	0 - 0xFFFFFFFF	0x30000
SE	**Source Endpoint**. Set/read the ZigBee application layer source endpoint value. This value will be used as the source endpoint for all data transmissions. SE is only supported in AT firmware.The default value 0xE8 (Data endpoint) is the Digi data endpoint	CRE	0 - 0xFF	0xE8
DE	**Destination Endpoint**. Set/read Zigbee application layer destination ID value. This value will be used as the destination endpoint all data transmissions. DE is only supported in AT firmware.The default value (0xE8) is the Digi data endpoint.	CRE	0 - 0xFF	0xE8
CI	**Cluster Identifier**. Set/read Zigbee application layer cluster ID value. This value will be used as the cluster ID for all data transmissions. CI is only supported in AT firmware.The default value0x11 (Transparent data cluster ID).	CRE	0 - 0xFFFF	0x11
NP	**Maximum RF Payload Bytes**. This value returns the maximum number of RF payload bytes that can be sent in a unicast transmission. If many-to-one and source routing are used (AR < 0xFF), or if APS security is used on a transmission, the maximum payload size is reduced further. Note: NP returns a hexadecimal value. (i.e. if NP returns 0x54, this is equivalent to 84 bytes)	CRE	0 - 0xFFFF	[read-only]

Node types that support the command: C=Coordinator, R=Router, E=End Device

Networking

Table 10-08. Networking Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
CH	**Operating Channel**. Read the channel number used for transmitting and receiving between RF modules. Uses 802.15.4 channel numbers. A value of 0 means the device has not joined a PAN and is not operating on any channel.	CRE	0, 0x0B - 0x1A (XBee) 0, 0x0B - 0x18 (XBee-PRO)	[read-only]
ID	**Extended PAN ID**. Set/read the 64-bit extended PAN ID. If set to 0, the coordinator will select a random extended PAN ID, and the router / end device will join any extended PAN ID. Changes to ID should be written to non-volatile memory using the WR command to preserve the ID setting if a power cycle occurs.	CRE	0 - 0xFFFFFFFFFFFFFFFF	0

Table 10-08. Networking Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
OP	**Operating Extended PAN ID**. Read the 64-bit extended PAN ID. The OP value reflects the operating extended PAN ID that the module is running on. If ID > 0, OP will equal ID.	CRE	0x01 - 0xFFFFFFFFFFFFFFFF	[read-only]
NH	**Maximum Unicast Hops**. Set / read the maximum hops limit. This limit sets the maximum broadcast hops value (BH) and determines the unicast timeout. The timeout is computed as (50 * NH) + 100 ms. The default unicast timeout of 1.6 seconds (NH=0x1E) is enough time for data and the acknowledgment to traverse about 8 hops.	CRE	0 - 0xFF	0x1E
BH	**Broadcast Hops.** Set/Read the maximum number of hops for each broadcast data transmission. Setting this to 0 will use the maximum number of hops.	CRE	0 - 0x20	0
OI	**Operating 16-bit PAN ID**. Read the 16-bit PAN ID. The OI value reflects the actual 16-bit PAN ID the module is running on. .	CRE	0 - 0xFFFF	[read-only]
NT	**Node Discovery Timeout**. Set/Read the node discovery timeout. When the network discovery (ND) command is issued, the NT value is included in the transmission to provide all remote devices with a response timeout. Remote devices wait a random time, less than NT, before sending their response.	CRE	0x20 - 0xFF [x 100 msec]	0x3C (60d)
NO	**Network Discovery options.** Set/Read the options value for the network discovery command. The options bitfield value can change the behavior of the ND (network discovery) command and/or change what optional values are returned in any received ND responses or API node identification frames. Options include: 0x01 = Append DD value (to ND responses or API node identification frames) 002 = Local device sends ND response frame when ND is issued.	CRE	0 - 0x03 [bitfield]	0
SC	**Scan Channels**. Set/Read the list of channels to scan. ***Coordinator*** - Bit field list of channels to choose from prior to starting network. ***Router/End Device*** - Bit field list of channels that will be scanned to find a Coordinator/ Router to join. Changes to SC should be written using WR command to preserve the SC setting if a power cycle occurs. Bit (Channel): 0 (0x0B) 4 (0x0F) 8 (0x13) 12 (0x17) 1 (0x0C) 5 (0x10) 9 (0x14) 13 (0x18) 2 (0x0D) 6 (0x11) 10 (0x15) 14 (0x19) 3 (0x0E) 7 (0x12) 11 (0x16) 15 (0x1A)	CRE	**XBee** 1 - 0xFFFF [bitfield] **XBee-PRO** 1 - 0x1FFE [bitfield] (bits 14, 15 not allowed)	0x3FFF.
SD	**Scan Duration**. Set/Read the scan duration exponent. Changes to SD should be written using WR command. ***Coordinator*** - Duration of the Active and Energy Scans (on each channel) that are used to determine an acceptable channel and Pan ID for the Coordinator to startup on. ***Router / End Device*** - Duration of Active Scan (on each channel) used to locate an available Coordinator / Router to join during Association. Scan Time is measured as:(# Channels to Scan) * (2 ^ SD) * 15.36ms - The number of channels to scan is determined by the SC parameter. The XBee can scan up to 16 channels (SC = 0xFFFF). Sample Scan Duration times (13 channel scan): If SD = 0, time = 0.200 sec SD = 2, time = 0.799 sec SD = 4, time = 3.190 sec SD = 6, time = 12.780 sec **Note**: SD influences the time the MAC listens for beacons or runs an energy scan on a given channel. The SD time is not a good estimate of the router/end device joining time requirements. ZigBee joining adds additional overhead including beacon processing on each channel, sending a join request, etc. that extend the actual joining time.	CRE	0 - 7 [exponent]	3
ZS	**ZigBee Stack Profile**. Set / read the ZigBee stack profile value. This must be set the same on all devices that should join the same network.	CRE	0 - 2	0
NJ	**Node Join Time**. Set/Read the time that a Coordinator/Router allows nodes to join. This value can be changed at run time without requiring a Coordinator or Router to restart. The time starts once the Coordinator or Router has started. The timer is reset on power-cycle or when NJ changes.	CR	0 - 0xFF [x 1 sec]	0xFF (always allows joining)
JV	**Channel Verification**. Set/Read the channel verification parameter. If JV=1, a router will verify the coordinator is on its operating channel when joining or coming up from a power cycle. If a coordinator is not detected, the router will leave its current channel and attempt to join a new PAN. If JV=0, the router will continue operating on its current channel even if a coordinator is not detected.	R	0 - Channel verification disabled 1 - Channel verification enabled	0
JN	**Join Notification**. Set / read the join notification setting. If enabled, the module will transmit a broadcast node identification packet on power up and when joining. This action blinks the Associate LED rapidly on all devices that receive the transmission, and sends an API frame out the UART of API devices. This feature should be disabled for large networks to prevent excessive broadcasts.	RE	0 - 1	0
AR	**Aggregate Routing Notification**. Set/read time between consecutive aggregate route broadcast messages. If used, AR should be set on only one device to enable many-to-one routing to the device. Setting AR to 0 only sends one broadcast	CR	0 - 0xFF	0xFF

