XBee-PRO 900HP DigiMesh Kit
Radio Frequency (RF) Module

User Guide
Revision history—90001496

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>January 2016</td>
<td>Initial release.</td>
</tr>
<tr>
<td>B</td>
<td>March 2016</td>
<td>Rebranded with minor updates.</td>
</tr>
</tbody>
</table>

Trademarks and copyright

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

© 2018 Digi International Inc. All rights reserved.

Disclaimers

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

Warranty

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

www.digi.com/howtobuy/terms

Send comments

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

Customer support

Digi Technical Support: Digi offers multiple technical support plans and service packages to help our customers get the most out of their Digi product. For information on Technical Support plans and pricing, contact us at +1 952.912.3444 or visit us at www.digi.com/support.
Contents

XBee-PRO 900HP DigiMesh Kit User Guide

Kit contents

Introduction to XBee modules

DigiMesh networks

Synchronous sleeping network .............................................................. 15
Mesh networking .................................................................................... 15

Getting Started with your DigiMesh kit

Plug in the XBee module ........................................................................ 17
How to unplug an XBee module ............................................................ 18
Download and install XCTU .................................................................... 18
Install XCTU - Windows ........................................................................ 18
Install XCTU - Linux ............................................................................... 18
Install XCTU - OSX ................................................................................ 19
Optional: Install XCTU updates ............................................................... 19
Optional: Manually install USB drivers .................................................. 19
Lab: Set up a simple DigiMesh network .................................................. 20
Step 1: Requirements .............................................................................. 20
Step 2: Connect the components ............................................................ 20
Step 3: Add the XBee modules to XCTU ................................................ 20
Step 4: Configure the XBee modules ...................................................... 21
Step 5: Check the network ..................................................................... 22
Step 6: Send wireless messages ............................................................. 23

How XBee devices work

How XBee devices communicate ............................................................ 26
Wireless communication .......................................................................... 26
Addressing .............................................................................................. 26
Network identifier .................................................................................. 27
Preamble identifier .................................................................................. 27
Radio frequency channels and channel masks ....................................... 28
Serial communication ............................................................................. 29
Operating modes .................................................................................... 30
Comparison of transparent and API modes ..............................................................30

XBee transparent mode

XBee transparent mode in detail ..............................................................................34
Command mode .......................................................................................................34
AT commands .........................................................................................................35
Use AT commands ..................................................................................................36

XBee API mode

API mode in detail ....................................................................................................39
Advantages of API mode ........................................................................................39
API frame structure .................................................................................................40
Start delimiter .........................................................................................................40
Length .....................................................................................................................40
Frame data .............................................................................................................40
Checksum ...............................................................................................................41
DigiMesh supported frames ..................................................................................42
Frame examples ......................................................................................................43
Operating mode configuration ...............................................................................48
API escaped operating mode (API 2) ....................................................................49
XBee frame exchange ..............................................................................................50
AT Command: configure a local XBee device .......................................................50
Transmit Request/Receive Packet: Transmit and receive wireless data ...............53
Lab: Transmit and receive data ..............................................................................54
Remote AT Command: Remotely configure an XBee module ................................59
Do more with API mode: XBee libraries .................................................................60

DigiMesh network setup

Set up a DigiMesh network ....................................................................................62
Routing ...................................................................................................................62
Route discovery ......................................................................................................63
Trace routing ...........................................................................................................64
Aggregate routes ....................................................................................................64
Use the AG command .............................................................................................64
Disable routing ........................................................................................................65
View the network ....................................................................................................65
Lab: Build and experiment with a DigiMesh network ............................................66
Step 1: Requirements ............................................................................................66
Step 2: Connect the components ..........................................................................66
Step 3: Configure the XBee modules .....................................................................66
Step 4: Set up the components ..............................................................................68
Step 5: Send messages via a bridge node ...............................................................68

Wireless data transmission

Transmission methods ............................................................................................72
Broadcast transmission ..........................................................................................72
Unicast transmission ...............................................................................................72
Lab: Transmit data wirelessly ................................................................................73
Step 1: Requirements ............................................................................................73
Step 2: Connect the components ................................................................. 74
Step 3: Configure the XBee modules ......................................................... 74
Step 4: Create a Java project .................................................................... 75
Step 5: Link libraries to the project ........................................................... 75
Step 6: Add the source code to the project ................................................. 77
Step 7: Set the port names and launch applications ..................................... 78
Step 8: Transmit data over the network ....................................................... 78
Step 9: Section summary of wireless data transmission ............................... 78
Step 10: Do more with wireless data transmission ....................................... 79

Low power and battery life

Low power devices and battery life .............................................................. 81
A real world scenario .................................................................................. 81
Design considerations for applications using sleep mode .......................... 81
Sleep modes ................................................................................................. 81
  Synchronous sleep modes ........................................................................... 82
  Asynchronous sleep modes ........................................................................ 82
Asynchronous sleep ..................................................................................... 82
  Synchronous sleep operation .................................................................... 83
  Synchronous sleep modes ........................................................................ 83
  Synchronous sleep parameters ................................................................. 84
  Designate a sleep coordinator ................................................................. 85
  Start a sleeping network .......................................................................... 87
Lab: Use synchronous sleep ....................................................................... 89
  Step 1: Requirements ............................................................................... 89
  Step 2: Connect the components ............................................................ 90
  Step 3: Configure the XBee modules ....................................................... 90
  Step 4: Test the sleep configuration ........................................................ 93
  Step 5: Section summary of synchronous sleep ...................................... 95
  Step 6: Do more with sleep mode ............................................................. 96

Inputs and outputs

XBee I/O pins ............................................................................................... 98
How XBee devices get sensor data .............................................................. 99
Sensors .......................................................................................................... 99
  How to configure a pin as an input .......................................................... 99
  How to obtain data from a sensor ........................................................... 100
Lab: receive digital data ............................................................................. 101
  Step 1: Requirements ............................................................................. 102
  Step 2: Connect the components ............................................................ 102
  Step 3: Configure the XBee modules ....................................................... 102
  Step 4: Create a Java project ................................................................ 104
  Step 5: Link libraries to the project ......................................................... 104
  Step 6: Add the source code to the project .............................................. 104
  Step 7: Set the port name and launch the application ............................... 107
  Step 8: Section summary of receiving digital data ................................... 107
  Step 9: Do more with receiving digital data ............................................. 108
Lab: Receive analog data ......................................................................... 108
  Step 1: Requirements ............................................................................. 109
  Step 2: Connect the components ............................................................ 109
  Step 3: Configure the XBee modules ....................................................... 109
Step 4: Create a Java project .................................................. 111
Step 5: Link libraries to the project ...................................... 112
Step 6: Add the source code to the project .............................. 113
Step 7: Set the port name and launch the application .......... 114
Step 8: Section summary of receiving analog data ............... 114
Step 9: Do more with receiving analog data .......................... 115

How XBee modules control devices ..................................... 115
Actuators ............................................................................ 116
Set pins for digital and analog actuators ............................... 116
How to configure a pin as an output .................................... 116
How to send actuations ....................................................... 117

Lab: Send digital actuations .................................................. 117
Step 1: Requirements ......................................................... 117
Step 2: Connect the components ......................................... 118
Step 3: Configure the XBee modules ................................. 118
Step 4: Create a Java project .............................................. 119
Step 5: Link libraries to the project .................................... 120
Step 6: Add the source code to the project ...................... 121
Step 7: Set the port name and launch the application .......... 122
Step 8: Section summary of sending digital actuations ....... 122
Step 9: Do more with sending digital actuations ................ 122

Lab: Send analog actuations .................................................. 123
Step 1: Requirements ........................................................ 123
Step 2: Connect the components ......................................... 123
Step 3: Configure the XBee modules ................................. 123
Step 4: Create a Java project .............................................. 124
Step 5: Link libraries to the project .................................... 125
Step 6: Add the source code to the project ...................... 126
Step 7: Set the port name and launch the application .......... 127
Step 8: Section summary of sending analog actuations ....... 127
Step 9: Do more with sending analog actuations ................ 127

Security and encryption

How to enable network security ........................................ 130
Lab: Encrypt a simple DigiMesh network ......................... 130
Test your encryption ....................................................... 130
Additional recommendations ........................................... 130

Signal strength and radio frequency range

Distance and obstacles ........................................................ 133
Factors affecting wireless communication ............................. 134
Signal strength and the RSSI pin ....................................... 135
Is RSSI the best indication of link quality? ......................... 137
Range test ............................................................................. 137
Example: perform a range test ......................................... 139
Step 1: Requirements ........................................................ 139
Step 2: Connect the components ......................................... 140
Step 3: Configure the XBee modules ................................. 140
Perform a range test ....................................................... 141
Step 5: Section summary of signal strength ....................... 141
Radio firmware

Firmware identification .......................................................... 143
Update radio firmware .................................................................. 143
  Update your device’s firmware .................................................. 144
Download new firmware ............................................................. 144

Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit

General ................................................................................. 146
  Cannot find the serial port for the module ............................... 146
  Can’t identify XBee modules .................................................. 146
  Error: Port is already in use .................................................. 146
  Error: Device driver was not successfully installed .................. 146
XCTU .................................................................................... 147
  Error upon installation of XCTU ............................................. 147
  Discovery process either does not find devices, or XCTU does not list serial ports ............................. 147
  XCTU reports errors for KY and DD settings after resetting to factory defaults .................... 148
DigiMesh network setup ................................................................ 148
  The modules are not found when checking the network ............ 148
Wireless data transmission ......................................................... 148
  Error: Parsing text error message ......................................... 148
  Error: Could not find the module in the network ................. 148
XBees Java library ................................................................... 148
  Warning message: RXTX version mismatch .............................. 149
  Invalid operating mode exception message ............................. 149
  Java lang unsatisfied exception message ............................... 149
  Interface in use exception message ....................................... 150
  Java lang no class definition exception message ..................... 150
  SLF4J class path contains multiple bindings message .......... 151
  XBee Java Library and transparent mode .............................. 151
Synchronous sleep ................................................................... 151
  A receiver module does not receive any message .................. 151
Range test .............................................................................. 151
  Error: There are not remote devices discovered for the selected local device ............... 151
  There are no remote devices to select .................................. 151
  The local RSSI and the number of packets received are always 0 ........................................ 152
  Remote RSSI is not included in the chart and the Remote RSSI control is disabled ............... 152

Additional resources

XBee buying guide ................................................................. 154
Where to buy XBee devices .......................................................... 154
  Find products from Digi and Digi distributors ......................... 154
  Find Digi products through resellers ....................................... 155
XCTU walkthrough ................................................................. 155
  XCTU overview ................................................................... 155
Application working modes .................................................... 158
  Add a module ....................................................................... 158
  Read settings ....................................................................... 159
  Change settings ..................................................................... 160
  Save settings ....................................................................... 160
Real projects with XBee modules ............................................... 160
  Community ......................................................................... 161
Digi’s XBee-PRO 900HP DigiMesh Kit is a great way to learn how to use XBee RF modules for device connectivity and mesh networking. This kit lets anyone get started in the world of DigiMesh. Whether you’re a hardware or software engineer, corporate technologist, educator, or student, you can quickly create wireless mesh networks.

XBee-PRO 900HP embedded modules provide best-in-class long-range wireless connectivity to devices. Supporting RF line-of-sight ranges from 1000 feet (305 m) indoors to 4 miles (6.5 km) outdoors with dipole antennas, and data rates of up to 200 Kbps, these modules are ideal for extended-range applications requiring increased data throughput.

XBee-PRO 900HP modules take advantage of the DigiMesh networking protocol. This innovative mesh protocol offers users added network stability through self-healing, self-discovering, and dense network operation. With support for sleeping routers, DigiMesh is ideal for power-sensitive applications that rely upon batteries or power-harvesting technology.

Follow this guide to:
- Assemble your kit components
- Set up device communications
- Build working control systems
- Design sensing networks with excellent battery life and robust security
Each section of this guide explains a basic topic related to XBees and DigiMesh with step-by-step examples that put into practice the concepts you have learned. The topics are arranged according to their complexity, from the most basic to the more powerful features. Certain examples use the Java programming language. These examples are easy for anyone to use, and those with some programming background can extend them to real-world scenarios.
**Kit contents**

Your XBee-PRO 900HP DigiMesh Kit contains the following components:

<table>
<thead>
<tr>
<th>Qty.</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>XBee Grove Development Boards</td>
</tr>
<tr>
<td>3</td>
<td>XBee-PRO 900HP RF modules</td>
</tr>
<tr>
<td>3</td>
<td>Micro USB cables</td>
</tr>
</tbody>
</table>
XBee modules come in two hardware footprints: through-hole and surface mount.

- Through-hole technology (THT) XBees include the 20-pin socket and require holes for mounting the component on the printed circuit board (PCB), although it is common for the carrier board to contain a female socket.

- Surface-mount technology (SMT) XBees include 37 pads. They are placed directly on the PCB, which means they do not require holes or sockets for mounting the component.

### Kit contents

<table>
<thead>
<tr>
<th>Qty.</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>XBee stickers</td>
</tr>
</tbody>
</table>

XBee-Pro 900HP DigiMesh Kit User Guide
Introduction to XBee modules

XBee modules are small RF (radio frequency) devices that transmit and receive data over the air using radio signals. Wireless capability is essential whenever you want to place sensors where no cables can be installed, or where such tethering is undesirable. XBee devices are highly configurable and support multiple protocols, which lets you choose the right technology for your application—whether you want to set up a pair of radios to swap data or design a large mesh network with multiple devices.

Here are some of the ways you can use XBee devices:

- Controlling a robot remotely or creating wearable electronics for people, pets, or wildlife, without hindering movement.
- Making a building smarter and more responsive to human interaction.
- Using XBee technology in industrial solutions. For example, XBee devices are used as sensors to monitor industrial tanks for liquid levels, temperature, and pressure, and to monitor and control complex machines such as wind turbines.
DigiMesh networks

DigiMesh is a proprietary networking topology for use in wireless end-point connectivity solutions. It supports advanced networking features including sleeping routers and dense mesh networks. DigiMesh supports multiple network topologies such as point-to-point, point-to-multipoint, and mesh networks. With support for sleeping routers, DigiMesh is ideal for power-sensitive applications that rely upon batteries or power-harvesting technology.

DigiMesh contains the following features:

- Self-healing: Any node may enter or leave the network at any time without causing the network as a whole to fail.
- Peer-to-peer architecture: No hierarchy and no parent-child relationships are needed.
- Easy to use: Mesh networking is simplified because it does not require hierarchy or parent-child relationships.
- Quiet: Routing overhead is reduced by using a reactive protocol similar to Ad-hoc On Demand Distance Vector routing (AODV).
- Route discovery: Routes are discovered and created only when needed, eliminating the need to maintain a network map.
- Selective acknowledgments: Only the destination node will reply to route requests.
- Reliable: Acknowledgments confirm successful delivery of data.
- Sleep modes: Supports low-power sleep modes with synchronized wake up as well as variable sleep and wake times.

Synchronous sleeping network .................................................................15
Mesh networking ..................................................................................15


**Synchronous sleeping network**

One advantage of DigiMesh is that it allows all nodes in the network to both route data and enter low-power states. The synchronous sleep feature makes it possible for all nodes in the network to synchronize their sleep and wake times. This allows some or all XBee modules in the network to be battery powered.

All modules go to sleep at the same time. This forms a cyclic sleeping network where we can define two different device types: sleeping router and sleep coordinator.

For more information on the synchronous sleep feature, see *Low power and battery life*.

**Mesh networking**

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

- **Routing.** With this technique, the message is propagated along a path by hopping from node to node until it reaches its final destination.

- **Ad-hoc network creation.** This is an automated process that creates an entire network of nodes on the fly, without any human intervention.

- **Self-healing.** This process automatically figures out if one or more nodes on the network is missing and reconfigures the network to repair any broken routes.

![Mesh Network Diagram](image)

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

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

---

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

This section provides everything you need to set up your XBee-PRO 900HP DigiMesh Kit and your first XBee DigiMesh application.

Plug in the XBee module ................................................................. 17
How to unplug an XBee module ...................................................... 18
Download and install XCTU ............................................................ 18
Lab: Set up a simple DigiMesh network .......................................... 20
Plug in the XBee module

This kit includes several XBee Grove Development Boards. For more information about this hardware, visit the XBee Grove Development Board documentation.

Follow these steps to connect the XBee devices to the boards included in the kit:

1. Plug one XBee-PRO 900HP DigiMesh Kit module into the XBee Grove Development Board.

   ! Make sure the board is NOT powered (either by the micro USB or a battery) when you plug in the XBee module.

2. Once the XBee module is plugged into the board (and not before), connect the board to your computer using the micro USB cables provided.

3. Ensure the loopback jumper is in the UART position.
How to unplug an XBee module

To disconnect your XBee module from the XBee Grove Development Board:
1. Disconnect the micro USB cable (or the battery) from the board so it is not powered.
2. Remove the XBee module from the board socket, taking care not to bend any of the pins.

Make sure the board is not powered when you remove the XBee module.

Download and install XCTU

This section contains download and install instructions based on operating system. XCTU is compatible with Linux, OSX, and Windows. It may be necessary to configure your system prior to installing XCTU for the first time.

Install XCTU - Windows

Follow the steps below to download and install XCTU on your computer.
2. Click Download XCTU.
3. Under Utilities, click the Windows installer link.
4. When the file has finished downloading, run the executable file and follow the steps in the XCTU Setup Wizard. A “What’s new” dialog appears when installation is complete.

Install XCTU - Linux

By default, access to the serial and USB ports in Linux is restricted to root and dialout group users. To access your XBee devices and use XCTU to communicate with them, your Linux user must belong to this group.

To add your Linux user to the dialout group:
1. Open a terminal console.
2. Execute this following command, where <user> is the user you want to add to the dialout group:

   ```
   sudo usermod -a -G dialout <user>
   ```
3. Log out and log in again with your user in the system.

Then download and install XCTU:
2. Click Download XCTU.
3. Under Utilities, click the Linux installer link.
4. When the file has finished downloading, run the executable file and follow the steps in the XCTU Setup Wizard. A “What’s new” dialog appears when installation is complete.
Install XCTU - OSX
OSX version 10.8 (Mountain Lion) and greater only allows you to install applications downloaded from the Apple Store. To install XCTU, you must temporarily disable this setting.

Follow these steps to enable installation of “unsigned” software:
1. Click the **Apple** icon in the top-left corner of your screen and choose **System Preferences**.
2. Click the **Security & Privacy** icon.
3. To edit security settings, click the **padlock** icon in the bottom left of the window.
4. Enter your Mac credentials and click **Unlock**. The **Allow applications downloaded from** dialog appears.
5. Click the **Anywhere** radio button and, in the confirmation window, click **Allow From Anywhere**.

**Note** We recommend you set this option back to **Mac App Store** or **Mac App Store and identified developers** once you have finished installing XCTU.

Download and install XCTU:
2. Click **Download XCTU**.
3. Under **Utilities**, click the OSX installer link.
4. When the file has finished downloading, unzip and run the executable file and follow the steps in the XCTU Setup Wizard. A “What’s new” dialog appears when installation is complete.

Optional: Install XCTU updates
When you start XCTU, you may be notified about software updates. You should always run the latest version of XCTU.

1. When a new version is available, a popup window appears in the bottom-right corner of XCTU.
2. Click on that window and follow the prompts to install the XCTU update.

You can also do the following:
- Check for updates and manually update the tool by clicking **Help > Check for XCTU Updates**.
- Check the XCTU version by clicking **Help > About XCTU**.

For more information about XCTU, see the [XCTU walkthrough](http://www.digi.com/xctu).

Optional: Manually install USB drivers
When you connect the XBee board to your computer for the first time, drivers are automatically installed. You can also install device drivers manually:

1. Download and install the appropriate USB drivers from the [Digi Support Site](http://www.digi.com/support).
2. Choose your operating system.
3. Download and run the file.
4. Follow the steps in the installation wizard.
Lab: Set up a simple DigiMesh network

Follow the steps in this section to set up your first DigiMesh network and start transmitting data between nodes. These steps include assembling the hardware, configuring the XBee modules for wireless communication, creating a network, and sending messages.

The initial network involves just two nodes, one of them sending broadcast messages. To demonstrate how easy it is to integrate additional XBee modules running DigiMesh topology, you can then connect a third module and see that it receives the messages as well.

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

Step 1: Requirements

For this setup you need the following hardware and software.

**Hardware**
- Three XBee Grove Development Boards
- Three micro USB cables
- One computer

**Software**
- XCTU 6.3.1 or later

*Tip* For more information about XCTU, see the XCTU walkthrough.

Step 2: Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

Step 3: Add the XBee modules to XCTU

Use XCTU to find your XBee modules and add them to the tool.

1. Click Discover radio modules from the toolbar.
2. In the Discover radio modules dialog, select the serial port(s) in which you want to look for radio modules. If you do not know the serial ports where your modules are attached, select all ports. Click Next.

3. In the Set port parameters window, maintain the default values and click Finish.

4. As XCTU locates radio modules, they appear in the Discovering radio modules… dialog box. Once the discovery process has finished, click Add selected devices.

5. At this point, assuming you have three modules connected to your computer, you should see something like this in the Radio Modules section on the left:

![Radio Modules](image)

Note The function, port number and the MAC address displayed for your modules need not match those shown in the picture.

**Step 4: Configure the XBee modules**

To transmit data wirelessly between your XBee modules, you must configure them to be in the same network.

1. Restore the default settings of all XBee modules with the Load default firmware settings button at the top of the Radio Configuration section.

