Digi XBee® XR 900

RF Module

User Guide

Revision history–90002474

Revision	Date	Description
A	November 2022	Initial release.

Trademarks and copyright

Disclaimers

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

Warranty

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

www.digi.com/howtobuy/terms

Customer support

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

- Product name and model
- Product serial number (s)
- Firmware version
- Operating system/browser (if applicable)
- Logs (from time of reported issue)
- Trace (if possible)
- Description of issue
- Steps to reproduce

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

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (Digi XBee® XR 900 RF Module User Guide, 90002474 A) in the subject line of your email.

Contents

XBIB-C development boards

Design notes

Pin signals

Mechanical drawings

Modes

I/O support

Serial communication

Networking

Network commissioning and diagnostics

Sleep support

AT commands

Operate in API mode

Frame descriptions

General Purpose Flash Memory

Update the firmware over-the-air

Regulatory information

Module support

Manufacturing information

About the XBee XR 900 RF Module

Applicable firmware and hardware

This user guide supports the following firmware:

- v.90xx

It supports the following hardware:

- XBee XR 900

Digi RF resources

There are many resources to further your understanding of Digi's RF devices. You can:

- Read the XBee Buying Guide.
- Ask questions on the Digi Support Forum.
- Search the Knowledge Base.
- Search the Resource Library.
- Read XBee-related posts on the Digi Blog.
- Explore hardware certifications.

One way to communicate with the XBee device is by using a software library. The libraries available for use with the XBee XR 900 RF Module include:

- XBee Java library
- XBee Python library
- XBee ANSI C library
- XBee mbed library

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

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

The XBee ANSI C Library project is a collection of portable ANSI C code for communicating with the devices in API mode.

The XBee mbed library is a ready-to-import mbed extension that dramatically reduces development time for XBee projects on mbed platforms.

Safety instructions

Safety instructions

XBee modules

- The XBee radio module cannot be guaranteed operation due to the radio link and so should not be used for interlocks in safety critical devices such as machines or automotive applications.
- The XBee radio module has not been approved for use in (this list is not exhaustive):
  - medical devices
  - nuclear applications
  - explosive or flammable atmospheres
- There are no user serviceable components inside the XBee radio module. Do not remove the shield or modify the XBee in any way. Modifications may exclude the module from any warranty and can cause the XBee radio to operate outside of regulatory compliance for a given country, leading to the possible illegal operation of the radio.
- Use industry standard ESD protection when handling the XBee module.
- Take care while handling to avoid electrical damage to the PCB and components.
- Do not expose XBee radio modules to water or moisture.
- Use this product with the antennas specified in the XBee module user guides.
- The end user must be told how to remove power from the XBee radio module or to locate the antennas 26 cm from humans or animals.

Consignes de sécurité

Modules XBee

- Le fonctionnement du module radio XBee ne peut pas être garanti en raison de la liaison radio et ne doit donc pas être utilisé pour les verrouillages dans des dispositifs critiques pour la sécurité tels que des machines ou des applications automobiles.
- Le module radio XBee n'a pas été approuvé pour une utilisation dans (cette liste n'est pas exhaustive) :
  - dispositifs médicaux
  - applications nucléaires
  - atmosphères explosives ou inflammables
- Il n'y a aucun composant réparable par l'utilisateur à l'intérieur du module radio XBee. Ne retirez pas la protection et ne modifiez en aucune façon le XBee. Les modifications peuvent exclure le module de toute garantie et peuvent entraîner le fonctionnement de la radio XBee en dehors de la conformité réglementaire pour un pays donné, ce qui peut entraîner un fonctionnement illégal de la radio.
- Utilisez la protection ESD standard de l'industrie lors de la manipulation du module XBee.
- Soyez prudent lors de la manipulation afin d'éviter des dommages électriques au circuit imprimé et aux composants.
- N'exposez pas les modules radio XBee à l'eau ou à l'humidité.

- Utilisez ce produit avec les antennes spécifiées dans les guides d'utilisation du module XBee.
- L'utilisateur final doit savoir comment couper l'alimentation du module radio XBee ou placer les antennes à 26 cm des humains ou des animaux.

Configure the XBee XR 900 RF Module

Configure the device using XCTU

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

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

XBee bootloader

You can update firmware on the XBee XR 900 RF Module serially. This is done by invoking the XBee bootloader and transferring the firmware image using XMODEM.

This process is also used for updating a local device's firmware using XCTU.

XBee devices use a modified version of Silicon Labs' Gecko bootloader. This bootloader version supports a custom entry mechanism that uses module pins DIN, $\overline{\text{DTR}}$/SLEEP_RQ, and $\overline{\text{RTS}}$.

To invoke the bootloader using hardware flow control lines, do the following:

1. Set $\overline{\text{DTR}}$/SLEEP_RQ low (CMOS0V) and RTS high.
2. Send a serial break to the DIN pin and power cycle or reset the module.
3. When the device powers up, set $\overline{\text{DTR}}$/SLEEP_RQ and DIN to low (CMOS0V) and $\overline{\text{RTS}}$ should be high.
4. Terminate the serial break and send a carriage return at 115200 baud to the device.
5. If successful, the device sends the Silicon Labs' Gecko bootloader menu out the DOUT pin at 115200 baud.
6. You can send commands to the bootloader at 115200 baud.

Note Disable hardware flow control when entering and communicating with the bootloader.

All serial communications with the module use 8 data bits, no parity bit, and 1 stop bit.

You can also invoke the bootloader from the XBee application by sending %P (Invoke Bootloader).

Send a firmware image

After invoking the bootloader, a menu is sent out the UART at 115200 baud. To upload a firmware image through the UART interface:

1. Look for the bootloader prompt **BL >** to ensure the bootloader is active.
2. Send an ASCII **1** character to initiate a firmware update.
3. After sending a **1**, the device waits for an XModem CRC upload of a .gbl image over the serial line at 115200 baud. Send the .gbl file to the device using standard XMODEM-CRC.

If the firmware image is successfully loaded, the bootloader outputs a “complete” string. Invoke the newly loaded firmware by sending a **2** to the device.

If the firmware image is not successfully loaded, the bootloader outputs an "aborted string". It returns to the main bootloader menu. Some causes for failure are:

- Over 1 minute passes after the command to send the firmware image and the first block of the image has not yet been sent.
- A power cycle or reset event occurs during the firmware load.

- A file error or a flash error occurs during the firmware load. The following table contains errors that could occur during the XMODEM transfer.

Error	Cause	Workaround
0x18	This error is observed when a serial upload attempt has been abruptly discontinued by invoking **Ctrl**+**C** and subsequently another attempt is made to upload a gbl by pressing **1** on the bootloader menu.	Press **2** on the bootloader menu. The bootloader performs a reboot and the menu gets displayed again. Now press **1** and begin uploading the gbl.

Software libraries

One way to communicate with the XBee XR 900 RF Module is by using a software library. The libraries available for use with the XBee XR 900 RF Module include:

- XBee Java library
- XBee Python library

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

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

XBee Network Assistant

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

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

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

To install the XBee Network Assistant:

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

XBee Multi Programmer

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

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

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

Download the XBee Multi Programmer application from: digi.com/support/productdetail?pid=5641

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

Specifications

The following tables provide general specifications for the hardware.

Power, sensitivity, and performance specifications

The following table describes the power, sensitivity, and performance specifications for the XBee XR 900 RF Module.

Specification	XBee XR 900 RF Module	Units
Supply voltage	2.1 – 3.6	VDC
Max RF power at 3.3 V	19	dBm
	79	mW
Receiver sensitivity at 10 kb/s	-113	dBm
TX current	110	mA
RX current	28	mA
Sleep current	1.5	µA
RF data rate	250, 110, and 10	kb/s
Indoor/urban range	Up to 2.5 km (1.5 mi)	
Outdoor RF line-of-sight range	Up to 14.5 km (9 mi)	

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

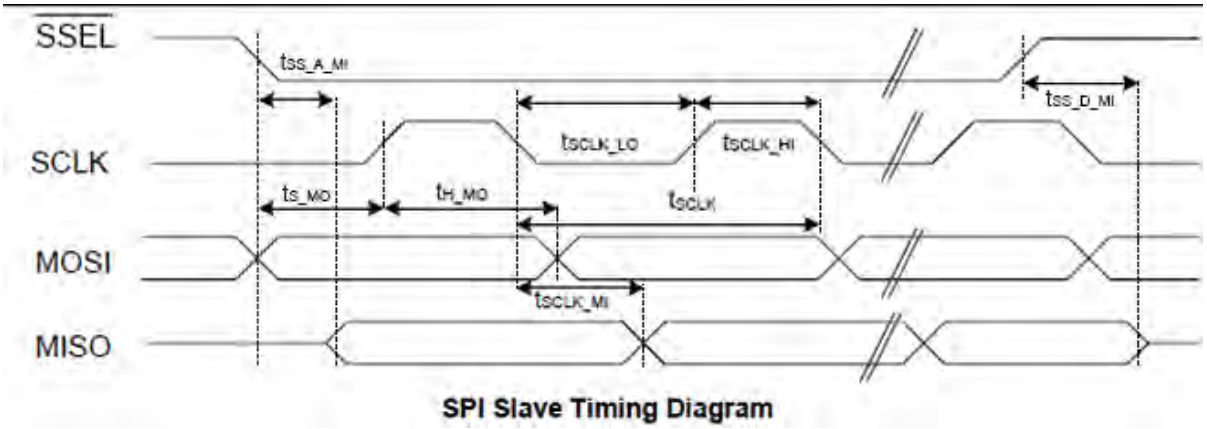
Serial interface specifications

This product supports two serial interfaces:

- UART (Universal Asynchronous Receiver Transmitter)
- SPI (Serial Peripheral Interface)

UART Baud Rate		Unit
Standard	921,600	b/s
Non-standard	967,680	b/s

SPI Timing	Min	Max	Unit
SCLK Speed		5	MHz
SCLK Period (t_{SCLK})	200		nS
SCLK High Time (t_{SCLK_HI})	63		nS
SCLK Low Time (t_{SCLK_LO})	63		nS
SSEL to MISO Active * ($t_{SS\_A\_MI}$)		70	nS
SSEL to MISO Disabled ($t_{SS\_D\_MI}$)		50	nS
MOSI Setup Time (t_{S_MO})	12.5		nS
MOSI Hold Time (t_{H_MO})	13		nS
SCLK Edge to MISO Update (t_{SCLK_MI})	43	108	nS



SPI Slave Timing Diagram

RF communications specifications

Specification	Condition	XBee value	XBee-PRO value
Frequency range		ISM 902 to 928 MHz	
RF data rate (software selectable)	Low data rate	10 kb/s	
	Middle data rate	110 kb/s	
	High data rate	250 kb/s	
Transmit power (software selectable)		Up to 13 dBm	
	New Zealand specific	Up to 13 dBm, at the high data rate (**BR** 2) it is capped at 7.5 dBm	
Maximum data throughput	High data rate	120 kb/s	
Channels		10 hopping sequences share 50 frequencies	
Available channel frequencies	Low/middle data rate	101[1]	
	High data rate	50	
Rural range line of sight	Low data rate	Up to 14.5 km (9 mi)	Up to 105 km (65 mi)[2]
Urban range line of sight	Low data rate	Up to 2.5 km (1.5 mi)	Up to 18 km (11 mi)[3]
Receiver sensitivity	Low data rate	-113 dBm	
	Middle data rate	-106 dBm	
	High data rate	-103 dBm	
Receiver IF selectivity	Low data rate ± 250 kHz	40 dB	
	Low data rate ± 500 kHz	50 dB	
	Middle data rate ± 250 kHz	30 dB	
	Middle data rate ± 500 kHz	40 dB	
	High data rate ± 500 kHz	30 dB	
	High data rate ± 1000 kHz	45 dB	

[1]The device hops on 50 channels selected using the **CM** command, from 101 available frequencies. For more details, see CM (Channel Mask).
[2]We estimate rural ranges based on a 14.5 km (9 mi) range test with dipole antennas.
[3]Range estimated assuming that the urban noise floor is approximately 15 dB higher than rural. The actual range depends on the setup and level of interference in your location.

Specification	Condition	XBee value	XBee-PRO value
Receiver RF selectivity	Below 900 MHz and above 930 MHz	> 50 dB	

General specifications

The following table describes the general specifications for the devices.

Specification	XBee XR micro-mount
Form factor	Micro-mount
Dimensions	Micro-mount: 1.36 cm x 1.93 cm x 0.241 cm (0.534 in x 0.760 in x 0.095 in)
MMT weight	1.2 grams
SMT weight	3.0 grams
TH RPSMA weight	6.1 grams
TH U.FL weight	3.0 grams
Operating temperature	-40 to 85 °C (industrial)
Antenna options	RF pad, chip antenna, or U.FL connector
Analog-to-digital converter (ADC)	4 10-bit analog inputs

Electrical characteristics

The following table lists the electrical characteristics for the XBee XR 900 RF Module. XBee XR 900 RF Modules have 15 General Purpose Input / Output (GPIO) ports available. The exact list depends on the device configuration as some GPIO pads are used for purposes such as serial communication.

Symbol	Parameter	Condition	Min	Typical	Max	Units
V_I	Input pins		-0.3		VCC +0.3	V
V_{IL}	Input low voltage				0.3 * VCC	V
V_{IH}	Input high voltage		0.7 * VCC			V
V_{OL}	Output low voltage	Sinking 3 mA VCC = 3.3 V			0.2 * VCC	V
V_{OH}	Output high voltage	Sourcing 3 mA VCC = 3.3 V	0.8 * VCC			V
I_{IN}	Input leakage current	High Z state I/O connected to Ground or VCC		0.1	100	nA
RPU	Internal pull-up resistor	Enabled		40		kΩ
RPD	Internal pull-down resistor	Enabled		40		kΩ
I_{OL}	Output source/sink current				50	mA
I_{OLT}	Total output current (for GPIO pads)				200	mA
* XBee power supply should be designed to operate at 1.8V or above.						

Regulatory conformity summary

Approval	XBee XR 900 RF Module	ID number
FCC (USA)	Yes	FCC ID: MCQ-XB9XR
ISED (Canada)	Yes	IC: 1846A-XB9XR
Australia	TBD	
New Zealand	TBD	
Brazil	TBD	
Europe	No	
UKCA	No	

Electro Static Discharge (ESD)

XBee XR 900 RF Module pins are tolerant to human-body model ± 1.0 kV.

Note Take care to limit all electrostatic discharges to the device.

Secure access

By default, the XBee XR 900 RF Module is easy to configure and allows for rapid prototyping. For deployment, you can encrypt networks to prevent unauthorized access. This can prevent entities outside of the network from accessing data on that network. Some customers may also desire a way to restrict communication between nodes from inside the same network.

There are two ways to secure your device against unauthorized access:

- Secure remote session
- Disable functionality

Secure session protects against external man-in-the middle attacks by requiring remote devices to authenticate before they are allowed to make configuration changes.

You can also disable device functionality in order to prevent unexpected malicious use of the product.

Secure Sessions

Secure Sessions provide a way to password-protect communication between two nodes on a network above and beyond the security of the network itself. With secure sessions, a device can 'log in', or create a session with another device that is encrypted and only readable by the two nodes involved. By restricting certain actions—such as remote AT commands or FOTA updates—to only be allowed over one of these secure sessions, you can make it so access to the network does not allow network configuration. A password must be set and the proper bits of SA (Secure Access) must be set to enable this feature.

The following definitions relate to secure Sessions:

Term	Definition
Client	The device that is attempting to log in and send secured data or commands is called the client.
Server	The device that is being logged into and will receive secured data or commands is called the server.
Secure Session	A secure connection between a server and a client where the pair can send and receive encrypted data that only they can decrypt.
Secure Remote Password (SRP)	Name of the authentication protocol used to create the secure connection between the nodes.
Salt	A random value generated as part of the authentication process.
Verifier	A value derived from a given salt and password.

Configure the secure session password for a device

For a device to act as a secure session server it needs to have a password configured. The password is configured on the server in the form of a salt and verifier used for the SRP authentication process. The salt and verifier can be configured in XCTU by selecting the **Secure Session Authentication** option.

We recommend using XCTU to set a password which will then generate the salt and verifier parameters, although the salt and verifier values can also be set manually. See *S (Secure Session Salt) and *V, *W, *X, *Y (Secure Session Verifier) for more information.

Note There is not an enforced password length. We recommend a minimum length of at least eight characters. The password should not exceed 64 characters, as it will exceed the maximum length of an API frame.

Start a secure session

A secure session can only be started in API mode. Once you have been authenticated you may send data in API mode or Transparent mode, but API mode is the recommended way to communicate.

To start a secure session:

1. Send a type Secure Session Control - 0x2E to your local client device with the address of the server device (not a broadcast address), the options bit field set to **0x00**, the timeout for the session, and the password that was previously set on the server.

2. The client and server devices will send/exchange several packets to authenticate the session.
3. When authentication is complete, the client device will output a Secure Session Response - 0xAE to indicate whether the login was a success or failure.

At this point if authentication was successful, the secure session is established and the client can send secured data to the server until the session times out.

Note A device can have one outgoing session—a session in which the node is a client—at a time. Attempting to start a new session while a session is already in progress automatically ends the previous session.

Note A device can have up to four incoming sessions—sessions in which the device is a server—at a time. Once that number has been reached, additional authentication requests are rejected until one of the active sessions ends.

End a secure session

A client can end a session by either waiting for the timeout to expire or by ending it manually. To end a session, send a Secure Session Control - 0x2E to the local client device with bit 0 of the options field set and with no password.

The device ends the outgoing secure session with the node whose address is specified in the type 0x2E frame. This frame can be sent even if the node does not have a session with the specified address—the device will send a message to the specified server prompting it to clear out any incoming session data related to the client (this can be used if the server and client fall out of sync. For example, if the client device unexpectedly loses power during a session.

Sending a type 0x2E frame with the logout option bit set, and the address field set to the broadcast address will end whatever outgoing session is currently active on the client and broadcast a request to all servers to clear any incoming session data related to that client.

Secured remote AT commands

Secure a node against unauthorized remote configuration

Secured Access is enabled by setting bits of SA (Secure Access). Additionally, an SRP Salt (***S**) and verifier (***V**, ***W**, ***X**, ***Y**) must be set. You can use XCTU to generate the salt and verifier based on a password.

Configure a node with a salt and verifier

In this example, the password is **pickle**.

1. The salt is randomly generated and the verifier is derived from the salt and password as follows:

   ***S = 0x1938438E**

   ***V = 0x0771F57C397AE4019347D36FD1B9D91FA05B2E5D7365A161318E46F72942A45D**

   ***W = 0xD4E44C664B5609C6D2BE3258211A7A20374FA65FC7C82895C6FD0B3399E73770**

***X =**
0x63018D3FEA59439A9EFAE3CD658873F475EAC94ADF7DC6C2C005b930042A0B74
***Y =**
0xAEE84E7A00B74DD2E19E257192EDE6B1D4ED993947DF2996CAE0D644C28E8307

Note The salt and verifier will not always be the same even if the same password is used to generate them.