Table 10-08. Networking Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
AI	**Association Indication**. Read information regarding last node join request: 0x00 - Successful completion - Coordinator started or Router/End Device found and joined with a parent. 0xAB - Attempted to join a device that did not respond. 0xAC - Secure join error - network security key received unsecured 0xAD - Secure join error - network security key not received 0xAF - Secure join error - joining device does not have the right preconfigured link key 0x21 - Scan found no PANs 0x22 - Scan found no valid PANs based on current SC and ID settings 0x23 - Valid Coordinator or Routers found, but they are not allowing joining (NJ expired) 0x27 - Node Joining attempt failed (typically due to incompatible security settings) 0x2A - Coordinator Start attempt failed' 0xFF - Scanning for a Parent 0x2B - Checking for an existing coordinator	CRE	0 - 0xFF [read-only]	--

Security

Table 10-09. Security Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
EE	**Encryption Enable**. Set/Read the encryption enable setting.	CRE	0 - Encryption disabled 1 - Encryption enabled	0
EO	**Encryption Options.** Configure options for encryption. Unused option bits should be set to 0. Options include: 0x01 - Send the security key unsecured over-the-air during joins 0x02 - Use trust center (coordinator only)r	CRE	0 - 0xFF	
NK	**Network Encryption Key**. Set the 128-bit AES network encryption key. This command is write-only; NK cannot be read. If set to 0 (default), the module will select a random network key.	C	128-bit value	0
KY	**Link Key**. Set the 128-bit AES link key. This command is write only; KY cannot be read. Setting KY to 0 will cause the coordinator to transmit the network key in the clear to joining devices, and will cause joining devices to acquire the network key in the clear when joining.	CRE	128-bit value	0

RF Interfacing

Table 10-010.RF Interfacing Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
PL	**Power Level**. Select/Read the power level at which the RF module transmits conducted power.	CRE	**XBee** (boost mode disabled) 0 = -8 dBm 1 = -4 dBm 2 = -2 dBm 3 = 0 dBm 4 = +2 dBm **XBee-PRO** 4 = 17 dBm **XBee-PRO (International Variant)** 4 = 10dBm	4
PM	**Power Mode**. Set/read the power mode of the device. Enabling boost mode will improve the receive sensitivity by 1dB and increase the transmit power by 2dB Note: Enabling boost mode on the XBee-PRO will not affect the output power. Boost mode imposes a slight increase in current draw. See section 1.2 for details.	CRE	0-1, 0= -Boost mode disabled, 1= Boost mode enabled.	1
DB	**Received Signal Strength**. This command reports the received signal strength of the last received RF data packet. The DB command only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link. DB can be set to 0 to clear it. The DB command value is measured in -dBm. For example if DB returns 0x50, then the RSSI of the last packet received was -80dBm.	CRE	0 - 0xFF Observed range for XBee-PRO: 0x1A - 0x58 For XBee: 0x 1A - 0x5C	

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

Serial Interfacing (I/O)

Table 10-011.Serial Interfacing Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
AP	API Enable. Enable API Mode. The AP command is only supported when using API firmware: 21xx (API coordinator), 23xx (API router), 29xx (API end device).	CRE	1 - 2 1 = API-enabled 2 = API-enabled (w/escaped control characters)	1
AO	**API Options**. Configure options for API. Current options select the type of receive API frame to send out the Uart for received RF data packets.	CRE	0 - Default receive API indicators enabled 1 - Explicit Rx data indicator API frame enabled (0x91)	0
BD	**Interface Data Rate**. Set/Read the serial interface data rate for communication between the module serial port and host. Any value above 0x07 will be interpreted as an actual baud rate. When a value above 0x07 is sent, the closest interface data rate represented by the number is stored in the BD register.	CRE	0x80 - 0xE1000 (non-standard rates up to 921kbps)	3
NB	**Serial Parity**. Set/Read the serial parity setting on the module.	CRE	0 = No parity 1 = Even parity 2 = Odd parity 3 = Mark parity	0
RO	**Packetization Timeout**. Set/Read number of character times of inter-character silence required before packetization. Set (RO=0) to transmit characters as they arrive instead of buffering them into one RF packet The RO command is only supported when using AT firmware: 20xx (AT coordinator), 22xx (AT router), 28xx (AT end device).	CRE	0 - 0xFF [x character times]	3
D7	**DIO7 Configuration**. Select/Read options for the DIO7 line of the RF module.	CRE	0 = Disabled 1 = CTS Flow Control 3 = Digital input 4 = Digital output, low 5 = Digital output, high 6 = RS-485 transmit enable (low enable) 7 = RS-485 transmit enable (high enable)	1
D6	**DIO6 Configuration.** Configure options for the DIO6 line of the RF module.	CRE	0 = Disabled 1 = RTS flow control 3 = Digital input 4 = Digital output, low 5 = Digital output, high	0

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

I/O Commands

Table 10-012.I/O Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
IR	**IO Sample Rate**. Set/Read the IO sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital IO functionality enabled (see D0-D8, P0-P2 commands). The sample rate is measured in milliseconds.	CRE	0 - 0xFFFF (ms)	0
IC	**IO Digital Change Detection**. Set/Read the digital IO pins to monitor for changes in the IO state. IC works with the individual pin configuration commands (D0-D8, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate IO sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0. Bit (IO pin): 0 (DIO0)4 (DIO4)8 (DIO8) 1 (DIO1) 5 (DIO5) 9 (DIO9) 2 (DIO2) 6 (DIO6) 10 (DIO10) 3 (DIO3) 7 (DIO7) 11 (DIO11)	CRE	: 0 - 0xFFFF	0
P0	**PWM0 Configuration**. Select/Read function for PWM0.	CRE	0 = Disabled 1 = RSSI PWM 3 - Digital input, monitored 4 - Digital output, default low 5 - Digital output, default high	1