2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBe A</th>
<th>XBe B</th>
<th>XBe C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>DH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Defines the destination address (high part) to transmit the data to.</td>
</tr>
</tbody>
</table>
3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

**Step 5: Check the network**

Once you have configured your XBee modules, use XCTU to verify that they are in the same network and can see each other.

1. Click the **Discover radio nodes in the same network** button of the first radio module.

   The device searches for radio modules in the same network.
When the discovery process is finished, XCTU lists discovered devices found within the network in the Discovering remote devices dialog. You do not need to add the remote device that has been discovered.

2. Close the dialog by clicking Cancel.

Step 6: Send wireless messages

Use the XCTU console or any serial port terminal application such as CoolTerm or TeraTerm (for Windows only) to configure SENDER to send a broadcast message every second. This example uses the XCTU console.

1. Before sending messages between the modules, disconnect RECEIVER_2. You only need two modules for this step.

2. To send broadcast messages, open XCTU if it is not already running on the computer where SENDER is connected.

3. Switch to the Consoles working mode.

   This working mode of XCTU allows you to communicate with the radio modules in the devices list. XCTU loads a list of consoles in the working area—one for each module of the devices list, sorted in a tabbed format.
4. If SENDER is not already there, add it to XCTU so it is listed in the Radio Modules list.
5. Open the serial connection of the radio module: Select the XBee in the Radio Modules section, and click the Open serial connection button.

The background changes to green to indicate that the connection is open.

6. Add a new packet with the message "Hello!"
7. To start sending this message every second, in the Send sequence box:
   a. Set 1000 ms as transmit interval.
   b. Select Loop infinitely.
   c. Click Start sequence.

You will see that the red RSSI LED of the board where RECEIVER_1 is attached blinks every second. This means that the module is successfully receiving the messages. You can also open the console of RECEIVER_1 and see that the messages are being received.

8. Plug in RECEIVER_2. You will see that its red RSSI LED also blinks every second because it has successfully joined the network. You can also open the console of RECEIVER_2 and see that the messages are being received.
How XBee devices work

This section describes how XBee devices communicate, and introduces two communication methods - wireless and serial communication. Both communication types are important in the function of XBee devices. For an example of wireless and serial communication, see Lab: Set up a simple DigiMesh network.

How XBee devices communicate ................................................................. 26
Wireless communication ........................................................................ 26
Serial communication ............................................................................ 29
How XBee devices communicate

XBee devices communicate with each other over the air, sending and receiving wireless messages. The devices only transfer those wireless messages; they cannot manage the received or sent data. However, they can communicate with intelligent devices via the serial interface.

XBee devices transmit data coming from the serial input over the air, and they send anything received wirelessly to the serial output. Whether for communication purposes or simply for configuring the device, a combination of both processes makes XBee communication possible. In this way, intelligent devices such as microcontrollers or PCs can control what the XBee device sends and manage incoming wireless messages.

With this information, you can identify the two types of wireless data transmission in an XBee communication process:

1. **Wireless communication**: This communication takes place between XBee modules. Modules that are supposed to work together need to be part of the same network and they must use the same radio frequency. All modules that meet these requirements can communicate wirelessly with each other.

2. **Serial communication**: This communication takes place between the XBee module and the intelligent device connected to it through the serial interface.

Wireless communication

XBee modules communicate with each other over the air, transmitting and receiving information via modulation of waves in the electromagnetic spectrum. In other words, they act as radio frequency (RF) devices. For data to transmit from one XBee module to another, both modules must be in the same network.

This section describes the key concepts to understand as you learn how to manage a network and transmit information between XBee modules.

Addressing

XBee device addresses are similar to postal and email addresses for people. Some addresses are unique, like an email address, but others are not. For example, several people can live at the same postal address.

Each XBee device is known by several different addresses, each of which serves a purpose.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>0013A20012345678</td>
<td>Always</td>
</tr>
<tr>
<td>16-bit</td>
<td>1234</td>
<td>Yes, but only within a network</td>
</tr>
<tr>
<td>Node identifier</td>
<td>Bob's module</td>
<td>Uniqueness not guaranteed</td>
</tr>
</tbody>
</table>
**64-bit address**

Every XBee device has a 64-bit address to distinguish it from others and prevent duplicate information. That address (also called MAC) is assigned to Digi by the IEEE and is guaranteed to be **unique**, so two modules cannot have the same address.

You can determine the value of the 64-bit address by reading the Serial Number High (SH) and Serial Number Low (SL) parameters on any module. It is also printed on the back of the module.

---

**Note** The concatenation of SH + SL forms the 64-bit or MAC address of the device. It is stored in the device's memory as two 32-bit values: the high part, SH, and the low part, SL. The high part is usually the same for all XBee devices (0013A200), as this is the prefix that identifies Digi devices. The low part is different for every device.

The 64-bit address of **000000000000FFFF** is reserved for sending a broadcast message.

**Node identifier**

The node identifier is a short string of text that allows users to address the module with a more human-friendly name. In this case, uniqueness is not guaranteed because you can assign the same node identifier to several modules.

You can read or set the value of the node identifier through the Node Identifier (NI) parameter.

**Network identifier**

DigiMesh networks are identified by a unique network identifier. This identifier is common among all XBee modules in the same network. Only modules with the same network identifier can communicate with each other.

You can set the value of the network identifier through the ID parameter. This setting allows multiple DigiMesh networks to coexist on the same physical channel.

---

Separate networks in physical range of each other should use different Preamble Identifier (HP) and/or Network Identifier (ID) parameters to avoid receiving data from the other network.

---

**Preamble identifier**

The preamble identifier minimizes interference between multiple groups of XBee modules operating in the same vicinity. This identifier is common among all XBee modules in the same network. Only modules with matching preamble identifiers can communicate with each other.
The preamble identifier is encoded in the preamble, as its name suggests, while the network identifier is encoded in the MAC header. Thus, when a module is receiving wireless data, it checks the value of the preamble identifier before the network identifier.

You can set the value of the preamble identifier through the **HP** parameter.

---

Separate networks in physical range of each other should use different Preamble Identifier (**HP**) and/or Network Identifier (**ID**) parameters to avoid receiving data from the other network.

---

**Radio frequency channels and channel masks**

To communicate with each other, radio frequency (RF) devices must operate in the same frequency. XBee devices support a different number of channels depending on the device’s region of operation. The Available Frequencies (**AF**) parameter specifies the available physical channels. Each bit represents a channel; when that bit is set to 1 the channel is available. See the "Regional frequencies and channels" table below.

You cannot explicitly establish the channel in which your device(s) work. Rather, you configure a list of channels to scan with the Channel Mask (**CM**) parameter.

**Note** All devices in a network must use an identical set of active channels specified in the Channel Mask (**CM**) parameter.

The Channel Mask parameter is a bitfield: each bit corresponds to a frequency as defined in the Available Frequencies setting. When a bit in the Channel Mask and the corresponding bit in the Available Frequencies are both set to 1, that physical channel is enabled and the module can select it as an active channel for communication.

You must enable a minimum number of channels with the Channel Mask parameter. This minimum value is determined by the Minimum Frequencies (**MF**) parameter and also depends on the module’s region of operation. See the "Minimum frequencies by region" table below.

**Regional frequencies and channels**

The Available Frequencies (**AF**) parameter specifies the available physical channels in the device’s region of operation. Each bit represents a channel; when that bit is set to 1 the channel is available.

<table>
<thead>
<tr>
<th>Region</th>
<th>AF</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA/Canada</td>
<td>00FFFFFFFFFFFFFFFF</td>
<td>0 - 63</td>
</tr>
<tr>
<td>Australia</td>
<td>00FFFFFFFFFE00000000</td>
<td>33 - 63</td>
</tr>
<tr>
<td>Brazil</td>
<td>00FFFFFFFFFE000000FF</td>
<td>0 - 11, 33 - 63</td>
</tr>
<tr>
<td>Singapore</td>
<td>00000FFE000000000000</td>
<td>45 - 55</td>
</tr>
</tbody>
</table>

**Minimum frequencies by region**

The Minimum Frequencies (**MF**) parameter determines the minimum number of channels that must be enabled with the Channel Mask (**CM**) parameter for proper operation. This value depends on the device’s region of operation.
How XBee devices work

Serial communication

An XBee module can operate as a stand-alone device or it can be attached to an intelligent device. For example, you can place several battery-powered XBee modules in remote locations to gather data such as temperature, humidity, light, or liquid level.

- When operating as a stand-alone device, an XBee module simply sends sensor data to a central node.
- When an XBee module is connected to an intelligent device (such as a computer, Arduino, or Raspberry Pi), it uses serial communication:
  - The intelligent device sends data through the serial interface to the XBee module to be transmitted to other devices over the air.
  - The XBee module receives wireless data from other devices, and then sends the data through the serial interface to the intelligent device.

The XBee modules interface to a host device such as a microcontroller or computer through a logic-level asynchronous serial port. They use a UART for serial communication with those devices.

For additional information about serial communication, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

Microcontrollers attached to an XBee module can process the information received by the module and thus monitor or even control remote devices by sending messages through their local XBee module. For prototyping, you can use external microcontrollers such as Arduino or Raspberry Pi, sockets, and breadboards.

<table>
<thead>
<tr>
<th>Region</th>
<th>MF - Minimum number of channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA/Canada</td>
<td>25</td>
</tr>
<tr>
<td>Australia</td>
<td>25</td>
</tr>
<tr>
<td>Brazil</td>
<td>25</td>
</tr>
<tr>
<td>Singapore</td>
<td>11</td>
</tr>
</tbody>
</table>

The boards included in this kit allow you to use the XBee modules in either mode:

- If you plug the modules into the boards and connect them to a computer or microcontroller using the micro USB cables, you can configure the XBee modules, test the connection, and send/receive data to/from other modules.
- If you plug the modules into the boards and connect them to a battery, the XBee modules work autonomously. For example, they can gather data from a sensor and send it to a central node.
Operating modes
XBee devices can use their local serial connection in very different ways. The "operating mode" establishes the way the host device communicates with an XBee module through the serial interface.

XBee modules support two different operating modes:
- Application Transparent ("transparent mode")
- Application Programming Interface ("API mode")

Application Transparent operating mode
This mode is called "transparent" because the radio passes information along exactly as it receives it. All serial data received by the radio module is sent wirelessly to a remote destination XBee module. When the other module receives the data, it is sent out through the serial port exactly as it was received. Transparent mode has limited functionality but is an easy way to get started with XBee devices.

To learn more about transparent mode, see XBee transparent mode.

API operating mode
Application Programming Interface (API) operating mode is an alternative to transparent mode. In API mode, a protocol determines the way information is exchanged. Data is communicated in packets (commonly called API frames). This mode allows you to form larger networks and is more appropriate for creating sensor networks to perform tasks such as collecting data from multiple locations, controlling devices remotely, or automating your home.

To learn more about API mode, see XBee API mode.

Comparison of transparent and API modes
XBee devices can use transparent or API operating mode to transmit data over the serial interface. You can use a mixture of devices running API mode and transparent mode in a network. The following
table provides a comparison of the two modes.

<table>
<thead>
<tr>
<th>Transparent operating mode</th>
<th>API operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>When to use:</strong></td>
<td><strong>When to use:</strong></td>
</tr>
<tr>
<td>■ Conditions for using API mode do not apply.</td>
<td>■ Sends wireless data to multiple destinations.</td>
</tr>
<tr>
<td>■ Sends wireless data to multiple destinations.</td>
<td>■ Configures remote XBee devices in the network.</td>
</tr>
<tr>
<td>■ Receives wireless data packets from multiple XBee devices, and the application needs to identify which devices send each packet.</td>
<td>■ Receiving I/O samples from remote XBee devices.</td>
</tr>
<tr>
<td>■ Receives I/O samples from remote XBee devices.</td>
<td>■ Must support multiple endpoints, clusters, and/or profiles (for Zigbee modules).</td>
</tr>
<tr>
<td>■ Uses Zigbee Device Object (ZDO) services (for Zigbee modules).</td>
<td>■ Received data includes the sender’s address.</td>
</tr>
</tbody>
</table>

Advantages:

- Provides a simple interface that makes it easy to get started with XBee devices.
- Easy for an application to support; what you send is exactly what other modules get, and vice versa.
- Works very well for two-way communication between XBee devices.

Advantages:

- Can set or read the configuration of remote XBee devices in the network.
- Can transmit data to one or multiple destinations; this is much faster than transparent mode where the configuration must be updated to establish a new destination.
- Received data includes transmission details and reasons for success or failure.
- Several advanced features, such as advanced networking diagnostics, and firmware upgrades.
### Transparent operating mode

**Disadvantages:**
- Cannot set or read the configuration of remote XBee devices in the network.
- Must first update the configuration to establish a new destination and transmit data.
- Cannot identify the source of received data, as it does not include the sender's address.
- Received data does not include transmission details or the reasons for success or failure.
- Does not offer the advanced features of API mode, including advanced networking diagnostics, and firmware upgrades.

### API operating mode

**Disadvantages:**
- Interface is more complex; data is structured in packets with a specific format.
- More difficult to support; transmissions are structured in packets that need to be parsed (to get data) or created (to transmit data).
- Sent data and received data are not identical; received packets include some control data and extra information.
XBee transparent mode

This section provides additional detail about XBee transparent mode. For a comparison of transparent and API modes, see Serial communication.

XBee transparent mode in detail ................................................................. 34
Command mode ......................................................................................... 34
XBee transparent mode in detail

When operating in transparent mode, an XBee module acts as a serial line replacement. All data received through the serial input is immediately transmitted over the air. When the XBee module receives wireless data, it is sent out through the serial interface exactly as it is received. In fact, communication in transparent mode yields the same result as if the two modules were connected by a wire, but wireless communication makes that physical wire unnecessary.

For two XBee modules to communicate, the sending module needs the address of the recipient. When working in transparent mode, you must configure this address in the module that is communicating. XBee modules can store the complete 64-bit address of the destination module. This address must be programmed in two parameters: Destination Address High (DH) and Destination Address Low (DL).

If you want modules A and B to communicate, configure the destination address (DH + DL) of XBee A as the MAC address (SH + SL) of XBee B, and vice versa.

<table>
<thead>
<tr>
<th>XBee B</th>
<th>XBee A</th>
</tr>
</thead>
<tbody>
<tr>
<td>SH Serial Number High</td>
<td>13A200</td>
</tr>
<tr>
<td>SL Serial Number Low</td>
<td>40A15861</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>XBee A</td>
<td>XBee B</td>
</tr>
<tr>
<td>SH Serial Number High</td>
<td>13A200</td>
</tr>
<tr>
<td>SL Serial Number Low</td>
<td>40A142E</td>
</tr>
</tbody>
</table>

Transparent mode has some limitations. For example, when you are working with several modules, you must configure the destination before sending each message. However, transparent mode provides an easy way to get started with XBee devices for the following reasons:

- Operation is very simple.
- What you send is exactly what the other modules get.
- Compatible with any device that can communicate over a serial interface.
- Works very well when facilitating communication between two XBee modules.

Command mode

An XBee device in Transparent mode simply passes information along exactly as it receives it. So, what you send is what other devices get. But sometimes you want to talk directly to the local device without sending data. For example, you may need to modify its configuration or alter the way it behaves. In that case, the XBee device needs to know that this communication should not be transmitted wirelessly.

Command mode is a state in which incoming characters are interpreted as commands. To get a device to switch into this mode, you must issue a unique string of text in a special way: ++++. When the device sees a full second of silence in the data stream followed by the string ++++ (without Enter or Return) and another full second of silence, it knows to stop sending data through and start accepting commands locally.
Guard time silence | Command sequence | Guard time silence
--- | --- | ---
One second before | +++ | One second after

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

Once the device is in Command mode, it listens for user input for a while. If 10 seconds go by without any user input, the device automatically drops out of Command mode and returns to Transparent mode.

AT commands
The purpose of command mode is to read or change the configuration of the local XBee device. Every module has a number of settings, like channel or network ID, that define its behavior. These settings are identified by two characters, for example, CH for channel, and ID for network ID.

When you want to read or set any setting of the XBee module, you must send it an AT command. Every AT command starts with the letters "AT" followed by the two characters that identify the command being issued and then by some optional configuration values.

```
“AT” Prefix  +  Setting  +  Space (optional)  +  Value (optional, HEX)  +  Carriage Return

ATID 2015<CR>
```

For example, to read and set the network ID setting:

```
// Enter command mode
+++OK

// Read the ID setting
ATID <Enter>
0

// Change the ID setting
ATID 2015 <Enter>
OK
```

Basic AT commands
- **AT**

This command checks the connection with the module. This is like asking "Are you there?" and the device replying "Yes." When you send this command, the module simply replies OK. If you don't see an OK in response, you have probably timed out of command mode. Type the +++ to go back into it.
XBee transparent mode

- **ATCN**
  This command explicitly exits the module from command mode. Remember that if you don't type anything for 10 seconds, the device automatically drops out of Command mode.

- **ATWR**
  This command writes the current configuration to non-volatile memory so that it persists the next time the device powers up. Otherwise, parameters are restored to previously saved values after the device is reset.

For the complete list of AT commands supported by the modules included in this kit, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

**Use AT commands**

In the first example in this kit, you used XCTU to configure some settings of each of your modules, such as the network ID. XCTU uses AT commands in the background to read and set the settings. For example, when you changed the value of that parameter and clicked the Write button, XCTU went into command mode using +++; changed the value of the setting with the ATID command, wrote the setting with the ATWR command, and finally exited command mode with the ATCN command.

XCTU simplifies the configuration of the XBee modules so you don't have to use command mode or AT commands to configure them. However, you can always configure an XBee module through any serial port terminal application or the XCTU console.

The following example demonstrates how you can perform some of the configuration steps outlined in the first lab but via command mode and using AT commands:

1. In the Consoles working mode of XCTU, click the **Open the serial connection with the radio module** button.
2. Use +++ to enter into command mode and wait for an OK response.
3. To set a register, type an AT command followed by the value you want to set; for example, ATID 2015; followed by a Return.
4. To read a register, type an AT command; for example, ATID; followed by a Return.
5. Use the ATWR command to write the new configuration to the module's memory.
6. Exit command mode with the ATCN command.

**Note** You should get an OK response after issuing each command to set parameters, write the changes, or exit from command mode. If not, you most likely took more than 10 seconds to issue the command and you have dropped out of command mode.
XBee API mode

This section provides additional detail about API mode and lets you put your knowledge into practice. For a comparison of transparent and API modes, see Serial communication.

API mode in detail .................................................................39
API frame structure ..............................................................40
DigiMesh supported frames ..................................................42
Operating mode configuration ..............................................48
XBee frame exchange ..........................................................50
Do more with API mode: XBee libraries ..................................60
API mode in detail

API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol.

By default, XBee devices are configured to work in transparent mode: all data received through the serial input is queued up for radio transmission and data received wirelessly is sent to the serial output exactly as it is received, with no additional information.

Because of this behavior, devices working in Transparent mode have some limitations:

1. To read or write the configuration of an device in Transparent mode, you must first transition the device into Command mode.
2. If a device needs to transmit messages to different devices, you must update its configuration to establish a new destination. The device must enter Command mode to set up the destination.
3. A device operating in Transparent mode cannot identify the source of a wireless message it receives. If it needs to distinguish between data coming from different devices, the sending devices must include extra information known by all the devices so it can be extracted later. To do this, you must define a robust protocol that includes all the information you think you need in your transmissions.

To minimize the limitations of the transparent mode, devices provide an alternative mode called Application Programming Interface (API). API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between modules without having to define your own protocol.

API mode provides a much easier way to perform the actions listed above:

1. Since there are different frames for different purposes (such as configuration and communication), you can configure a device without entering Command mode.
2. Since the data destination is included as part of the API frame structure, you can use API mode to transmit messages to multiple devices.
3. The API frame includes the source of the message so it is easy to identify where data is coming from.

Advantages of API mode

- Configure local and remote XBee devices in the network.
- Manage wireless data transmission to one or multiple destinations.
- Identify the source address of each received packet.
XBee API mode

- Receive success/failure status of each transmitted packet.
- Obtain the signal strength of any received packet.
- Perform advanced network management and diagnosis.
- Perform advanced functions such as remote firmware update, ZDO, ZCL and so on.

API frame structure

The structured data packets in API mode are called frames. They are sent and received through the serial interface of the device and contain the wireless message itself as well as some extra information such as the destination/source of the data or the signal quality.

When a device is in API mode, all data entering and leaving the module through the serial interface is contained in frames that define operations or events within the device.

An API frame has the following structure:

<table>
<thead>
<tr>
<th>Start delimiter</th>
<th>Length</th>
<th>Frame data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0x7E</td>
<td>MSB</td>
<td>LSB</td>
<td>API-specific structure</td>
</tr>
</tbody>
</table>

Note: MSB represents the most significant byte, and LSB represents the least significant byte.

Any data received through the serial interface prior to the start delimiter is silently discarded by the XBee. If the frame is not received correctly, or if the checksum fails, the data is also discarded and the module indicates the nature of the failure by replying with another frame.

Start delimiter

The start delimiter is the first byte of a frame consisting of a special sequence of bits that indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.

Length

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

Frame data

This field contains the information received or to be transmitted. Frame data is structured based on the purpose of the API frame:
<table>
<thead>
<tr>
<th>Start delimiter</th>
<th>Length</th>
<th>Frame type</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0x7E</td>
<td>MSB</td>
<td>LSB</td>
<td>API frame type</td>
<td>Frame-type-specific data</td>
</tr>
</tbody>
</table>

**Note** MSB represents the most significant byte, and LSB represents the least significant byte.

- **Frame type** is the API frame type identifier. It determines the type of API frame and indicates how the information is organized in the Data field.
- **Data** contains the data itself. The information included here and its order depends on the type of frame defined in the Frame type field.