2. Enforce secure access for Remote AT Commands by setting Bit 1 of the **SA** command:

   SA = **0x02**

3. Write the configuration to flash using WR (Write).

WARNING! Make sure that this step is completed. If your device resets for any reason and ***S**, ***V**, ***W**, ***X**, ***Y** and **SA** are not written to flash they will revert to defaults, rendering the node open to insecure access.

4. From now on, any attempt to issue a Remote AT Command Request - 0x17 to this device will be rejected with a **0x0B** status unless a secure session is established first.

Remotely configure a node that has been secured

In the example above a node is secured against unauthorized remote configuration. In this instance, the secured node acts as a Secure Session Server (remote). The sequence below describes how a Secure Session Client (local) can authenticate and securely configure the server remotely.

Establish a secure session using the password that was set on the server node

1. Generate a Secure Session Control - 0x2E.

- The destination address must match the 64-bit address (**SH** + **SL**) of the remote server.
- Since you are logging in, leave the options field as **0x00**.
- Set a five minute timeout, which should give sufficient time for ad hoc configuration. The units are in tenths of a second, so **0x0BB8** gives you five minutes.
- The options are set for a fixed duration, so after the five minutes expire, both the server and client emit a modem status indicating the session ended.
- Enter the original password used to generate the verifier from the random salt above.

2. Pass the type 0x2E Control frame into the serial interface of the local client:

- For example, to log into a Secure Session server at address **0013A200 417B2162** for a five minute duration using the password **pickle**, use the following frame:
  7E 00 12 2E 00 13 A2 00 41 7B 21 62 00 0B B8 70 69 63 6B 6C 65 A2

3. Wait for a Secure Session Response - 0xAE to indicate the session establishment succeeded or failed with the reason.

- The address of the remote that is responding and the status is included in the response.
- For example, the response to the request above is as follows:
  7E 00 0B AE 00 00 13 A2 00 41 7B 21 62 **00** 5D. The 0**x00** status indicates success.

4. Send remote AT Commands to the remote server using the Remote AT Command Request - 0x17 with bit 4 of the Command Options field set. Bit 4 indicates the AT command should be sent securely.

Send data to a secured remote node

The process to send secured data is very similar to remotely configuring a node. The following steps show how a client node can authenticate with a server node and send data securely.

1. Send a Secure Session Control - 0x2E to the client node with:
   - The server's 64-bit address.
   - The desired timeout.
   - The options field set to **0x00** for fixed timeout login or to **0x04** for inter-packet timeout refresh login.
   - The password of the server node.
2. Wait for the Secure Session Response - 0xAE to determine if the the authentication was successful.
3. Data can now be sent securely with Transmit Request - 0x10 and Explicit Addressing Command Request - 0x11 provided that:
   - Bit 4 in the transmit options field is set to indicate that the data should be sent encrypted.
4. The returned Receive Packet - 0x90 and Explicit Receive Indicator - 0x91 receive options fields should also have bit 4 set.

Note The maximum payload per transmission size is reduced by four bytes due to the additional encryption overhead. NP (Maximum Packet Payload Bytes) will not reflect this change when the session is going on.

A node can be secured against emitting data out the serial port that was received insecurely via the **SA** command. This means that a remote node will not emit any serial data if it was received insecurely (TO (Transmit Options) bit 4 was not set). This includes any data in Transparent mode, **0x80**, **0x90** and **0x91** frames.

End a session from a server

If bit 3 of AZ (Extended API Options) is set, the server emits an extended modem status (whenever a client establishes a session with it) that includes the 64-bit address of the client. Using these statuses the MCU connected to the server can keep track of sessions established with the server. To end a session from the server do the following:

1. Send a Secure Session Control - 0x2E to the server node with:
   - The client's 64-bit address.
   - The options field set to **0x02** for server side session termination.
   - Set the timeout to **0x0000**.

2. Wait for the Secure Session Response - 0xAE to determine if the termination was successful.
   - The client will emit a modem status **0x3C** (Session Ended).
   - The server will also emit a modem status (or an extended modem status depending on **AZ**) of **0x3C** (Session Ended).

Note The 64-bit address can be set to the broadcast address to end all incoming sessions.

Note This functionality can be used to end orphaned client-side sessions–in case the server unexpectedly reset for some reason.

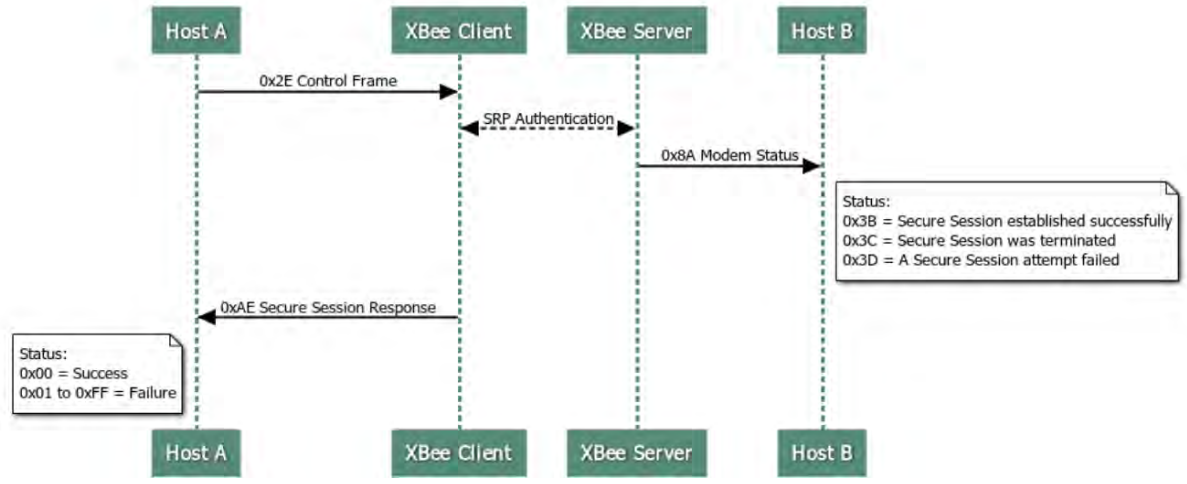
Secure Session API frames

Secure Session can only be established from a node that is operating in API mode. The server-side can be in Transparent mode, but the client must be in API mode. Once a session has been established between a client and server node, the client can be transitioned to Transparent mode; and if bit 4 of **TO** is set, the client will encrypt data sent in Transparent mode for the duration of session.

There are four frames that are used for controlling and observing a secure session.

- Secure Session Control - 0x2E: This frame is passed to the client that wishes to log into or out of a server. Any attempt to use the Control frame will generate a response frame.
- Secure Session Response - 0xAE: This frame returns the status of the previously sent 0x2E frame indicating whether it was successful or not.
- Modem Status - 0x8A: The server will also emit a modem status whenever an attempt succeeds, fails, or was terminated. The client will also emit modem statuses if the session times out.
- Extended Modem Status - 0x98: If bit 3 of **AZ** is set then modem statuses will be replaced with extended modem statuses. These frames will contain the status that caused them to be emitted as well as the address of the node that initiated the session, the session options, and the timeout value.

Frame exchanges:



Secure transmission failures

This section describes the error messages you can see when trying to send a secure packet.

Data Frames - 0x10 and 0x11 frames

- Response frame type: Extended Transmit Status - 0x8B

Possible error statuses:

Status	Description	Reason
0x34	No Secure Session Connection	The sending node does not have an active session with the destination node.
0x35	Encryption Failure	The encryption process failed. Only likely to be seen when using manual SRP and when an invalid encryption parameter was passed in.

Remote AT Commands- 0x17 frames

- Response frame type: Remote AT Command Response- 0x97

Possible error statuses:

Status	Description	Reason
0x0B	No Secure Session Connection	The sending node does not have an active session with the destination node.
0x0C	Encryption Error	There was an internal encryption error on the radio.
0x0D	TO Bit Not Set	The client has a session with the server but forgot to set the **TO** bit.

XBIB-C development boards

This section describes the XBIB-C development boards and how to interact with them.

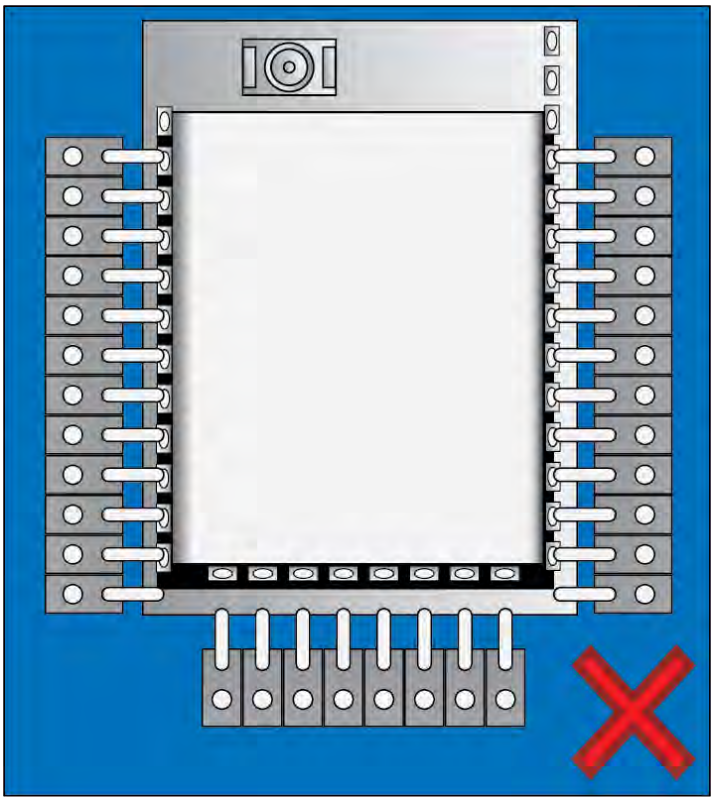
XBIB-C Micro Mount reference

This picture shows the XBee-C Micro Mount development board and the table that follows explains the callouts in the picture.

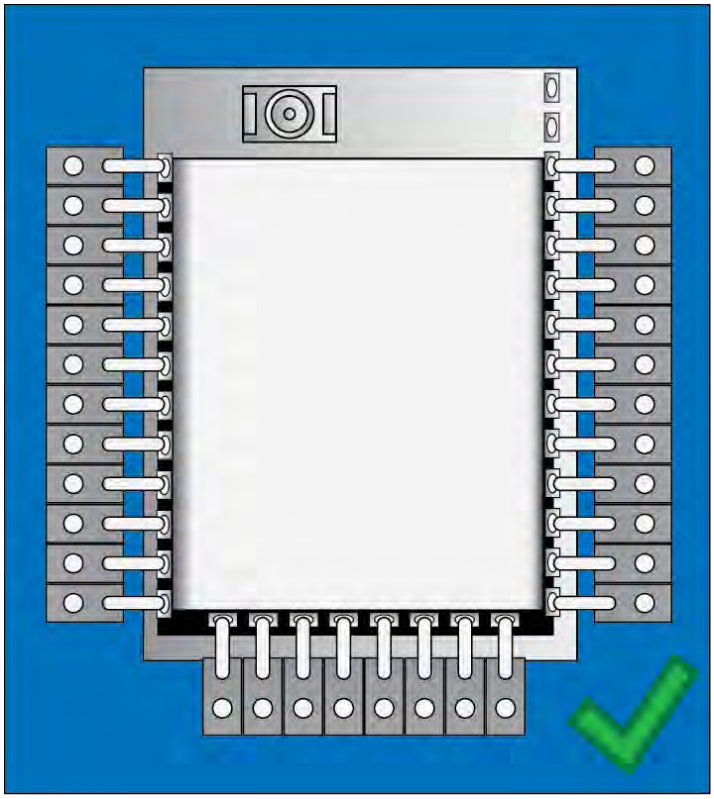
Note This board is sold separately.

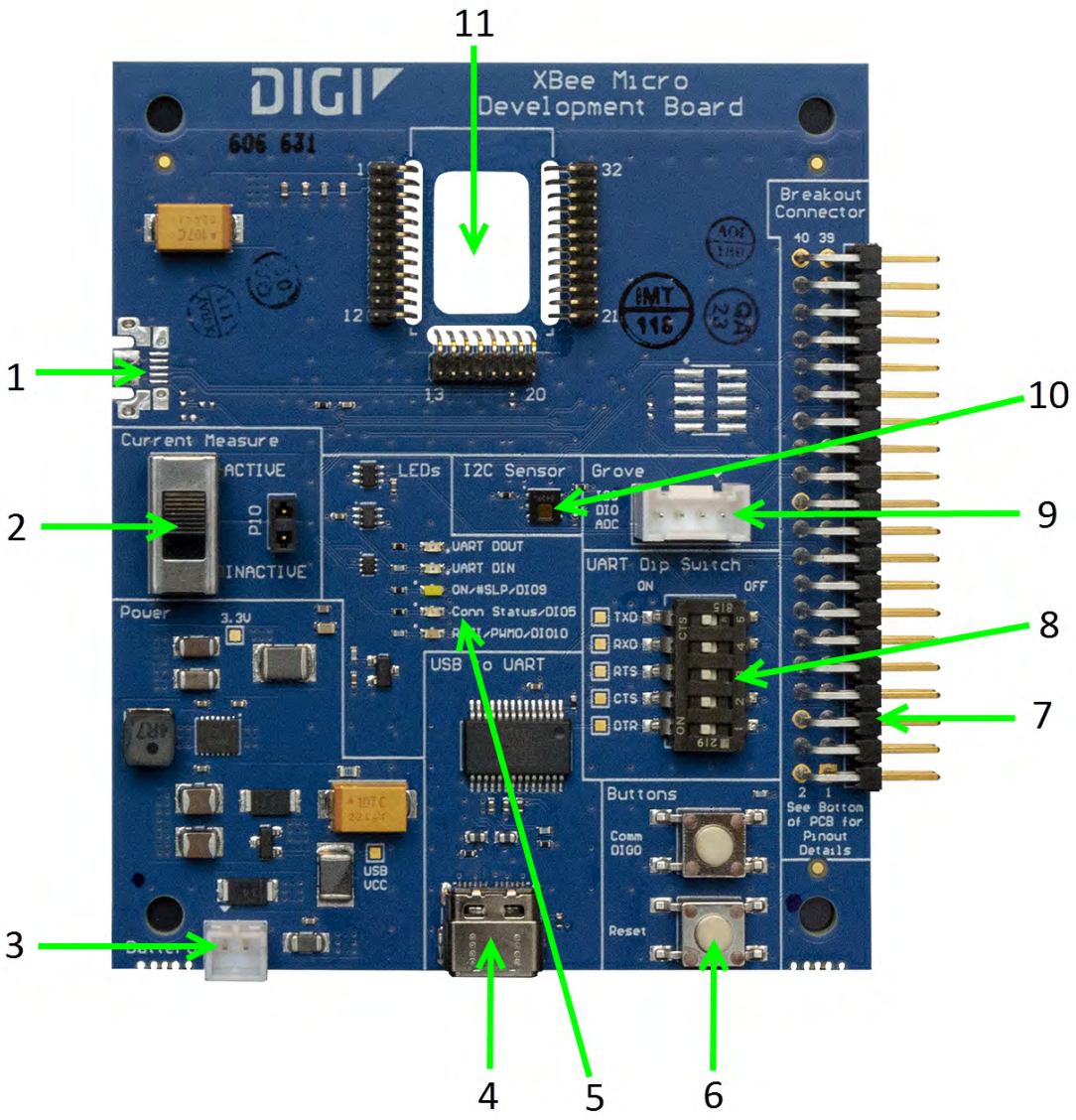
WARNING! Connecting one pin off will destroy the unit under test! Always disconnect USB and power before changing one unit for another. See the following images for examples.

Incorrect



Correct




11
1
2
3
4
5
6
7
8
9
10
XBee Micro
Development Board
Breakout
Connector
Current Measure
ACTIVE
INACTIVE
LEDs
I2C Sensor
Grove
UART Dip Switch
ON
OFF
TXD
RXD
RTS
CTS
DTR
Power
3.3V
USB
VCC
Buttons
Comm
DIO0
Reset
See Bottom
of PCB for
Pinout
Details

Number	Item	Description
1	Secondary USB (USB MICRO B)	Secondary USB Connector for possible future use. Not used.
2	Current Measure	Large switch controls whether current measure mode is active or inactive. When inactive, current can freely flow to the VCC pin of the XBee. When active, the VCC pin of the XBee is disconnected from the 3.3 V line on the development board. This allows current measurement to be conducted by attaching a current meter across the jumper P10.
3	Battery Connector	If desired, you can attach a battery to provide power to the development board. The voltage can range from 2 V to 5 V. The positive terminal is on the left.
4	USB-C Connector	Connects to your computer. This is connected to a USB to UART conversion chip that has the five UART lines passed to the XBee device. The UART Dip Switch can be used to disconnect these UART lines from the XBee.
5	LED indicator	Red: UART DOUT (modem sending serial/UART data to host) Green: UART DIN (modem receiving serial/UART data from host) White: ON/SLP/DIO9 Blue: Connection Status/DIO5 Yellow: RSSI/PWM0/DIO10
6	User Buttons	Comm DIO0 Button connects the Commissioning/DIO0 pin on the XBee Connector through to a 10 Ω resistor to GND when pressed. RESET Button Connects to the RESET pin on the XBee Connector to GND when pressed.
7	Breakout Connector	This 40-pin connector can be used to connect to various XBee pins as shown on the silkscreen on the bottom of the board.
8	UART Dip Switch	This dip switch allows the user to disconnect any of the primary UART lines on the XBee from the USB to UART conversion chip. This allows for testing on the primary UART lines without the USB to UART conversion chip interfering. Push Dip switches to the right to disconnect the USB to UART conversion chip from the XBee.
9	Grove Connector	This connector can be used to attach I2C enabled devices to the development board. Note that I2C needs to be available on the XBee in the board to use this functionality. Pin 1: I2C_CLK/XBee DIO1 Pin2: I2C_SDA/XBee DIO11 Pin3: VCC Pin4: GND
10	Temp/Humidity Sensor	This as a Texas Instruments HDC1080 temperature and humidity sensor. This part is accessible through I2C. Be sure that the XBee

Number	Item	Description
		that is inserted into the development board has I2C if access to this sensor is desired.
11	XBee Socket	This is the socket for the XBee (Micro form factor).