Table 10-012.I/O Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
P1	**DIO11 Configuration**. Configure options for the DIO11 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P2	**DIO12 Configuration**. Configure options for the DIO12 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P3	**DIO13 Configuration**. Set/Read function for DIO13. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
D0	**AD0/DIO0 Configuration**. Select/Read function for AD0/DIO0.	CRE	1 - Commissioning button enabled 2 - Analog input, single ended 3 - Digital input 4 - Digital output, low 5 - Digital output, high	1
D1	**AD1/DIO1 Configuration**. Select/Read function for AD1/DIO1.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D2	AD2/DIO2 Configuration. Select/Read function for AD2/DIO2.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D3	**AD3/DIO3 Configuration**. Select/Read function for AD3/DIO3.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D4	**DIO4 Configuration**. Select/Read function for DIO4.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D5	**DIO5 Configuration**. Configure options for the DIO5 line of the RF module.	CRE	0 = Disabled 1 = Associated indication LED 3 = Digital input 4 = Digital output, default low 5 = Digital output, default high	1

Table 10-012.I/O Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
D8	**DIO8 Configuration**. Set/Read function for DIO8. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
LT	**Assoc LED Blink Time**. Set/Read the Associate LED blink time. If the Associate LED functionality is enabled (D5 command), this value determines the on and off blink times for the LED when the module has joined a network. If LT=0, the default blink rate will be used (500ms coordinator, 250ms router/end device). For all other LT values, LT is measured in 10ms.	CRE	0x14 - 0xFF (200 - 2550 ms)	0
PR	Set/read the bit field that configures the internal pull-up resistor status for the I/O lines. "1" specifies the pull-up resistor is enabled. "0" specifies no pullup.(30k pull-up resistors) Bits:" 0 - DIO4 (Pin 11) 1 - AD3 / DIO3 (Pin 17) 2 - AD2 / DIO2 (Pin 18) 3 - AD1 / DIO1 (Pin 19) 4 - AD0 / DIO0 (Pin 20) 5 - RTS / DIO6 (Pin 16) 6 - DTR / Sleep Request / DIO8 (Pin 9) 7 - DIN / Config (Pin 3) 8 - Associate / DIO5 (Pin 15) 9 - On/Sleep / DIO9 (Pin 13) 10 - DIO12 (Pin 4) 11 - PWM0 / RSSI / DIO10 (Pin 6) 12 - PWM1 / DIO11 (Pin 7)	CRE	0 - 0x1FFF	0 - 0x1FFF
RP	**RSSI PWM Timer.** Time RSSI signal will be output after last transmission. When RP = 0xFF, output will always be on.	CRE	0 - 0xFF [x 100 ms]	0x28 (40d)
CB	**Commissioning Pushbutton**. This command can be used to simulate commissioning button presses in software. The parameter value should be set to the number of button presses to be simulated. For example, sending the ATCB1 command will execute the action associated with 1 commissioning button press.	CRE		
%V	**Supply Voltage.** Reads the voltage on the Vcc pin. Divide the read value by 1024 A %V reading of 0x900 (2304 decimal) represents 2700mV or 2.70V.	R	-	-

Diagnostics

Table 10-013.Diagnostics Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
VR	**Firmware Version**. Read firmware version of the module. The firmware version returns 4 hexadecimal values (2 bytes) "ABCD". Digits ABC are the main release number and D is the revision number from the main release. "B" is a variant designator. XBee and XBee-PRO ZB modules return: 0x2xxx versions. XBee and XBee-PRO ZNet modules return: 0x1xxx versions. ZNet firmware is not compatible with ZB firmware.	CRE	0 - 0xFFFF [read-only]	Factory-set
HV	**Hardware Version**. Read the hardware version of the module.version of the module. This command can be used to distinguish among different hardware platforms. The upper byte returns a value that is unique to each module type. The lower byte indicates the hardware revision. XBee ZB and XBee ZNet modules return the following (hexadecimal) values: 0x19xx - XBee module 0x1Axx - XBee-PRO module	CRE	0 - 0xFFFF [read-only]	Factory-set

1. Node types that support the command:C = Coordinator, R = Router, E = End Device

AT Command Options

Table 10-014.AT Command Options Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
CT	**Command Mode Timeout.** Set/Read the period of inactivity (no valid commands received) after which the RF module automatically exits AT Command Mode and returns to Idle Mode.	CRE	2 - 0x028F [x 100 ms]	0x64 (100d)
CN	**Exit Command Mode.** Explicitly exit the module from AT Command Mode.	CRE	--	--
GT	**Guard Times**. Set required period of silence before and after the Command Sequence Characters of the AT Command Mode Sequence (GT + CC + GT). The period of silence is used to prevent inadvertent entrance into AT Command Mode.	CRE	1 - 0x0CE4 [x 1 ms] (max of 3.3 decimal sec)	0x3E8 (1000d)
CC	**Command Sequence Character**. Set/Read the ASCII character value to be used between Guard Times of the AT Command Mode Sequence (GT + CC + GT). The AT Command Mode Sequence enters the RF module into AT Command Mode. The CC command is only supported when using AT firmware: 20xx (AT coordinator), 22xx (AT router), 28xx (AT end device).	CRE	0 - 0xFF	0x2B ('+' ASCII)

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

Sleep Commands

Table 10-015.Sleep Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
SM	**Sleep Mode** Sets the sleep mode on the RF module	E	0-Sleep disabled 1-Pin sleep enabled 4-Cyclic sleep enabled 5 - Cyclic sleep, pin wake	0
SN	**Number of Sleep Periods.** Sets the number of sleep periods to not assert the On/Sleep pin on wakeup if no RF data is waiting for the end device. This command allows a host application to sleep for an extended time if no RF data is present	CRE	1 - 0xFFFF	1
SP	**Sleep Period.** This value determines how long the end device will sleep at a time, up to 28 seconds. (The sleep time can effectively be extended past 28 seconds using the SN command.) On the parent, this value determines how long the parent will buffer a message for the sleeping end device. It should be set at least equal to the longest SP time of any child end device.	CRE	0x20 - 0xAF0 x 10ms (Quarter second resolution)	0x20
ST	**Time Before Sleep** Sets the time before sleep timer on an end device.The timer is reset each time serial or RF data is received. Once the timer expires, an end device may enter low power operation. Applicable for cyclic sleep end devices only.	E	1 - 0xFFFE (x 1ms)	0x1388 (5 seconds)
SO Command	**Sleep Options**. Configure options for sleep. Unused option bits should be set to 0. Sleep options include: 0x02 - Always wake for ST time 0x04 - Sleep entire SN * SP time Sleep options should not be used for most applications. See Sleep Mode chapter for more information.	E	0 - 0xFF	0
WH	**Wake Host**. Set/Read the wake host timer value. If the wake host timer is set to a non-zero value, this timer specifies a time (in millisecond units) that the device should allow after waking from sleep before sending data out the UART or transmitting an IO sample. If serial characters are received, the WH timer is stopped immediately.	E	0 - 0xFFFF (x 1ms)	