**Checksum**

Checksum is the last byte of the frame and helps test data integrity. It is calculated by taking the hash sum of all the API frame bytes that came before it, excluding the first three bytes (start delimiter and length).

**Note** Frames sent through the serial interface with incorrect checksums will never be processed by the module and the data will be ignored.

**Calculate the checksum of an API frame**

1. Add all bytes of the packet, excluding the start delimiter 0x7E and the length (the second and third bytes).
2. From the result, keep only the lowest 8 bits.
3. Subtract this quantity from 0xFF.

Example: Checksum calculation

To calculate the checksum for the given frame:

<table>
<thead>
<tr>
<th>Start Delimiter</th>
<th>Length</th>
<th>Frame Data</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E</td>
<td>00</td>
<td>0F</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Add all bytes excluding the start delimiter and the length: $17 + 01 + 00 + 13 + A2 + 00 + 40 + AD + 14 + 2E + FF + FE + 02 + 44 + 42 = 481$
2. From the result, keep only the lowest 8 bits: 81.
3. Subtract that result from 0xFF: FF - 81 = 7E

In this example, 0x7E is the checksum of the frame.

**Verify the checksum of a given API frame**

1. Add all bytes including the checksum (do not include the delimiter and length).
2. If the checksum is correct, the last two digits on the far right of the sum will equal FF.

Example: Checksum verification

In our example above, we want to verify the checksum is 7E.

<table>
<thead>
<tr>
<th>Start Delimiter</th>
<th>Length</th>
<th>Frame Type</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E</td>
<td>00</td>
<td>17</td>
<td>01A01204AD12FE0442</td>
<td>7E</td>
</tr>
</tbody>
</table>

1. Add all data bytes and the checksum: $17 + 01 + 00 + 13 + A2 + 00 + 40 + AD + 14 + 2E + FF + FE + 02 + 44 + 42 + 7E = 4FF$
2. Since the last two far right digits of 4FF are FF, the checksum is correct.

**DigiMesh supported frames**

Support for API frame types depends on the type of XBee device you are using. The XBee DigiMesh modules included in this kit support the following API frames.

Transmit data frames are sent through the serial input, with data to be transmitted wirelessly to remote XBee modules:

<table>
<thead>
<tr>
<th>API ID</th>
<th>Frame name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08</td>
<td>AT Command</td>
<td>Queries or sets parameters on the local XBee module</td>
</tr>
<tr>
<td>0x09</td>
<td>AT Command Queue Parameter Value</td>
<td>Queries or sets parameters on the local XBee module without applying changes</td>
</tr>
<tr>
<td>0x10</td>
<td>Transmit Request</td>
<td>Transmits wireless data to the specified destination</td>
</tr>
<tr>
<td>0x11</td>
<td>Explicit Addressing Command Frame</td>
<td>Allows application layer fields (endpoint and cluster ID) to be specified for a data transmission</td>
</tr>
<tr>
<td>0x17</td>
<td>Remote AT Command Request</td>
<td>Queries or sets parameters on the specified remote XBee module</td>
</tr>
</tbody>
</table>

Receive data frames are received through the serial output, with data received wirelessly from remote XBee modules:
**API ID** | **Frame name** | **Description**
---|---|---
0x88 | AT Command Response | Displays the response to previous AT command frame
0x8A | Modem Status | Displays event notifications such as reset, association, and disassociation
0x8B | Transmit Status | Indicates data transmission success or failure
0x8D | Route information packet | Displays route information for a wireless transmission
0x8E | Aggregate addressing update | Indicates the node updated its DH and DL parameters due to the execution of AG command
0x90 | Receive Packet | Sends wirelessly received data out the serial interface (AO = 0)
0x91 | Explicit Rx Indicator | Sends wirelessly received data out the serial interface when explicit mode is enabled (AO ≠ 0)
0x92 | IO Data Sample Rx Indicator | Sends wirelessly received IO data out the serial interface
0x95 | Node Identification Indicator | Displays received node identification message when explicit mode is disabled (AO = 0)
0x97 | Remote AT Command Response | Displays the response to previous remote AT command requests

**Note** To learn more about the structure of some of these frames, see go to the XBee-PRO 900HP and XSC RF Modules User Guide.

### Frame examples

The following examples of sent and received API frames are expressed in hexadecimal format.

**Example: 0x10 - Transmit Request**
The following frame is a Transmit Request frame with the following characteristics:

```
7E 00 13 10 01 00 13 A2 00 40 DA 9D 23 FF FE 00 00 48 65 6C 6C 6F 6E
```

- The frame ID is 0x01, so the sender will receive a Transmit Status frame with the result of the transmission.
- The destination XBee module has a 64-bit address of **00 13 A2 00 40 DA 9D 23**.
- It does not specify any option.
- The data to transmit is 'Hello' (**48 65 6C 6C 6F**).
**XBee API mode**

**DigiMesh supported frames**

<table>
<thead>
<tr>
<th>Frame fields</th>
<th>Offset</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>MSB 1</td>
<td>0x00</td>
<td>Number of bytes between the length and the checksum</td>
</tr>
<tr>
<td></td>
<td>LSB 2</td>
<td>0x13</td>
<td></td>
</tr>
</tbody>
</table>
### XBee API mode

**DigiMesh supported frames**

<table>
<thead>
<tr>
<th>Frame fields</th>
<th>Offset</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame type</td>
<td>3</td>
<td>0x10</td>
<td>0x10 - Indicates this is a <em>Transmit Request</em> frame</td>
</tr>
<tr>
<td>Frame ID</td>
<td>4</td>
<td>0x01</td>
<td>Identifies the data frame for the host to correlate with a subsequent <em>Transmit Status</em> (0x8B) frame Setting Frame ID to '0' will disable response frame</td>
</tr>
<tr>
<td>64-bit Destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSB</td>
<td>5</td>
<td>0x00</td>
<td>Set to the 64-bit address of the destination XBee</td>
</tr>
<tr>
<td>6</td>
<td>0x13</td>
<td></td>
<td>The following addresses are also supported:</td>
</tr>
<tr>
<td>7</td>
<td>0xA2</td>
<td></td>
<td>- 0x000000000000FFFF - Broadcast address</td>
</tr>
<tr>
<td>8</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0x40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0xDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0x9D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSB</td>
<td>12</td>
<td>0x23</td>
<td></td>
</tr>
<tr>
<td>16-bit Destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSB</td>
<td>13</td>
<td>0xFF</td>
<td>Set to 0xFFFF</td>
</tr>
<tr>
<td>LSB</td>
<td>14</td>
<td>0xFE</td>
<td></td>
</tr>
<tr>
<td>Broadcast Radius</td>
<td>15</td>
<td>0x00</td>
<td>Sets the maximum number of hops a broadcast transmission can occur. If set to '0', the broadcast radius will be set to the maximum hops value</td>
</tr>
<tr>
<td>Options</td>
<td>16</td>
<td>0x00</td>
<td>Bitfield of supported transmission options Bit values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bit 0 - Disable ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bit 1 - Disable Route Discovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bit 2 - Enable Unicast NACK messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bit 3 - Enable Unicast Trace Route messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bits 6,7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 01 - Point-Multipoint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 10 - Repeater mode (directed broadcast)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 11 - DigiMesh (not available on 10k product)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All other bits must be set to '0'. If this bitfield is '0', then the TO parameter will be used</td>
</tr>
</tbody>
</table>
## XBee API mode

<table>
<thead>
<tr>
<th>Frame fields</th>
<th>Offset</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF Data</td>
<td>MSB</td>
<td>0x48</td>
<td>Data that is sent to the destination XBee</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>0x65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0x6C</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>LSB</td>
<td>0x6F</td>
<td>Hash sum of frame data bytes</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>0x6E</td>
<td></td>
</tr>
</tbody>
</table>

### Example: 0x91 - Explicit Rx Indicator

The following frame is an Explicit Rx Indicator frame with the following characteristics:

- The XBee module that sent this data has a 64-bit address of **00 13 A2 00 40 DA 9D 05**.
- The endpoint of the source that initiated the transmission is **E8** and the destination endpoint is **E8**.
- The Cluster ID the data is addressed to is **00 11**.
- The Profile ID the data is addressed to is **C1 05**.
- The packet was acknowledged because the Receive options value is **01**.
- The received data 'Hello' is **(48 65 6C 6C 6F)**.

<table>
<thead>
<tr>
<th>Frame fields</th>
<th>Offset</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start delimiter</td>
<td>0</td>
<td>0x7E</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>MSB 1</td>
<td>0x00</td>
<td>Number of bytes between the length and the checksum</td>
</tr>
<tr>
<td></td>
<td>LSB 2</td>
<td>0x17</td>
<td></td>
</tr>
<tr>
<td>Frame fields</td>
<td>Offset</td>
<td>Example</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------</td>
<td>---------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Frame data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame type</td>
<td>3</td>
<td>0x91</td>
<td>0x91 - Indicates this is a <em>Explicit Rx Indicator</em> frame</td>
</tr>
<tr>
<td>64-bit Source Address</td>
<td>4</td>
<td>0x00</td>
<td>64-bit address of sender</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0xA2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0xDA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0x9D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSB</td>
<td>0x05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit Source Network Address</td>
<td>MSB</td>
<td>0xFF</td>
<td>Set to 0xFFFE</td>
</tr>
<tr>
<td></td>
<td>LSB</td>
<td>0xFE</td>
<td></td>
</tr>
<tr>
<td>Source Endpoint</td>
<td>14</td>
<td>0xE8</td>
<td>Endpoint of the source that initiated the transmission</td>
</tr>
<tr>
<td>Destination Endpoint</td>
<td>15</td>
<td>0xE8</td>
<td>Endpoint of the destination the message is addressed to</td>
</tr>
<tr>
<td>Cluster ID</td>
<td>16</td>
<td>0x00</td>
<td>Cluster ID the message was addressed to</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>0x11</td>
<td></td>
</tr>
<tr>
<td>Profile ID</td>
<td>18</td>
<td>0xC1</td>
<td>Profile ID the message was addressed to</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>0x05</td>
<td></td>
</tr>
</tbody>
</table>
### Operating mode configuration

The API Enable (AP) parameter configures the XBee module to operate using a frame-based API instead of the default Transparent mode. It allows you to select between the two supported API modes and the default transparent operation.

<table>
<thead>
<tr>
<th>Mode</th>
<th>AP value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>0</td>
<td>API modes are disabled and the module operates in transparent mode</td>
</tr>
<tr>
<td>API 1</td>
<td>1</td>
<td>API mode without escaped characters</td>
</tr>
<tr>
<td>API 2</td>
<td>2</td>
<td>API mode with escaped characters</td>
</tr>
</tbody>
</table>

The only difference between API 1 and API 2 is that API 2 operating mode requires that frames use escape characters (bytes).

Configuration of the serial XBee communication—whether it is transparent, API non-escaped (API 1), or API escaped (API 2)—does not prevent wireless communication between XBee modules. Since only the payload portion of the API frame is transmitted over the air, the receiving XBee modules will alter the packet information based on their AP setting, allowing an API non-escaped module to successfully communicate with others working in API escaped or Transparent mode.
Note Devices working in Transparent mode and modules set to API non-escaped (API 1) operation can communicate with devices configured to work in API escaped mode (API 2).

API escaped operating mode (API 2)

API non-escaped (API 1) operation relies solely on the start delimiter and length bytes to differentiate API frames. If bytes in a packet are lost, the length count will be off, and the next API frame (packet) will also be lost. API escaped (API 2) operation involves escaping character sequences in an API frame in order to improve reliability, especially in noisy RF environments.

The basic frame structure of both API modes is the same, but in API escaped (API 2) mode, all bytes except for the start delimiter must be escaped if needed. The following data bytes must be escaped in API 2 mode:
- 0x7E: Start delimiter
- 0x7D: Escape character
- 0x11: XON
- 0x13: XOFF

API 2 mode guarantees all the 0x7E bytes received are start delimiters: this character cannot be part of any of the other frame fields (length, data, or checksum) since it must be escaped.

To escape a character:
1. Insert 0x7D, the escape character.
2. Append it with the byte to be escaped, XORed with 0x20.

In API 2 mode, the length field does not include any escape character in the frame and the checksum is calculated with non-escaped data.

Example: Escape an API frame
To express the following API non-escaped frame in API 2 mode:

<table>
<thead>
<tr>
<th>Start Delimiter</th>
<th>Length</th>
<th>Frame Type</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E</td>
<td>00</td>
<td>0F</td>
<td>1A</td>
<td>6D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0E</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4E</td>
<td></td>
</tr>
</tbody>
</table>

The 0x13 byte must be escaped:
1. Insert a 0x7D.
2. XOR the byte 0x13 with 0x20: 13 ⊕ 20 = 33.

This is the resulting frame. Note that the length and checksum are the same as the non-escaped frame.
**XBee frame exchange**

Now that you understand how API mode works and how API frames are structured, the next step is to learn about how frames are exchanged when you perform certain common operations such as configuring an XBee module or transmitting wireless data.

The following section provides examples using XCTU. You can use the Frames interpreter tool of the XCTU console to view detailed API frame structure.

**AT Command: configure a local XBee device**

To query or set the value of the local XBee—that is, the device directly connected to an intelligent device such as a microcontroller or PC via the serial interface—you must use AT parameters and commands. These are the same AT parameters and commands that are available in Transparent/Command mode, but included in an AT Command (0x08) frame. The response containing the result of the operation is sent back in an AT Command Response (0x88) frame.

The following image shows the API frame exchange that takes place at the serial interface when sending either an AT Command (0x08) or an AT Command Queue Parameter Value (0x09) request.

During an API frame exchange, the following process occurs:

1. An AT Command (0x08) frame is sent to the device through the serial input. This frame contains configuration instructions or queries parameters on the local XBee device.

2. The XBee device processes the command and returns an AT Command Response (0x88) through its serial output. If the frame ID of the AT Command frame is 0, this response is not sent.

This section demonstrates how to read the Node Identifier (NI) of your local XBee module configured in API mode. To do this, you create an AT command frame to read the NI parameter, send it to the XBee module, and analyze the response.

<table>
<thead>
<tr>
<th>Start Delimiter</th>
<th>Length</th>
<th>Frame Type</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E</td>
<td>00</td>
<td>0F</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>7</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6D</td>
</tr>
</tbody>
</table>
**Step 1: Configure the XBee module**

Before creating and sending the frame, configure the XBee module as follows:

<table>
<thead>
<tr>
<th>Param</th>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
</table>
| NI    | XBEE_A  | Defines the node identifier, a human-friendly name for the module.  
|       |         | The default NI value is a blank space. Make sure to delete the space when you change the value. |

**Step 2: Open the XCTU console**

1. Switch to the **Consoles working mode**.
2. Open the serial connection with the radio module.

**Step 3: Generate the AT command frame**

These instructions describe how to generate an AT command frame using the XCTU Frame Generator tool.

1. Click **Add new frame to the list**.
2. Open the **Frames Generator** tool.
3. In the **Frame type** section, select 0x08 - AT Command.
4. In the **AT command** section, select the **ASCII** tab and type NI.
5. Click **OK**.
6. Click **Add frame**.
Step 4: Send the AT command frame

After you have created an AT command frame, you must send it to the local XBee module to receive a response containing the configured **NI** value.

1. Select the frame in the XCTU **Send frames** section.
2. Click **Send selected packet**.

The **Frames log** indicates that one frame has been sent (blue) and another has been received (red).

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Len</th>
<th>Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>21:21:03.648</td>
<td>4</td>
<td>AT Command</td>
</tr>
<tr>
<td>1</td>
<td>21:21:03.710</td>
<td>11</td>
<td>AT Command Response</td>
</tr>
</tbody>
</table>
**Step 5: Analyze the response**

Once you have sent the frame, you can analyze the responses on the receiving end.

1. Select the frame received (AT Command Response) to see its details in the Frame details section.

2. Analyze its details and verify that it contains the NI value of your module.
   - **Frame type**: The received frame is an AT Command Response.
   - **Frame ID**: This AT Command Response frame is the answer to the sent AT Command request because both have the same value (1).
   - **Status**: The value was successfully read because the status is OK.
   - **Response**: This received frame contains the value of the NI parameter previously requested in the AT Command frame, XBEE_A

3. Disconnect the console by clicking Close the serial connection.

**Transmit Request/Receive Packet: Transmit and receive wireless data**

A Transmit Request frame encapsulates data with its remote destination and some transmission options. The wireless data received by an XBee module is included in a Receive Packet frame along with the remote transmitter and options for receipt.

Two more frames use Explicit addressing. They require that you specify application layer addressing fields (endpoints, cluster ID, profile ID).

For more information about Explicit addressing, see the Zigbee communication in depth chapter.

The following image shows the API exchanges that take place at the serial interface when transmitting wireless data to another XBee module.
1. The intelligent device (host) sends a Transmit Request (0x10) or an Explicit Addressing Command Frame (0x11) to XBee A through the serial input to transmit data to XBee B.

2. XBee A wirelessly transmits the data in the frame to the module configured as destination in the same frame; in this case, the destination is XBee B.

3. The remote XBee B module receives the wireless data and sends out through the serial output a Receive Packet (0x90) or an Explicit Rx Indicator (0x91), depending on the value of API Options (AO) setting. These frames contain the data received over the air and the source address of the XBee module that transmitted it, in this case XBee A.

4. The remote XBee B module transmits a wireless acknowledge packet with the status to the sender, XBee A.

5. The sender XBee A module sends out a Transmit Status (0x8B) through its serial output with the status of the transmission to XBee B.

The Transmit Status (0x8B) frame is always sent at the end of a wireless 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.

To send data using an explicit frame:
- The source and destination endpoints must be E8.
- The cluster ID must be 0011.
- The profile ID must be C105.

To receive an explicit frame, the API Options (AO) parameter must be configured to API Explicit Rx Indicator - 0x91 [1]. If this setting is API Rx Indicator - 0x90 [0], a Receive Packet (0x90) will be received instead of an Explicit Rx Indicator (0x91).

**Lab: Transmit and receive data**

This section describes how to transmit data to another XBee module using the XCTU console. The steps include creating a Transmit Request frame with the message you want to transmit to the other module and sending the frame serially to the local XBee module. You can then analyze the responses, both in the local and the remote module.

If you get stuck, see **Troubleshooting for XBee ZigBee Mesh Kit**.

**Step 1: Configure the XBee modules**

Before creating and sending the frame, configure the XBee modules as follows:
<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios on your network.</td>
</tr>
</tbody>
</table>
| NI    | SENDER | RECEIVER | Defines the node identifier, a human-friendly name for the module.  
⚠️ The default NI value is a blank space. Make sure to delete the space when you change the value. |

**Step 2: Open the XCTU console**

1. Switch to the **Consoles working mode**.
2. Open the serial connection with the radio module.
3. Change to the console of the other XBee module.
4. Open the serial connection with the radio module.
5. 

**Step 3: Generate the Transmit Request frame**

This topic describes how to generate a Transmit Request frame using the XCTU SENDER console.

1. Go to the **SENDER** console and detach it to see two consoles at the same time.
2. In the SENDER console, click **Add new packet to the list**.
3. Open the **Frames Generator** tool.
4. In the **Protocol control**, select **Zigbee**.
5. In the **Frame type** control, select **0x10 - Transmit Request**.
6. In the **64-bit dest. address** box, type the 64-bit address of the RECEIVER module.
7. In the **RF data** box, click the **ASCII** tab and type the message "Hello, this is SENDER!"
8. Click **OK**.
9. Click **Add frame**.

![XBee API Frame generator](image)

**Step 4: Send the Transmit Request frame**

After you have created a Transmit Request frame, you must send it.

1. Select the frame in the XCTU **Send frames** section.

2. Click **Send selected packet**.

The **Frames log** indicates that one frame has been sent (blue) and another has been received (red).
Additionally, the RECEIVER console indicates that another packet has been received.

**Step 5: Analyze the responses**
Once you have sent the frames, you can analyze the responses on the receiving end.

1. Select the received frame (Transmit Status) in the SENDER console to view the frame details on the right panel. Verify that the message was sent successfully.
   - **Frame type**: The received frame is a Transmit Status.
   - **Frame ID**: Since both frames have the same Frame ID, this is the response for the Transmit Request frame.
   - **Status**: The Success status indicates that the message was sent successfully.
<table>
<thead>
<tr>
<th>Frame details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>7E</strong></td>
</tr>
<tr>
<td><strong>Length</strong></td>
</tr>
<tr>
<td>00 07 (7)</td>
</tr>
<tr>
<td><strong>Frame type</strong></td>
</tr>
<tr>
<td>8B <strong>(Transmit Status)</strong></td>
</tr>
<tr>
<td><strong>Frame ID</strong></td>
</tr>
<tr>
<td>01 (1)</td>
</tr>
<tr>
<td><strong>16-bit dest. address</strong></td>
</tr>
<tr>
<td>FF FE</td>
</tr>
<tr>
<td><strong>Tx. retry count</strong></td>
</tr>
<tr>
<td>00 (0)</td>
</tr>
<tr>
<td><strong>Delivery status</strong></td>
</tr>
<tr>
<td>00 <strong>(Success)</strong></td>
</tr>
<tr>
<td><strong>Discovery status</strong></td>
</tr>
<tr>
<td>00 (No discovery overhead)</td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
</tr>
<tr>
<td>76</td>
</tr>
</tbody>
</table>
2. Analyze the details of the Receive Packet for RECEIVER. Verify that the message is the one you typed and the sender’s address belongs to SENDER.
   - **Frame type:** The received frame is a Receive Packet
   - **64-bit source address:** This field displays the 64-bit address of the sender module, SENDER.
   - **Receive options:**
     - It is a DigiMesh message (0xC1 = 1100 0001).
     - It is a unicast transmission (0xC1 = 1100 0001).
     - The packet was acknowledge (0xC1 = 1100 0001).
   - **RF data:** The message of the packet is "Hello, this is SENDER!".