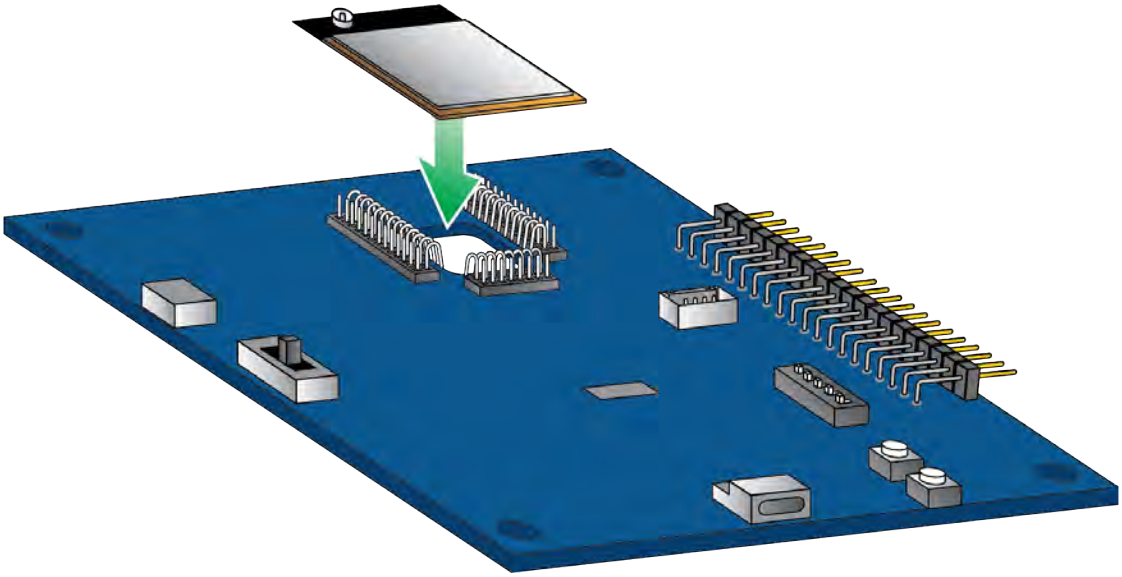
Attach the XBee XR 900 RF Module

It is important to attach the device to the board correctly.

WARNING! Make sure the board is NOT powered when you plug in the XBee XR 900 RF Module. Never insert or remove the XBee XR 900 RF Module while the power is on!

Make sure the XBee XR 900 RF Module is oriented correctly and not upside down when you attach it to the board.

Micro



Design notes

XBee XR 900 RF Modules do not require any external circuitry or specific connections for proper operation. However, there are some general design guidelines that we recommend to build and troubleshoot a robust design.

Power supply design

A poor power supply can lead to poor device performance, especially if you do not keep the supply voltage within tolerance or if it is excessively noisy. To help reduce noise, place a 1.0 µF and 47 pF capacitor as near as possible to the VCC connection on the XBee XR 900 RF Module (pad 2 for micro and surface-mount, and pin 1 for through-hole). Adding a 10 µF decoupling capacitor is also recommended. If you are using a switching regulator for the power supply, switch the frequencies above 500 kHz. Limit the power supply ripple to a maximum 50 mV peak to peak. For best results, place the lower capacitance capacitors closest to the XBee XR 900 RF Module.

Board layout

We design XBee XR 900 RF Modules to be self-sufficient and have minimal sensitivity to nearby processors, crystals or other printed circuit board (PCB) components. Keep power and ground traces thicker than signal traces and make sure that they are able to comfortably support the maximum current specifications. There are no other special PCB design considerations to integrate XBee XR 900 RF Modules, with the exception of antennas.

Antenna performance

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

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

- Metal poles
- Metal studs
- Structure beams
- Concrete, which is usually reinforced with metal rods
- Batteries
- Tall electrolytic capacitors

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

- Vehicles
- Elevators
- Ventilation ducts
- Refrigerators
- Microwave ovens

Use the following additional guidelines for optimal antenna performance:

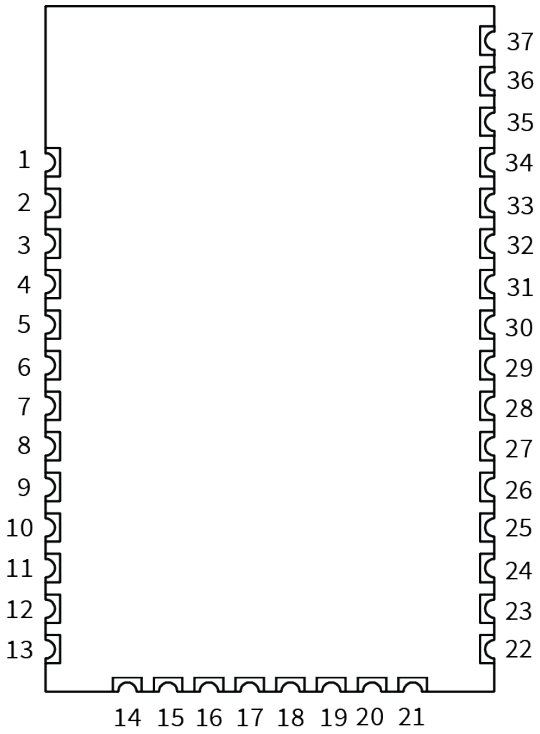
- Do not place XBee XR 900 RF Modules with the chip antenna or the embedded antenna inside a metal enclosure.
- Do not place any ground planes or metal objects above or below the antenna.

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

Pin signals

Pin signals for the surface-mount XBee XR 900 RF Module

The following drawing shows the surface-mount (SMT) pin locations.



The following table shows the pin signals and their descriptions for the surface-mount device.

Pin#	Name	Direction	Default state	Description
1	GND	-	-	Ground.
2	VCC	-	-	Power supply.
3	DOUT /DIO13	Both	Output	UART data out /GPIO.
4	DIN / $\overline{\text{CONFIG}}$ /DIO14	Both	Input	UART data in /GPIO.
5	DIO12	Both		GPIO.
6	$\overline{\text{RESET}}$	Input		Device reset.
7	RSSI PWM/DIO10	Both	Output	RX signal strength Indicator /GPIO.
8	PWM1/DIO11/I2C SDA	Both	Disabled	Pulse width modulator/GPIO/I2C SDA.
9	[reserved]	-	Disabled	Do not connect.
10	$\overline{\text{DTR}}$/SLEEP_RQ	Both	Input	Pin sleep control

Pin#	Name	Direction	Default state	Description
	/DIO8			Line/GPIO.
11	GND	-	-	Ground.
12	SPI_$\overline{\text{ATTN}}$/ $\overline{\text{BOOTMODE}}$/DIO19	Output	Output	Serial peripheral interface attention . Do not tie low on reset.
13	GND	-	-	Ground.
14	SPI_CLK /DIO18	Input	Input	Serial peripheral interface clock/GPIO.
15	SPI_$\overline{\text{SSEL}}$/DIO17	Input	Input	Serial peripheral interface not select/GPIO.
16	SPI_MOSI/DIO16	Input	Input	Serial peripheral interface data in/GPIO.
17	SPI_MISO/DIO15	Output	Output	Serial peripheral interface data out/GPIO.
18	[reserved]	-	Disabled	Do not connect.
19	[reserved]	-	Disabled	Do not connect.
20	[reserved]	-	Disabled	Do not connect.
21	[reserved]	-	Disabled	Do not connect.
22	GND	-	-	Ground.
23	[reserved]	-	Disabled	Do not connect.
24	DIO4	Both	Disabled	GPIO.
25	$\overline{\text{CTS}}$/DIO7	Both	Output	Clear to send flow control/GPIO.
26	ON/$\overline{\text{SLEEP}}$/DIO9	Both	Output	Device status indicator/GPIO
27	[reserved]	-	Disabled	Do not connect or connect to Ground.
28	ASSOCIATE/DIO5	Both	Output	Associate Indicator/GPIO.

Pin#	Name	Direction	Default state	Description
29	$\overline{\text{RTS}}$/DIO6	Both	Input	Request to send flow control /GPIO.
30	AD3/DIO3	Both	Disabled	Analog input/GPIO.
31	AD2/DIO2	Both	Disabled	Analog input/GPIO
32	AD1/DIO1/I2C SCL	Both	Disabled	Analog input/GPIO/I2C SCL.
33	AD0 /DIO0	Both	Input	Analog input / GPIO / Commissioning button.
34	[reserved]	-	Disabled	Do not connect.
35	GND	-v	-	Ground.
36	RF	Both	-	RF I/O for RF pad variant.
37	[reserved]	-	Disabled	Do not connect.

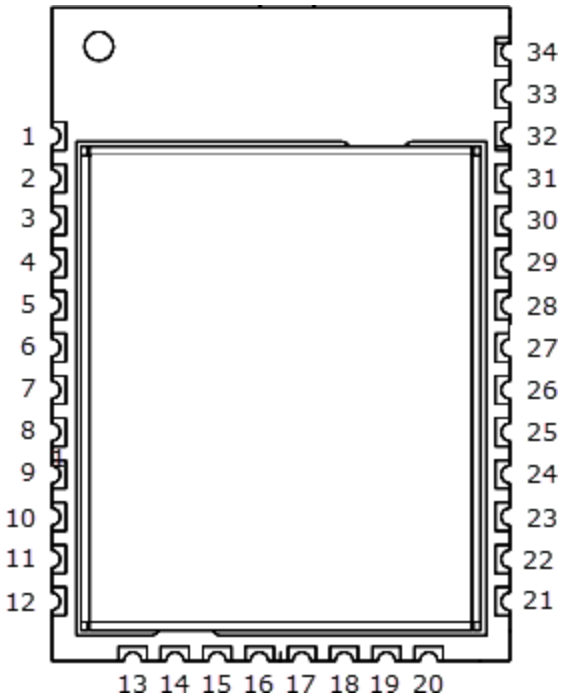
Signal direction is specified with respect to the device.
This is a complete list of functionalities. See the applicable software manual for available functionalities.

Note There are a possible three RF test points located on the bottom of the device. Do not connect these test points. For more information, see Recommended footprint.

See Design notes for details on pin connections.

Pin signals for the micro-mount XBee XR 900 RF Module

The following drawing shows the micro pin locations.



The following table shows the pin signals and their descriptions for the XBee XR 900 RF Module device.

Pin#	Name	Direction	Default state	Description
1	GND	-	-	Ground.
2	VCC	-	-	Power supply.
3	DOUT /DIO13	Both	Output	UART data out /GPIO.
4	DIN / $\overline{\text{CONFIG}}$ /DIO14	Both	Input	UART data in /GPIO.
5	DIO12	Both		GPIO.
6	$\overline{\text{RESET}}$	Input		Device reset.
7	RSSI PWM/DIO10	Both	Output	RX signal strength Indicator /GPIO.
8	PWM1/DIO11/I2C SDA	Both	Disabled	Pulse width modulator/GPIO/I2C SDA.
9	$\overline{\text{DTR}}$/SLEEP_RQ /DIO8	Both	Input	Pin sleep control Line/GPIO.

Pin#	Name	Direction	Default state	Description
10	GND	-	-	Ground.
11	SPI_$\overline{\text{ATTN}}$/ $\overline{\text{BOOTMODE}}$/DIO19	Output	Output	Serial peripheral interface attention Do not tie low on reset.
12	GND	-	-	Ground.
13	SPI_CLK /DIO18	Input	Input	Serial peripheral interface clock/GPIO.
14	SPI_$\overline{\text{SSEL}}$/DIO17	Input	Input	Serial peripheral interface not select/GPIO.
15	SPI_MOSI/DIO16	Input	Input	Serial peripheral interface data in/GPIO.
16	SPI_MISO/DIO15	Output	Output	Serial peripheral interface data out/GPIO.
17	[reserved]	-	Disabled	Do not connect.
18	[reserved]	-	Disabled	Do not connect.
19	[reserved]	-	Disabled	Do not connect.
20	[reserved]	-	Disabled	Do not connect.
21	GND	-	-	Ground.
22	[reserved]	-	Disabled	Do not connect.
23	DIO4	Both	Disabled	GPIO.
24	$\overline{\text{CTS}}$/DIO7	Both	Output	Clear to send flow control/GPIO.
25	ON/$\overline{\text{SLEEP}}$/DIO9	Both	Output	Device status indicator/GPIO.
26	ASSOCIATE/DIO5	Both	Output	Associate Indicator/GPIO.
27	$\overline{\text{RTS}}$/DIO6	Both	Input	Request to send flow control /GPIO.
28	AD3/DIO3	Both	Disabled	Analog input/GPIO.

Pin#	Name	Direction	Default state	Description
29	AD2/DIO2	Both	Disabled	Analog input/GPIO.
30	AD1/DIO1/I2C SCL	Both	Disabled	Analog input/GPIO/I2C SCL.
31	AD0 /DIO0	Both	Input	Analog input / GPIO / Commissioning button.
32	GND	-	-	Ground.
33	RF	Both	-	RF I/O for RF pad variant.
34	GND	-	-	Ground.

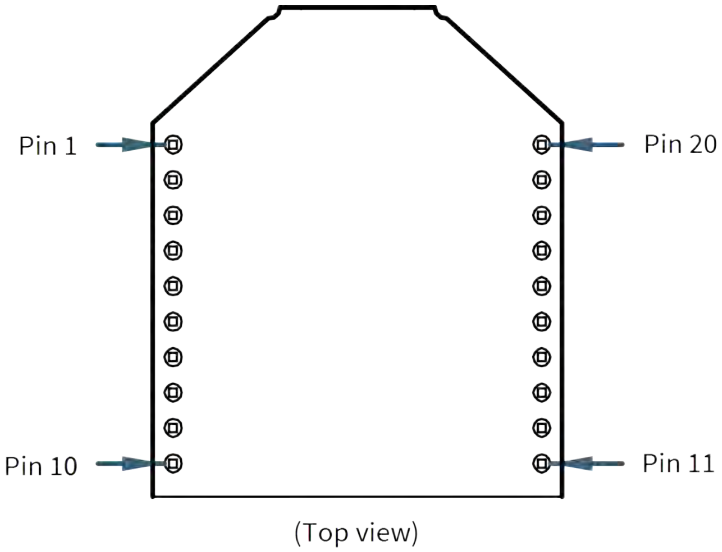
Signal direction is specified with respect to the device.
This is a complete list of functionalities. See the applicable software manual for available functionalities.

Note There are three RF test points located on the bottom of the device. Do not connect these test points. For more information, see Recommended footprint.

See Design notes for details on pin connections.

Pin signals for the through-hole XBee XR 900 RF Module

The following drawing shows the through-hole pin locations.



The following table shows the pin signals and their descriptions for the XBee XR 900 RF Module though-hole device.

Pin#	Name	Direction	Default state	Description
1	VCC	-	-	Power supply.
2	DOUT/DIO13	Both	Output	UART data out/GPIO.
3	DIN/$\overline{\text{CONFIG}}$/DIO14	Both	Intput	UART data in/GPIO.
4	DIO12/SPI_MISO	Both	-	GPIO/SPI data out.
5	$\overline{\text{RESET}}$	Input	-	Device reset.
6	RSSI PWM/DIO10	Both	Output	RX signal Indicator strength/GPIO.
7	PWM1/DIO11/I2C SDA	Both	Disabled	Pulse width modulator/GPIO/I2C SDA.
8	[reserved]	-	Disabled	Do not connect.
9	$\overline{\text{DTR}}$/SLEEP_RQ/DIO8	Both	Input	Pin sleep control Line/GPIO.
10	GND	-	-	Ground.
11	DIO4/SPI_MOSI	Both	Disabled	GPIO/Serial peripheral interface data in.

Pin#	Name	Direction	Default state	Description
12	CTS/DIO7	Both	Output	Clear to send flow control/GPIO.
13	ON/SLEEP/DIO9	Both	Output	Device status indicator/GPIO.
14	[reserved]	-	Disabled	Do not connect or connect to Ground.
15	ASSOCIATE/DIO5	Both	Output	Associate Indicator/GPIO.
16	RTS/DIO6	Both	Input	Request to send flow control/GPIO.
17	AD3/DIO3/SPI_SSEL	Both	Disabled	Analog input/GPIO/SPI not select.
18	AD2/DIO2/SPI_CLK	Both	Disabled	Analog input/GPIO/SPI clock.
19	AD1/DIO1/SPI_ATTN/I2C SCL	Both	Disabled	Analog input/GPIO/SPI attention/I2C SCL.
20	AD0 /DIO0	Both	Input	Analog input/GPIO/ Commissioning button.
Signal direction is specified with respect to the device. This is a complete list of functionalities. See the applicable software manual for available functionalities. See Design notes for details on pin connections.				
* The I2C functionality will be software enabled in a future release.				

Recommended pin connections

The only required pin connections for two-way communication are VCC, GND, DOUT and DIN. To support serial firmware updates and recovery, you must connect VCC, GND, DOUT, DIN, RTS, and DTR.

For applications that need to ensure the lowest sleep current, never leave unconnected inputs floating. Use internal or external pull-up or pull-down resistors, or set the unused I/O lines to outputs.

Connect the UART for flow control

The RS-232 and UART specification has been around for a long time. Despite its longevity, confusion remains about how to connect pinouts. There are two different types of communication devices:

- Data Communications Equipment (DCE)
- Data Terminal Equipment (DTE)

The majority of products such as microprocessors and computers are DTE, and they function as the local master of a system. Modems or other communication devices are DCE. In a schematic, each line receives a single name and is usually named from the DTE perspective with the name of the DCE/modem device.

SPI naming was a significant improvement with Master Input and Slave Output (MISO) and Master Output and Slave Input (MOSI). These names help remind us that the device must be hooked up differently depending on if it is a master or slave. One must be the master and the other the slave. If applying the same thinking of the SPI to UART we would get Computer or Microprocessor DTE = Master and DCE or Modem = Slave device. If we were to take SPI naming and UART naming and mix the two, we would get this result:

- TXD = MTSR where the master transmits and the slave receives like MOSI.
- RXD = MRST where the master receives and the slave transmits like MISO.
- RTS = MRSC where the master indicates it is ready to receive data and the slave receives that it is clear to send.
- CTS = MCSR where the master receives it is clear to send and slave indicates it is ready to receive data.

This is not a real naming convention. It is just a fun way to clear up some of the confusion. The problem with the abbreviations above is that it is still confusing and hard to remember which is and input and which is an output. **R** stands for receive in the top two lines, an input. **R** in MRSC is for RTS, an output, in the bottom two for RTS and CTS. We could switch the R and C so that R would always be an input and T/C would always be an output, but that would mix DTE and DCE.