Execution Commands

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

Table 10-01. Execution Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
AC	**Apply Changes**. Applies changes to all command registers causing queued command register values to be applied. For example, changing the serial interface rate with the BD command will not change the UART interface rate until changes are applied with the AC command. The CN command and 0x08 API command frame also apply changes.	CRE	-	

Table 10-01. Execution Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
WR	**Write**. Write parameter values to non-volatile memory so that parameter modifications persist through subsequent resets. Note: Once WR is issued, no additional characters should be sent to the module until after the "OK\r" response is received. The WR command should be used sparingly. The EM250 supports a limited number of write cycles.“	CRE	--	--
RE	**Restore Defaults**. Restore module parameters to factory defaults.	CRE	--	--
FR	**Software Reset.** Reset module. Responds immediately with an OK status, and then performs a software reset about 2 seconds later.	CRE	--	--
NR	**Network Reset**. Reset network layer parameters on one or more modules within a PAN. Responds immediately with an “OK” then causes a network restart. All network configuration and routing information is consequently lost. *If NR = 0*: Resets network layer parameters on the node issuing the command. *If NR = 1*: Sends broadcast transmission to reset network layer parameters on all nodes in the PAN.	CRE	0 - 1	--
SI	**Sleep Immediately**. Cause a cyclic sleep module to sleep immediately rather than wait for the ST timer to expire.	E	-	-
ND	**Node Discover.** Discovers and reports all RF modules found. The following information is reported for each module discovered. MY<CR> SH<CR> SL<CR> NI<CR> (Variable length) PARENT_NETWORK ADDRESS (2 Bytes)<CR> DEVICE_TYPE<CR> (1 Byte: 0=Coord, 1=Router, 2=End Device) STATUS<CR> (1 Byte: Reserved) PROFILE_ID<CR> (2 Bytes) MANUFACTURER_ID<CR> (2 Bytes) <CR> After (NT * 100) milliseconds, the command ends by returning a <CR>. ND also accepts a Node Identifier (NI) as a parameter (optional). In this case, only a module that matches the supplied identifier will respond. If ND is sent through the API, each response is returned as a separate AT_CMD_Response packet. The data consists of the above listed bytes without the carriage return delimiters. The NI string will end in a "0x00" null character. The radius of the ND command is set by the BH command.	CRE	optional 20-Byte NI or MY value	--
DN	**Destination Node.** Resolves an NI (Node Identifier) string to a physical address (case-sensitive). The following events occur after the destination node is discovered: <AT Firmware> 1. DL & DH are set to the extended (64-bit) address of the module with the matching NI (Node Identifier) string. 2. OK (or ERROR)\r is returned. 3. Command Mode is exited to allow immediate communication <API Firmware> 1. The 16-bit network and 64-bit extended addresses are returned in an API Command Response frame. If there is no response from a module within (NT * 100) milliseconds or a parameter is not specified (left blank), the command is terminated and an “ERROR” message is returned. In the case of an ERROR, Command Mode is not exited. The radius of the DN command is set by the BH command.	CRE	up to 20-Byte printable ASCII string	--
IS	**Force Sample** Forces a read of all enabled digital and analog input lines.	CRE	--	--
1S	**XBee Sensor Sample**. Forces a sample to be taken on an XBee Sensor device. This command can only be issued to an XBee sensor device using an API remote command.	RE	-	-

Node types that support the command: C = Coordinator, R = Router, E = End Device

11. OEM Support

This chapter provides customization information for the XBee/XBee-PRO ZB modules. In addition to providing an extremely flexible and powerful API, the XBee and XBee-PRO ZB modules are a robust development platform that have passed FCC and ETSI testing. Developers can customize default parameters, or even write or load custom firmware for Ember's EM250 chip.

X-CTU Configuration Tool

Digi provides a Windows X-CTU configuration tool for configuring module parameters and updating firmware. The XCTU has the capability to do the following:

- Discover all XBee devices in the network
- Update firmware on a local module (requires USB or serial connection)
- Read or write module configuration parameters on a local or remote device
- Save and load configuration profiles containing customized settings.

Contact Digi support for more information about the X-CTU.

Customizing XBee ZB Firmware

Once module parameters are tested in an application and finalized, Digi can manufacture modules with specific, customer-defined configurations for a nominal fee. These custom configurations can lock in a firmware version or set command values when the modules are manufactured, eliminating the need for customers to adjust module parameters on arrival. Alternatively, Digi can program custom firmware, including Ember's EZSP UART image, into the modules during manufacturing. Contact Digi to create a custom configuration.

Design Considerations for Digi Drop-In Networking

XBee/XBee-PRO embedded RF modules contain a variety of features that allow for interoperabilitywith Digi's full line of Drop-in Networking products. Interoperability with other "DIN" products can offer these advantages:

- Add IP-connectivity to your network via Cellular, Ethernet or WiFi with a ConnectPort X Gateway.
- Extend the range of your network with the XBee Wall Router.
- Make deployment easy by enabling the Commissioning Pushbutton (pin 20) and AssociateLED (pin 15) to operate with the Network Commissioning Tool software.
- Interface with standard RS-232, USB, Analog & Digital I/O, RS-485, and other industrial devices using XBee Adapters.
- Monitor and manage your network securely from remote locations with Connectware Manager software.

We encourage you to contact our technical representatives for consideration, implementation, or design review of your product for interoperability with Digi's Drop-in Networking solutions.

XBee Bootloader

XBee modules use a modified version of Ember's boot loader. This bootloader version supports a custom entry mechanism that uses module pins DIN (pin 3), $\overline{\text{DTR}}$ / SLEEP_RQ (pin 9), and $\overline{\text{RTS}}$ (pin 16). To invoke the boot loader, do the following:

1. Set $\overline{\text{DTR}}$ / SLEEP_RQ low (TTL 0V) and RTS high.

2. Send a serial break to the DIN pin and power cycle or reset the module.

3. When the module powers up, $\overline{\text{DTR}}$ / SLEEP_RQ and DIN should be low (TTL 0V) and RTS should be high.

4. Terminate the serial break and send a carriage return at 115200bps to the module.

5. If successful, the module will send the Ember boot loader menu out the DOUT pin at 115200bps.

6. Commands can be sent to the boot loader at 115200bps.

Note: Hardware flow control should be disabled when entering and communicating with the EM250 bootloader.

Programming XBee Modules

Firmware on the XBee and XBee-PRO ZB modules can be updated through one of two means:

- Serially
- SIF header.

Each method is described below.