![Frame details]

3. Disconnect both consoles by clicking **Close the serial connection**.

**Remote AT Command: Remotely configure an XBee module**

Working in API mode also allows you to configure remote XBee modules wirelessly. Any AT command or parameter that can be issued locally can also be sent wirelessly for execution on a remote XBee module.

The following image shows the API frame exchanges that take place at the serial interface when sending a Remote AT Command Request (0x17) to remotely read or set an XBee parameter.
1. The intelligent device (host) sends a Remote AT Command Request (0x17) to XBee A through the serial input to configure the remote XBee B.
2. XBee A wirelessly transmits the AT Command in the frame to the module configured as destination in the same frame; in this case, the destination is XBee B.
3. XBee B receives the AT command and processes the command to wirelessly return the result to the sender, XBee A.
4. XBee A sends out a Remote AT Command Response (0x97) through its serial output with the result of the AT command processed by XBee B. If the frame ID of the Remote AT Command frame is '0', this response is not sent.

Do more with API mode: XBee libraries

Let's say you want to write an application to enable an intelligent device to monitor and manage an XBee network. You can write your own code to work with API mode, and you can also take advantage of existing software libraries that already parse the API frames. Depending on your preferred programming language and the intelligent device connected to the serial interface of the XBee module, you can choose from a variety of available libraries.

Here are some of the libraries available:

- **XBee mbed Library** is a ready-to-import mbed extension to develop XBee projects on the mbed platforms. For details, go to https://developer.mbed.org/teams/Digi-International-Inc/code/XBeeLib/.

- **Digi XBee Ansi C Library** is a collection of portable ANSI C code for communicating with XBee modules in API mode. For details, go to https://github.com/digidotcom/xbee_ansic_library/.

- **XBee-arduino** is an Arduino library for communicating with XBees in API mode. To learn more, go to https://code.google.com/p/xbee-arduino/.

- **XBee Java Library** is an easy-to-use library developed in Java that allows you to interact with XBee modules working in API mode. For details, go to XBee Java Library documentation.

In this kit, you use the **XBee Java Library** to learn about the XBee features and capabilities offered in API operating mode. You create several Java applications to control and monitor XBee modules connected to your computer via the XBee Grove Development Board.
DigiMesh network setup

This section describes how routing works in networks. Once you learn about routing, you can build on everything you have learned so far by building a DigiMesh network and sending data over the network.

Set up a DigiMesh network ................................................................. 62
Routing ......................................................................................... 62
View the network ................................................................. 65
Lab: Build and experiment with a DigiMesh network ........................................ 66
Set up a DigiMesh network

Forming a DigiMesh network is straightforward, since all nodes are standard routers by default and there are no parent-child relationships.

1. Ensure that the following settings have the same values in all nodes:
   - Channel Mask (CM)
   - Preamble ID (HP)
   - Network ID (ID)

2. Configure the Channel Mask (CM) parameter to enable at least the number of channels specified by the Minimum Frequencies (MF) parameter.

To add a new node to an already-formed network, set the same values for those three parameters. When you add the node, it automatically joins the DigiMesh network.

There are other settings that affect the behavior of a DigiMesh network, such as those related to the sleep feature. For details, see Sleep modes.

To learn about additional configuration parameters, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

Routing

The basic function of a network is to transfer data from one node to another. In the simplest data communication, data is transmitted directly from the source node to the destination node. However, direct communication may not be possible if the two nodes are far apart or in a difficult environment.

In this case, data must travel to another node within the radio range, which then passes it on to another node, and so on until the data reaches the desired destination node.

Routing is the process of receiving data destined for another node and passing it on. Each of the intermediary nodes between source and destination is called "hop."
Routing allows the range of a network to be extended beyond the distances supported by direct radio communication.

**Route discovery**

A message is normally routed along an already-discovered route. But if this route does not exist, the nodes involved in transmitting the data initiate a route discovery. Once completed, the message is sent along the calculated route. The route discovery process is based on the **Ad-hoc On demand Distance Vector** (AODV) routing protocol.

This algorithm uses routing tables in each node to store the next hop for a destination node. By sending a message to the next hop address, the message will either reach its destination or be forwarded to an intermediate router which will route the message to its destination.

The route discovery process finds the best available route to the destination when sending a message.

When a source node A must discover a route to a destination node B, route discovery involves the following steps:

1. Source node A sends a route request broadcast. Any router that receives the broadcast and is not the destination is called an intermediate node.
2. Intermediate nodes update and broadcast the request if it has a better route quality back to the source node A.

3. When the destination node B receives a route request, it sends a unicast route reply back to A along the path specified by the received route request.

4. This is done for every route request received, regardless of route quality.

5. The source node A may receive multiple route replies. It selects the route with the best round-trip route quality, and uses that route for every transmission to the same destination B.

**Note** XBee-PRO 900HP modules support a maximum of 128 routing table entries. If a network contains more nodes than available routing table entries, established routes are overwritten with new routes, causing route discoveries to occur more regularly. This could result in larger delays and decreased network performance.

---

**Trace routing**

Knowing the route of a DigiMesh unicast transmission is especially useful when setting up a network or when diagnosing problems within a network. This information can be obtained by enabling the **Unicast Trace Route bit** in the Transmit Options field of the Transmit Request (0x10) and Explicit Addressing Command (0x11) API frames.

When unicast data is sent with the Trace Route option enabled, each intermediate node of the route transmits a route information packet (0x8D) back along the route to the unicast originator.

---

**Route information packets (0x8D) are not guaranteed to arrive in the same order as the unicast packet.**

On a weak route, the transmission of route information packets may fail before arriving at the unicast source.

---

Route information packet (0x8D) frames contain:

- The source and destination addresses of the unicast transmission.
- The address of the node that generates the route information packet.
- The address of the next hop in the route to the destination node.

---

Enable the Unicast Trace Route option only for occasional diagnostic purposes and not for normal operations, as it can generate a large number of route information packets.

---

**Aggregate routes**

In some cases, many or all nodes in a network have to transmit data to a central aggregator node. In a new DigiMesh network, the overhead of these nodes discovering routes to the aggregator node can be extensive and may result in poor network performance.

To eliminate this overhead, you can use the **AG** command to automatically build routes to an aggregate node.

**Use the AG command**

When deploying a DigiMesh network, issue the **AG** command on the desired aggregator node to cause all nodes in the network to build routes to that node. You can also use the command to automatically
update the **DH/DL** parameters to match the MAC address of the aggregator node:

1. The **AG** command requires a 64-bit parameter to indicate the new value of the **DH/DL** parameters on the modules receiving the broadcast. Use 0xFFFE if you do not want to update **DH/DL** values.

2. XBee devices in API mode that receive the **AG** command send an Aggregate Addressing Update (0x8E) frame out the serial interface, indicating they have updated the destination address.

3. All XBee devices that receive an **AG** broadcast update their routing table information to create a route to the sending device, regardless of whether their **DH/DL** address is updated. This routing information is used for future unicast transmissions.

**Disable routing**

DigiMesh nodes can turn routing off. This means that these nodes cannot route messages to other nodes of the network. This may be useful if you want to reduce the packet traffic in your network.

To disable routing on a node, you can set its Routing/Messaging Mode (CE) setting to Non-Routing Module [2].

**View the network**

To better understand how a DigiMesh network is formed, use XCTU's Network view to discover and visualize the topology and interconnections of the network.

To learn more about the network view, see **How-to: Visualize your network.**
Lab: Build and experiment with a DigiMesh network

This section describes how to set up a simple DigiMesh network and transmit data between the nodes. You assemble the hardware and connect it to your computer, configure the XBee modules for wireless communication, create a network, and start sending messages. The network is formed initially by two nodes. To demonstrate the easy integration of XBee modules running DigiMesh topology, move one of them away until it does not receive a message. Then place a third module between the others to act as a bridge and relay the messages.

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

Step 1: Requirements
For this setup you need the following hardware and software.

**Hardware**
- Three XBee-PRO 900HP modules
- Three XBee Grove Development Boards
- Three micro USB cables
- One computer
- Two power supplies to power XBee Development Grove boards (battery, USB power adapter, laptop)

**Software**
- XCTU 6.2.0 or later

*Tip* For more information about XCTU, see the XCTU walkthrough.

Step 2: Connect the components
To get started, connect the components and start XCTU.
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

Step 3: Configure the XBee modules
To transmit data wirelessly between your XBee modules, you must configure them to be in the same network.
1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

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

3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

**How to identify the XBee modules**

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one by clicking the **Read** button and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.
Step 4: Set up the components

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

1. Keep the SENDER’s XBee Grove Development Board connected to your computer.
2. Connect RECEIVER’s board to a portable battery or to a laptop.
3. Plug BRIDGE’s board to another portable battery or to a laptop.

Step 5: Send messages via a bridge node

You can now start sending messages. Follow these steps to configure your SENDER node to send messages to your RECEIVER node, and then add your BRIDGE node.

Start by configuring SENDER to send a broadcast message every second. You can use the XCTU console or any serial port terminal application such as CoolTerm or TeraTerm (for Windows only). This tutorial uses the XCTU console.

1. Before sending messages between the modules, disconnect BRIDGE. You only need two modules for this step.
2. To send broadcast messages, open XCTU if it is not already running in the computer where SENDER is connected.
3. Switch to the Consoles working mode.
   - This working mode of XCTU allows you to communicate with the radio modules in the devices list. XCTU loads a list of consoles in the working area—one for each module of the devices list, sorted in a tabbed format.
4. If SENDER is not already there, add it to XCTU so it is listed in the Radio Modules list.
5. Open the serial connection of the radio module: select the XBee in the **Radio Modules** section, and click the **Open serial connection** button.

The background changes to green to indicate that the connection is open.

6. Add a new packet with the message "Hello!"

7. To start sending this message every second, in the **Send sequence** box:
   a. Set 1000 ms as transmit interval.
   b. Select **Loop infinitely**.
   c. Click **Start sequence**.

Notice that the red RSSI LED of RECEIVER blinks every second. This means that the module is successfully receiving the messages.

8. Move RECEIVER away from SENDER until the RSSI LED of RECEIVER does not blink anymore. This means that the module is out of range so it cannot receive messages from SENDER.

---

**Warning:** Although the Power Level (PL) of all the XBee modules is configured to the minimum value, modules still have a long range. You may have to move RECEIVER away several meters until the RSSI LED does not blink anymore.
9. Place BRIDGE between SENDER and RECEIVER and plug it in. BRIDGE joins the network, creates a bridge between the other two nodes and relays messages from SENDER to RECEIVER.
Wireless data transmission

This section explains wireless transmission and guides you through a lab to illustrate how it works.

Transmission methods ................................................................. 72
Lab: Transmit data wirelessly ..................................................... 73
Transmission methods

An XBee can communicate with multiple devices or with just one device:
- Broadcast transmissions are sent to many or all modules in the network.
- Unicast transmissions route wireless data from one XBee to one destination module.

Broadcast transmission

Broadcast means to transmit the same data to all nodes on a network. These transmissions are propagated throughout the entire network so that all possible nodes receive the transmission. To accomplish this, the originating node and all nodes that receive a broadcast transmission will transmit the data the number of times specified by the Broadcast Multi-Transmits (MT) setting plus 1 (the default value of MT is 3).

The broadcast address is a 64-bit address with the lowest 16 bits set to 1 and the upper bits set to 0, 000000000000FFFF.

Unicast transmission

A unicast transmission consists of sending messages to a single node on the network that is identified by a unique 64-bit address. The destination XBee module could be an immediate neighbor of the sender, or be several hops away.

When transmitting while using DigiMesh unicast communications, reliable delivery of data is accomplished using retries and acknowledgments.
- **MAC retries/acknowledgements** are used for transmission between adjacent nodes. The maximum number of retries before receiving an acknowledge from a neighbor is determined by the Unicast Retries (RR) parameter. Wireless data can be sent up to RR + 1 times.

- **Network retries/acknowledgements** are used across the entire route. The value of Mesh Unicast Retries (MR) parameter is the maximum number of attempts that wireless data is re-transmitted across the route. Wireless data can be transmitted up to MR + 1 times until a network acknowledgment is received.

If a non-existent 64-bit address is used as a destination address, address discovery will be attempted and the Transmit Status (0x8B) message will show a delivery status code of 0x25 (Route not found) and a discovery status code of 0x02 (Route discovery).

**Note** To learn more about advanced data transmission topics, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

---

**Lab: Transmit data wirelessly**

In this lab, you create an application in the Java programming language to transmit data between the nodes of the network. To simplify that application, you use the XBee Java Library, an easy-to-use API that allows you to interact with XBee modules.

Once you have everything set up, send a message to a specific device (unicast) or to all devices of the network (broadcast).

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

**Step 1: Requirements**

For this setup you need the following hardware and software.

**Hardware**

- Three XBee-PRO 900HP DigiMesh Kit modules
- Three XBee Grove Development Boards
Wireless data transmission

Lab: Transmit data wirelessly

- Three micro USB cables
- One computer

**Software**
- XCTU 6.3.1 or later
- XBee Java Library (XBJL-X.Y.Z.zip release file)
- Java Virtual Machine 6 or later
- A Java IDE (such as Eclipse or NetBeans)

**Tip** For more information about XCTU, see the [XCTU walkthrough](#).

### Step 2: Connect the components
To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in [Plug in the XBee module](#).
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

### Step 3: Configure the XBee modules
Configure each of the three XBee modules to work in API mode and assign each a different role.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>XBee C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will connect to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>NI</td>
<td>XBEE_A</td>
<td>XBEE_B</td>
<td>XBEE_C</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
</tbody>
</table>

3. Write the settings of all XBee modules with the Write radio settings button at the top of the Radio Configuration section.

**Step 4: Create a Java project**

**Option 1: Eclipse**
- Select File > New, and click the Java Project.
- The New Java Project window appears. Enter the Project name.
- Click Next.

or

**Option 2: NetBeans**
- Select File > New project....
- The New Project window appears. In the Categories frame, select Java > Java Application from the panel on the right, and click Next.
- Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
- Click Finish to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

**Step 5: Link libraries to the project**

This topic describes how to link the XBee Java Library, the RXTX library (including the native one), and the logger library to the project.
Wireless data transmission

Lab: Transmit data wirelessly

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

**Option 1: Eclipse**

a. Go to the **Libraries** tab of the New Java Project window.
b. Click **Add External JARs....**
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the xbee-java-library-X.Y.Z.jar file.
d. Click **Add External JARs...** again.
e. Go to the **extra-libs** folder and select the following files:
   - rxtx-2.2.jar
   - slf4j-api-x.y.z.jar
   - slf4j-nop-x.y.z.jar
f. Expand the rxtx-2.2.jar file of the Libraries tab list, select **Native library location**, and click **Edit....**
g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```
a. Click **OK** to add the path to the native libraries.
b. Click **Finish**.

or

**Option 2: NetBeans**

a. From **Projects** view, right-click your project and go to **Properties**.
b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.
c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the xbjlib-X.Y.X.jar file.
d. Click **Add JAR/Folder** again.
e. Go to the **extra-libs** folder and select the following files:
   - rxtx-2.2.jar
   - slf4j-api- x.y.z.jar
   - slf4j-nop- x.y.z.jar
f. Select **Run** in the left tree of the **Properties** dialog.
g. In the **VM Options** field, add the following option:

```bash
-Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>/extra-libs/native/Windows\win32
```

where:

- `<path_where_the_XBee_Java_Library_is_unzipped>` is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
- `Windows\win32` is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don’t know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```bash
java -version
```

h. Click **OK**.

**Step 6: Add the source code to the project**

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: `MainApp.java`.
2. Add the Java source file with Eclipse or NetBeans.

**Option 1: Eclipse**

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: `MainApp`.

d. Click **Finish**.

e. The `MainApp.java` file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (*Move 'MainApp.java' to package '...'*) to resolve the error.

**Option 2: NetBeans**

a. In the **Projects** view, select the project and right-click.

b. From the context menu, select **New > Java Class**... The New Java Class wizard opens.

c. Modify the **Class Name** to be `MainApp`.

d. Click **Finish**.

e. The `MainApp.java` file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (*Move class to correct folder*) to resolve the error.
Step 7: Set the port names and launch applications

For this step, set the port names for all three XBee modules, duplicate the project for XBEE_B and XBEE_C, then launch the applications.

1. Change the port name in the Java source code to match the port XBEE_A is connected to.

   ```java
   // TODO: Replace with the port where your module is connected
   private static final String PORT = "COM1";
   // TODO: Replace with the baud rate of your module.
   private static final int BAUD_RATE = 9600
   ```

2. Duplicate the Java project for XBEE_B and rename it to `XBeeTransmitDataB`.
3. Change the port name in the second project's source code to match the port XBEE_B is connected to.
4. Duplicate the Java project for XBEE_C and rename it to `XBeeTransmitDataC`.
5. Change the port name in the second project's source code to match the port XBEE_C is connected to.
6. Launch the three applications.

Step 8: Transmit data over the network

Follow these steps to transmit data to other XBee modules in your network with the unicast or broadcast method.

1. Send messages to a specific XBee module (unicast) or to all (broadcast) using the following pattern:
   - **Unicast:** `NODE_IDENTIFIER: message`
   - **Broadcast:** `ALL: message`

   For example, to send the message "Hi XBee nodes" to all nodes of the network:

   ```
   ALL: Hi XBee nodes
   ```

Step 9: Section summary of wireless data transmission

In this section, you have learned the following:

- An XBee sending a transmission in API mode not only transmits a raw message but also some extra information—such as the address of the source XBee module—packaged in what is called an API frame.
- In API mode, you don't need to set the DH and DL parameters of the receiver device because the destination address is already included in the API frame.
- API mode allows you to easily work with multiple destinations without needing to re-configure the sender module to establish a new destination module before sending the data.
- You can use the XBee Java Library to simplify and improve the use of the API operating mode.
- Depending on the number of devices that will receive the message, there are two types of transmissions:
  - Unicast sends a message to one node identified by a unique address.
  - Broadcast sends the same message to all possible nodes on the network.

**Step 10: Do more with wireless data transmission**

If you’re ready to work more extensively with data transmission, try the following:

- Extend the network by adding more XBee-PRO 900HP DigiMesh Kit modules so you can chat with other devices.

  **Note** To find the best channel to acquire more modules, see *Where to buy XBee devices*.

- Use Arduino or Raspberry Pi instead of a computer to transmit data wirelessly.
Low power and battery life

This section introduces the key concepts you need to know to take advantage of the power saving capabilities of XBee devices. It also provides a lab that lets you put the concepts to work and see the results.

- Low power devices and battery life ................................................................. 81
- Sleep modes ........................................................................................................ 81
- Asynchronous sleep ......................................................................................... 82
- Synchronous sleep ............................................................................................ 83
- Lab: Use synchronous sleep ........................................................................... 89
Low power devices and battery life

Wireless RF devices are typically battery-powered, which can pose a challenge: depending on the location of the device, it may be difficult or expensive to replace the battery.

XBee modules are low-power devices. They can put themselves into a temporary sleep state in which they consume virtually no current. During sleep, the device is almost completely turned off and is sometimes incapable of sending or receiving data until it wakes up.

A real world scenario

Extending battery life is important in many real world scenarios. For example, if you had several greenhouses, each with a temperature sensor connected to an XBee module, battery life would be critical. Fully charged batteries would only power the modules for one day.

There are several ways to maximize battery life. For example:

- Putting the modules into a cycle where they sleep for one second and then wake for one second before sleeping again could double the battery life to two days.
- Cyclically sleep for 59 seconds and then waking for one second might keep the same batteries going for 60 days. Taking this further, you could potentially extend the battery life for years.

Design considerations for applications using sleep mode

Before using sleep mode you must take into consideration the structure of your project and your XBee network. Some applications, like the greenhouse example, are particularly suited to sleep mode. In that scenario, the modules only send data periodically and are not expected to receive data. The modules can therefore be sleeping most of the time and wake up only to send the temperature value.

By contrast, there are applications where devices send and receive data frequently. These applications need to use sleep mode differently—for example, by triggering sleeping and waking with events. Continue learning about sleep mode so you can optimize its capabilities for your needs.

Sleep modes

Putting XBee modules into a temporary sleep state preserves battery life when using wireless networks. DigiMesh modules support five sleep modes that are classified as synchronous or
asynchronous.

**Note** Asynchronous sleep modes should not be used in a synchronous sleeping network, and vice versa.

### Synchronous sleep modes

Synchronous modes allow all XBee modules to sleep and wake at the same time.

**Synchronous sleep support (SM = 7)**

An XBee module in synchronous sleep support mode synchronizes with a sleeping network, but does not sleep. At any time, a sleep support node responds to new nodes that are attempting to join the sleeping network with a sync message. It only transmits normal data when the other nodes in the sleeping network are awake. Sleep support nodes are especially useful as preferred sleep coordinator nodes and as aids in adding new nodes to a sleeping network. To enable synchronous sleep support, set the Sleep Mode (SM) parameter to Sleep Support [7].

**Synchronous cyclic sleep mode (SM = 8)**

An XBee module in synchronous cyclic sleep mode sleeps for a programmed time, wakes in unison with other nodes, exchanges data and sync messages, and then returns to sleep. While asleep, it cannot receive wireless or serial data. To enable synchronous sleep support, set the Sleep Mode (SM) parameter to Synchronized Cyclic Sleep [8].

### Asynchronous sleep modes

Asynchronous modes allow you to control the sleep state of a single XBee module.

**Asynchronous pin sleep (SM = 1)**

Pin sleep allows an external microcontroller to determine when the XBee module should sleep and when it should wake by controlling the Sleep_RQ pin (pin 9). When Sleep_RQ is asserted (high) by connecting it to 3.3 volts, the XBee module finishes any operation and enters a low power state. The XBee module wakes when the Sleep_RQ pin is de-asserted (low).

**Asynchronous cyclic sleep (SM = 4)**

Cyclic sleep allows the XBee module to sleep for a specified time (SP) and wake for a short time (ST) before returning to sleep again. If the XBee module receives serial or wireless data while awake, it extends the time before it returns to sleep by the amount specified by the ST parameter. Otherwise, it enters sleep mode immediately. Parameters SO, SN, SP and ST control the sleep cycle.

**Asynchronous cyclic sleep with pin wake-up (SM = 5)**

The cyclic sleep with pin wake-up is a slight variation of the cyclic sleep mode that allows the XBee module to be woken prematurely by de-asserting the Sleep_RQ pin. The XBee module does not sleep when the Sleep_RQ is de-asserted. Parameters SO, SN, SP and ST control the sleep cycle.

### Asynchronous sleep

XBee devices configured to sleep asynchronously enter low power mode only when their configuration requires it; they do not wait for a synchronization message.

- Do not use modules operating in an asynchronous sleep mode to route data.
- We strongly encourage you to configure asynchronous sleeping modules as non-routing nodes (CE = 2). This prevents the node from attempting to route data.

DigiMesh supports three different asynchronous sleep modes:
- Asynchronous pin sleep (SM = 1).
  Pin sleep allows an external microcontroller to determine when the device should sleep and when it should wake by controlling the Sleep_RQ pin (pin 9). When Sleep_RQ is asserted (high) by connecting it to 3.3 volts, the device will finish any operation and enter a low power state. The device wakes when the Sleep_RQ pin is de-asserted (low).
- Asynchronous cyclic sleep (SM = 4).
  Cyclic sleep allows the module to sleep for a specified time (SP) and wake for a short time (ST) before returning to sleep again.
  If the device receives serial or wireless data while awake, it extends the time before it returns to sleep by the amount specified by the ST parameter. Otherwise, it enters sleep mode immediately.
  The SO, SN, SP and ST parameters control the sleep cycle; see Configuration Parameters.
- Asynchronous cyclic sleep with pin wake-up (SM = 5).
  The cyclic sleep with pin wake up is a slight variation of the cyclic sleep mode that allows the device to be woken prematurely by de-asserting the Sleep_RQ pin. The module will not sleep when the Sleep_RQ is de-asserted.
  The SO, SN, SP and ST parameters control the sleep cycle (See, Configuration Parameters).

Note To learn more about asynchronous sleep networks, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

Synchronous sleep

The synchronous sleep feature of DigiMesh makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time, forming a cyclic sleeping network. Nodes synchronize by receiving a special wireless packet called a sync message sent by a node acting as a sleep coordinator. This allows some or all XBee modules in the network to be battery powered.

Synchronous sleep operation

All modules go to sleep at the same time. This forms a cyclic sleeping network where we can define two different device types: sleep coordinator and sleeping router.

Sleep coordinator

A node in the DigiMesh network acts as the sleep coordinator. This device:
- Defines the sleeping parameters for the entire network: sleep and wake times for every cycle.
- Broadcasts a synchronization message at the beginning of each wake period.
- Cannot sleep.

A node in the network can become a coordinator through a nomination and election process.

Sleeping router
A sleeping router is a full-featured DigiMesh node configured as a low-power device. This device:

- Sleeps for a programmed time.
- Wakes in unison with other nodes.
- Exchanges data and synchronization messages, and then returns to sleep.
- Cannot receive wireless messages or data from the serial interface while asleep.

During normal operations:
1. The sleep coordinator sends a synchronization message as a broadcast to all nodes at a beginning of every wake cycle.
2. This message contains synchronization information and the wake and sleep times for the current cycle.
3. All cyclic sleep nodes \((SM = 8)\) receiving a sync message remains awake for the wake time and then sleep for the specified sleep period.

**Synchronous sleep modes**

DigiMesh supports two different synchronous sleep modes:

- Synchronous sleep support \((SM = 7)\)
- Synchronous cyclic sleep \((SM = 8)\)

**Synchronous sleep support**

An XBee in synchronous sleep support mode will synchronize itself with a sleeping network but will not sleep.

At any time, a sleep support node will respond with a sync message to new modules which are attempting to join the sleeping network. It will only transmit normal data when the other nodes in the sleeping network are awake.
Sleep support nodes are especially useful when used as preferred sleep coordinator nodes and as aids in adding new modules to a sleeping network.
You can enable synchronous sleep support by setting the Sleep Mode (SM) parameter to Sleep Support [7].

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

**Synchronous cyclic sleep**
An XBee in synchronous cyclic sleep mode sleeps for a programmed time, wakes in unison with other nodes, exchanges data and sync messages, and then returns to sleep. While asleep, it cannot receive wireless or serial data.
You can enable synchronous sleep support by setting the Sleep Mode (SM) parameter to Synchronized Cyclic Sleep [8].

**Synchronous sleep parameters**
The following parameters control and detail the status of the synchronous cyclic sleep network.

**Configuration parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO</td>
<td>Sleep Options</td>
<td>Bitmask to define options for sleep mode behavior:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 0: Preferred sleep coordinator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 1: Non-sleep coordinator; if set, the module never acts as a sleep coordinator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 2: Enable API sleep status messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 3: Disable early wake-up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 4: Node Type Equality; if set, no distinction is made between sleep support (SM = 7) and cyclic sleep (SM = 8) nodes for the purpose of nomination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bit 5: Disable coordinator rapid sync deployment mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All other bits are reserved.</td>
</tr>
<tr>
<td>SN</td>
<td>Number of Cyclic Sleep Periods</td>
<td>Configures the number of sleep periods multiplier.</td>
</tr>
</tbody>
</table>
### Status parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| SS        | Sleep Status       | Describes the current sleep status of the module. The bits are defined as follows:  

- bit 0: Network awake  
- bit 1: Current sleep coordinator  
- bit 2: Ever received sync since it was powered on  
- bit 3: Sync received this cycle  
- bit 4: Overriding sync pending after a sleep settings modification  
- bit 5: Nomination requested by user (using the Commissioning button or the CB2 command)  
- bit 6: Currently in deployment mode  

All other bits are reserved. |
| OS        | Operating Sleep Time | The current sleep time that the module is using:  

- In a synchronized network, all nodes should report the same value  
- An unsynchronized node reports the value of its Sleep Period (SP) parameter |
| OW        | Operating Wake Time | The current wake time that the module is using:  

- In a synchronized network, all nodes should report the same value  
- An unsynchronized node reports the value of its Wake Time (ST) parameter |
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>Missed Sync Messages</td>
<td>The number of complete wake cycles have elapsed on the module since the last received sync message (or sent in the case of sleep coordinators)</td>
</tr>
<tr>
<td>SQ</td>
<td>Missed Sleep Sync Count</td>
<td>The total number of wake cycles that have elapsed on the module in which a sync message was not received</td>
</tr>
</tbody>
</table>

Generally, network sleep and wake times are specified by the Sleep Time (SP) and Wake Time (ST) of the network’s sleep coordinator. These parameters are only used at start up until the node is synchronized with the network. When a module has synchronized with the network, its sleep and wake times can be queried with the Operating Sleep Time (OS) and Operating Wake Time (OW) parameters respectively.

A newly powered unsynchronized sleeping node polls for a synchronized message and then sleeps for the period specified by its Sleep Time (SP), repeating this cycle until it becomes synchronized by receiving a sync message. Once it receives a sync message, the node synchronizes itself with the network.

### Designate a sleep coordinator

There are four ways for a DigiMesh node to become a sleep coordinator.

#### Preferred sleep coordinator option

The preferred sleep coordinator bit (bit 0) of the Sleep Options (SO) parameter configures the device to always act as a sleep coordinator.

Enable this option only on one device in the network because any node configured this way always sends a sync message at the beginning of every wake cycle.

The following are good candidates for the preferred sleep coordinator:

- A node that is centrally located in the network. This position minimizes the number of hops a sync message must take to get across the network.
- A sleep support node.
- A node that is mains powered.

The preferred sleep coordinator option can be used when setting up a network for the first time:

1. When starting a network, configure a module as a sleep coordinator so it begins sending sleep messages.
2. After the node is set up, disable the preferred sleep coordinator bit.

The preferred sleep coordinator bit should be used with caution. The advantages of using the option become weaknesses when used on a node that is not positioned or configured properly.
Nomination and election

Nomination is an optional process that can occur on a node in the event that contact with the network sleep coordinator is lost. Only XBee devices with the non-sleep coordinator bit (bit 1) of the Sleep Options (SO) set to 0 participate in the nomination process. This process occurs automatically if a node has missed three or more sync messages and it has the non-sleep coordinator bit set to 0. In this case, the node eventually nominates itself after a number of cycles without a sync. A nominated node will begin acting as the new network sleep coordinator.

An election takes place if multiple nodes in the network nominate themselves as a sleep coordinator. Seniority is determined by four factors (in order of priority):

a. Newer sleep parameters. A node using newer sleep parameters (SP / ST) is considered senior to a node using older sleep parameters.

b. Preferred Sleep Coordinator. A node acting as a preferred sleep coordinator is senior to other nodes.

c. Sleep support (SM = 7) nodes are senior to cyclic sleep (SM = 8) nodes. This behavior can be modified using the SO parameter.

d. Serial number. In the event that the above factors do not resolve seniority, the node with the higher serial number is considered senior.

Commissioning button

If the Commissioning button functionality is enabled, a device can be immediately nominated as a sleep coordinator by:

- Pressing the Commissioning button twice.
- Issuing the Commissioning Pushbutton (CB) command with a parameter of 2.

A node nominated in this manner is still subject to the election process described above.

A node configured as a non-sleep coordinator (bit 1 of SO enabled) will ignore Commissioning button nomination requests.

Changing sleep parameters

Any sleep-compatible node in the network that does not have the non-sleep coordinator sleep option set can be used to make changes to the network’s sleep and wake times.

If a node’s SP and/or ST are changed to values different from those the network is using, that node will become the sleep coordinator and begin sending sync messages with the new sleep parameters at the beginning of the next wake cycle.

For normal operations, a module uses the sleep and wake parameters it gets from the sleep sync message, not those specified in its SP and ST parameters. The SP and ST parameters are not updated with the values of the sync message. You can query the node’s current network sleep and wake times with the OS and OW commands.

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

- Network Hops (NH)
- Network Delay Slots (NN)
- Network Options (NQ)
- Mesh Unicast Retries (MR)
In most applications, these network parameters should only be configured during deployment.

---

**Start a sleeping network**

You can set up XBee modules with their sleep parameters pre-configured prior to deployment:

1. If you are going to use a preferred sleep coordinator in the network, deploy it first.
2. Otherwise, select an XBee module for deployment, power it on, and press the Commissioning button twice. This causes the module to begin emitting synchronization messages.
3. Verify that this module is emitting sync messages by watching its associate LED. A slow blink (500 ms blink time) indicates that the XBee module is acting as a sleep coordinator.
4. Next, power on an XBee module in range of the sleep coordinator or other modules that have synchronized with the network. If a synchronized module is asleep, wake it by pressing the Commissioning button once.
5. Wait a cycle for the module to synchronize itself.
6. Verify that the XBee module synchronizes with the network. The associate LED blinks (250 ms blink time) when the module is awake and synchronized.
7. Continue this process until all modules are deployed.

**Note** To learn more about synchronous sleep networks, go to the XBee-PRO 900HP and XSC RF Modules User Guide.

---

**Lab: Use synchronous sleep**

Use the topics in this section to get started using synchronous sleep mode in your DigiMesh network. If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

**Step 1: Requirements**

For this setup you need the following hardware and software.

**Hardware**

- Three XBee-PRO 900HP DigiMesh Kit modules
- Three XBee Grove Development Boards
- Three micro USB cables
- One computer

**Software**

- XCTU 6.3.1 or later

**Tip** For more information about XCTU, see the XCTU walkthrough.
Step 2: Connect the components
To get started, connect the components and start XCTU.
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

Step 3: Configure the XBee Modules
To transmit data wirelessly between your XBee modules, you must configure them to be in the same network. For a sleeping network, you need a sleep coordinator. For this tutorial, configure the sleep support node as the sleep coordinator; the other two sleep synchronously.
1. Restore the default settings of all XBee modules with the Load default firmware settings button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>XBee C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>DH</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>Defines the destination address (high part) to transmit the data to.</td>
</tr>
<tr>
<td>DL</td>
<td>FFFF</td>
<td>—</td>
<td>—</td>
<td>Defines the destination address (low part) to transmit the data to. The address 000000000000FFFF can be used to send a broadcast message.</td>
</tr>
<tr>
<td>NI</td>
<td>SENDER</td>
<td>RECEIVER_1</td>
<td>RECEIVER_2</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
<tr>
<td>D4</td>
<td>Digital Input [3]</td>
<td>—</td>
<td>—</td>
<td>Sets the DIO4/AD4 pin as digital input in the sleep coordinator. This pin is connected to a button</td>
</tr>
<tr>
<td>IC</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>Configures the sender to transmit an IO sample when pin DIO4 (where the button is connected) changes. 00001000 (binary) = 10 (hexadecimal). For more information on how to configure this parameter to monitor the pins, see How to obtain data from a sensor.</td>
</tr>
</tbody>
</table>
3. Write the settings of all XBee modules with the Write radio settings button at the top of the Radio Configuration section.

**How to identify the XBee modules**

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one by clicking the Read button and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.
Step 4: Test the sleep configuration

With this configuration, when SENDER’s user button is pressed or released its value is sent to RECEIVER_1 and RECEIVER_2. However, it only receives this sample when the module wakes up. To verify, perform the following steps in XCTU:

1. Select RECEIVER_1 module (receiver).
2. Switch to Consoles working mode.

3. Open the serial connection with the module.

4. Press and release the user button of the board where SENDER is attached.
5. While RECEIVER_1 and RECEIVER_2 are asleep, their corresponding On/Sleep LED is off. It turns on when they wake up.
6. When RECEIVER_1 wakes up, two IO samples (IO Data Sample RX Indicator) will be received from SENDER: one for the button press and other for the button release.
7. Select one frame and check its details in the right panel. You will see the value of the user button (DIO4_AD4) and other details related to the frame.
8. Open the serial connection with RECEIVER_2 and verify the same frames are received when the SENDER’s user button is pressed and released.

![Image of serial connection with frames and details](image)

**Note** Remember to close the serial connection with the modules when you complete this lab.

**Step 5: Section summary of synchronous sleep**

In this section, you have learned the following:

- DigiMesh modules can go into a temporary sleep state in which they consume virtually no current.
- In DigiMesh, all the nodes in the network can go to sleep at the same time. This forms a synchronous cyclic sleep network.
- Sleep nodes synchronize by receiving a wireless synchronization message with the wake and sleep times for the current cycle.
- The sleep coordinator is the node of the sleeping network that broadcasts the sync message at the beginning of each wake period.
- A node in the network can become a coordinator through a nomination and election process.
- While an XBee module is in sleep mode, there is no data transmission or reception. So, if you try to communicate with the module when it is asleep, XCTU displays a warning message saying that the module must be reset in order to wake up.
- DigiMesh supports two different synchronous sleep modes:
  - Synchronous sleep support (SM = 7). When this mode is configured, the module synchronizes itself with a sleeping network but it does not sleep.
  - Synchronous cyclic sleep (SM = 8). An XBee module in this mode wakes in unison with other nodes, exchanges data and sync messages, and then returns to sleep.

- Generally, network sleep and wake times are specified by the Sleep Time (SP) and Wake Time (ST), respectively, of the network’s sleep coordinator.

- When sleeping synchronously, the sleep and wake times of an XBee module can be queried with the Operating Sleep Time (OS) and Operating Wake Time (OW) parameters, respectively.

**Step 6: Do more with sleep mode**

If you're ready to work more extensively with sleep mode, try the following:

- Change the Sleep Period (SP) and Wake Time (ST) values of SENDER and observe how the sleep cycle changes for the whole network.

- Connect a battery to RECEIVER_1 (XBee B), RECEIVER_2 (XBee C), or both and move the modules away from SENDER (XBee A).

- Change RECEIVER_1 Sleep Mode (SM) to be a sleep support node (SM = 7), set Sleep Options (SO) of SENDER and RECEIVER_1 to 0, and press the commissioning button twice or use any other technique described in the to select the sleep coordinator.

- Combine the synchronous sleep feature with a real sensor to create a low-power sensor network.
Inputs and outputs

All XBee modules have a set of pins that can be used to connect sensors or actuators and configure them for specific behavior. Each XBee radio has the capability to directly gather sensor data and transmit it without the use of an external microcontroller.

With these pins you can, for example, turn on a light by sending information to an XBee module connected to an actuator, or measure the outside temperature by obtaining data from a temperature sensor attached to your XBee module.

Learn about I/O pins, sensors, actuators in this section, then put your knowledge to work by using sensors.

XBee I/O pins ................................................................. 98
How XBee devices get sensor data ...................................................... 99
Lab: receive digital data ................................................................. 101
Lab: Receive analog data ............................................................... 108
How XBee modules control devices ................................................. 115
Lab: Send digital actuations .......................................................... 117
Lab: Send analog actuations .......................................................... 123
**XBee I/O pins**

The following table shows the I/O pins of XBee-PRO 900HP modules:

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Physical pin #</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIO0, AD0</td>
<td>20</td>
<td>D0</td>
</tr>
<tr>
<td>DIO1, AD1</td>
<td>19</td>
<td>D1</td>
</tr>
<tr>
<td>DIO2, AD2</td>
<td>18</td>
<td>D2</td>
</tr>
<tr>
<td>DIO3, AD3</td>
<td>17</td>
<td>D3</td>
</tr>
<tr>
<td>DIO4</td>
<td>11</td>
<td>D4</td>
</tr>
<tr>
<td>DIO5</td>
<td>15</td>
<td>D5</td>
</tr>
<tr>
<td>DIO6</td>
<td>16</td>
<td>D6</td>
</tr>
<tr>
<td>DIO7</td>
<td>12</td>
<td>D7</td>
</tr>
<tr>
<td>DIO8</td>
<td>9</td>
<td>D8</td>
</tr>
<tr>
<td>DIO9</td>
<td>13</td>
<td>D9</td>
</tr>
<tr>
<td>DIO10, PWM0, RSSI</td>
<td>6</td>
<td>P0</td>
</tr>
<tr>
<td>DIO11, PWM1</td>
<td>7</td>
<td>P1</td>
</tr>
<tr>
<td>DIO12</td>
<td>4</td>
<td>P2</td>
</tr>
<tr>
<td>DIO13</td>
<td>2</td>
<td>P3</td>
</tr>
<tr>
<td>DIO14</td>
<td>3</td>
<td>P4</td>
</tr>
</tbody>
</table>

(D = digital; I = input; O = output; AD = analog input; PWM = pulse-width modulation)

*Note* The number and type of IOs available can vary between different module variants.
How XBee devices get sensor data

XBee devices are often used to form sensor networks. In a sensor network, the main device—also called the local XBee device—receives data from the sensors attached to the remote XBee devices.

To receive that data, you must configure the remote XBee devices to "listen" on the particular pin where the sensor is connected and to send the data to the main XBee device.

Sensors

A sensor is a device that detects events or changes and provides a corresponding output, generally as an electrical signal.

There are two types of sensors: digital and analog. A motion sensor is a digital sensor because it can return two discrete values: movement detected or movement not detected. Other digital sensors might provide a binary value. A digital compass, for example, may provide your current heading by sending a 9-bit value with a range from 0 to 359. On the other hand, a thermometer is an analog sensor because the voltage output changes gradually as the temperature changes.

Setting pins for digital and analog sensors

Configure the pin of your XBee module according to the sensor that is connected to it:

- If you connect a digital sensor, configure the pin as Digital Input.
- If you connect an analog sensor, configure the pin as Analog to Digital Converter (ADC).

Note For more information about sensors, see the How XBee devices get sensor data section.

How to configure a pin as an input

Configure a pin for digital input

You can configure a pin through XCTU. If your sensor reads digital values (like a doorbell) and is connected to the DIO1/AD1 pin, configure the D1 parameter as Digital Input [3]:

---

*XBee-PRO 900HP DigiMesh Kit User Guide*
Configure a pin for analog input
If your sensor reads analog values (like a temperature sensor) and is connected to the DIO1/AD1 pin, configure the D1 parameter as ADC [2]:

How to obtain data from a sensor
There are two ways to obtain sensor information:
- Queried sampling to immediately read all enabled digital and analog input pins.
- Automatic sampling to transmit the sensor data periodically or whenever a digital pin changes.

In both cases, the information is sent to the other module called IO sample. It contains which inputs (DIO lines or ADC channels) have sampling enabled and the value of all the enabled digital and analog inputs.

Queried sampling (IS)
The Force Sample (IS) command forces a read of all enabled digital and analog input pins. You can send it locally or to a remote device.
Use the XCTU console or any serial port terminal application to send this command.
When the module sends the IS command, the receiving device reads all enabled digital IO and analog input channels and returns their value. If the module transmits the IS command locally, it sends the IO data out the serial interface. If the module transmits the IS command to a remote XBee module, it sends the remote IO data over the air to the requester module.

Automatic sampling
Once you have set up the pin, the remote module must be configured to automatically transmit the sensor information to the main XBee module. The remote XBee module needs to know:
1. Where to transmit the sensor data: define this information for the module receiving this information by the destination address (DH + DL) parameters.
2. When to transmit the sensor data:
   - Periodically: The XBee can send the information read from the sensor at a specified interval.
   - By change detection: When a pin or several pins change status.

Configure parameters IO Sampling Rate (IR) and Digital IO Change Detection (IC) to automatically transmit the sensor data.

Note These two features can work in combination with each other, depending on your requirements. For example, you could choose to receive an IO sample every minute (IR) but also when a certain pin changes state (IC).
**IO Sampling Rate (IR)**

The IR parameter sets the I/O sample rate: that is, how frequently to report the current pin state and transmit it to the destination address. The rate is set in milliseconds using hexadecimal notation. The value 0 disables the feature.

For example, if you want to transmit the sensor info every minute, set this parameter to EA60 (1 minute = 60 seconds = 60000 ms = EA60 hex).

Use XCTU to configure the sample rate interval.

**Note** Sleeping devices, configured to send samples periodically, transmit the first sample immediately after waking up, and then continue sending periodic I/O samples at the IR rate, until the Time Before Sleep (ST) timer expires and the device can resume sleeping.

**Digital IO Change Detection (IC)**

The IC parameter allows you to set which pins to monitor for change detection. When the state of the monitored pin(s) changes, a sample is immediately sent to the destination address.

Use XCTU to set the value of IC parameter.

To select which pins to monitor, assign a binary value to IC parameter based on the following pattern:

<table>
<thead>
<tr>
<th>DIO12</th>
<th>DIO11</th>
<th>DIO10</th>
<th>DIO9</th>
<th>DIO8</th>
<th>DIO7</th>
<th>DIO6</th>
<th>DIO5</th>
<th>DIO4</th>
<th>DIO3</th>
<th>DIO2</th>
<th>DIO1</th>
<th>DIO0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

For example, if you want to monitor DIO1, the value would be 0000000000010, which is 2 in hexadecimal notation. If you want to monitor DIO2, DIO8, DIO3 and DIO1, the value would be 1000100001010 (binary) = 110A (hexadecimal). The value 0 disables the feature.

The Digital IO Change Detection (IC) feature only works for digital pins, so you will not receive anything if the value of an analog pin changes.

If an XBee module is sleeping, changes in any of the monitored pins will not wake the module.

---

**Lab: receive digital data**

This section teaches you how to create an XBee digital sensor network. Since the kit does not contain a real sensor, you simulate a digital sensor with the user button of the XBee Grove Development Board.

**Note** If you have a Grove sensor, you can connect it to the board for a more realistic example of a sensor network.

Two XBee modules, connected to the sensors, will be the senders. These XBee modules will transmit the status of the user button to the other module (the receiver) every time it is pressed or released.
**Step 1: Requirements**
For this setup you need the following hardware and software.

**Hardware**
- Three XBee-PRO 900HP DigiMesh Kit modules
- Three XBee Grove Development Boards
- Three micro USB cables
- One computer

**Software**
- XCTU 6.3.1 or later
- XBee Java Library (XBJL-X.Y.Z.zip release file)
- Java Virtual Machine 6 or later
- A Java IDE (such as Eclipse or NetBeans)

**Tip** For more information about XCTU, see the XCTU walkthrough.

**Step 2: Connect the components**
If you have a Grove sensor (not included in the kit), plug it into the Grove DIO4 connector of the sender XBee's board. Follow the steps to connect your modules:
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

**Step 3: Configure the XBee modules**
SENDER_1 and SENDER_2 will send the digital value from the user button to RECEIVER every time the value changes (that is, when the button is pressed or released).

Set the destination address (DH + DL) of the senders (SENDER_1 and SENDER_2) to the MAC address (SH + SL) of the receiver (RECEIVER). Additionally, configure the pin where the button is connected (DIO4/AD4) as a digital input, and set the DIO change detect (IC) to monitor the same pin.

1. Restore the default settings of all XBee modules with the Load default firmware settings button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>XBee C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>DH</td>
<td>—</td>
<td>0013A200</td>
<td>0013A200</td>
<td>Defines the destination address (high part) to transmit the data to. The value of this setting should be the Serial Number High (SH) of the RECEIVER module.</td>
</tr>
<tr>
<td>DL</td>
<td>—</td>
<td>SL of XBee A</td>
<td>SL of XBee A</td>
<td>Defines the destination address (low part) to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the RECEIVER module.</td>
</tr>
<tr>
<td>NI</td>
<td>RECEIVER</td>
<td>SENDER_1</td>
<td>SENDER_2</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The default NI value is a blank space. Make sure to delete the space when you change the value.</td>
</tr>
<tr>
<td>IC</td>
<td>—</td>
<td>10</td>
<td>10</td>
<td>Configures the senders to transmit an IO sample when pin DIO4 (where the button is connected) changes. 00010000 (binary) = 10 (hexadecimal). For further information on how to configure this parameter to monitor the pins, see How to obtain data from a sensor.</td>
</tr>
</tbody>
</table>

*Note* The dash (—) in the table means to keep the default value. Do not change the default value.
3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

**Step 4: Create a Java project**

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **ReceiveDigitalData**.

**Option 1: Eclipse**

a. Select *File > New*, and click the *Java Project*.

b. The *New Java Project* window appears. Enter the Project name.

c. Click *Next*.

or

**Option 2: NetBeans**

a. Select *File > New project*....

b. The New Project window appears. In the Categories frame, select *Java > Java Application* from the panel on the right, and click *Next*.

c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.

d. Click *Finish* to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

**Step 5: Link libraries to the project**

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the **XBJL_X.Y.Z.zip library**.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

**Option 1: Eclipse**

a. Go to the *Libraries* tab of the New Java Project window.

b. Click *Add External JARs*....

c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbee-java-library-X.Y.Z.jar** file.

d. Click *Add External JARs*... again.

e. Go to the *extra-libs* folder and select the following files:
   
   - rtxt-2.2.jar
   - slf4j-api-x.y.z.jar
   - slf4j-nop-x.y.z.jar

f. Expand the **rtxt-2.2.jar** file of the Libraries tab list, select *Native library location*, and click *Edit*....

---

*XBeep-PRO 900HP DigiMesh Kit User Guide*
g. Click **External folder...** to navigate to the `extra-libs\native\Windows\win32` folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Replace `Windows\win32` with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

   ```
   java -version
   ```

h. Click **OK** to add the path to the native libraries.

i. Click **Finish**.

or

**Option 2: NetBeans**

a. From **Projects** view, right-click your project and go to **Properties**.

b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.

c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the `xbjlib-X.Y.X.jar` file.

d. Click **Add JAR/Folder** again.

e. Go to the **extra-libs** folder and select the following files:
   - `rxx-2.2.jar`
   - `slf4j-api- x.y.z.jar`
   - `slf4j-nop- x.y.z.jar`

f. Select **Run** in the left tree of the **Properties** dialog.

g. In the **VM Options** field, add the following option:

   ```
   -Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
   ```

   where:
   - `<path_where_the_XBee_Java_Library_is_unzipped>` is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
   - `Windows\win32` is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

   ```
   java -version
   ```

h. Click **OK**.

**Step 6: Add the source code to the project**

Follow these steps to add the source to the project.
1. Open the following source code, select all, and copy it to the clipboard: `MainApp.java`.
2. Add the Java source file with Eclipse or NetBeans.

**Option 1: Eclipse**

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: **MainApp**.

d. Click **Finish**.

e. The `MainApp.java` file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

**Option 2: NetBeans**

a. In the **Projects** view, select the project and right-click.

b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.

c. Modify the **Class Name** to be **MainApp**.

d. Click **Finish**.

e. The `MainApp.java` file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

**Option 1: Eclipse**

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: **MainApp**.

d. Click **Finish**.

e. The `MainApp.java` file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

**Option 2: NetBeans**

a. In the **Projects** view, select the project and right-click.

b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.

c. Modify the **Class Name** to be **MainApp**.

d. Click **Finish**.

e. The `MainApp.java` file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (Move class to correct folder) to resolve the error.

Step 7: Set the port name and launch the application
For this step, set the port name and launch the application.
1. Change the port name in the Java source code to match the port that the RECEIVER module (XBee A) is connected to.