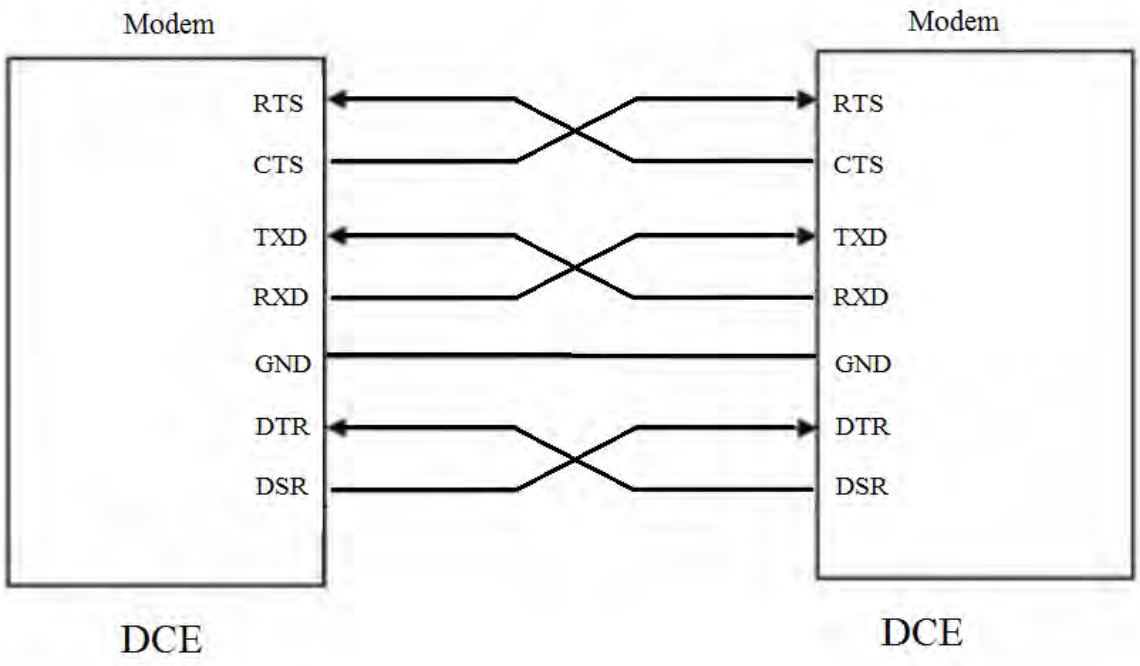
Never mix DTE and DCE conventions on a single component. Either choose:

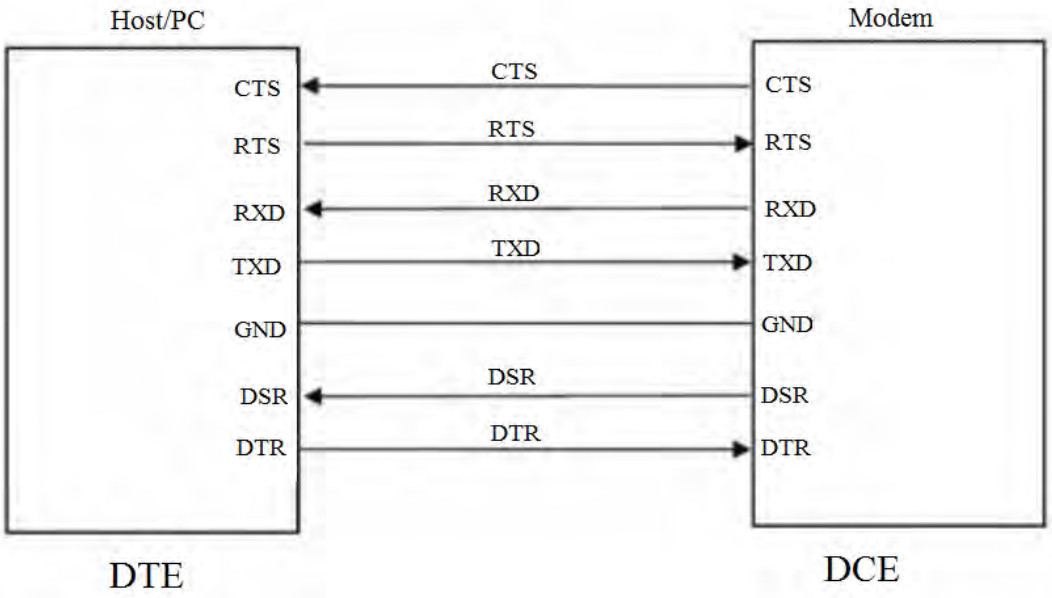
- [TXD, RTS, DTR are outputs and RXD, CTS, DSR are inputs (DTE)] or
- [TXD, RTS, DCD are inputs and RXD, CTS, DSR are outputs (DCE)]

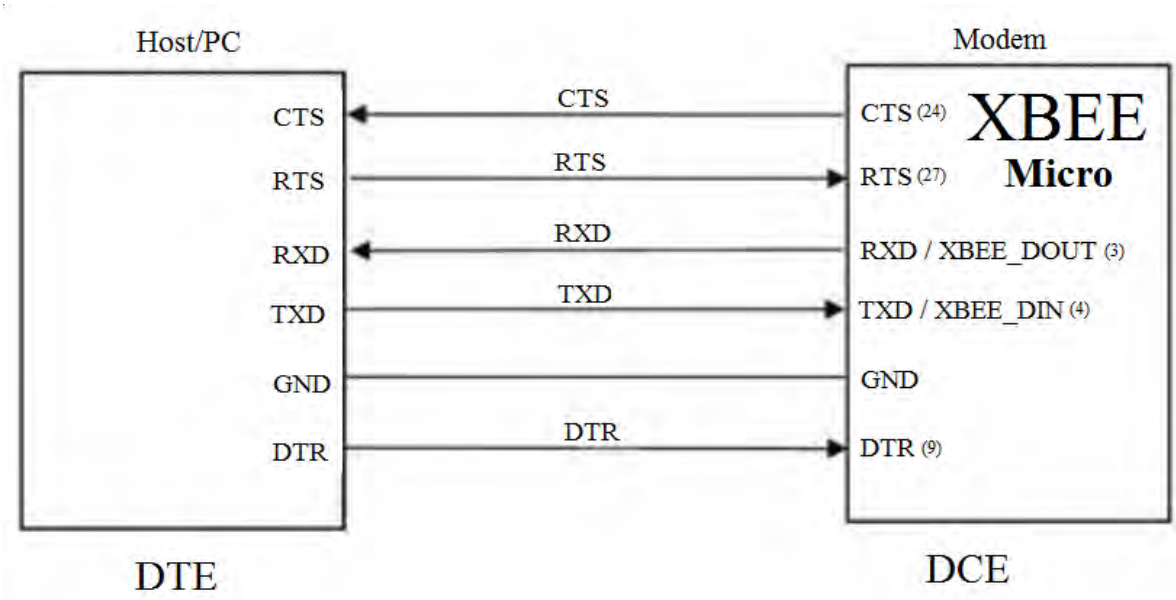
Not knowing if the format is DCE or DTE allows a 50% chance of successfully connecting the wires for the first try. Mixing the signals changes it to a 100% guarantee if the directions are assumed to be one or the other.

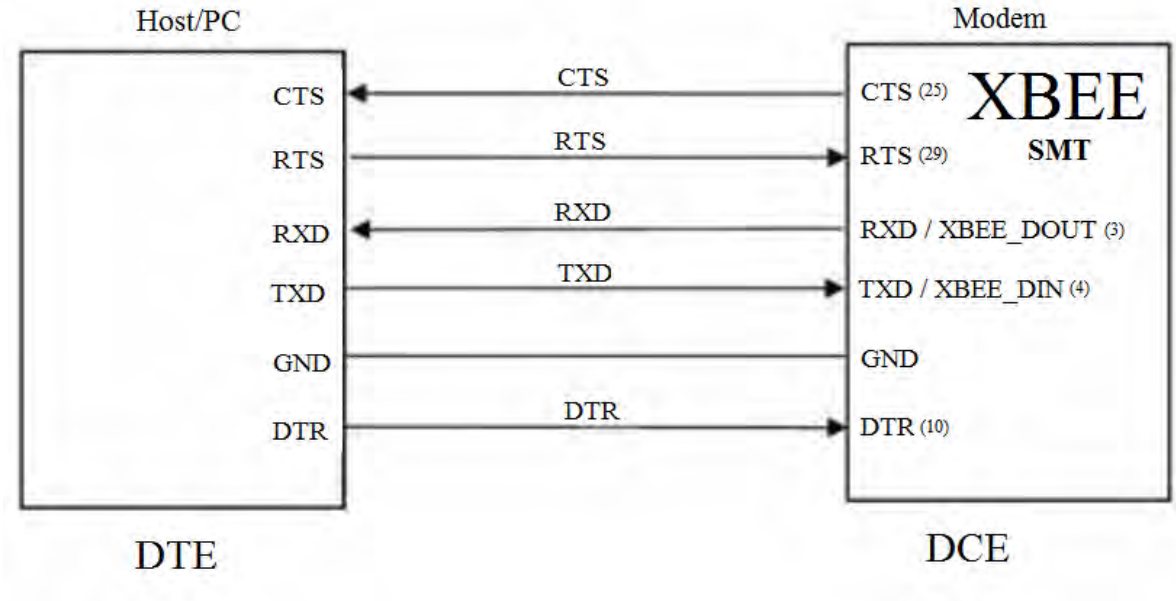
XBee UART lines are in DCE format, where CTS is an output and RTS, DTR are inputs. Signals DIN and DOUT are not the normal name of TXD or RXD. Knowing the direction of the data lines increases the chance of correct hookup on these critical lines to near 100%. However, there has still been ample confusion on RTS and CTS. Below are some diagrams on how to connect RS-232/UARTs in general as well as with Digi XBee products.

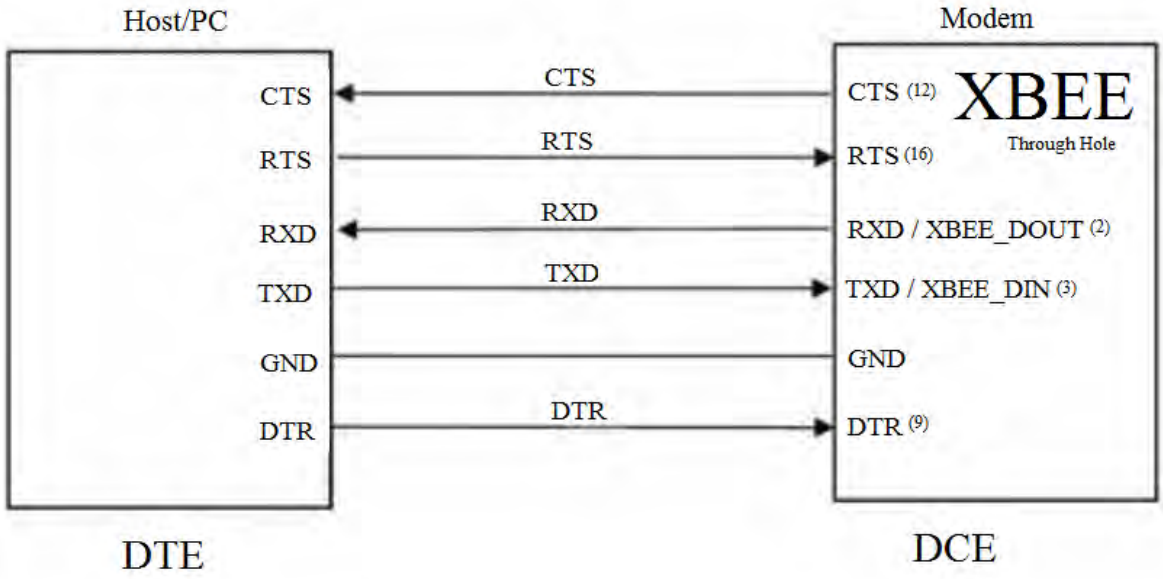
There are other signal lines, like DTR, DSR and DCD which have loose definitions of the original use, but these often are used as other signals and are not uniform across the market. As such, we will mention them but not cover them in detail.

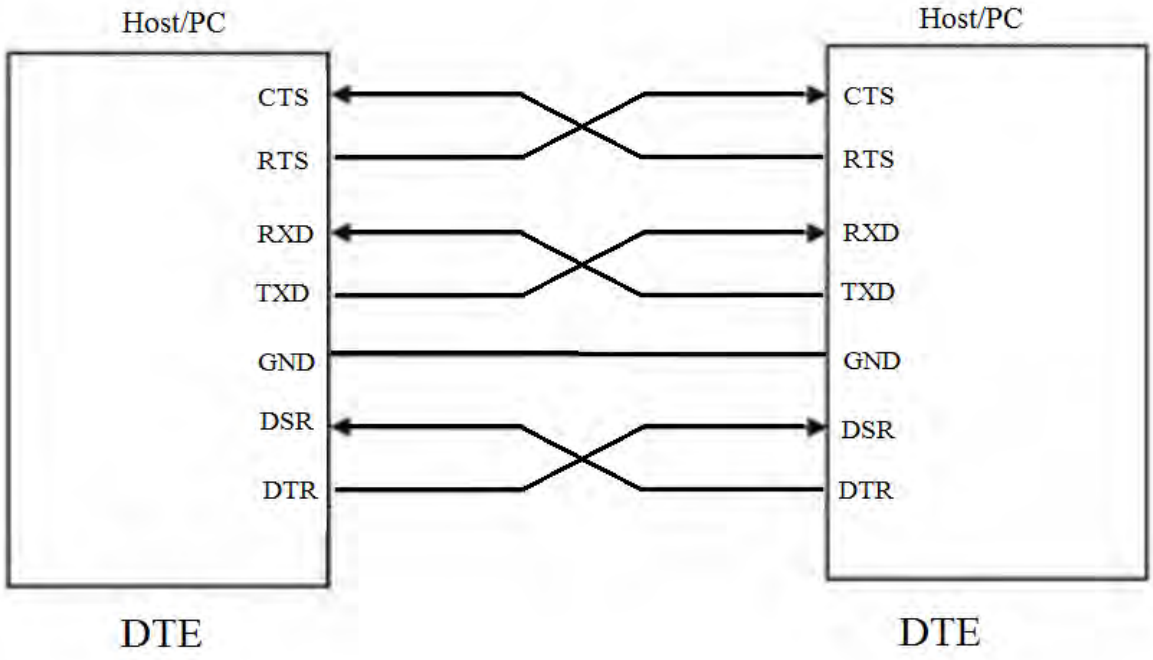

Modem
RTS
CTS
TXD
RXD
GND
DTR
DSR
DCE
Modem
RTS
CTS
TXD
RXD
GND
DTR
DSR
DCE


Host/PC
CTS
RTS
RXD
TXD
GND
DSR
DTR
DTE
CTS
RTS
RXD
TXD
GND
DSR
DTR
Modem
CTS
RTS
RXD
TXD
GND
DSR
DTR
DCE


Host/PC
Modem
CTS
RTS
RXD
TXD
GND
DTR
CTS
RTS
RXD
TXD
DTR
CTS (24)
RTS (27)
RXD / XBEE_DOUT (3)
TXD / XBEE_DIN (4)
GND
DTR (9)
XBEE
Micro
DTE
DCE


Host/PC
Modem
CTS
RTS
RXD
TXD
GND
DTR
CTS
RTS
RXD
TXD
DTR
CTS (25)
RTS (29)
RXD / XBEE_DOUT (3)
TXD / XBEE_DIN (4)
GND
DTR (10)
XBEE
SMT
DTE
DCE


Host/PC
Modem
CTS
RTS
RXD
TXD
GND
DTR
CTS
RTS
RXD
TXD
DTR
CTS (12)
RTS (16)
RXD / XBEE_DOUT (2)
TXD / XBEE_DIN (3)
GND
DTR (9)
XBEE
Through Hole
DTE
DCE


Host/PC
Host/PC
CTS
RTS
RXD
TXD
GND
DSR
DTR
CTS
RTS
RXD
TXD
GND
DSR
DTR
DTE
DTE

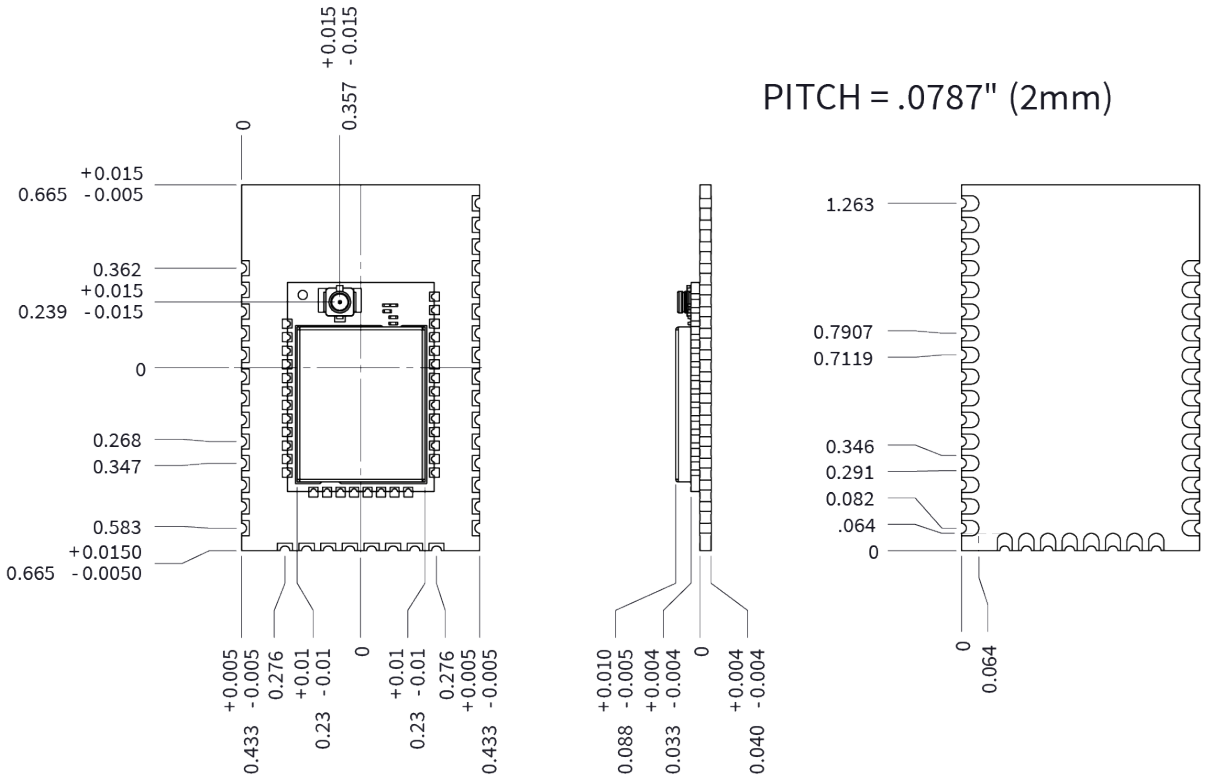
Mechanical drawings

The following mechanical drawings of the XBee XR 900 RF Modules show all dimensions in inches.