Serial Firmware Updates

Serial firmware updates make use of the XBee custom bootloader which ships in all units. This modified bootloader is based on Ember's standalone bootloader, but with a modified entry mechanism. The modified entry mechanism uses module pins 3, 9, and 16 (DIN, DTR, and RTS respectively).

The X-CTU program can update firmware serially on the XBee and XBee-PRO ZB modules. Contact Digi support for details.

If an application requires custom firmware to update the XBee firmware serially, the following steps are required.

Invoke XBee Bootloader

See the "XBee Bootloader" section above for steps to invoke the bootloader

Send Firmware Image

After invoking the bootloader, the Ember bootloader will send the bootloader menu characters out the UART at 115200 bps. The application should do the following to upload a firmware image.

1. Look for the bootloader prompt "BL >" to ensure the bootloader is active

2. Send an ASCII "1" character to initiate a firmware update

3. After sending a "1", the EM250 waits for an XModem CRC upload of an .ebl image over the serial line at 115200 bps. The .ebl file must be sent to the EM250 in order.

If no serial transaction is initiated within a 60 second timeout period, the bootloader times out and returns to the menu. If the upload is interrupted with a power cycle or reset event, the EM250 will If no transaction is initiated within 60 seconds, the bootloader times out and returns to the menu. If the upload is interrupted with a power cycle or reset event, the EM250 will detect an invalid application image and enter bootloader mode. The entire ebl image should be uploaded again to recover. If an error occurs while uploading, the EM250 bootloader returns an error code from the following table:

Hex Error Code	Description
0x21	The bootloader encountered an error while trying to parse the Start of Header (SOH) character in the XModem frame.
0x22	The bootloader detected an invalid checksum in the XModem frame.
0x23	The bootloader encountered an error while trying to parse the high byte of the CRC in the XModem frame.
0x24	The bootloader encountered an error while trying to parse the low byte of the CRC in the XModem frame.

Hex Error Code	Description
0x25	The bootloader encountered an error in the sequence number of the current XModem frame.
0x26	The frame that the bootloader was trying to parse was deemed incomplete (some bytes missing or lost).
0x27	The bootloader encountered a duplicate of the previous XModem frame.
0x41	No .ebl header was received when expected.
0x42	Header failed CRC.
0x43	File failed CRC.
0x44	Unknown tag detected in .ebl image.
0x45	Invalid .ebl header signature.
0x46	Trying to flash odd number of bytes.
0x47	Indexed past end of block buffer.
0x48	Attempt to overwrite bootloader flash.
0x49	Attempt to overwrite SIMEE flash.
0x4A	Flash erase failed.
0x4B	Flash write failed.
0x4C	End tag CRC wrong length.
0x4D	Received data before query request/response

SIF Firmware Updates

The XBee/XBee-PRO modules have a 2x5 SIF header that can be used with Ember's InSight tools to upload firmware onto the modules. These tools include a USB device (USBLink) and Ethernet-enabled InSight Adapters. Contact Ember for details.

Warning: If programming firmware through the SIF interface, be aware that uploading firmware through the SIF header can potentially erase the XBee bootloader. If this happens, serial firmware updates will not work.

(The pinout for the SIF headers are shown in chapter 1.)

Writing Custom Firmware

The XBee/XBee-PRO module can be used as a hardware development platform for the EM250. Custom firmware images can be developed around the EmberZNet 2.5.x and 3.x mesh stacks (for the EM250) and uploaded to the XBee.

Warning: If programming firmware through the SIF interface, be aware that uploading firmware through the SIF header can potentially erase the XBee bootloader. If this happens, serial firmware updates will not work.

Regulatory Compliance

XBee modules are FCC and ETSI certified for operation on all 16 channels. The EM250 output power can be configured up to 3dBm with boost mode enabled.

XBee-PRO modules are certified for operation on 14 of the 16 band channels (channels 11 - 24). The scan channels mask of XBee-PRO devices must be set in the application to disable the upper two channels (i.e. 0x01FFF800). The XBee-PRO contains power compensation circuitry to adjust the output power near 18dBm or 10dBm depending on the part number. For best results, the EM250 should be configured with an output power level of 0dBm (or -2dBm if boost mode is enabled). The end product is responsible to adhere to these requirements.

Enabling GPIO 1 and 2

Most of the remaining sections in this chapter describe how to configure GPIO 1 and 2 to function correctly in custom applications that run on the XBee and XBee-PRO modules. In order for GPIO

pins 1 and 2 to be configurable, the application must set the GPIO_CFG register to enable GPIO 1 and 2. Bits 4 - 7 in the GPIO_CFG register control the functionality of various GPIO lines. The following table lists values for these bits that enable GPIO 1 and 2. Other functionality is affected by these settings. See the EM250 datasheet from Ember for a complete listing of functionality.

GPIO_CFG[7:4]Enabled Functionality	Enabled Functionality
0000	GPIO 0, 1, 2, 3, 9, 10, 11, 12
0111	0111GPIO 0, 1, 2, 3, 12
1010	GPIO 0, 1, 2, 3
1101	GPIO 0, 1, 2, 3, 11, 12

Example 1

The following code enables GPIO 0, 1, 2, 3, 9, 10, 11, and 12 and maintains all other GPIO_CFG bits.

```
int16u x;
x = GPIO_CFG;
x &= (0xFF0F);// Clear bits 4 - 7
GPIO_CFG = x;
```

Example 2

The following code enables GPIO 0, 1, 2, 3, and 12 and maintains all other GPIO_CFG bits.

```
int16u x;
x = GPIO_CFG;
x &= (0xFF0F);// Clear bits 4 - 7
x |= 0x0070;// Set bits 4 - 7 to 0111 as shown in the table above.
GPIO_CFG = x;
```

Detecting XBee vs XBee-PRO

For some applications, it may be necessary to determine if the code is running on an XBee or an XBee-PRO device. The GPIO1 pin on the EM250 is used to identify the module type (see table 1-03 in chapter 1). GPIO1 is connected to ground on the XBee module. The following code could be used to determine if a module is an XBee or XBee-PRO:

```
GPIO_DIRCLRL = GPIO(1);// Set GPIO1 as an input
GPIO_PUL |= GPIO(1);// Enable GPIO1 pullup resistor
ModuleIsXBeePro = (GPIO_INL & GPIO(1));//ModuleIsXBeePro > 0 if XBee-PRO, =0 if non-PRO.
```

Ensuring Optimal Output Power

XBee modules manufactured before February 2008 had an incorrect configuration setting that caused the default output power mode to be set incorrectly. Digi's ZB and ZNet firmware compensate for this by setting the output power mode in the application firmware.