```java
// TODO: Replace with the port where your receiver module is connected
private static final String PORT = "COM1";
// TODO: Replace with the baud rate of your receiver module.
private static final int BAUD_RATE = 9600
```

2. Launch the application on your computer. Notice that every time you press or release the SENDER_1 (XBee B) or SENDER_2 (XBee C) user button, RECEIVER (XBee A) receives its status.
3. Press the button and check the received status. The output of the application should be similar to the following:

```
WARNING: RXTX Version mismatch
 Jar version = RXTX-2.2pre1
 native lib Version = RXTX-2.2pre2

Listening for IO samples... Press the user button of any remote device.

Digital data from '0013A20012345678': Low (button pressed)
Digital data from '0013A20012345678': High (button released)
Digital data from '0013A20012345678': Low (button pressed)
Digital data from '0013A20012345678': High (button released)
```

Step 8: Section summary of receiving digital data
In this section, you have learned that:
All XBee modules have a set of pins you can use to connect and configure sensors or actuators.

A sensor is a device that provides a corresponding output as a response to events or changes in quantities. There are two types:
- Digital sensors return discrete values such as on/off.
- Analog sensors can return a wide variety of values such as the temperature of a room.

If, as in this example, you want to read data from a digital sensor, you must configure the selected IO as Digital Input.

You can obtain digital data from a sensor by configuring the remote XBee to transmit the IO data:
- To the local device by setting the DH and DL parameters to the MAC of the receiver module.
- When a digital pin changes (using the IC or Digital IO Change Detection parameter) as you did in this example, or periodically (using the IR or IO Sampling Rate parameter).

Although the Digital IO Change Detection (IC) parameter is configured to monitor one or more pins, changes in these pins do not wake an end device while it is sleeping.

The data sent from one module to the other is called IO Sample. It contains the inputs (DIO lines or ADC channels) for which sampling has been enabled. It also contains the value of all enabled digital and analog inputs.

**Step 9: Do more with receiving digital data**

If you're ready to work more extensively with receiving digital data, try the following:

- Modify the example and add sleep support to one or both modules connected to the button (Sender_1, Sender_2). They can sleep for three seconds, then wake for one second and send the button status.
- Instead of using the button on the board, connect a digital Grove sensor (for example, a motion sensor) to the board.
- Form a larger sensor network by adding more XBee modules and configuring them to send digital data to RECEIVER.

**Lab: Receive analog data**

This section demonstrates how to create an XBee sensor network. Since the kit does not contain a real sensor, you will simulate an analog sensor with the potentiometer of the XBee Grove Development Board.

**Note** If you have a Grove sensor, you can connect it to the board for a more realistic example of a sensor network.

One of the modules is configured as sleep coordinator, the other two as sleeping routers. The routers, where the sensor is connected, are the senders. Both sleep for 7.5 seconds and then wake for 2.5 seconds to transmit the value of the potentiometer to the sleep coordinator (the receiver).

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.
**Step 1: Requirements**
For this setup you need the following hardware and software.

**Hardware**
- Three XBee-PRO 900HP DigiMesh Kit modules
- Three XBee Grove Development Boards
- Three micro USB cables
- One computer

**Software**
- XCTU 6.3.1 or later
- XBee Java Library (XBJL-X.Y.Z.zip release file)
- Java Virtual Machine 6 or later
- A Java IDE (such as Eclipse or NetBeans)

*Tip* For more information about XCTU, see the [XCTU walkthrough](#).

**Step 2: Connect the components**
If you have a Grove sensor (not included in the kit), plug it into the Grove AD2 connector of the XBee B or XBee C (senders) board. Follow the steps to connect your modules:
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in [Plug in the XBee module](#).
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

**Step 3: Configure the XBee modules**
XBee B and XBee C, the sleeping routers, will sleep for 7.5 seconds. After this sleep period, they will wake for 2.5 seconds and send the analog value read from the potentiometer to XBee A, the sleep coordinator. They will then enter low-power mode for another 7.5 seconds.

Set the destination address \((DH + DL)\) of the senders (XBee B and XBee C) to the MAC address \((SH + SL)\) of the receiver (XBee A). Additionally, configure the pin where the potentiometer is connected \((DIO3/AD3)\) as an analog input, and set the sample rate parameter \((IR)\) to 7.5 seconds.
1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>XBee C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios on your network.</td>
</tr>
<tr>
<td>DH</td>
<td>—</td>
<td>0013A200</td>
<td>0013A200</td>
<td>Defines the destination address (high part) to transmit the data to. The value of this setting should be the Serial Number High (SH) of the RECEIVER module.</td>
</tr>
<tr>
<td>DL</td>
<td>—</td>
<td>SL of XBee A</td>
<td>SL of XBee A</td>
<td>Defines the destination address (low part) to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the RECEIVER module.</td>
</tr>
<tr>
<td>NI</td>
<td>RECEIVER</td>
<td>SENDER_1</td>
<td>SENDER_2</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
</tbody>
</table>

⚠️ The default NI value is a blank space. Make sure to delete the space when you change the value.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D3</td>
<td>—</td>
<td>ADC [2]</td>
<td>ADC [2]</td>
<td>Sets the DIO3/AD3 pin as ADC in XBee B and XBee C. This pin is connected to a potentiometer.</td>
</tr>
<tr>
<td>IR</td>
<td>—</td>
<td>1D4C</td>
<td>1D4C</td>
<td>Configures XBee B and XBee C to send IO samples every 7.5 seconds (7500 ms = 1D4C in hexadecimal).</td>
</tr>
</tbody>
</table>
### Inputs and outputs

**Lab: Receive analog data**

<table>
<thead>
<tr>
<th>Param</th>
<th>XBe A</th>
<th>XBe B</th>
<th>XBe C</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO</td>
<td>001</td>
<td>—</td>
<td>—</td>
<td>Establishes XBe A to be the preferred sleep coordinator.</td>
</tr>
</tbody>
</table>
| SP    | 2EE   | —     | —     | Defines the duration of time spent sleeping.  
2EE (hexadecimal) = 750 (decimal) x 10 ms = 7.5 seconds. |
| ST    | 9C4   | —     | —     | Defines the period of inactivity (no serial or RF data received) before going to sleep.  
9C4 (hexadecimal) = 2500 (decimal) x 1 ms = 2.5 seconds. |

**Note** The dash (—) in the table means to keep the default value. Do not change the default value.

**Note** If you are using a Grove sensor and have connected it to the Grove AD2 connector, you must configure the D2 parameter as ADC [2] instead of the D3.

3. Write the settings of all XBe modules with the **Write radio settings** button at the top of the Radio Configuration section.

### Step 4: Create a Java project

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **ReceiveAnalogData**.

**Option 1: Eclipse**

a. Select **File > New**, and click the **Java Project**.

b. The **New Java Project** window appears. Enter the Project name.

c. Click **Next**.

or

**Option 2: NetBeans**

a. Select **File > New project**.

b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.

c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.

d. **Click Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.
Step 5: Link libraries to the project

This topic describes how to link the XBee Java Library, the RXTX library (including the native one), and the logger library to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the Libraries tab of the New Java Project window.
b. Click Add External JARs....
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the xbee-java-library-X.Y.Z.jar file.
d. Click Add External JARs... again.
e. Go to the extra-libs folder and select the following files:
   - rxtx-2.2.jar
   - slf4j-api-X.Y.Z.jar
   - slf4j-nop-X.Y.Z.jar
f. Expand the rxtx-2.2.jar file of the Libraries tab list, select Native library location, and click Edit....
g. Click External folder... to navigate to the extra-libs\native\Windows\win32 folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
h. Replace Windows\win32 with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

   java -version

i. Click OK to add the path to the native libraries.
j. Click Finish.

or

Option 2: NetBeans

a. From Projects view, right-click your project and go to Properties.
b. In the categories list on the left, go to Libraries and click Add JAR/Folder.
c. In the Add JAR/Folder window, search the folder where you unzipped the XBee Java Library and open the xbjlib-X.Y.Z.jar file.
d. Click Add JAR/Folder again.
e. Go to the **extra-libs** folder and select the following files:
   - rxtx-2.2.jar
   - slf4j-api-x.y.z.jar
   - slf4j-nop-x.y.z.jar
f. Select **Run** in the left tree of the **Properties** dialog.
g. In the **VM Options** field, add the following option:
   ```
   -Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
   ```
   where:
   - `<path_where_the_XBee_Java_Library_is_unzipped>` is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
   - `Windows\win32` is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don’t know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:
     ```
     java -version
     ```
h. Click **OK**.

### Step 6: Add the source code to the project

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: **MainApp.java**.
2. Add the Java source file with Eclipse or NetBeans.

**Option 1: Eclipse**

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: **MainApp**.

d. Click **Finish**.

e. The **MainApp.java** file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

**Option 2: NetBeans**

a. In the **Projects** view, select the project and right-click.

b. From the context menu, select **New > Java Class...**. The New Java Class wizard opens.

c. Modify the **Class Name** to be **MainApp**.

d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option *(Move class to correct folder)* to resolve the error.

### Step 7: Set the port name and launch the application

For this step, set the port name and launch the application.

1. Change the port name in the Java source code to match the port that the receiver (XBee A) is connected to.

   ```java
   // TODO: Replace with the port where your receiver module is connected
   private static final String PORT = "COM1";
   // TODO: Replace with the baud rate of your receiver module.
   private static final int BAUD_RATE = 9600
   ```

   **Note** If you are using a Grove sensor connected to AD2, make this additional code change to select the correct IO Line:

   ```java
   // Analog line to monitor.
   private static final IOLine LINE = IOLine.DIO2_AD2;
   ```

2. Launch the application on your computer. Every ten seconds, you will receive the value of the potentiometer connected to SENDER_1 and SENDER_2. Rotate them and see that the values change.

3. The output of the application should be similar to the following:

   ```
   +-------------------------------+
   | Receive Analog Data Sample    |
   +-------------------------------+
   WARNING: RXTX Version mismatch
   Jar version = RXTX-2.2pre1
   native lib Version = RXTX-2.2pre2
   Analog data from '0013A20011111111': 227
   Analog data from '0013A20022222222': 0
   Analog data from '0013A20011111111': 113
   Analog data from '0013A20022222222': 1023
   ```

### Step 8: Section summary of receiving analog data

In this section, you have learned that:
All XBee modules have a set of pins that you can use to connect and configure sensors or actuators.

A sensor is a device that provides a corresponding output as a response to events or changes in quantities. There are two types:
- Digital sensors return discrete values such as on/off.
- Analog sensors can return a wide variety of values such as the temperature of a room.

If, as in this example, you want to read data from an analog sensor, you must configure the selected IO as Analog to Digital Converter (ADC).

You can obtain analog data from a sensor by configuring the remote XBee module to transmit the IO data:
- To the local device, by setting the DH and DL parameters to the MAC of the receiver module.
- Periodically, as you did in this example (using the IR or IO Sampling Rate parameter).

Note: Remember that the Digital IO Change Detection (IC) feature only works for digital pins, so in this case you would not receive any data.

In this case, the data sent from one module to the other is called IO Sample. It contains the inputs (DIO lines or ADC channels) for which sampling has been enabled. It also contains the value of all enabled digital and analog inputs.

A sleeping end device will transmit periodic IO samples at the IR rate until the Time Before Sleep (ST) timer expires and the device can resume sleeping.

**Step 9: Do more with receiving analog data**

If you are ready to work more extensively with receiving analog data, try the following:

- Instead of using the potentiometer on the board, connect an analog Grove sensor to the board to create a home automation system. You can monitor a number of factors, such as:
  - Temperature
  - Humidity
  - Luminance
  - CO2
  - UV
  - Gas

You can find more analog sensors at SeeedStudio.

**How XBee modules control devices**

There are many reasons to create a sensor network—that is, to collect data from multiple nodes and bring it to a central location. There are also many reasons you may want to take remote commands from a central location and create real events in multiple physical locations.
An XBee module is capable of receiving commands that set its digital and analog output pins to trigger real-world events without the use of an external microcontroller. By itself, an XBee device can power an LED, sound a small buzzer, or even operate a tiny motor. If you use a relay, you can operate many more devices directly from the XBee module.

**Actuators**

An actuator is a device that is responsible for controlling a mechanism or system. The XBee device offers some simple output functions so that basic actuations can take place. For example, you can send digital information directly to an XBee device and direct it to turn on a light or start up a motor.

**Set pins for digital and analog actuators**

Configure the pin of your XBee device according to the actuator that is connected to it:

- If you connect a digital actuator, configure the pin as Digital Output.
- If you connect an analog actuator, configure the pin as PWM (analog output).

For more information about sensors, see How XBee modules control devices.

**How to configure a pin as an output**

**Configure a pin for digital output**

If you want to control a digital device, for example to switch a street lamp on or off, connect the device to a pin that supports digital output. XBee DigiMesh modules have 13 digital outputs (from D0 to D9 and P0 to P2). In addition, configure that pin as Digital Output Low (Digital Out, Low [4]) or Digital Output High (Digital Out, High [5]), depending on the desired default state.
Configure a pin for analog output

If you want to control an analog device—for example, to change the brightness of an LED—connect it to a pin that supports analog output (PWM). XBee-PRO 900HP DigiMesh Kits have two PWM outputs (P0 and P1). Configure that pin as PWM Output [2].

Note: PWM stands for pulse-width modulation, which is a way for digital devices to simulate an analog signal using a digital-to-analog converter (DAC). They do this by switching the pin between high and low with different duty cycles.

A full 3.3 volts from the XBee device has a 100% duty cycle. That is, the PWM output pin will always be high. If you want something halfway between off and on (1.65 V) the DAC keeps the pin high for 50% of the time and low for the remaining 50%. This happens very fast—about 16,000 times per second.

How to send actuations

Once the pin of a remote device is properly configured, you can send any actuation to that XBee device. You can use XCTU or the XBee Java Library.

To send an actuation in XCTU, create a Remote AT Command frame in the API console, configuring the following parameters:

- 64-bit destination address: the remote device’s MAC address.
- AT command (ASCII): the AT command corresponding to the IO line you want to change.
  - For digital output lines, D0-D9, P0-P1. For example, if you want to send an actuation to DIO3, type D3.
  - For analog output lines (PWM), M0-M1. For example, if you want to send an actuation to PWM1, type M1.
- Parameter value (HEX): the new value for the selected IO line.
  - For digital output lines: 04 for off, 05 for on.
  - For analog output lines: a hexadecimal value between 00 and 03FF.

Lab: Send digital actuations

This section helps you put your knowledge of sensors and actuators to work by controlling devices with XBee modules and sending various kinds of outputs.

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

Step 1: Requirements

For this setup you need the following hardware and software.

Hardware

- Two XBee-PRO 900HP DigiMesh Kit modules
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer

**Software**

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-X.Y.Z.zip release file)
- Java Virtual Machine 6 or later
- A Java IDE (such as Eclipse or NetBeans)

**Tip** For more information about XCTU, see the [XCTU walkthrough](#).

**Step 2: Connect the components**

To get started, connect the components and start XCTU.
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in [Plug in the XBee module](#).
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

**Step 3: Configure the XBee modules**

Follow these steps to configure XBee A to send digital actuations to XBee B every second. The LED of the receiver dims with each signal received.
1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>NI</td>
<td>SENDER</td>
<td>RECEIVER</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>⚠️</strong> The default NI value is a blank space. Make sure to delete the space when you change the value.</td>
</tr>
<tr>
<td>D4</td>
<td>—</td>
<td>Digital Out, High [5]</td>
<td>Sets the DIO4/AD4 pin as Digital Output High in XBee B. The LED is connected to this pin.</td>
</tr>
</tbody>
</table>

*Note* The dash (—) in the table means to keep the default value. Do not change the default value.

3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

**Step 4: Create a Java project**

Create an empty Java project named using Eclipse or NetBeans with the following project name: **SendDigitalActuations**.

**Option 1: Eclipse**

a. Select **File > New**, and click the **Java Project**.

b. The **New Java Project** window appears. Enter the Project name.

c. Click **Next**.

or

**Option 2: NetBeans**

a. Select **File > New project....**

b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.

c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
d. Click **Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

**Step 5: Link libraries to the project**

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the **XBJL_X.Y.Z.zip library**.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

**Option 1: Eclipse**

a. Go to the **Libraries** tab of the New Java Project window.

b. Click **Add External JARs...**

c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the *xbee-java-library-X.Y.Z.jar* file.

d. Click **Add External JARs...** again.

e. Go to the **extra-libs** folder and select the following files:
   - *rxtx-2.2.jar*
   - *slf4j-api-x.y.z.jar*
   - *slf4j-nop-x.y.z.jar*

f. Expand the *rxtx-2.2.jar* file of the Libraries tab list, select **Native library location**, and click **Edit...**

g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

   ```
   java -version
   ```

h. Click **OK** to add the path to the native libraries.

i. Click **Finish**.

**Option 2: NetBeans**

a. From **Projects** view, right-click your project and go to **Properties**.

b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.

c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the *xbjlib-X.Y.X.jar* file.

d. Click **Add JAR/Folder** again.
e. Go to the extra-libs folder and select the following files:
   - rxtx-2.2.jar
   - slf4j-api-x.y.z.jar
   - slf4j-nop-x.y.z.jar
f. Select Run in the left tree of the Properties dialog.
g. In the VM Options field, add the following option:

```
-Djava.library.path=<path_where_the_XBee_Java.Library_is_unzipped>\extra-libs\native\Windows\win32
```

where:
- `<path_where_the_XBee_Java.Library_is_unzipped>` is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
- `Windows\win32` is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don’t know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```
h. Click OK.

**Step 6: Add the source code to the project**

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: MainApp.java.
2. Add the Java source file with Eclipse or NetBeans.

**Option 1: Eclipse**

a. In the Package Explorer view, select the project and right-click.
b. From the context menu, select New > Class. The New Java Class wizard opens.
c. Type the Name of the class: MainApp.
d. Click Finish.
e. The MainApp.java file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (Move 'MainApp.java' to package '...') to resolve the error.

**Option 2: NetBeans**

a. In the Projects view, select the project and right-click.
b. From the context menu, select New > Java Class... The New Java Class wizard opens.
c. Modify the Class Name to be MainApp.
d. Click Finish.
e. The MainApp.java file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (Move class to correct folder) to resolve the error.

**Step 7: Set the port name and launch the application**

For this step, set the port name and launch the application.

**Step 8: Section summary of sending digital actuations**

In this section, you have learned that:

- All XBee modules have a set of pins you can use to connect and configure sensors or actuators.
- An actuator is a device that controls a mechanism or system. For instance, you can use an XBee module connected to an actuator to send digital information to another XBee module so that it raises or lowers your window blinds.
- You can create a network with a central node that sends orders to remote nodes. This network allows you to trigger real-world events wirelessly, such as switching on all the lights at home, via actuators connected to remote node(s).
- The default state of a pin that supports digital output depends on the values of the DIO setting:
  - Digital Output, Low [4]: the output is set by default to low.
  - Digital Output, High [5]: the output is set by default to high.

**Step 9: Do more with sending digital actuations**

If you’re ready to work more extensively with actuations, try the following:

- Add sensors to your network, as explained in Lab: receive digital data and Lab: Receive analog data. Then control your actuators depending on the value returned. For example, switch the RECEIVER’s (XBee B) LED on when a button connected to SENDER (XBee A) is pressed, or when the value of the SENDER’s potentiometer exceeds a defined threshold.
- Add sensors to your network, as it is explained in the Lab: receive digital data and Lab: Receive analog data examples, and control your actuators depending on the value returned. For example, switch the XBee B LED on when a button connected to XBee A is pressed, or when the value of the XBee A potentiometer exceeds a defined threshold.
- Instead of controlling an LED, connect a relay to one of the digital output pins to create a home automation system. You can:
  - Switch lights on/off.
  - Switch the irrigation system of your garden on/off.
  - Raise/lower the blinds.
  - Control your garage door.
- Extend the network by adding more XBee modules connected to different devices.
Control all your devices remotely with a smartphone application connected to an XBee Gateway. See Related products.

Lab: Send analog actuations
This section describes how to create a Java application that will dim the LED of a remote XBee module. XBee A, the sender, transmits the analog actuation to XBee B, which is connected to the LED. If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

Step 1: Requirements
For this setup you need the following hardware and software.

Hardware
- Two XBee-PRO 900HP DigiMesh Kit modules
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer

Software
- XCTU 6.2.0 or later
- XBee Java Library (XBJL-X.Y.Z.zip release file)
- Java Virtual Machine 6 or later
- A Java IDE (such as Eclipse or NetBeans)

Tip For more information about XCTU, see the XCTU walkthrough.

Step 2: Connect the components
To get started, connect the components and start XCTU.
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

Step 3: Configure the XBee modules
Follow these steps to configure XBee A to send analog actuations to XBee B every second. The LED of the receiver blinks with each signal received.
1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.

2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A</th>
<th>XBee B</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>NI</td>
<td>SENDER</td>
<td>RECEIVER</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>![Note](The default NI value is a blank space. Make sure to delete the space when you change the value.)</td>
</tr>
<tr>
<td>P0</td>
<td>—</td>
<td>PWM0 Output [2]</td>
<td>Sets the PWM0 pin as PWM Output in XBee B. The LED is connected to this pin.</td>
</tr>
</tbody>
</table>

**Note** The dash (—) in the table means to keep the default value. Do not change the default value.

3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

### Step 4: Create a Java project

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **SendAnalogActuations**.

**Option 1: Eclipse**

a. Select **File > New**, and click the **Java Project**.

b. The **New Java Project** window appears. Enter the Project name.

c. Click **Next**.

or

**Option 2: NetBeans**

a. Select **File > New project....**

b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.
c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.

d. Click Finish to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

Step 5: Link libraries to the project

This topic describes how to link the XBee Java Library, the RXTX library (including the native one), and the logger library to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the Libraries tab of the New Java Project window.
b. Click Add External JARs....
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the xbee-java-library-X.Y.Z.jar file.
d. Click Add External JARs... again.
e. Go to the extra-libs folder and select the following files:
   - rtxt-2.2.jar
   - slf4j-api-x.y.z.jar
   - slf4j-nop-x.y.z.jar
f. Expand the rtxt-2.2.jar file of the Libraries tab list, select Native library location, and click Edit....
g. Click External folder... to navigate to the extra-libs\native\Windows\win32 folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   i. Replace Windows\win32 with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

a. Click OK to add the path to the native libraries.
b. Click Finish.

or

Option 2: NetBeans

a. From Projects view, right-click your project and go to Properties.
b. In the categories list on the left, go to Libraries and click Add JAR/Folder.
c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the *xbjlib-X.Y.Z.jar* file.

d. Click **Add JAR/Folder** again.

e. Go to the **extra-libs** folder and select the following files:
   - `rtxt-2.2.jar`
   - `slf4j-api-x.y.z.jar`
   - `slf4j-nop-x.y.z.jar`

f. Select Run in the left tree of the **Properties** dialog.

g. In the **VM Options** field, add the following option:

   ```
   -Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
   ```

   where:
   - `<path_where_the_XBee_Java_Library_is_unzipped>` is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
   - Windows\win32 is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

   ```
   java -version
   ```

h. Click OK.

### Step 6: Add the source code to the project

Follow these steps to add the source to the project:

1. Open the following source code, select all, and copy it to the clipboard: *MainApp.java*.
2. Add the Java source file with Eclipse or NetBeans.

**Option 1: Eclipse**

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: *MainApp*.

d. Click **Finish**.

e. The *MainApp.java* file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (Move 'MainApp.java' to package '...') to resolve the error.

**Option 2: NetBeans**

a. In the **Projects** view, select the project and right-click.

b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.
c. Modify the **Class Name** to be **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

**Step 7: Set the port name and launch the application**
For this step, set the port name and launch the application.
1. Change the port name in the Java source code to match the port that the SENDER module (XBee A) is connected to.

   ```java
   // TODO: Replace with the port where your sender module is connected.
   private static final String PORT = "COM1";
   // TODO: Replace with the baud rate of your sender module.
   private static final int BAUD_RATE = 9600
   ```

2. Launch the application on your computer. Notice that the LED of the RECEIVER module (XBee B) dims.

**Step 8: Section summary of sending analog actuations**
In this section, you have learned that:

- All XBee modules have a set of pins that can be used to connect and configure sensors or actuators.
- An actuator is a device that controls a mechanism or system. For instance, you can use an XBee module connected to an actuator to send digit information to another XBee module so that it raises or lowers your window blinds.
- You can create a network with a central node that sends orders to remote nodes. This network will allow you to trigger real-world events wirelessly, such as switching on all the lights at home via actuators connected to remote node(s).
- You can configure the selected pin as Analog Output (PWM Output) in order to send analog orders to dim an LED of a remote device, as in this example.
- The XBee-PRO 900HP DigiMesh Kits have two PWM outputs: **P0** and **P1**.
- PWM stands for pulse-width modulation, a way for digital devices to simulate outputting analog signals using a digital-to-analog converter (DAC).

**Step 9: Do more with sending analog actuations**
If you're ready to work more extensively with analog actuations, try the following:
Add sensors to your network, as it is explained in Lab: receive digital data and Lab: Receive analog data. Then control your actuators depending on the value returned. For example, dim the RECEIVER’s (XBee B) LED based on the value read from the potentiometer connected to SENDER (XBee A).

- Continue the home automation exercise explained in the Lab: Send digital actuations section by adding some of the following features:
  - Regulate the light intensity of a room.
  - Manage the temperature of your heating system.
  - Manipulate the speed the blinds go up and down.
  - Regulate the pressure of the irrigation system in your garden.

- Extend the network by adding more XBee modules connected to different devices.
- Control all your devices remotely with a smartphone application connected to an XBee Gateway. See Related products.
Security and encryption

Sometimes the information transmitted in an XBee network needs to be protected. For example, an XBee network transferring financial information must be carefully protected against external agents. XBee modules can be configured for secure communication via encryption keys.

Note When you enable security on a device, the information you transmit is encrypted before it is sent. Likewise, information you receive must first be decrypted in order to be readable. Consider your project requirements before enabling security on an XBee network, because this encryption/decryption process can result in a slight increase in both latency and packet size.

How to enable network security ................................................................. 130
Lab: Encrypt a simple DigiMesh network ..................................................... 130
How to enable network security

To enable secure communication, configure the following parameters with the same value in all the devices of the network.

1. Set the AES Encryption Enable (EE) parameter to 1.
2. Set the AES Encryption Key (KY) parameter to any 32 hexadecimal character string. Setting this parameter enables the encryption/decryption handshake. Once this parameter has been set, it is impossible to retrieve the actual value.

Lab: Encrypt a simple DigiMesh network

Follow these steps to add a security level to your network by encrypting communication between the three XBee modules. Note that this feature is applicable for both AT and API operating modes.

If you get stuck, see Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit.

1. Follow the steps in Lab: Set up a simple DigiMesh network.
2. When you have added the modules to XCTU and changed the value of the corresponding settings, the next step is to enable security on each module. To do so, set the EE and KY parameters as follows and write the new values:

   ![EE Encryption Enable](Enabled [1])
   ![KY AES Encryption Key](AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA)

Once you have done this, the wireless data is encrypted and the communication secure.

Test your encryption

Once you have completed the previous steps, try these methods to see how it's working:

1. Send a message from SENDER to the other two XBee devices.
   
   You will see that the message was received correctly, but because of the encryption/decryption process, you won’t notice any difference in the way the XBee modules display the information.

2. Now try enabling the encryption on one device but not on the others. Or, enable encryption on the three devices but with a different key in one of them.

   In both of these instances, the receiver modules will not receive the data.

Additional recommendations

Many external agents can compromise the integrity of your security configuration. To maintain the integrity of your secure network, remember the following:

- Safeguard the value of your KY parameter. Do it virtually, but physically as well. Most communication security holes are due to inadequate information protection.
- The key you choose should not be easy to guess. While an encryption key can be any 32-bit hexadecimal value, you can create a more complex key by combining hexadecimal characters.
- If you are developing an application that performs radio configuration, do not set the value of the encryption key (KY parameter) programmatically. Instead, use XCTU to set the value directly on the module. Otherwise, the key could be obtained by anyone reading the data stream on the serial port.
Signal strength and radio frequency range

This section describes how obstacles and other factors can impact how well the devices in your network communicate. Once you learn about the factors that can impact your signal and wireless communications, you can try performing a range test.

Distance and obstacles ................................................................. 133
Factors affecting wireless communication ................................... 134
Signal strength and the RSSI pin .................................................. 135
Range test .................................................................................. 137
Example: perform a range test .................................................. 139
**Distance and obstacles**

Basic communication systems involve the following components:

- Transmitting element
- Receiving device
- Environment through which communication is occurring
- Antennas or other focusing elements

RF communication can be compared to simple audio communication: our vocal cords transmit sound waves that may be received by someone’s eardrum. We can use a megaphone to focus and direct the sound waves in order to make the communication more efficient.

The **transmitter**’s role in wireless communication is to feed a signal to an antenna for transmission. A radio transmitter encodes data in RF waves with a certain signal strength (power output) to project the signal to a receiver.

The **receiver** gets and decodes data that comes through the receiving antenna. The receiver performs the task of accepting and decoding designated RF signals while rejecting unwanted ones.

**Antennas** are devices that focus energy in a particular direction, similar to the way the megaphone focuses voice energy. Antennas can provide different radiation patterns depending on the design and application. How much the energy is focused in a given direction is referred to as antenna gain.

The space between the transmitter and receiver is the system’s **environment**. Attaining RF line-of-sight (LOS) between sending and receiving antennas is essential to achieving long range in wireless communication. There are two types of LOS that are generally used to describe an environment:

- **Visual LOS** is the ability to see from one site to the other. It requires only a straight linear path between two points.

- **RF LOS** requires not only visual LOS, but also a football-shaped path called a **Fresnel zone** that is free of obstacles so data can travel optimally from one point to another. The Fresnel zone can be thought of as a tunnel between two sites that provides a path for RF signals.
Factors affecting wireless communication

Although the communication distance specified for some XBee devices can be up to 25 miles or more, this value may be affected by factors that may decrease the quality of the signal:

- **Some materials can reflect the radio frequency waves, causing interference** with other waves and loss of signal strength. In particular, metallic or conductive materials are great reflectors, although almost any surface can reflect the waves and interfere with other radio frequency waves.

- **Radio waves can be absorbed by objects in their path, causing loss of power** and limiting transmission distance.

- **Antennas can be adjusted to increase the distance that data can travel in a wireless communication system.** The more focus the antenna can apply, the more range the system will yield. High-gain antennas can achieve greater range than low-gain antennas, although they cover less area.

A flashlight can help illustrate the principle. Some flashlights allow the user to adjust the beam of light by twisting the lens to focus or spread the beam of light. When the lens spreads—or diffuses—the beam of light, that beam of light travels a shorter distance than when the lens is twisted to focus the beam of light.

- **Line-of-sight can help increase reliability of the signal.**

To achieve the greatest range, the football-shaped path in which radio waves travel (Fresnel zone) must be free of obstructions. Buildings, trees, or any other obstacles in the path will decrease the communication range. If the antennas are mounted just off the ground, over half of the Fresnel zone ends up being obstructed by the curvature of the earth, resulting in significant reduction in range. To avoid this problem, mount the antennas high enough off the ground that the earth does not interfere with the central diameter of the Fresnel zone.
Signal strength and the RSSI pin

The Received Signal Strength Indicator (RSSI) measures the amount of power present in a radio signal. It is an approximate value for signal strength received on an antenna. Measuring the signal strength at the receiving antenna is one way to determine the quality of a communication link. If a distant transmitter is moved closer to a receiver, the strength of the transmitted signal at the receiving antenna increases. Likewise, if a transmitter is moved farther away, signal strength at the receiving antenna decreases.

The RSSI is measured in dBm. A greater negative value (in dBm) indicates a weaker signal. Therefore, -50 dBm is better than -60 dBm.

XBee module's pin 6 can be configured as an RSSI pin that outputs a PWM (pulse-width modulation) signal representing this value. To do so, configure P0 as RSSI [1]:

![Diagram of signal strength and radio frequency range]
The XBee Grove Development Board includes an LED connected to the XBee module's pin 6. When this pin is configured as the RSSI pin, the LED lights every time the connected XBee module receives data. Its intensity represents the RSSI value of the last-received data: a brighter light means a higher RSSI value and better signal quality.

Configure the amount of time the RSSI pin is active, and therefore the amount of time the LED will remain lit, by modifying the RSSI PWM Timer (RP) setting:

<table>
<thead>
<tr>
<th>RP</th>
<th>RSSI PWM Timer</th>
<th>1E</th>
<th>x 100 ms</th>
</tr>
</thead>
</table>

RP value is expressed in hexadecimal notation. For example, a configured value of 0x1E is equivalent to 30 in decimal and means that the pin will be active for three seconds (30*100=3000ms.) So the LED will light for a total of three seconds, representing the last RSSI value.

After the RP time has elapsed and no data has been received, the pin will be set to low and the LED will not light until more data is received. The pin will also be set to low at power-up until the first data packet is received. A value of 0xFF permanently enables the pin; when configured in this way, it will always reflect the RSSI value of the last-received data packet.

Although the luminosity variations of the RSSI LED may be difficult to distinguish, the LED can be used to verify successful receipt of data packets. Each time the XBee module
receives data, the LED is solid during the configured time.

---

**Note: Received Signal Strength (DB) parameter**

The RSSI value can also be obtained by reading the XBee DB parameter value. It represents the RSSI absolute value of the last received data packet expressed in hexadecimal notation.

<table>
<thead>
<tr>
<th>DB Received Signal Strength</th>
<th>3E</th>
</tr>
</thead>
</table>

---

**Is RSSI the best indication of link quality?**

One thing to keep in mind is that the RSSI is only an indication of the RF energy detected at the antenna port. The power level reported could be artificially high because it may include energy from background noise and interference as well as the energy from the desired signal. This situation is worse in an interference-prone environment where it is possible to get consistently high RSSI readings, yet still have communication errors.

If the application is attempting to measure "link reliability" and not just "signal strength," it may be helpful to factor in "% packets received" or similar data.

**Tip** A range test is always a good idea, as it allows you to measure link performance in terms of signal strength and packet success rate. This will help you determine the reliability of your RF system. For more information, see Example: perform a range test.

---

**Range test**

Since the communication between XBee modules takes place over the air, the quality of the wireless signal can be affected by many factors: absorption, reflection of waves, line-of-sight issues, antenna style and location, etc.

A range test demonstrates the real-world RF range and link quality between two XBee modules in the same network. Performing a range test will give an initial indication of the expected communication performance of the kit components.

When deploying an actual network, multiple range tests are recommended to analyze varying conditions in your application.

XCTU allows you to perform a range test with at least one XBee module connected to your computer (local) and another remote XBee module, both in the same network. The range test involves sending data packets from the local XBee module to the remote and waiting for the echo to be sent from the remote to the local. During this process, XCTU counts the number of packets sent and received by the local module and measures the signal strength of both sides (RSSI):

- RSSI is the Received Signal Strength Indicator value.
- Every sent packet from the local XBee module should be received again as an echo by the same local XBee module.
There are two types of range tests:

- **Loopback cluster** (0x12): The range test is performed using explicit addressing frames/packets directed to the Cluster ID 0x12 on the data endpoint (0xE8) which returns the received data to the sender. Not all XBee variants support the Loopback Cluster. XCTU Range Test tool displays an error when this method is selected and the XBee module does not support it.

- **Hardware loopback**: The range test is performed using the serial port/USB hardware loopback capabilities. To use this type, the remote module must be configured to work in transparent mode and the loopback jumper must be closed before starting. This causes any received data to be transmitted back to the sender.

---

**Note** The local XBee module (the one attached to your computer) can be configured to use API or transparent mode. The RSSI value of the remote device can only be read when the local XBee module is working in API mode.

Once the range test process has started, XCTU represents the retrieved data in three ways:

- **RSSI Chart** represents the RSSI values of the local and remote devices during the range test session. The chart also contains the percentage of success for the total packets sent.

- **Local and Remote instant RSSI value** display the instant RSSI value of the local and remote devices. This value is retrieved for the last packet sent/received.
Packet summary displays the total number of packets sent, packets received, transmission errors, and packets lost. It also displays the percentage of success sending and receiving packets during the range test session.

Example: perform a range test

Follow the steps in this section to perform a range test with XCTU using the loopback cluster of your XBee modules.

The steps show you how to review the RSSI of both local and remote modules and the number of packets successfully sent and received by the local module during the range test session.

Step 1: Requirements

Hardware
- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two
Software
- XCTU 6.3.1 or later

Step 2: Connect the components
To get started, connect the components and start XCTU.
1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in Configuration working mode.

Step 3: Configure the XBee modules
Both modules must be in the same network in order to communicate during the range test.
The XBee-PRO 900HP modules support the loopback cluster (0x12). Using this type of range test is advantageous because you don’t have to close the loopback jumper of the remote module and the module can work in any operating mode.
To configure your modules to perform the range test, follow these steps:
1. Restore the default settings of all XBee modules with the Load default firmware settings button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

<table>
<thead>
<tr>
<th>Param</th>
<th>XBee A (local)</th>
<th>XBee B (remote)</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>2015</td>
<td>2015</td>
<td>Defines the network that a radio will attach to. This must be the same for all radios in your network.</td>
</tr>
<tr>
<td>NI</td>
<td>LOCAL_XBEE</td>
<td>REMOTE_XBEE</td>
<td>Defines the node identifier, a human-friendly name for the module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default NI value is a blank space. Make sure to delete the space when you change the value.</td>
</tr>
</tbody>
</table>
3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

**Perform a range test**

**Step 5: Section summary of signal strength**
In this section, you have learned the following:
- There are two types of line-of-site (LOS) that describe an environment:
  - Visual LOS describes the ability to see from one place to the other. It requires only a straight linear path.
  - RF LOS requires visual LOS and a Fresnel zone free of obstacles for data to travel optimally.
- Almost any surface, especially metallic or conductive materials, can cause interference and a resulting loss of quality in transmitted data.
- Adjust antennas to increase the distance data can travel. Place them high up off the ground to help increase their range.
- The RSSI or Received Signal Strength Indicator measures the amount of power present in a radio signal. It is measured in dBm, and its depends on the protocol.
- XBee pin 6 (P0) can be configured as RSSI. The Grove board includes an LED connected to it which visually represents the RSSI value of the last-received data by varying its light intensity.
- The RSSI PWM Timer (RP) parameter helps you configure the amount of time the RSSI pin is active.
- The Received Signal Strength (DB) parameter represents the absolute RSSI value, in hexadecimal notation, of the last data packet received.
- A range test determines the real-world RF range and link quality between two XBee modules in the same network by sending data packets from the local device to the remote device and waiting for an echo.
- There are two types of range tests: loopback cluster and hardware loopback. The loopback cluster test is preferable, as it doesn’t require you to change the loopback jumper of the remote module’s board; however, this type of test is not supported by all XBee variants.
Radio firmware

Radio firmware is the program code stored in the radio module's persistent memory that provides the control program for the device. The firmware programmed may determine the protocol of the radio (802.15.4, Zigbee, DigiMesh, or Wi-Fi) if several are compatible, or, in some cases, the role of the module or its operating mode.

Update the firmware of an XBee module locally by connecting the module to your computer or over the air if the module is remotely located. Both methods are supported by XCTU.

Firmware identification ................................................................. 143
Update radio firmware ................................................................. 143
Download new firmware ............................................................... 144
Firmware identification

Identify the firmware of an XBee module using three elements:

- **Product family** indicates the XBee type. The product family of the XBee module is printed on the back of the module.

- **Function set** determines the available functionality. For some modules this may include choose transparent or API mode; or whether the device is an end device, router, or coordinator. There are also function selections that allow you to choose firmware for several of Digi's special sensor and adapter modules.

- **Version** is a unique number used to identify the firmware release. The firmware version of an XBee module is reported by the Firmware Version (VR) parameter.

![VR Firmware Version]

XBee firmware version numbers have five hexadecimal digits using "ABCDE" convention. "A" is an optional digit and if it is not present, it assumes a 0.

![Firmware information]

**Update radio firmware**

Upgrading the radio firmware of an XBee device can be a common task. For example, you need to update firmware regularly if you want to keep your modules up-to-date and take advantage of improvements and new features implemented in the latest version. You can also change the firmware when you need to use a different function set.
XCTU allows you to upgrade or change the firmware of XBee devices physically attached to your computer or located in remote places over the air using the **Update firmware** tool.

XCTU includes a set of radio firmware files that you can use to update your modules. The Update firmware tool filters these available firmware versions to only list those that are compatible with the selected XBee module.

### Update your device's firmware

Use XCTU to update your module's radio firmware:

1. Start XCTU.
2. Switch to **Configuration working mode**.
3. Select a radio module from the device list.
4. Click the **Update firmware** button. A dialog box appears displaying the available and compatible firmware for the selected module.
5. Choose the firmware family, function, and version. If you don't remember the firmware version that is currently flashed in your module, click **Select current** to automatically select it.
6. Click **Update**. A dialog box displays update progress.

**Note** If **Maintain current module configuration** is checked, XCTU attempts to reconfigure the module with the current setting values once the new firmware has been flashed.

### Download new firmware

Digi periodically releases new versions of radio firmware that fix issues, improve functionality, or add new features. Digi also launches new XBee modules in the market that require new radio firmware to be configured with XCTU. These firmware files might not be included with XCTU and need to be downloaded.

XCTU has the ability to download and install the radio firmwares library from the application itself. By default, XCTU is configured to automatically look for new radio firmware inside Digi's update site when XCTU is started. You can manually launch this process and install firmware previously downloaded or provided by Digi.

1. In XCTU, select **Help > Update the Radio Firmware Library**.
   - To look for new firmware inside Digi's update site, select **Remote server**.
   - To add a local XBee firmware file to the XCTU library, select **Local file** and specify the path where the file is located.
2. Click **OK** to start. A dialog displays the status of the download process.

When the process finishes, a new dialog displays the list of downloaded firmware.

**Note** Downloading the firmware does not automatically update attached modules.
Troubleshooting for XBee-PRO 900HP DigiMesh Kit

If you encounter problems while working on your XBee-PRO 900HP DigiMesh Kit, try the following troubleshooting tips.

General .......................................................... 146
XCTU ................................................................ 147
DigiMesh network setup .................................. 148
Wireless data transmission .............................. 148
XBee Java library ............................................. 148
Synchronous sleep .......................................... 151
Range test ...................................................... 151
General

Cannot find the serial port for the module
You can remove the XBee Grove Development Board from the USB port and view which port name no longer appears in your port list. The name that no longer appears is your XBee board.
To use XCTU to determine the correct serial port:
1. Open XCTU, Click the Discover radio modules button.
2. Select all ports to be scanned.
3. Click Next and then Finish.
Once the discovery process completes, a new window notifies you of the devices discovered and the details. The serial port and the baud rate appear in the Port label.

Can't identify XBee modules
Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.
When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.

Error: Port is already in use
The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed
Sometimes when you connect the XBee Grove Development Board into your computer, the operating system cannot install the driver automatically. If you get this error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.
If this does not fix the problem, see Optional: Manually install USB drivers.
**XCTU**

**Error upon installation of XCTU**
XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

**Discovery process either does not find devices, or XCTU does not list serial ports**
There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.

- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.
XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.
- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

DigiMesh network setup

The modules are not found when checking the network

Ensure that all modules have the same Channel Mask (CM) and Network ID (ID) values. Reset the modules and try again.

Wireless data transmission

Error: Parsing text error message

If you see the "Error parsing the text..." error when sending a message, this indicates that you typed the message incorrectly. Remember to follow this pattern when sending a message:
- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_C:

  XBEE_C: Hi XBee

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes in the network:

  ALL: Hi XBees

Error: Could not find the module in the network

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the list of devices found by the application. If it is not there, launch the application again.

XBee Java library

This section describes how to troubleshoot potential problems with the XBee Java library.
Warning message: RXTX version mismatch
If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

Invalid operating mode exception message
If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

Could not determine operating mode:
```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not determine operating mode.
    at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
    at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```
In this case, the library cannot access the module. Check that the PORT constant is correct.