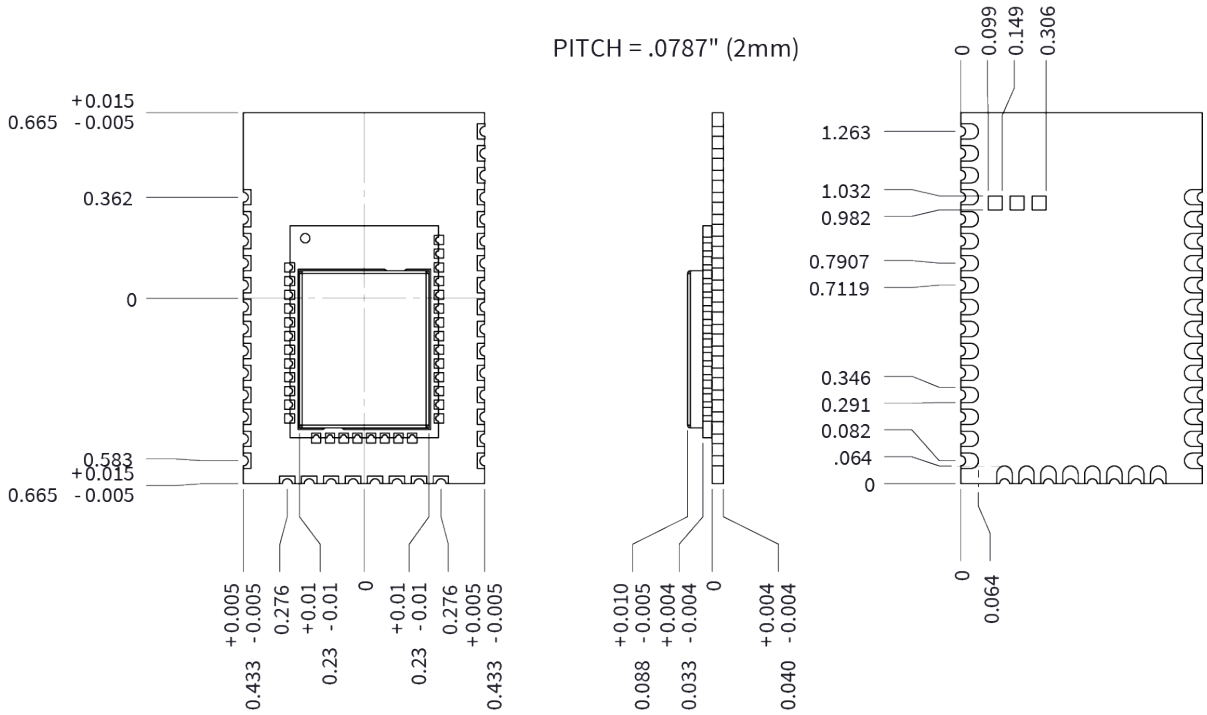
XBee XR 900 RF Module surface-mount antennas

The following mechanical drawings are for the XBee XR 900 RF Module surface-mount antennas.

XBee XR 900 RF Module surface-mount - U.FL/RF pad antenna



XBee XR 900 RF Module surface-mount - embedded antenna

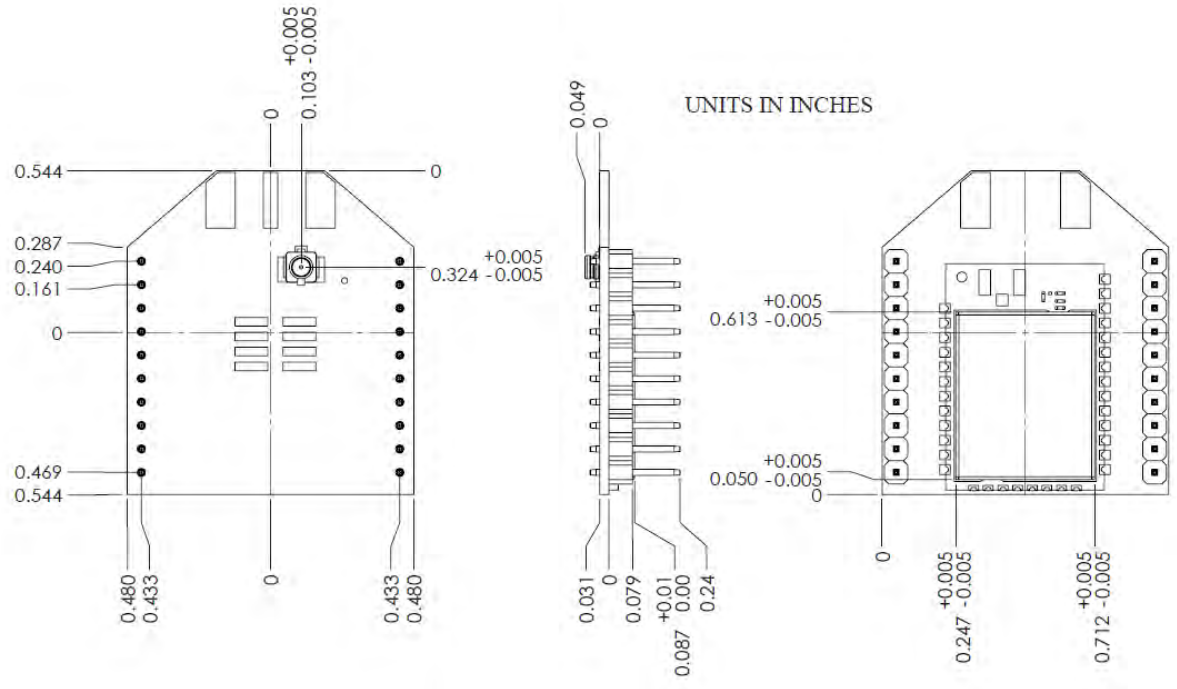


XBee XR 900 RF Module through-hole antennas

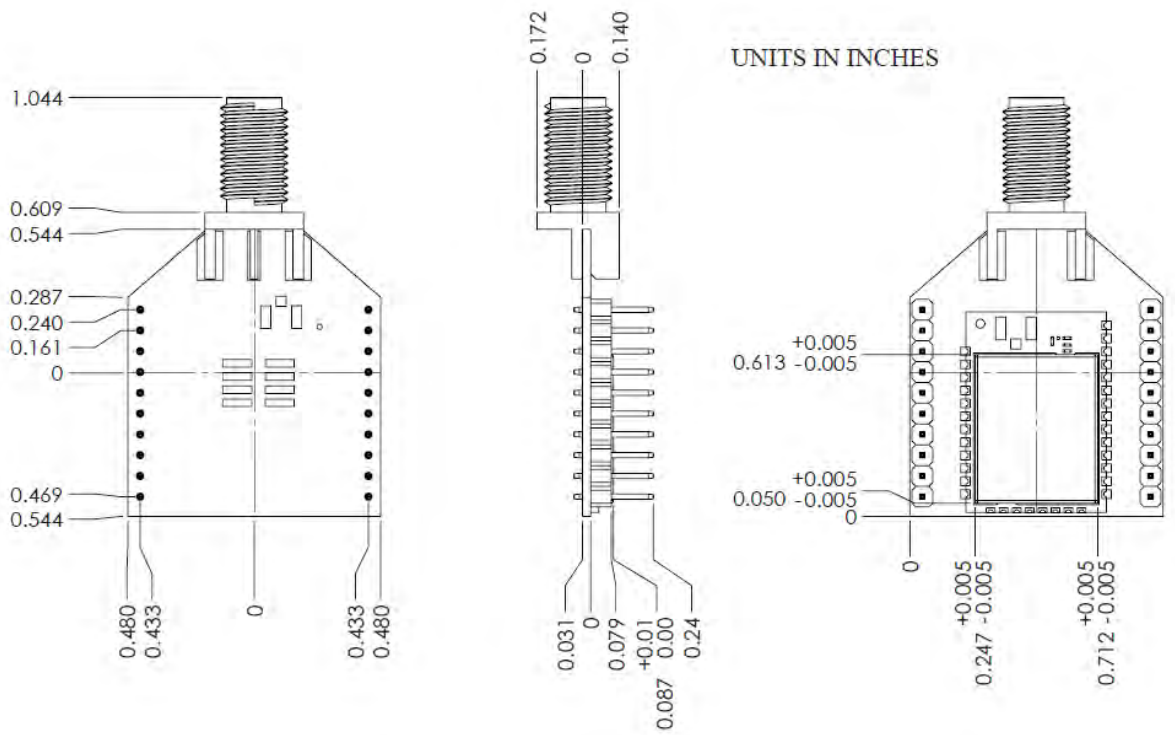
The following mechanical drawings are for the XBee XR 900 RF Module through-hole antennas.

XBee XR 900 RF Module through-hole - PCB antenna

XBee XR 900 RF Module through-hole - U.FL antenna



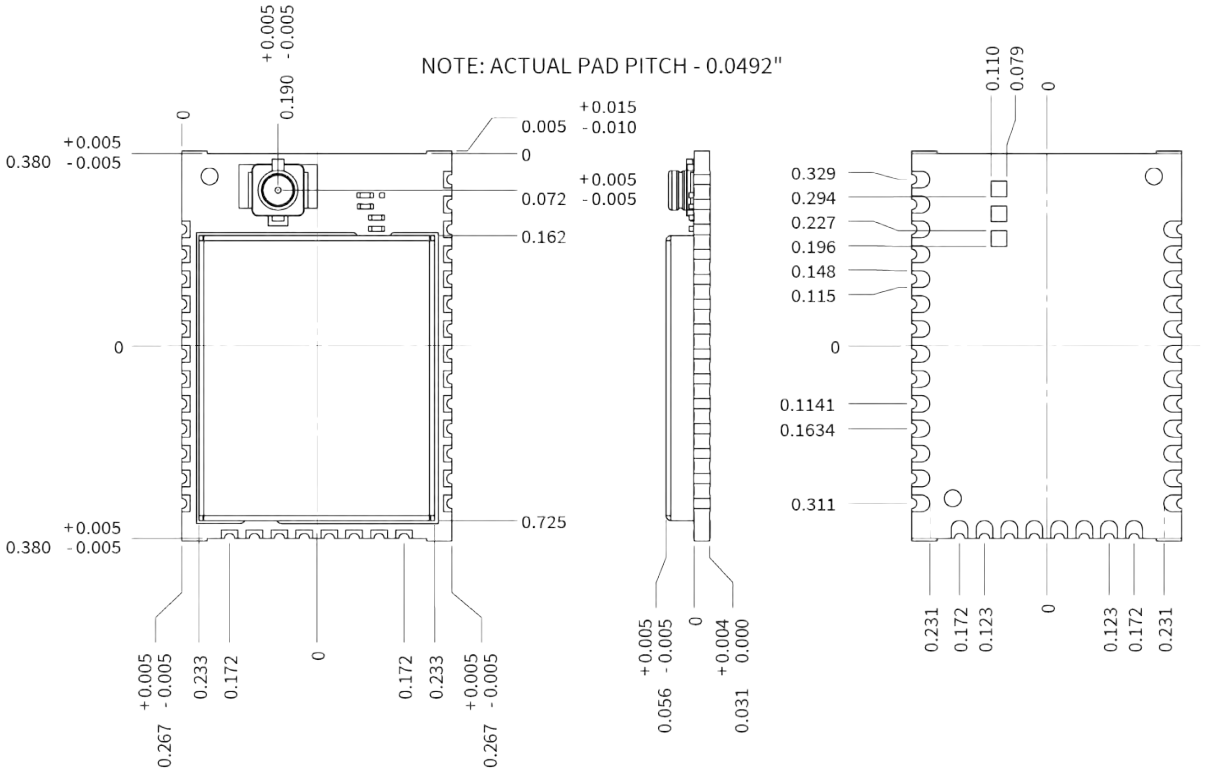
XBee XR 900 RF Module through-hole - RPSMA antenna



XBee XR 900 RF Module micro antennas

The following mechanical drawing is for theXBee XR 900 RF Module micro antennas.

U.FL/RF Pad

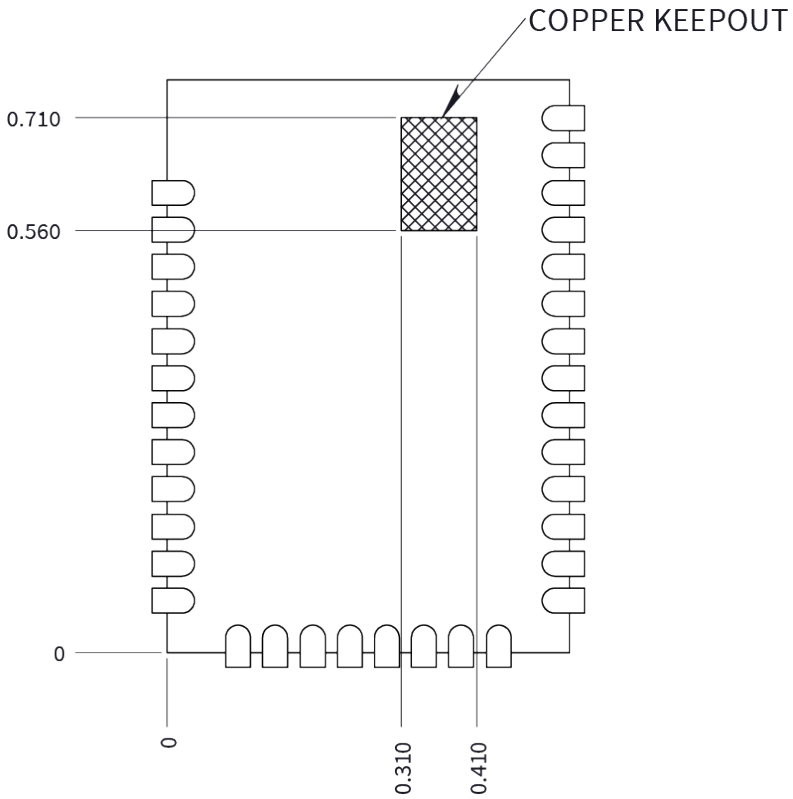


Copper keepout for test points

The following keepouts are required for all surface-mount or micro-mount devices. These keepouts are in addition to the other keepouts if using a PCB or chip antenna.

While the underside of the device is mostly coated with solder resist, we recommended the copper layer directly below the device be left open to avoid unintended contacts. Copper or vias must not interfere with the three exposed RF test points on the bottom of the device as shown in the following diagrams. These devices have a ground plane in the middle on the back side for shielding purposes, which can be affected by copper traces directly below the device.

Copper keepout for the XBee XR



Modes

Transparent operating mode

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

API operating mode

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

The advantages of API operating mode include:

Command mode

Command mode is a state in which the firmware interprets incoming characters as commands. It allows you to modify the device's configuration using parameters you can set using AT commands. When you want to read or set any parameter of the XBee XR 900 RF Module using this mode, you have to send an AT command. Every AT command starts with the letters **AT** followed by the two characters that identify the command and then by some optional configuration values.

The operating modes of the XBee XR 900 RF Module are controlled by the AP (API Enable) setting, but Command mode is always available as a mode the device can enter while configured for any of the operating modes.

Command mode is available on the UART interface for all operating modes.

You cannot use the SPI interface to enter Command mode.

Enter Command mode

When using the default configuration values for GT (Guard Times) and CT (Command Mode Timeout), you must enter **+++** preceded and followed by one second of silence—no input—to enter Command mode. However, both **GT** and **CC** are configurable. This means that the silence before and after the escape sequence—**GT**—and the escape characters themselves—**CC**—can be changed. For example, if **GT** is **5DC** and **CC** is **31**, then Command mode can be entered by typing **111** preceded and followed by 1.5 seconds of silence. When the entrance criteria are met the device responds with **OK\r** on UART signifying that it has entered Command mode successfully and is ready to start processing AT commands.

If configured to operate in Transparent operating mode, when entering Command mode the XBee XR 900 RF Module knows to stop sending data and start accepting commands locally.

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

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

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

Troubleshooting

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

There are two alternative ways to enter Command mode:

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

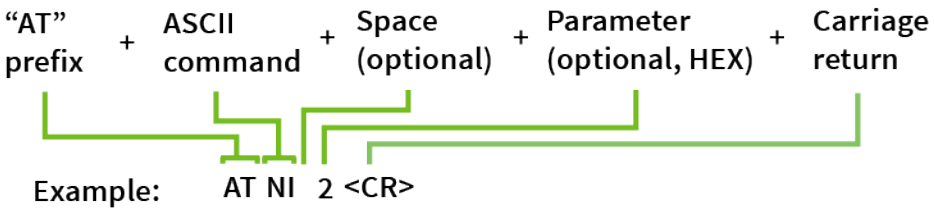
Note You must assert $\overline{\text{RTS}}$ for both of these methods, otherwise the device enters the bootloader.

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

Send AT commands

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

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



The preceding example changes NI (Node Identifier) to **2**.

Multiple AT commands

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

Note The behavior of the comma is the same as the behavior of the <CR> in the previous example except that the next command following the comma is not preceded by AT. The only real purpose of the comma is to reduce keystrokes.

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

Parameter format

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

Response to AT commands

When using AT commands to set parameters the XBee XR 900 RF Module responds with **OK<cr>** if successful and **ERROR<cr>** if not.

Apply command changes

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

1. Send AC (Apply Changes).
2. Send WR (Write). In this case, changes are only applied following a reset. The **WR** command by itself does not apply changes.
   or:
3. Exit Command mode. You can exit Command mode in two ways: Either enter the **CN** command or wait for Command mode to timeout as specified by the **CT** parameter.

Make command changes permanent

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

Send an RE (Restore Defaults) followed by **WR** to restore parameters back to their factory defaults. The next time the device is reset the default settings are applied.

Exit Command mode

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

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

Transmit mode

Transmit mode is the mode in which the device is transmitting data. This typically happens after data is received from the serial port.

Receive mode

This is the default mode for the XBee XR 900 RF Module. The device is in Receive mode when it is neither transmitting data nor sleeping. If a destination node receives a valid RF data packet, the destination node transfers the data to its serial transmit buffer.

I/O support

The following topics describe analog and digital I/O line support, line passing and output control.

Digital I/O support

Digital I/O is available on lines DIO0 through DIO9 (D0 (DIO0/AD0/Commissioning Button Configuration) - D9 (DIO9/ON_SLEEP Configuration)) and DIO12 through DIO19 (P2 (DIO12/TH_SPI_MISO Configuration) - P9 (DIO19/SPI_ATTN Configuration)).