Custom applications should call the emberSetTxPowerMode() function to set the output power mode as shown below:

XBee Applications

```
emberSetTxPowerMode(EMBER_TX_POWER_MODE_DEFAULT);  or
emberSetTxPowerMode(EMBER_TX_POWER_MODE_BOOST);
```

XBee-PRO Applications:

emberSetTxPowerMode(EMBER_TX_POWER_MODE_ALTERNATE); or

emberSetTxPowerMode(EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE);

XBee-PRO modules must also set a couple of IO lines to enable output power compensation. This is shown below. Once the IO lines are initialized (after powerup), the XBee will enable the power amplifier and LNA as needed.

On Powerup:

```
/* GPIO 2 should be set low for at least 10 milliseconds when coming up from power cycle. */
GPIO_DIRSETL = GPIO(2);// Set GPIO 2 as an output
GPIO_CLRL = GPIO(2);// Drive GPIO 2 low


/* After at least 10ms, GPIO 2 should be set high to power the output power compensation circuitry.
   At the same time GPIO 1 should be configured as an output and set low to enable the output power
   compensation circuitry. */
GPIO_DIRSETL = GPIO(1) | GPIO(2);// Set GPIO 1,2 as outputs
GPIO_CLRL = GPIO(1);// Drive GPIO 1 low
GPIO_SETL = GPIO(2);// Drive GPIO 2 high
```

Improving Low Power Current Consumption

To improve low power current consumption, the XBee should set a couple of unused IO lines as output low. This can be done during application initialization as shown below.

XBee (non-PRO) Initialization:

```
/* GPIO 1 and 2 are not used in the XBee (non-PRO) and should be set as outputs and driven low to
   reduce current draw. */
GPIO_DIRSETL = GPIO(1) | GPIO(2);// Set GPIO 1,2 as outputs
GPIO_CLRL = GPIO(1) | GPIO(2);// Set GPIO 1,2 low
```

XBee-PRO modules should disable the power compensation circuitry when sleeping to reduce current draw. This is shown below.

When sleeping (end devices):

```
/* The power compensation shutdown line on XBee-PRO modules (GPIO 1) should be set high when
   entering sleep to reduce current consumption. */
GPIO_SETL = GPIO(1);
```

When waking from sleep (end devices):

```
/* The power compensation shutdown line on XBee-PRO (GPIO 1) should be set low to enable the
   power compensation circuitry and LNA. */
GPIO_CLRL = GPIO(1);
```

Appendix A: Definitions

Definitions

Table A-01. Terms and Definitions

ZigBee Node Types	
Coordinator	A node that has the unique function of forming a network. The coordinator is responsible for establishing the operating channel and PAN ID for an entire network. Once established, the coordinator can form a network by allowing routers and end devices to join to it. Once the network is formed, the coordinator functions like a router (it can participate in routing packets and be a source or destination for data packets). -- One coordinator per PAN -- Establishes/Organizes PAN -- Can route data packets to/from other nodes -- Can be a data packet source and destination -- Mains-powered Refer to the XBee coordinator section for more information.
Router	A node that creates/maintains network information and uses this information to determine the best route for a data packet. A router must join a network before it can allow other routers and end devices to join to it. A router can participate in routing packets and is intended to be a mains-powered node. -- Several routers can operate in one PAN -- Can route data packets to/from other nodes -- Can be a data packet source and destination -- Mains-powered Refer to the XBee router section for more information.
End device	End devices must always interact with their parent to receive or transmit data. (See 'joining definition.) They are intended to sleep periodically and therefore have no routing capacity. An end device can be a source or destination for data packets but cannot route packets. End devices can be battery-powered and offer low-power operation. -- Several end devices can operate in one PAN -- Can be a data packet source and destination -- All messages are relayed through a coordinator or router -- Lower power modes
ZigBee Protocol	
PAN	Personal Area Network - A data communication network that includes a coordinator and one or more routers/end devices.

Table A-01. Terms and Definitions

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

Appendix B: Agency Certifications

United States FCC

The XBee 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 fufill FCC Certification, the OEM must comply with the following regulations:

1.The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product. [Figure A-01]

2.XBee RF Module may only be used with antennas that have been tested and approved for use with this module [refer to the antenna tables in this section].

OEM Labeling Requirements

WARNING: The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product enclosure that displays the contents shown in the figure below.

Required FCC Label for OEM products containing the XBee RF Module

> Contains FCC ID: OUR-XBEE2*
>
> 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 PRO RF Module

> Contains FCC ID:MCQ-XBEEPRO2*
>
> The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (*i.*) this device may not cause harmful interference and (*ii.*) this device must accept any interference received, including interference that may cause undesired operation.

FCC Notices

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

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

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

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

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

FCC-Approved Antennas (2.4 GHz)

The XBee and XBee-PRO RF Module can be installed utilizing antennas and cables constructed with standard connectors (Type-N, SMA, TNC, etc.) if the installation is performed professionally and according to FCC guidelines. For installations not performed by a professional, non-standard connectors (RPSMA, RPTNC, etc.) must be used.

The modules are FCC approved for fixed base station and mobile applications on channels 0x0B-0x1A for Xbee Series2 and on channels 0x0B - 0x18 for Xbee ZNet-PRO 2.5 . If the antenna is mounted at least 20cm (8 in.) from nearby persons, the application is considered a mobile application. Antennas not listed in the table must be tested to comply with FCC Section 15.203 (Unique Antenna Connectors) and Section 15.247 (Emissions).

XBee RF Modules: XBee RF Modules have been tested and approved for use with all the antennas listed in the tables below. (Cable-loss IS required when using gain antennas as shown below.)