Unsupported operating mode:
```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported operating mode:
at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```
This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

Java lang unsatisfied exception message
If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

"no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":
```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver
    at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
    at java.lang.Runtime.loadLibrary0(Runtime.java:849)
    at java.lang.System.loadLibrary(System.java:1087)
    at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
    at com.digi.xbee.api.connection.serial.SerialPortRxTx.open(SerialPortRxTx.java:161)
    at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
    at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```
This exception indicates that the RXTX native library is not linked to the rtxx-2.2.jar file. See the second step of the Link the libraries to the project section.

"Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):
```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\rxtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver
    Exception in thread "main" java.lang.UnsatisfiedLinkError:
```

---

Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit
XBee Java library
Troubleshooting for XBee-PRO 900HP DigiMesh Kit kit

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

Interface in use exception message

If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

Java lang no class definition exception message

If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (slf4j-api-1.7.7.jar) is not linked to the project. See the Link the libraries to the project section to know which libraries you have to link.
SLF4J class path contains multiple bindings message

If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

- xbee-java-library-X.Y.Z.jar
- rtxx-2.2.jar
- slf4j-api-x.y.z.jar
- slf4j-nop-x.y.z.jar

XBee Java Library and transparent mode

The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Synchronous sleep

A receiver module does not receive any message

Ensure that the module is joined to the network. Verify the following:

- The Associate LED of RECEIVER_1 and RECEIVER_2 is blinking (250 ms blink time) when the modules are awake.
- The Associate LED of SENDER is also blinking when the network is awake but more slowly (500 ms blink time). If not, reset the XBees and wait a few seconds or try to reconfigure all the modules.

Range test

Error: There are not remote devices discovered for the selected local device

The local device you have selected has no remote devices. Click the Discover remote devices button and XCTU will discover devices on the local device’s network.

There are no remote devices to select

If there are no remote XBee modules to select in the Radio Range Test dialog, try one of the following resolutions.

Check cables

The USB cables should be firmly and fully attached to both the computer and the XBee development board. When attached correctly, the association LED on the adapter is lit.

Check that the XBee module is fully seated in the XBee Grove Development Board

When the XBee module is correctly installed, it is pushed fully into the board and no air or metal is visible between the plastic of the adapter socket and the XBee module headers. Also, check that all
ten pins on each side of the XBee module are in a matching hole in the socket.

**Check the XBee module orientation**
The angled "nose" of the XBee module should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development board.

**Check that the XBee modules are in the same network**
Check that the Network ID (ID), Preamble ID (HP), and Channel Mask (CM) settings have the same value for both XBee modules.

**Restore default settings**
If the XBee modules are properly connected and in the same network, restore default settings and configure them again.

**The local RSSI and the number of packets received are always 0**

**Check cables**
The USB cables should be firmly and fully attached to both the computer and the XBee development board. When attached correctly, the association LED on the adapter is lit.

**Check that the XBee module is fully seated in the XBee Grove Development Board**
When the XBee module is correctly installed, it is pushed fully into the board and no air or metal is visible between the plastic of the adapter socket and the XBee module headers. Also, check that all ten pins on each side of the XBee module are in a matching hole in the socket.

**Check the XBee module orientation**
The angled "nose" of the XBee module should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development board.

**Remote RSSI is not included in the chart and the Remote RSSI control is disabled**
The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.
To display the remote RSSI value, reconfigure the local module to work in API mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
</table>
Additional resources

Wireless connectivity offers almost unlimited options for making our surroundings smarter, more efficient, and more connected. Now that you have completed the activities in this kit, here are some additional resources to help you explore XBee modules.

XBee buying guide ................................................................. 154
Where to buy XBee devices ...................................................... 154
XCTU walkthrough ............................................................... 155
Real projects with XBee modules ............................................. 160
Related products .................................................................. 162
XBee buying guide

You have become familiar with the XBee modules included in the kit, but Digi makes a large variety of modules with different features and for different functions. So, which module is best suited for your applications? Why are there different types of antennas? Should you use a "PRO" version, or is a regular XBee module enough? In this guide, we look over the different XBee options to help you answer these questions.

Note that the XBee module you select affect the parameters of your application:
- The location of your application affects the operating frequency of the XBee modules.
- To get greater range, you may select an external antenna, a different operating frequency, or even an XBee-PRO.
- Power consumption is an important factor to consider.
- The required network topology also impacts the type of module you need.

To help you select an XBee module based on your requirements, see the XBee Family Features Comparison table.

The following sections review the different options available for XBee radios and how they affect wireless communication.

Note Not all options are available for every XBee device.

Where to buy XBee devices

We are committed to providing our customers with local sales and support across the globe. With our reach extending to more than 70 countries worldwide, we work with a number of channel partners in local countries to provide you with excellent sales and support.

Advantages of working with our network of international channel partners include:
- Local language contacts within the organization.
- In-country and local language technical support and customer service.
- In-country RMA support.
- In-country product delivery.
- Understanding of local businesses and industry segments.

Contact us online or by phone at 1-952-912-3444 for more information about which channel partner is best suited to your individual needs.

Find products from Digi and Digi distributors

Digi products are available from many sources worldwide.
- You can find Digi products through distributors. Find a distributor now.
- You can also find Digi products in our official Digi online store:
  - Americas online store (U.S., Canada, and Latin America)
  - EMEA online store (Europe, Middle East, and Africa)
  - Japan online store
  - U.S. Government Sales
Find Digi products through resellers

You can purchase XBees and other Digi networking products from online retailers:

- Adafruit
- Fry's
- Maker Shed
- Microcenter
- Parallax
- RobotShop
- Seeed Studio
- Solarbotics
- Sparkfun
- TrossenRobotics

XCTU walkthrough

This walkthrough describes the layout and basic concepts of the XCTU tool.

XCTU overview

XCTU is divided into five main sections: the menu bar, main toolbar, devices list, working area, and status bar.
Menu bar
The menu bar is located at the top of the application. You can use the menu bar to access all XCTU features, tools, and working modes.

Main toolbar
The main toolbar is located at the top of the application and is divided into three sections.

- The first section contains two icons used to add radio modules to the radio modules list. See Add radio modules to XCTU.

- The second section contains the static XCTU functionality that does not require a radio module. This section includes the XCTU tools, the XCTU configuration, the feedback form, and the help and updates functions. See XCTU tools and Configure XCTU.
The third section contains tabs corresponding to the three XCTU working modes. To use this functionality, you must have added one or more radio modules to the list. See XCTU working modes.

**Devices list**

The radio modules list, or devices list, is located on the left side of the tool and displays the radio modules that are connected to your computer. If you know the serial port configuration of a radio module, you can add it to the list directly. You can also use the discovery feature of XCTU to find radio modules connected to your PC and add them to the list. See Add radio modules to XCTU.

Depending on the protocol of the local radio modules added, you can also add remote radio modules to the list using the module's search feature.

**Working area**

The working area is the largest section and is located at the right side of the application. The contents of the working area depend on the working mode selected in the toolbar. To interact with the controls displayed in the working area, you must have added one or more radio modules to the list and one of the modules must be selected.
**Status bar**
The status bar is located at the bottom of the application and displays the status of specific tasks, such as the firmware download process.

Application working modes
A working mode represents a layout which displays operations you can perform with a radio module. Typically, the working mode functionality appears in the working area. The tool has 4 working modes:

- **Configuration mode**: Allows you to configure the selected radio module from the list.
- **Consoles mode**: Allows you to interact or communicate with the selected radio module.
- **Network mode**: Allows you to discover and see the network topology of 802.15.4, Zigbee and DigiMesh protocols.
- **Remote Manager mode**: Allows you to learn about the Digi Remote Manager® platform, create an account, and access your personal Remote Manager page.

**Note** To serve our customers most effectively, Digi International Inc. is consolidating its cloud services, Digi Device Cloud and Digi Remote Manager®, under the Remote Manager name. This phased process does not affect device functionality or the functionality of the web services and other features. However, you will find instances of both Device Cloud and Digi Remote Manager in some documentation, firmware, and user interfaces.

You can only select one working mode at a time. The Configuration mode is the default selection when you start XCTU.

**Add a module**
To work and interact with a radio module, you must plug it into a USB adapter and plug that adapter into one of your computer’s USB ports. Once you have connected the module, you must add it to the

...
list of devices. There are two different methods:

1. If you know the serial configuration of your radio module, click the **Add radio module** button to add it directly.

   ![Add radio module button]

   When the dialog opens, you must select the serial port the radio module is connected to and configure the serial settings of the port.

2. If you don’t know the serial configuration of your radio module, or which port is connected to, or if you want to add more than one module, click the **Discover radio modules** button to use the discovery utility.

   ![Discover radio modules button]

   In the Discover radio devices dialog, select the serial port in which you want to look for radio modules. If you do not know the serial port where your module is attached, select all ports. Click **Next**, and then click **Finish**.

   As radio modules are found, they appear in the Discovering radio modules… dialog box. Once the discovery process has finished, click **Add selected devices**.

### Read settings

If you are in the Configuration working mode, when you select a radio module from the list of devices XCTU displays the settings of that module in the working area. XCTU automatically reads the values and completes all the fields.

![Reading radio module settings dialog]

At any time, you can read the settings of the selected radio module. To do so, click the **Read module settings** button and XCTU reads the firmware settings and refreshes the values.

![Refresh button]

The previous button reads all the settings, but if you want to read only a specific setting you can click the **Refresh** button that appears on the right side of the setting control.
Change settings
When you change the value of a setting, the background color of the control changes depending on the status of its value. The color legend appears next to the firmware information panel and reads the following:

- **Gray**: The value of the setting is written in the radio module and matches the default value.
- **Blue**: The value of the setting is written in the radio module but is different than the default value.
- **Green**: The value of the setting has changed but it has not been written in the radio module yet.
- **Red**: The value of the setting is not valid.

Whenever you change the value of a setting, you must save the changes in the module.

Save settings
If you have changed the value of any firmware setting, click the **Write module settings** button to write the new values to the radio module.

As with the reading process, the previous button writes all the settings that have been modified. If you want to write only a specific setting, click the **Write** button that appears on the right side of the setting control.

Real projects with XBee modules
Explore the links below for real-world projects made with XBee technology.
Community

XBee Projects
The largest collection of XBee projects on the web.

Digi XBee examples and guides
Learn more about wirelessly connecting XBees to sensors, outputs, motors, lights, and the Internet.

Industrial solutions

Wireless Tank Monitoring with 1844myfuels
Wireless ultrasonic sensors connected to XBee modules enable up-to-the-minute monitoring of farm silo levels.

Devergy Expands Solar Power Possibilities in Africa
Devergy uses XBee technology for the communication network, where hundreds of nodes are connected with XBee modules—making the solar micro-grids smart, cost effective and manageable.

Tracking Hand Washing Decreases the Spread of Infection at Hospitals
Hand washing is one of the most important daily routines to avoid the spreading of bacteria and disease, especially in hospitals and healthcare centers.
XBee helps Libelium monitor harsh environments
Embedded XBee and XBee-PRO modules enable low-cost, low-power remote monitoring of isolated and difficult-to-access sensors.

For more industrial solutions, visit www.digi.com/industries.

Related products
See the following products from Digi International.

**XBee Gateway**
The low-cost XBee-to-IP solution enables remote connectivity, configuration, and management of XBee networks with Digi Remote Manager®. All XBee data sent to the gateway is automatically available to online applications via Remote Manager. Additionally, this gateway can run custom Python applications that communicate and manage your XBee network.

**XBee RF Modems**
XBee RF Modems are small, low-power devices using XBee RF modules to communicate with systems using RS-232, RS-485, and USB interfaces. You can easily make existing wired systems wireless with this out-of-box solution. XBee RF modems are ideal for extended-range applications with a high data throughput.