Function	MMT Pin	SMT Pin	TH Pin	AT Command
DIO0	31	33	20	D0 (DIO0/ADC0/Commissioning Configuration)
DIO1	30	32	19	D1 (DIO1/ADC1/TH_SPI_ATTN Configuration)
DIO2	29	31	18	D2 (DIO2/ADC2/TH_SPI_CLK Configuration)
DIO3	28	30	17	D3 (DIO3/ADC3/TH_SPI_SSEL Configuration)
DIO4	23	24	11	D4 (DIO4/TH_SPI_MOSI Configuration)
DIO5	26	28	15	D5 (DIO5/Associate Configuration)
DIO6	27	29	16	D6 (DIO6/RTS Configuration)
DIO7	24	25	12	D7 (DIO7/CTS Configuration)
DIO8	9	10	9	D8 (DIO8/DTR/SLP_Request Configuration)
DIO9	25	26	13	D9 (DIO9/ON_SLEEP Configuration)
DIO12	5	5	4	P2 (DIO12/TH_SPI_MISO Configuration)
DIO13	3	3	2	P3 (DIO13/UART_DOUT Configuration)
DIO14	4	4	3	P4 (DIO14/UART_DIN Configuration)
DIO15	16	17	-	P5 (DIO15/SPI_MISO Configuration)
DIO16	15	16	-	P6 (DIO16/SPI_MOSI Configuration)
DIO17	14	15	-	P7 (DIO17/SPI_SSEL Configuration)
DIO18	13	14	-	P8 (DIO18/SPI_CLK Configuration)
DIO19	11	12	-	P9 (DIO19/SPI_ATTN Configuration)

Digital sampling is enabled on these pins if configured as 3, 4, or 5 with the following meanings:

- 3 is digital input.
  - Use PR (Pull-up/Down Resistor Enable) to enable internal pull up/down resistors for each digital input. Use PD (Pull Up/Down Direction) to determine the direction of the internal pull up/down resistor. All disabled and digital input pins are pulled up by default.
- 4 is digital output low.
- 5 is digital output high.

Analog I/O support

Analog input is available on **D0** through **D3**. Configure these pins to **2** (ADC) to enable analog sampling.

Function	MMT Pin	SMT Pin	TH Pin	AT Command
ADC0	31	33	20	D0 (DIO0/ADC0/Commissioning Configuration)
ADC1	30	32	19	D1 (DIO1/ADC1/TH_SPI_ATTN Configuration)
ADC2	29	31	18	D2 (DIO2/ADC2/TH_SPI_CLK Configuration)
ADC3	28	30	17	D3 (DIO3/ADC3/TH_SPI_SSEL Configuration)

AV (Analog Voltage Reference) specifies the analog reference voltage used for the 10-bit ADCs. Analog sample data is represented as a 2-byte value. For a 10-bit ADC, the acceptable range is from **0x0000** to **0x03FF**. To convert this value to a useful voltage level, apply the following formula:

ADC / 1023 (vREF) = Voltage

Example

An ADC value received is 0x01AE; to convert this into a voltage the hexadecimal value is first converted to decimal (0x01AE = 430). Using the default **AV** reference of 1.25 V, apply the formula as follows:

430 / 1023 (1.25 V) = 525 mV

Monitor I/O lines

You can monitor pins you configure as digital input, digital output, or analog input and generate I/O sample data. If you do not define inputs or outputs, no sample data is generated.

Typically, I/O samples are generated by configuring the device to sample I/O pins periodically (based on a timer) or when a change is detected on one or more digital pins. These samples are always sent over the air to the destination address specified by DH (Destination Address High) and DL (Destination Address Low).

You can also gather sample data using on-demand sampling, which allows you to collect the state of the device's I/O pins by issuing an AT command. You can do this on either a local or remote device via an AT command request.

The three methods to generate sample data are:

- Periodic sample (IR (Sample Rate))
  - Periodic sampling based on a timer
  - Samples are taken immediately upon wake (excluding pin sleep)
  - Sample data is sent to **DH**+**DL** destination address
  - Can be used with line passing
  - Requires API mode on receiver
- Change detect (IC (DIO Change Detect))
  - Samples are generated when the state of specified digital input pin(s) change
  - Sample data is sent to **DH**+**DL** destination address
  - Can be used with line passing
  - Requires API mode on receiver

- On-demand sample (IS (Immediate Sample))
  - Immediately query the device's I/O lines
  - Can be issued locally in Command Mode
  - Can be issued locally or remotely in API mode

These methods are not mutually exclusive and you can use them in combination with each other.

I/O sample data format

Regardless of how I/O data is generated, the format of the sample data is always represented as a series of bytes in the following format:

Bytes	Name	Description
1	Sample sets	Number of sample sets. There is always one sample set per frame.
1	Analog channel mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. If a bit is set, then a corresponding 2-byte analog data set is included. bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3
2	Digital data set	Each bit in the digital data set corresponds to a bit in the digital channel mask and indicates the digital state of the pin, whether high (1) or low (0). If the digital channel mask is 0x0000, then these two bytes are omitted as no digital I/O lines are enabled.
2	Analog data set (multiple)	Each enabled ADC line in the analog channel mask will have a separate 2-byte value based on the number of ADC inputs on the originating device. The data starts with AD0 and continues sequentially for each enabled analog input channel up to AD3. If the analog channel mask is 0x00, then no analog sample bytes is included.

API frame support

I/O samples generated using Periodic I/O sampling (**IR**) and Digital I/O change detection (**IC**) are transmitted to the destination address specified by **DH** and **DL**. In order to display the sample data, the receiver must operate in API mode (**AP** = **1** or **2**). The sample data is represented as an I/O sample API frame.

See I/O Sample Indicator - 0x92 for more information on the frame's format and an example.

On-demand sampling

You can use IS (Immediate Sample) to query the current state of all digital I/O and ADC lines on the device and return the sample data as an AT command response. If no inputs or outputs are defined, the command returns an ERROR.

On-demand sampling can be useful when performing initial deployment, as you can send **IS** locally to verify that the device and connected sensors are correctly configured. The format of the sample

data matches what is periodically sent using other sampling methods. You can also send **IS** remotely using a remote AT command. When sent remotely from a gateway or server to each sensor node on the network, on-demand sampling can improve battery life and network performance as the remote node transmits sample data only when requested.

If you send **IS** using Command mode, then the device returns a carriage return delimited list containing the I/O sample data. If **IS** is sent either locally or remotely via an API frame, the I/O sample data is presented as the parameter value in the AT command response frame (Local AT Command Response - 0x88 or Remote AT Command Response- 0x97).

Example: Command mode

An **IS** command sent in Command mode returns the following sample data:

Output	Description
01	One sample set
0C0C	Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 **11**00 0000 **11**00b = DIO2, 3, 10, 11)
03	Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 00**11**b = AD0, 1)
0408	Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 **01**00 0000 **1**000b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03D0	Analog sample data for AD0
0124	Analog sample data for AD1

Example: Local AT command in API mode

The **IS** command sent to a local device in API mode would use a Local AT Command Request - 0x08 or Queue Local AT Command Request - 0x09 frame:

7E 00 04 08 53 49 53 08

The device responds with a Local AT Command Response - 0x88 that contains the sample data:

Output	Field	Description
7E	Start Delimiter	Indicates the beginning of an API frame
00 0F	Length	Length of the packet
88	Frame type	AT Command response frame
53	Frame ID	This ID corresponds to the Frame ID of the 0x08 request
49 53	AT Command	Indicates the AT command that this response corresponds to 0x49 0x53 = **IS**
00	Status	Indicates success or failure of the AT command **00** = OK

Output	Field	Description
		if no I/O lines are enabled, this will return 01 (ERROR)
01	I/O sample data	One sample set
0C 0C		Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 **11**00 0000 **11**00b = DIO2, 3, 10, 11)
03		Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 00**11**b = AD0, 1)
04 08		Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 **01**00 0000 **10**00b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03 D0		Analog sample data for AD0
01 24		Analog sample data for AD1
68	Checksum	Can safely be discarded on received frames

Example: Remote AT command in API mode

The **IS** command sent to a remote device with an address of 0013A200 12345678 uses a Remote AT Command Request - 0x17:

7E 00 0F 17 87 00 13 A2 00 12 34 56 78 FF FE 00 49 53 FF

The sample data from the device is returned in a Remote AT Command Response- 0x97 frame with the sample data as the parameter value:

Output	Field	Description
7E	Start Delimiter	Indicates the beginning of an API frame
00 19	Length	Length of the packet
97	Frame type	Remote AT Command response frame
87	Frame ID	This ID corresponds to the Frame ID of the 0x17 request
0013A200 12345678	64-bit source	The 64-bit address of the node that responded to the request
0000	16-bit source	The 16-bit address of the node that responded to the request
49 53	AT Command	Indicates the AT command that this response corresponds to 0x49 0x53 = **IS**
00	Status	Indicates success or failure of the AT command **00** = **OK** if no I/O lines are enabled, this will return **01** (ERROR)

Output	Field	Description
01	I/O sample data	One sample set
0C 0C		Digital channel mask, indicates which digital lines are sampled (0x0C0C = 0000 1100 0000 1100b = DIO2, 3, 10, 11)
03		Analog channel mask, indicates which analog lines are sampled (0x03 = 0000 0011b = AD0, 1)
04 08		Digital sample data that corresponds with the digital channel mask 0x0408 = 0000 0100 0000 1000b = DIO3 and DIO10 are high, DIO2 and DIO11 are low
03 D0		Analog sample data for AD0
01 24		Analog sample data for AD1
50	Checksum	Can safely be discarded on received frames

Periodic I/O sampling

Periodic sampling allows a device to take an I/O sample and transmit it to a remote device at a periodic rate.

Source

Use IR (Sample Rate) to set the periodic sample rate for enabled I/O lines.

- To disable periodic sampling, set **IR** to **0**.
- For all other **IR** values, the device samples data when **IR** milliseconds elapse and transmits the sampled data to the destination address.

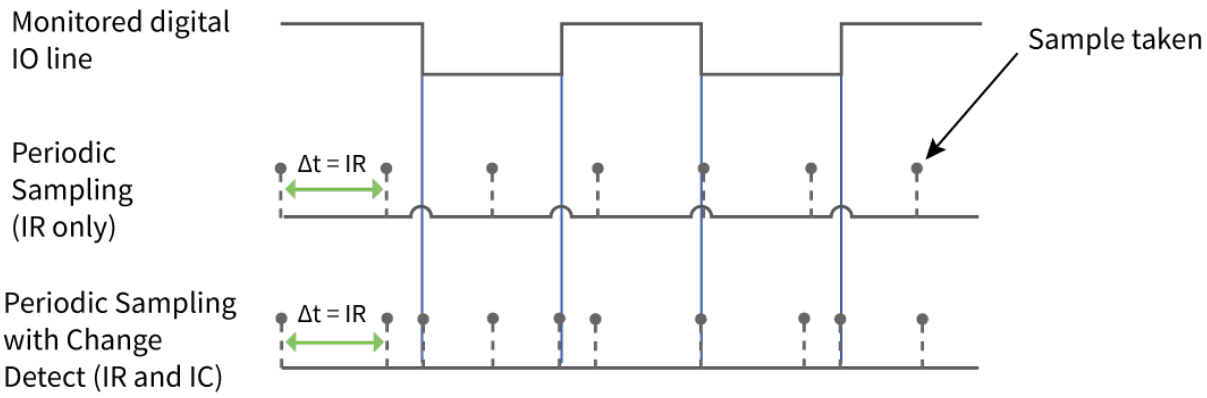
Destination

If the receiving device is operating in API operating mode the I/O sample data format is emitted out of the serial port. Devices that are in Transparent operating mode discard the I/O data samples they receive unless you enable line passing.

Digital I/O change detection

You can configure devices to transmit a data sample immediately whenever a monitored digital I/O pin changes state. IC (DIO Change Detect) is a bitmask that determines which digital I/O lines to monitor for a state change. If you set one or more bits in **IC**, the device transmits an I/O sample as soon as it observes a state change on the monitored digital I/O line(s) using edge detection.

The figure below shows how I/O change detection can work in combination with Periodic I/O sampling to improve sampling accuracy. In the figure, the gray dashed lines with a dot on top represent samples taken from the monitored DIO line. The top graph shows only periodic IR samples, the bottom graph shows a combination of **IR** periodic samples and **IC** detected changes. In the top graph, the humps indicate that the sample was not taken at that exact moment and needed to wait for the next **IR** sample period.



Note Use caution when combining change detect sampling with sleep modes. **IC** only causes a sample to be generated if a state change occurs during a wake period. If the device is sleeping when the digital transition occurs, then no change is detected and an I/O sample is not generated. Use periodic sampling with **IR** in conjunction with **IC** in this instance, since **IR** generates an I/O sample upon wakeup and ensures that the change is properly observed.

I/O line passing

Line passing allows you to affect the output pins of one device by sampling the I/O pins of another. To support line passing, you must configure a device to generate I/O sample data using periodic sampling (IR (Sample Rate)) and/or change detection (IC (DIO Change Detect)).

On the device that receives I/O samples, enable line passing setting IA (I/O Input Address) with the address of the device that has the appropriate inputs enabled. This effectively binds the outputs to a particular device's input. This does not affect the ability of the device to receive I/O line data from other devices–only its ability to update enabled outputs. Set **IA** to **0xFFFF** (broadcast address) to affect the output using input data from any device on the network.

Digital line passing

Each digital pin has an associated timeout value. When an I/O sample is received that affects a digital output pin, the pin returns to its configured state after the timeout period expires. For pins **D0** through **D9**, the associated timeout commands are T0 (D0 Timeout) through T9 (D9 Timeout). For pins **P0** through **P2**, the associated timeout commands are Q0 (P0 Timeout) through Q2 (P2 Timeout).

Digital line passing is only available on pins **D0** through **P2**. You cannot use UART and SPI pins for line passing.

Example: Digital line passing

A sampling XBee XR 900 RF Module is configured with the following settings (where DH/DL specifies the address of the node that outputs the the inputs from the sampling node):

AT command	Parameter value
D2 (DIO2/AD2/TH_SPI_CLK Configuration)	3 (digital input)

AT command	Parameter value
IR (Sample Rate)	0x7D0 (2 seconds)
DH (Destination Address High)	0013A200
DL (Destination Address Low)	12345678

Every two seconds, an I/O sample is generated and sent to the address specified by **DH** and **DL**. The receiver is configured with the following settings:

AT command	Parameter value
D2 (DIO2/AD2/TH_SPI_CLK Configuration)	4 (digital output low)
T2 (D2 Output Timeout)	0x64 (10 seconds)
IA (I/O Input Address)	00103A20012345678

When this device receives an incoming I/O sample, if the source address matches the one set by **IA**, the device sets the output of **D2** to match the input of **D2** of the receiver. This output level holds for ten seconds before the pin returns to a digital output low state.

Output sample data

If a device receives an I/O sample whose address matches that set by IA (I/O Input Address), it triggers line passing. Line passing operates whether the receiving device is operating in API or Transparent mode.

By default, if the receiver is configured for API mode, it outputs the I/O sample frame in addition to affecting output pins. You can suppress the I/O sample frame output by setting IU (I/O Output Enable) to **0**. This only suppresses I/O samples that trigger line passing, a sample generated from a device whose address does not match the **IA** address is sent regardless of **IU**.

I/O behavior during sleep

When the device sleeps (**SM !** = **0**) the I/O lines are optimized for a minimal sleep current.

Digital I/O lines

Digital I/O lines set as digital output high or low maintain those values during sleep. Disabled or input pins continue to be controlled by the **PR**/**PD** settings. Peripheral pins (with the exception of $\overline{\text{CTS}}$) are set low during sleep and SPI pins are set high. Peripheral and SPI pins resume normal operation upon wake.

Digital I/O lines that have been set using I/O line passing hold their values during sleep, however the digital timeout timer (**T0** through **T9**, and **Q0** through **Q2**) are suspended during sleep and resume upon wake.

Serial communication

Serial interface

The XBee XR 900 RF Module interfaces to a host device through a serial port. The device can communicate through its serial port:

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

Serial receive buffer

When serial data enters the XBee XR 900 RF Module through the serial port, the device stores the data in the serial receive buffer until it can be processed.

When the XBee runs out of space to store incoming data, new data coming in is discarded. This is unlikely to happen unless the serial port is much faster than the RF data rate for a long period of time.

Other operations that may slow down transmissions–and increase the likelihood of dropping data–are flash write operations, route discoveries, retransmissions, and processing RF receptions.

Hardware flow control can prevent data from being discarded. When the XBee de-asserts $\overline{\text{CTS}}$, that tells the host to stop sending data because it is almost out of space to store it. As long as the host honors $\overline{\text{CTS}}$, no data will be lost.

Serial transmit buffer

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

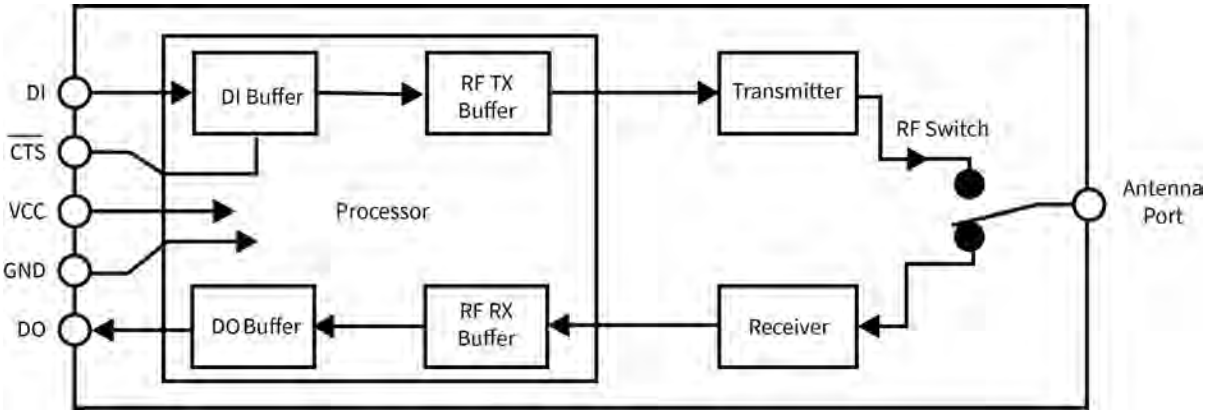
UART operation

Devices that have a UART interface connect directly to the pins of the XBee XR 900 RF Module as shown in the following figure. The figure shows system data flow in a UART-interfaced environment. Low-asserted signals have a horizontal line over the signal name.

Flow control

The XBee XR 900 RF Module maintains buffers to collect serial and RF data that it receives. The serial receive buffer collects incoming serial characters and holds them until the device can process them. The serial transmit buffer collects the data it receives via the RF link until it transmits that data out the serial port. The following figure shows the process of device buffers collecting received serial data.

Use D6 (DIO6/RTS Configuration) and D7 (DIO7/CTS Configuration) to set flow control.



Clear-to-send (CTS) flow control

If you enable CTS flow control (D7 (DIO7/CTS Configuration)), when the serial receive buffer is more than **FT** bytes full, the device de-asserts CTS (sets it high) to signal to the host device to stop sending serial data. The device reasserts CTS after the serial receive buffer has less than **FT** bytes in it. See FT (Flow Control Threshold) to configure and read this threshold.