Table A-01. antennas approved for use with the XBee RF Modules

YAGI CLASS ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-Y6NF	Yagi (6-element)	8.8 dBi	Fixed	2 m	N/A
A24-Y7NF	Yagi (7-element)	9.0 dBi	Fixed	2 m	N/A
A24-Y9NF	Yagi (9-element)	10.0 dBi	Fixed	2 m	N/A
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed	2 m	N/A
A24-Y12NF	Yagi (12-element)	12.0 dBi	Fixed	2 m	N/A
A24-Y13NF	Yagi (13-element)	12.0 dBi	Fixed	2 m	N/A
A24-Y15NF	Yagi (15-element)	12.5 dBi	Fixed	2 m	N/A
A24-Y16NF	Yagi (16-element)	13.5 dBi	Fixed	2 m	N/A
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5 dBi	Fixed	2 m	N/A
A24-Y18NF	Yagi (18-element)	15.0 dBi	Fixed	2 m	N/A
OMNI-DIRECTIONAL ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-C1	Surface Mount integral chip	-1.5 dBi	Fixed/Mobile	20 cm	N/A
A24-F2NF	Omni-directional (Fiberglass base station)	2.1 dBi	Fixed/Mobile	20 cm	N/A
A24-F3NF	Omni-directional (Fiberglass base station)	3.0 dBi	Fixed/Mobile	20 cm	N/A
A24-F5NF	Omni-directional (Fiberglass base station)	5.0 dBi	Fixed/Mobile	20 cm	N/A
A24-F8NF	Omni-directional (Fiberglass base station)	8.0 dBi	Fixed	2 m	N/A
A24-F9NF	Omni-directional (Fiberglass base station)	9.5 dBi	Fixed	2 m	N/A
A24-F10NF	Omni-directional (Fiberglass base station)	10.0 dBi	Fixed	2 m	N/A
A24-F12NF	Omni-directional (Fiberglass base station)	12.0 dBi	Fixed	2 m	N/A
A24-F15NF	Omni-directional (Fiberglass base station)	15.0 dBi	Fixed	2 m	N/A
A24-W7NF	Omni-directional (Base station)	7.2 dBi	Fixed	2 m	N/A
A24-M7NF	Omni-directional (Mag-mount base station)	7.2 dBi	Fixed	2 m	N/A
PANEL CLASS ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-P8SF	Flat Panel	8.5 dBi	Fixed	2 m	N/A
A24-P8NF	Flat Panel	8.5 dBi	Fixed	2 m	N/A
A24-P13NF	Flat Panel	13.0 dBi	Fixed	2 m	N/A
A24-P14NF	Flat Panel	14.0 dBi	Fixed	2 m	N/A
A24-P15NF	Flat Panel	15.0 dBi	Fixed	2 m	N/A
A24-P16NF	Flat Panel	16.0 dBi	Fixed	2 m	N/A
A24-P19NF	Flat Panel	19.0 dBi	Fixed	2m	1.5 dB

Table A-02. antennas approved for use with the XBee-PRO RF Modules

YAGI CLASS ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-Y6NF	Yagi (6-element)	8.8 dBi	Fixed	2 m	7.8dB
A24-Y7NF	Yagi (7-element)	9.0 dBi	Fixed	2 m	8 dB
A24-Y9NF	Yagi (9-element)	10.0 dBi	Fixed	2 m	9 dB
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed	2 m	10 dB
A24-Y12NF	Yagi (12-element)	12.0 dBi	Fixed	2 m	11 dB
A24-Y13NF	Yagi (13-element)	12.0 dBi	Fixed	2 m	11 dB
A24-Y15NF	Yagi (15-element)	12.5 dBi	Fixed	2 m	11.5 dB
A24-Y16NF	Yagi (16-element)	13.5 dBi	Fixed	2 m	12.5 dB
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5 dBi	Fixed	2 m	12.5 dB
A24-Y18NF	Yagi (18-element)	15.0 dBi	Fixed	2 m	14 dB
OMNI-DIRECTIONAL ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-C1	Surface Mount integral chip	-1.5dBi	Fixed/Mobile	20 cm	-
A24-F2NF	Omni-directional (Fiberglass base station)	2.1 dBi	Fixed/Mobile	20 cm	-
A24-F3NF	Omni-directional (Fiberglass base station)	3.0 dBi	Fixed/Mobile	20 cm	.3 dB
A24-F5NF	Omni-directional (Fiberglass base station)	5.0 dBi	Fixed/Mobile	20 cm	2.3 dB
A24-F8NF	Omni-directional (Fiberglass base station)	8.0 dBi	Fixed	2 m	5.3 dB
A24-F9NF	Omni-directional (Fiberglass base station)	9.5 dBi	Fixed	2 m	6.8 dB
A24-F10NF	Omni-directional (Fiberglass base station)	10.0 dBi	Fixed	2 m	7.3 dB
A24-F12NF	Omni-directional (Fiberglass base station)	12.0 dBi	Fixed	2 m	9.3dB
A24-F15NF	Omni-directional (Fiberglass base station)	15.0 dBi	Fixed	2 m	12.3dB
A24-W7NF	Omni-directional (Base station)	7.2 dBi	Fixed	2 m	4.5 dB
A24-M7NF	Omni-directional (Mag-mount base station)	7.2 dBi	Fixed	2 m	4.5 dB
PANEL CLASS ANTENNAS					
Part Number	**Type (Description)**	**Gain**	**Application***	**Min. Separation Required**	**Cable-loss**
A24-P8SF	Flat Panel	8.5 dBi	Fixed	2 m	8.2 dB
A24-P8NF	Flat Panel	8.5 dBi	Fixed	2 m	82 dB
A24-P13NF	Flat Panel	13.0 dBi	Fixed	2 m	12.7 dB
A24-P14NF	Flat Panel	14.0 dBi	Fixed	2 m	13.7 dB
A24-P15NF	Flat Panel	15.0 dBi	Fixed	2 m	14.7 dB
A24-P16NF	Flat Panel	16.0 dBi	Fixed	2 m	15.7 dB
A24-P19NF	Flat Panel	19.0 dBi	Fixed	2m	18.7 dB

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

RF Exposure

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

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

Europe (ETSI)

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

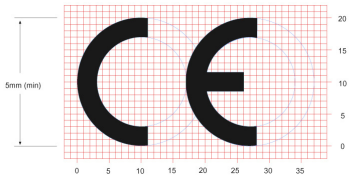
If the XBee RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product to the European harmonized EMC and low-voltage/safety standards. A Declaration of Conformity must be issued for each of these standards and kept on file as described in Annex II of the R&TTE Directive.

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

OEM Labeling Requirements

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

Figure B-01. 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 5mm except where this is not possible on account of the nature of the apparatus.
- The CE marking must be affixed visibly, legibly, and indelibly.

Restrictions

Power Output: The power output of the XBee RF Module must not exceed 10 dBm. The power level is set using the PL command and the PL parameter must equal "0" (10 dBm).

France: France imposes restrictions on the 2.4 GHz band. Go to www.art-telecom.Fr or contact MaxStream for more information.

Norway: Norway prohibits operation near Ny-Alesund in Svalbard. More information can be found at the Norway Posts and Telecommunications site (www.npt.no).

Declarations of Conformity

Digi has issued Declarations of Conformity for the XBee RF Modules concerning emissions, EMC and safety. Files are located in the 'documentation' folder of the Digi CD.

Important Note

Digi does not list the entire set of standards that must be met for each country. Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. For more information relating to European compliance of an OEM product incorporating the XBee RF Module, contact Digi, or refer to the following web sites:

CEPT ERC 70-03E - Technical Requirements, European restrictions and general requirements: Available at www.ero.dk/.