RTS flow control

If you set D6 (DIO6/RTS Configuration) to enable RTS flow control, the device does not send data in the serial transmit buffer out the DOUT pin as long as RTS is de-asserted (set high). Do not de-assert RTS for long periods of time or the serial transmit buffer will fill. If the device receives an RF data packet and the serial transmit buffer does not have enough space for all of the data bytes, it discards the entire RF data packet.

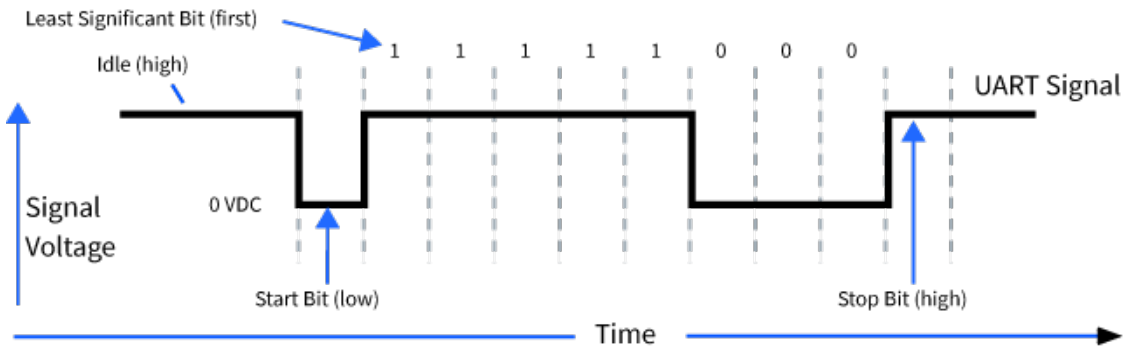
If the device sends data out the UART when RTS is de-asserted (set high) the device could send up to five characters out the UART port after RTS is de-asserted.

Serial data

A device sends data to the XBee XR 900 RF Module's UART as an asynchronous serial signal. When the device is not transmitting data, the signals should idle high.

For serial communication to occur, you must configure the UART of both devices (the microcontroller and the XBee XR 900 RF Module) with compatible settings for the baud rate, parity, start bits, stop bits, and data bits.

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



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

SPI operation

This section specifies how SPI is implemented on the device, what the SPI signals are, and how full duplex operations work.

SPI communications

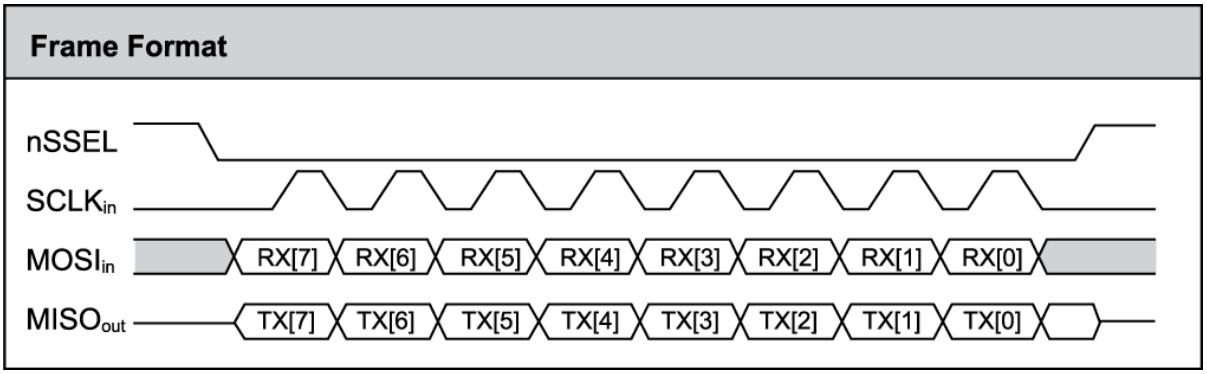
The XBee XR 900 RF Module supports SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The following table shows the signals that the SPI port uses on the device.

Signal	Direction	Function
SPI_MOSI (Master Out, Slave In)	Input	Inputs serial data from the master
SPI_MISO (Master In, Slave Out)	Output	Outputs serial data to the master
SPI_SCLK (Serial Clock)	Input	Clocks data transfers on MOSI and MISO
SPI_$\overline{\text{SSEL}}$ (Slave Select)	Input	Enables serial communication with the slave
SPI_$\overline{\text{ATTN}}$ (Attention)	Output	Alerts the master that slave has data queued to send. The XBee XR 900 RF Module asserts this pin as soon as data is available to send to the SPI master and it remains asserted until the SPI master has clocked out all available data.

In this mode:

- SPI clock rates up to 5 MHz (burst) are possible.
- Data transmission format is most significant bit (MSB) first; bit 7 is the first bit of a byte sent over the interface.
- Frame Format mode 0 is used. This means CPOL= 0 (idle clock is low) and CPHA = 0 (data is sampled on the clock's leading edge).
- The SPI port only supports API Mode (**AP** = 1).

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



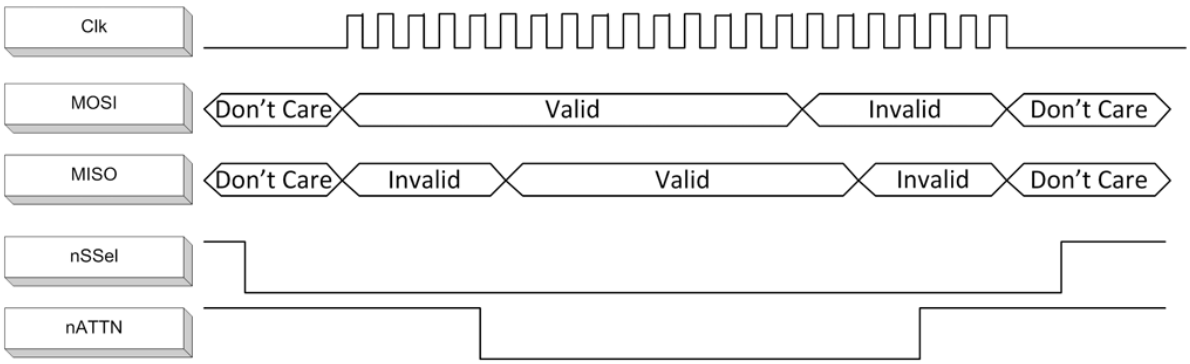
SPI mode is chip to chip communication. We do not supply a SPI communication interface on the XBee development evaluation boards included in the development kit.

Full duplex operation

When using SPI on the XBee XR 900 RF Module the device uses API operation without escaped characters to packetize data. The device ignores **AP** configuration because SPI does not operate in any other mode. SPI is a full duplex protocol, even when data is only available in one direction. This means that whenever a device receives data, it also transmits, and that data is normally invalid. Likewise, whenever a device transmits data, invalid data is probably received. To determine whether or not received data is invalid, the firmware places the data in API packets.

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

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



Low power operation

Sleep modes generally work the same on SPI as they do on UART. However, the addition of SPI mode provides an option to configure another pin as a sleep pin.

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

If SLEEP_REQUEST is not configured as a peripheral and SPI_SSEL is configured as a peripheral, then pin sleep is controlled by SPI_SSEL rather than by SLEEP_REQUEST. Asserting

SPI_SSEL by driving it low either wakes the device or keeps it awake. Negating SPI_SSEL by driving it high puts the device to sleep.

SPI_SSELcan be configured to both control sleep and to indicate that the SPI master has selected a particular slave device. This configuration provides an advantage where the pin sleep implementation on SPI mode requires one less physical pin. It does have the disadvantage that it puts the device to sleep whenever the SPI master unintentionally negates SPI_SSEL.

To effectively use the pin sharing configuration, the user/design must have control of the SPI_SSELpin to the extent that it can control pin sleep. This makes the SLEEP_REQUEST pin available for a different purpose. Without control of SPI_SSEL while using it for sleep request, the device may go to sleep at inopportune times.

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

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

Select the SPI port

To force SPI mode on through-hole devices, hold DOUT/DIO13 low while resetting the device until SPI_ATTN asserts. This causes the device to disable the UART and go straight into SPI communication mode. Once configuration is complete, the device queues a modem status frame to the SPI port, which causes the SPI_ATTN line to assert. The host can use this to determine that the SPI port is configured properly.

On surface-mount devices, forcing DOUT low at the time of reset has no effect. To use SPI mode on the SMT modules, assert the SPI_SSEL low after reset and before any UART data is input.

Forcing DOUT low on TH devices forces the device to enable SPI support by setting the following configuration values to 1 (peripheral):

Through-hole	Micro and Surface-mount	SPI signal
D1 (AD1/DIO1/TH_SPI_ATTN Configuration)	P9 (DIO19/SPI_ATTN Configuration)	ATTN
D2 (DIO2/AD2/TH_SPI_CLK Configuration)	P8 (DIO18/SPI_CLK Configuration)	SCLK
D3 (DIO3/AD3/TH_SPI_SSEL Configuration)	P7 (DIO17/SPI_SSEL Configuration)	SSEL
D4 (DIO4/TH_SPI_MOSI Configuration)	P6 (DIO16/SPI_MOSI Configuration)	MOSI
P2 (DIO12/TH_SPI_MISO Configuration)	P5 (DIO15/SPI_MISO Configuration)	MISO

Note The ATTN signal is optional–you can still use SPI mode if you disable the SPI_ATTN pin (**D1** on through-hole or **P9** on surface-mount devices).

As long as the host does not issue a **WR** command, these configuration values revert to previous values after a power-on reset. If the host issues a **WR** command while in SPI mode, these same parameters are written to flash, and after a reset the device continues to operate in SPI mode.

If the UART is disabled and the SPI is enabled in the written configuration, then the device comes up in SPI mode without forcing it by holding DOUT low. If both the UART and the SPI are configured (P3 (DIO13/DOUT Configuration) through P9 (DIO19/SPI_ATTN Configuration) are set to **1**) at the time of reset, then output goes to the UART until the host sends the first input to the SPI interface. As soon as the first input comes on the SPI port, then all subsequent output goes to the SPI port and the UART is disabled.

After the first input arrives on the SPI port, all subsequent output goes to the SPI port and the UART is disabled.

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

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

Force UART operation

If you configure a device with only the SPI enabled and no SPI master is available to access the SPI slave port, you can recover the device to UART operation by holding DIN / CONFIG low at reset time. DIN/CONFIG forces a default configuration on the UART at 9600 baud and brings up the device in Command mode on the UART port. You can then send the appropriate commands to the device to configure it for UART operation. If you write those parameters, the device comes up with the UART enabled on the next reset.

Networking

Network identifiers

You define DigiMesh networks with a unique network identifier. Use the ID (Network ID) command to set this identifier. For devices to communicate, you must configure them with the same network identifier. For devices to communicate, the HP (Preamble ID) and **ID** commands must be equal on all devices in the network.

The **ID** command directs the devices to talk to each other by establishing that they are all part of the same network. The **ID** and **HP** parameters allow multiple DigiMesh networks to co-exist within RF range of each other.

Delivery methods

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

The XBee XR 900 RF Module supports three delivery methods:

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

Point-to-multipoint

To select point-to-multipoint, set the transmit options to 0x40.

In Transparent mode, use the **TO** (Transmit Options) command to set the transmit options.

In API mode, use the Transmit Request (0x10) and Explicit Addressing Command (0x11) frames to set the transmit options. However, if the transmit options in the API frame are zero, then the transmit options in the **TO** command apply.

Point-to-multipoint transmissions occur between two adjacent nodes within RF range. No route discovery and no routing occur for these types of transmissions. The networking layer is entirely skipped.

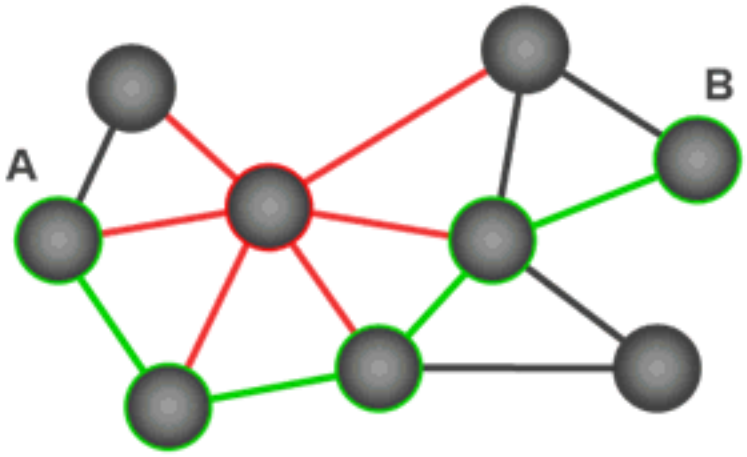
Point-to-multipoint has an advantage over DigiMesh for two adjacent devices due to less overhead. However, it cannot work over multiple hops.

DigiMesh networking

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

- **Routing**. With this technique, the message is propagated along a path by hopping from node to node until it reaches its final destination.
- **Ad-hoc network creation**. This is an automated process that creates an entire network of nodes on the fly, without any human intervention (other than initiating a transmission to discover a route to the designated destination).
- **Self-healing**. This process automatically figures out if one or more nodes on the network is broken, impeded by the environment, or powered off and repairs any broken routes.
- **Peer-to-peer architecture**. No hierarchy and no parent-child relationships are needed.

- **Quiet protocol**. Routing overhead will be reduced by using a reactive protocol similar to AODV. (This means there are no polls, beacons, or keep-alives to keep nodes connected.)
- **Route discovery**. Rather than maintaining a network map, routes will be discovered and created only when needed.
- **Selective acknowledgments**. Only the destination node will reply to route requests.
- **Reliable delivery**. Reliable delivery of data is accomplished by means of acknowledgments and retries as needed.



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

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

Broadcast addressing

All of the routers in a network receive and repeat broadcast transmissions. Broadcast transmissions do not use ACKs, so the sending device sends the broadcast multiple times. By default, the sending device sends a broadcast transmission four times. The transmissions become automatic retries without acknowledgments. This results in all nodes repeating the transmission four times as well.

In order to avoid RF packet collisions, the network inserts a random delay before each router relays the broadcast message. You can change this random delay time with the **NN** parameter.

Sending frequent broadcast transmissions can quickly reduce the available network bandwidth. Use broadcast transmissions sparingly.

The broadcast address is a 64 bit address with the lowest 16 bits set to 1. The upper bits are set to 0. To send a broadcast transmission:

- Set **DH** to 0.
- Set **DL** to 0xFFFF.

In API operating mode, this sets the destination address to 0x000000000000FFFF.

Unicast addressing

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

Packet tracking keeps the receiving device from handling a given packet multiple times. However, even for duplicate packets, the receiving node sends an ACK if it is requested. This is done because the sending node may not have received the ACK for the same packet previously sent.

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

If a device sends a unicast that uses both MAC and NWK retries and acknowledgments:

- Use MAC retries and acknowledgments for transmissions between adjacent devices in the route.
- Use NWK retries and acknowledgments across the entire route.

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

Route discovery

Route discovery is a process that occurs when:

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

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

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

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

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

Routing

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

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

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

Routers

You can use the **CE** command to configure devices in a DigiMesh network to act as routers or end devices. All devices in a DigiMesh network act as routers by default. Any devices that you configure as routers actively relay network unicast and broadcast traffic.

Repeater/directed broadcast

A directed broadcast is most useful in an environment where an aggregator needs to send messages to more than 128 nodes in a round robin fashion. Since the routing table only holds 128 entries, every unicast will result in a route discovery, which is a broadcast and multiple unicasts back to the aggregator. Then the aggregator has extra wait time to ensure that the best route is selected for unicasts.

In such an environment, we recommend a directed broadcast because it is less traffic and faster than route discovery.

The MAC layer is the building block that is used to build repeater capability. To implement Repeater mode, we use a network layer header that comes after the MAC layer header in each packet. In this network layer there is additional packet tracking to eliminate duplicate broadcasts.

Each node that receives a directed broadcast does one of the following actions:

1. Rebroadcasts the packet
2. Sends the packet out the serial port
3. Discards the packet

If the receiving node has already seen the packet, it is discarded.

Otherwise, If the receiving node is the ultimate destination, the message is sent out the serial port.

Otherwise, if the receiving node is configured as a router—which is the default—and the maximum number of hops is not reached, the message is sent—repeated—**MT** + 1 times.

Encryption

The XBee XR 900 RF Module supports 256-bit keys and AES encryption. CTR mode encryption is the default. You can use the **C8** command to specify ECB encryption with a 128-bit key for compatibility with older products.

Unlike DigiMesh 2.4, the entire frame, including the MAC and NWK layers are encrypted.

Maximum payload

In addition to the MAC, NWK, and APP headers, the XR product supports up to 256 bytes of payload.

If more than that is sent in API mode, a transmit status frame will be sent to indicate that the frame is too large.

If more than that is sent in transparent mode, the firmware will break it up into chunks <= 256 bytes.

Network commissioning and diagnostics

We call the process of discovering and configuring devices in a network for operation, "network commissioning." Devices include several device discovery and configuration features. In addition to configuring devices, you must develop a strategy to place devices to ensure reliable routes. To accommodate these requirements, modules include features to aid in placing devices, configuring devices, and network diagnostics.

Local configuration

You can configure devices locally using serial commands in Command mode or API mode, or remotely using remote AT commands. Devices that are in API mode can send configuration commands to set or read the configuration settings of any device in the network.

Remote configuration

When you do not have access to the device's serial port, you can use a separate device in API mode to remotely configure it. To remotely configure devices, use the following steps.

Send a remote command

To send a remote command, populate the Remote AT Command Request - 0x17 with:

1. The 64-bit address of the remote device.
2. The correct command options value.
3. Optionally, the command and parameter data.
4. If you want a command response, set the Frame ID field to a non-zero value.
    a. To distinguish which remote command response is associated with a given remote command request, use a different non-zero value Frame ID for each request.

XCTU has a Frames Generator tool that can assist you with building and sending a remote AT frame; see Frames generator tool in the *XCTU User Guide*.

Apply changes on remote devices

When you use remote commands to change the command parameter settings on a remote device, you must apply the parameter changes or they do not take effect. For example, if you change the **BD** parameter, the actual serial interface rate does not change on the remote device until you apply the changes. You can apply the changes using remote commands in one of three ways:

1. Set the apply changes option bit in the API frame.
2. Send an **AC** command to the remote device.
3. Send the **WR** command followed by the **FR** command to the remote device to save the changes and reset the device.

Remote command response

If a local device sends a command request to a remote device, and the API frame ID is non-zero, the remote device sends a remote command response transmission back to the local device.