R&TTE Directive - Equipment requirements, placement on market: Available at www.ero.dk/.

Approved Antennas

When integrating high-gain antennas, European regulations stipulate EIRP power maximums. Use the following guidelines to determine which antennas to design into an application.

XBee OEM Module

The following antennas types have been tested and approved for use with the XBee Module:

Antenna Type: Yagi

RF module was tested and approved with 15 dBi antenna gain with 1 dB cable-loss (EIRP Maimum of 14 dBm). Any Yagi type antenna with 14 dBi gain or less can be used with no cable-loss.

Antenna Type: Omni-Directional

RF module was tested and approved with 15 dBi antenna gain with 1 dB cable-loss (EIRP Maimum of 14 dBm). Any Omni-Directional type antenna with 14 dBi gain or less can be used with no cable-loss.

Antenna Type: Flat Panel

RF module was tested and approved with 19 dBi antenna gain with 4.8 dB cable-loss (EIRP Maimum of 14.2 dBm). Any Flat Panel type antenna with 14.2 dBi gain or less can be used with no cable-loss.

XBee RF Module

The following antennas have been tested and approved for use with the embedded XBee RF Module:

- Dipole (2.1 dBi, Omni-directional, Articulated RPSMA, Digi part number A24-HABSM)
- Chip Antenna (-1.5 dBi)
- Attached Monopole Whip (1.5 dBi)

XBee-PRO RF Module

The following antennas have been tested and approved for use with the embedded XBee-PRO RF Module:

- Dipole (2.1 dBi, Omni-directional, Articulated RPSMA, Digi part number A24-HABSM)
- Chip Antenna (-1.5 dBi)
- Attached Monopole Whip (1.5 dBi).

Canada (IC)

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:

Contains Model XBee Radio, IC: 4214A-XBEE2

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.

If it contains an XBee-PRO OEM Module, the clearly visible label on the outside of the final product enclosure must display the following text:

Contains Model XBee PRO Radio, IC: 1846A-XBEEPRO2

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

Transmitters for Detachable Antennas

This device has been designed to operate with the antennas listed in table A-3 and having a maximum of 17.5 dB. Antennas not included in this list or having a gain greater than 17.5 dB are strictly prohibited for use with this device. The required antenna impedance is 50 Ω

Detachable Antenna

To reduce potential radio interference to other users, the antenna type and gained shuold be so chosen that the equivaleny istropically radiated power (e.i.r.p.) is not more than permitted for successful communication

Appendix C: Migrating from ZNet 2.5 to XBee ZB

The following paragraph contains the significant differences in XBee ZB compared to its predecessor, ZNet 2.5.

- Command Set
- Firmware Versions
- New Features.

Command Set

The following ZNet 2.5 commands have changed for XBee ZB:

- ZA - Set / read the ZigBee Addressing enable command. This command was required in ZNet 2.5 to enable application level addressing commands SE, DE, CI. XBee ZB does not support ZA. The SE, DE, and CI values always determine the application level addressing values.
- AI - Read the association status. AI now includes several new values.

Firmware Versions

ZNet 2.5 supported coordinator and router/end device targets. Due to flash constraints, XBee ZB split the router/end device target into 2 different targets - router, and end device. There is not a router/end device target.

New Features

ZB offers many new and improved features over ZNet 2.5, including:

- Data transmissions are directly resolved to APS unicasts. This provides the ability to send and receive ZDO commands.
- NH command configures the unicast transmission timeout. This command can extend the number of unicast hops dramatically over the 6-8 hop limit that existed in ZNet 2.5.
- ZS command allows ZigBee stack profile to be set as required.

Appendix D: XBee ZB Firmware Matrix

Overview of Features

The XBee ZB firmware supports the following versions:

- 204x - AT Coordinator
- 214x - API Coordinator
- 224x - AT Router
- 234x - API Router
- 284x - AT End Device
- 294x - API End Device.

The supported features of each firmware version are listed in the table below:

Feature	204x (AT Crd)	214x (API Crd)	224x (AT Rtr)	234x (API Rtr)	284x (AT End-Dev)	294x (API End-Dev)
Aggregator Capable	X	X	X	X		
Allows Joining	X	X	X	X		
AT Cmd Mode	X		X		X	
API		X		X		X
Channel Verification on Join (JV)			X	X		
Commissioning Button	X	X	X	X	X	X
IO Sampling (IS)			X		X	X
Receives unicast traffic from devices in the network	X	X	X	X	X	X
Receives broadcast traffic from devices in the network	X	X	X	X	X	X
Responds to Remote Commands	X	X	X	X	X	X
Responds to Network Discovery	X	X	X	X	X	X
RF data transmissions / receptions are always routed through the parent					X	X
Routes Data	X	X	X	X		

Sends Network Discovery (ND)	X	X	X	X	X	
Sends Network Reset (NR1)	X	X	X	X	X	
Sends Remote Commands		X		X		X
Sleep Modes					X	X

Appendix E: Additional Information

1-Year Warranty

XBee RF Modules from Digi, Inc. (the "Product") are warranted against defects in materials and workmanship under normal use, for a period of 1-year from the date of purchase. In the event of a product failure due to materials or workmanship, Digi will repair or replace the defective product. For warranty service, return the defective product to MaxStream, shipping prepaid, for prompt repair or replacement.

The foregoing sets forth the full extent of MaxStream's warranties regarding the Product. Repair or replacement at MaxStream's option is the exclusive remedy. THIS WARRANTY IS GIVEN IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, AND DIGI SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL DIGI, ITS SUPPLIERS OR LICENSORS BE LIABLE FOR DAMAGES IN EXCESS OF THE PURCHASE PRICE OF THE PRODUCT, FOR ANY LOSS OF USE, LOSS OF TIME, INCONVENIENCE, COMMERCIAL LOSS, LOST PROFITS OR SAVINGS, OR OTHER INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT, TO THE FULL EXTENT SUCH MAY BE DISCLAIMED BY LAW. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES. THEREFORE, THE FOREGOING EXCLUSIONS MAY NOT APPLY IN ALL CASES. This warranty provides specific legal rights. Other rights which vary from state to state may also apply.

Contact Digi

Free and unlimited technical support is included with every Digi Radio Modem sold. For the best in wireless data solutions and support, please use the following resources:

Technical Support:	Phone.	(866) 765-9885 toll-free U.S.A. & Canada (801) 765-9885 Worldwide
	Live Chat.	www.digi.com
	Online Support.	http://www.digi.com/support/eservice/login.jsp

Digi's office hours are 8:00 am - 5:00 pm [U.S. Mountain Time]