When the local device receives a remote command response transmission, it sends a remote command response API frame out its UART. The remote command response indicates:

1. The status of the command, which is either success or the reason for failure.
2. In the case of a command query, it includes the register value.

The device that sends a remote command does not receive a remote command response frame if:

1. It could not reach the destination device.
2. You set the frame ID to 0 in the remote command request.

Build aggregate routes

In many applications, many or all of the nodes in the network must transmit data to a central aggregator node. In a new DigiMesh network, the overhead of these nodes discovering routes to the aggregator node can be extensive and taxing on the network. To eliminate this overhead, you can use the **AG** command to automatically build routes to an aggregate node in a DigiMesh network.

To send a unicast, devices configured for Transparent mode (**AP** = 0) must set their **DH**/**DL** registers to the MAC address of the node that they need to transmit to. This is normally the MAC address of the aggregator node. This can be a tedious process. A simple and effective method of setting DH/DL to the address of the aggregator is to use the **AG** command.

Upon deploying a DigiMesh network, you can issue the **AG** command on the desired aggregator node to cause all nodes in the network to build routes to the aggregator node.

The **AG** command requires a 64-bit parameter, which is the current DH/DL of the device(s) that needs their DH/DL updated. However, if you do not want to update the **DH**/**DL** of the device receiving the **AG** broadcast you can use the invalid address of 0xFFFE. The receiving nodes that are configured in API mode output an Aggregator Update API frame (0x8E) if they update their **DH**/**DL** address; for a description of the frame, see Aggregate Addressing Update - 0x8E.

All devices that receive an **AG** broadcast update their routing table information to build a route to the sending device, regardless of whether or not their **DH**/**DL** address is updated. The devices use this routing information for future DigiMesh unicast transmissions.

DigiMesh routing examples

Example one

In a scenario where you deploy a network, and then you want to update the **DH** and **DL** registers of all the devices in the network so that they use the MAC address of the aggregator node, which has the MAC address 0x0013A200 4052C507, you could use the following technique.

1. Deploy all devices in the network with the default **DH**/**DL** of 0xFFFF.
2. Serially, send an ATAGFFFF command to the aggregator node so it sends the broadcast transmission to the rest of the nodes.

All the nodes in the network that receive the **AG** broadcast set their **DH** to 0x0013A200 and their **DL** to 0x4052C507 because their DH/DL values match 0xFFFF. These nodes also automatically build a route to the aggregator node.

Example two

If you want all of the nodes in the network to build routes to an aggregator node with a MAC address of 0x0013A200 4052C507 without affecting the **DH** and **DL** registers of any nodes in the network:

1. Send the ATAGFFFE command to the aggregator node. This sends an **AG** broadcast to all of the nodes in the network. None of the nodes in the network will update DH/DL because none of the will have DH/DL set to 0xFFFE.

2. All of the nodes internally update only their routing table information to contain a route to the aggregator node.
3. None of the nodes update their **DH** and **DL** registers because none of the registers are set to the 0xFFFE address.

Replace nodes

You can use the **AG** command to update the routing table and **DH**/**DL** registers in the network after you replace a device. To update only the routing table information without affecting the **DH** and **DL** registers, use the process in example two, above.

To update the **DH** and **DL** registers of the network, use the following example.

Example

This example shows how to cause all devices to update their **DH** and **DL** registers to the MAC address of the sending device. In this case, assume you are using a device with a serial number of 0x0013A200 4052C507 as a network aggregator, and the sending device has a MAC address of 0x0013A200 F5E4D3B2 To update the **DH** and **DL** registers to the sending device's MAC address:

1. Replace the aggregator with 0x0013A200 F5E4D3B2.
2. Send the ATAG0013A200 4052C507 command to the new device.

Test links between adjacent devices

It often helps to test the quality of a link between two adjacent modules in a network. You can use the Test Link Request Cluster ID to send a number of test packets between any two devices in a network. To clarify the example, we refer to "device A" and "device B" in this section.

To request that device B perform a link test against device A:

1. Use device A in API mode (**AP** = 1) to send an Explicit Addressing Command (0x11) frame to device B.
2. Address the frame to the Test Link Request Cluster ID (0x0014) and destination endpoint: 0xE6.
3. Include a 12-byte payload in the Explicit Addressing Command frame with the following format:

Number of bytes	Field name	Description
8	Destination address	The address the device uses to test its link. For this example, use the device A address.
2	Payload size	The size of the test packet. Use the **NP** command to query the maximum payload size for the device.
2	Iterations	The number of packets to send. This must be a number between 1 and 4000.

4. Device B should transmit test link packets.
5. When device B completes transmitting the test link packets, it sends the following data

packet to device A's Test Link Result Cluster (0x0094) on endpoint (0xE6).

6. Device A outputs the following information as an API Explicit RX Indicator (0x91) frame:

Number of bytes	Field name	Description
8	Destination address	The address the device used to test its link.
2	Payload size	The size of the test packet device A sent to test the link.
2	Iterations	The number of packets that device A sent.
2	Success	The number of packets that were successfully acknowledged.
2	Retries	The number of MAC retries used to transfer all the packets.
1	Result	0x00 - the command was successful. 0x03 - invalid parameter used.
1	RR	The maximum number of MAC retries allowed.
1	maxRSSI	The strongest RSSI reading observed during the test.
1	minRSSI	The weakest RSSI reading observed during the test.
1	avgRSSI	The average RSSI reading observed during the test.

Example

Suppose that you want to test the link between device A (**SH**/**SL** = 0x0013A200 40521234) and device B (**SH**/**SL**=0x0013A 200 4052ABCD) by transmitting 1000 40-byte packets:

Send the following API packet to the serial interface of device A.

In the following example packet, whitespace marks fields, bold text is the payload portion of the packet:

7E 0020 11 01 0013A20040521234 FFFE E6 E6 0014 C105 00 00 **0013A2004052ABCD 0028 03E8** EB

When the test is finished, the following API frame may be received:

7E 0027 91 0013A20040521234 FFFE E6 E6 0094 C105 00 **0013A2004052ABCD 0028 03E8 03E7 0064 00 0A 50 53 52** 9F

This means:

- 999 out of 1000 packets were successful.
- The device made 100 retries.
- **RR** = 10.
- maxRSSI = -80 dBm.
- minRSSI = -83 dBm.
- avgRSSI = -82 dBm.

If the Result field does not equal zero, an error has occurred. Ignore the other fields in the packet.

If the Success field equals zero, ignore the RSSI fields.

The device that sends the request for initiating the Test link and outputs the result does not need to be the sender or receiver of the test. It is possible for a third node, "device C", to request device A to perform a test link against device B and send the results back to device C to be output. It is also possible for device B to request device A to perform the previously mentioned test. In other words, the frames can be sent by either device A, device B or device C and in all cases the test is the same: device A sends data to device B and reports the results.

Trace route option

In many networks, it is useful to determine the route that a DigiMesh unicast takes to its destination, particularly when you set up a network or want to diagnose problems within a network.

Note Because of the large number of Route Information Packet frames that a unicast with trace route enabled can generate, we suggest you only use the trace route option for occasional diagnostic purposes and not for normal operations.

The Transmit Request (0x10 and 0x11) frames contain a trace route option, which transmits routing information packets to the originator of the unicast using the intermediate nodes.

When a device sends a unicast with the trace route option enabled, the unicast transmits to its destination devices, which forward the unicast to its eventual destination. The destination device transmits a Route Information Packet (0x8D) frame back along the route to the unicast originator.

The Route Information Packet frame contains:

- Addressing information for the unicast
- Addressing information for the intermediate hop
- Timestamp
- Other link quality information

For a full description of the Route Information Packet frame, see Route Information - 0x8D.

Trace route example

Suppose that you successfully unicast a data packet with trace route enabled from device A to device E, through devices B, C, and D. The following sequence would occur:

- After the data packet makes a successful MAC transmission from device A to device B, device A outputs a Route Information Packet frame indicating that the transmission of the data packet from device A to device E was successful in forwarding one hop from device A to device B.
- After the data packet makes a successful MAC transmission from device B to device C, device B transmits a Route Information Packet frame to device A. When device A receives the Route Information packet, it outputs it over its serial interface.
- After the data packet makes a successful MAC transmission from device C to device D, device C transmits a Route Information Packet frame to device A (through device B). When device A receives the Route Information packet, it outputs it over its serial interface.
- After the data packet makes a successful MAC transmission from device D to device E, device D transmits a Route Information Packet frame to device A (through device C and device B). When device A receives the Route Information packet, it outputs it over its serial interface.

There is no guarantee that Route Information Packet frames will arrive in the same order as the route taken by the unicast packet. On a weak route, it is also possible for the transmission of Route Information Packet frames to fail before arriving at the unicast originator.

NACK messages

Transmit Request (0x10 and 0x11) frames contain a negative-acknowledge character (NACK) API option (Bit 2 of the Transmit Options field).

If you use this option when transmitting data, when a MAC acknowledgment failure occurs on one of the hops to the destination device, the device generates a Route Information Packet (0x8D) frame and sends it to the originator of the unicast.

This information is useful because it allows you to identify and repair marginal links.

RSSI indicators

The received signal strength indicator (RSSI) measures the amount of power present in a radio signal. It is an approximate value for signal strength received on an antenna.

You can use the **DB** command to measure the RSSI on a device. **DB** returns the RSSI value measured in -dBm of the last packet the device received. This number can be misleading in multi-hop DigiMesh networks. The **DB** value only indicates the received signal strength of the last hop. If a transmission spans multiple hops, the **DB** value provides no indication of the overall transmission path, or the quality of the worst link, it only indicates the quality of the last link.

To determine the **DB** value in hardware:

1. Set **PO** to 1 to enable the RSSI pulse-width modulation (PWM) functionality.
2. Use the DIO10/RSSI/PWM0 module pin (pin 7). When the device receives data, it sets the RSSI PWM duty cycle to a value based on the RSSI of the packet it receives.

This value only indicates the quality of the last hop of a multi-hop transmission. You could connect this pin to an LED to indicate if the link is stable or not.

Associate LED

The Associate pin (pin 26) provides an indication of the device's status. To take advantage of these indications, connect an LED to the Associate pin.

To enable the Associate LED functionality, set the **D5** command to 1; it is enabled by default. If enabled, the Associate pin is configured as an output. This section describes the behavior of the pin.

The pin functions as a power indicator.

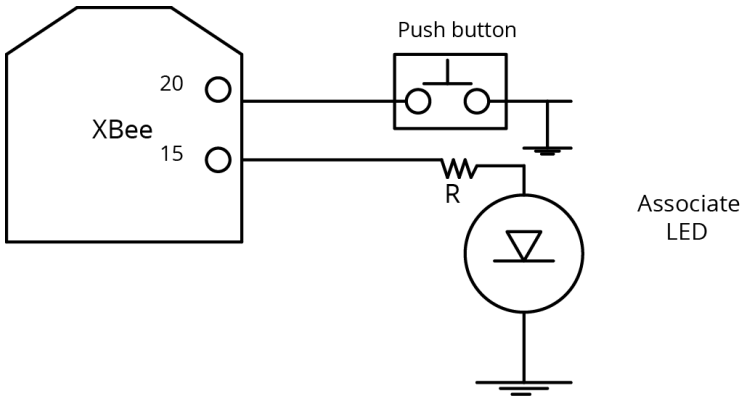
Use the **LT** command to override the blink rate of the Associate pin. If you set **LT** to 0, the device uses the default blink time of 250 ms.

The following table describes the Associate LED functionality.

Sleep mode	LED Status	Meaning
0	On, blinking	The device has power and is operating properly
1, 4, 5	Off	The device is asleep
1, 4, 5	On, blinking	The device has power, is awake and is operating properly

The Commissioning Pushbutton

The XBee XR 900 RF Module supports a set of commissioning and LED functions to help you deploy and commission devices. These functions include the Commissioning Pushbutton definitions and the associated LED functions. The following diagram shows how the hardware can support these features.



To support the Commissioning Pushbutton and its associated LED functions, connect a pushbutton and an LED to device pins 20 and 15 respectively.

Definitions

To enable the Commissioning Pushbutton functionality on pin 20, set the **D0** command to 1. The functionality is enabled by default.

You must perform the designated number of button presses within two seconds. If any number of commissioning button presses occur while the device is asleep, it will wake up until the sleep cycle is finished or for 30 seconds, whichever occurs first.

The following table provides the pushbutton definitions.

Button presses	Action
1	Sends a Node Identification broadcast transmission. All devices that receive this transmission blink their Associate LED rapidly for one second. Additionally, receiving devices that are operating in API mode also send a Node Identification frame (0x95) out their UART.
2	This function only applies for synchronous sleep networks. Two button presses nominate a node as the sleep coordinator by sending out a sync message. If the sending node has seniority over the current sleep coordinator, the sending node becomes the sleep coordinator. Otherwise, the current sleep coordinator retains that role.
4	Restores the node to default configuration. If custom defaults are in use, they will be applied on top of the factory defaults. Unlike RE (Restore Defaults), this function not only restores the default configuration, but it also applies those changes.

Use the Commissioning Pushbutton

Use the **CB** command to simulate button presses in software. Send **CB** with a parameter set to the number of button presses to perform. For example, if you send **ATCB1**, the device performs the action(s) associated with a single button press.

Node Identification Indicator - 0x95 is similar to Remote AT Command Response- 0x97 – it contains the device's address, node identifier string (**NI** command), and other relevant data. All devices in API operating mode that receive the Node Identification Indicator frame send it out their UART as a Node Identification Indicator frame.

Node discovery

Node discovery has three variations as shown in the following table:

Commands	Syntax	Description
Node Discovery	**ND**	Seeks to discover all nodes in the network (on the current Network ID).
Directed Node Discovery	**ND** <NI String>	Seeks to discover if a particular node named <NI String> is found in the network.
Destination Node	**DN** <NI String>	Sets **DH**/**DL** to point to the MAC address of the node whose <NI String> matches.

The node discovery command (without an NI string designated) sends out a broadcast to every node in the Network ID. Each node in the network sends a response back to the requesting node.

When the node discovery command is issued in Command mode, all other AT commands are inhibited until the node discovery command times out, as determined by the **N?** parameter. After the timeout, an extra CR is output to the terminal window, indicating that new AT commands can be entered. This is the behavior whether or not there were any nodes that responded to the broadcast.

When the node discovery command is issued in API mode, the behavior is the same except that other commands may be executed while waiting for the pending command to complete and the response is output in API mode. If no nodes respond, there will be no responses at all to the node discover command.

Discover all the devices on a network

You can use the **ND** (Network Discovery) command to discover all devices on a network. When you send the **ND** command:

1. The device sends a broadcast **ND** command through the network.
2. All devices that receive the command send a response that includes their addressing information, node identifier string and other relevant information. For more information on the node identifier string, see NI (Node Identifier).

ND is useful for generating a list of all device addresses in a network.

When a device receives the network discovery command, it waits a random time before sending its own response. You can use the **NT** command to set the maximum time delay on the device that you use to send the **ND** command.

- The device that sends the **ND** includes its **NT** setting in the transmission to provide a random delay window for all devices in the network. When devices respond at random intervals during the **NT** window, fewer collisions occur and more responses can be obtained.
- The default **NT** value is 0x82 (13 seconds).

Directed node discovery

The directed node discovery command (**ND** with an **NI** string parameter) sends out a broadcast to find a node in the network with a matching **NI** string. If such a node exists, it sends a response with its information back to the requesting node.

In Transparent mode, the requesting node outputs an extra carriage return following the response from the designated node and the command terminates; it is then ready to accept a new AT command. In the event that the requested node does not exist or is too slow to respond, the requesting node outputs an ERROR response after **N?** expires.

In API mode, the response from the requesting node will be output in API mode and the command will terminate immediately. If no response comes from the requested node, the requesting node outputs an error response in API mode after **N?** expires. The device's software assumes that each node has a unique **NI** string.

The directed node discovery command terminates after the first node with a matching **NI** string responds. If that **NI** string is duplicated in multiple nodes, the first responding node may not always be the same node or the desired node.

Destination Node

The Destination Node command (**DN** with an **NI** string parameter) sends out a broadcast containing the **NI** string being requested. The responding node with a matching **NI** string sends its information back to the requesting node. The local node then sets **DH**/**DL** to match the address of the responding node. As soon as this response occurs, the command terminates successfully. If the device is in AT Command mode, an OK string is output and Command mode exits. In API mode, you may enter another AT command.

If an **NI** string parameter is not provided, the **DN** command terminates immediately with an error. If a node with the given NI string does not respond, the **DN** command terminates with an error after **N?** times out.

In Transparent mode, unlike **ND** (with or without an **NI** string), **DN** does not cause the information from the responding node to be output; rather it simply sets **DH**/**DL** to the address of the responding node.

In API mode, the response from the requesting node outputs in API mode and the command terminates immediately. If no response comes from the requested node, the requesting node outputs an error response in API mode after **N?** expires.

The device's software assumes that each node has a unique **NI** string. The directed destination node command terminates after the first node with a matching **NI** string responds. If that **NI** string is duplicated in multiple nodes, **DH**/**DL** may not be set to the desired value.

Discover devices within RF range

The **FN** (Find Neighbor) command works the same as the **ND** (Node Discovery) except that it is limited to neighboring devices (devices that are only one hop away). See FN (Find Neighbors) for details.

- You can use the **FN** (Find Neighbors) command to discover the devices that are immediate neighbors (within RF range) of a particular device.
- **FN** is useful in determining network topology and determining possible routes.

You can send **FN** locally on a device in Command mode or you can use a local Local AT Command Request - 0x08.

To use **FN** remotely, send the target node a Remote AT Command Request - 0x17 using **FN** as the name of the AT command.

The device you use to send **FN** transmits a zero-hop broadcast to all of its immediate neighbors. All of the devices that receive this broadcast send an RF packet to the device that transmitted the **FN** command. If you sent **FN** remotely, the target devices respond directly to the device that sent the **FN** command. The device that sends **FN** outputs a response packet in the same format as an Local AT Command Response - 0x88.

Sleep support

Sleep is implemented to support installations where a mains power source is not available and a battery is required. In order to increase battery life, the device sleeps, which means it stops operating. It can be woken by a timer expiration or a pin.

Sleep modes

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

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

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

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

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

Note Synchronous and asynchronous sleep modes are incompatible. Synchronous and asynchronous sleep nodes should not be configured in the same network. Asynchronous sleep does not apply in a mesh network. It can only operate over one hop where a designated node holds messages for the sleeping node.

Transmissions sent to an asynchronously sleeping device are not buffered and will be lost if the receiving device is not actively awake when the transmission occurred. If you require two-way communication, you can use Indirect Messaging for P2MP unicast messages. For more information, see Indirect messaging and polling.

Asynchronous sleep modes

Use the asynchronous sleep modes to control the sleep state on a device by device basis.

Do not use devices operating in asynchronous sleep mode to route data.

We strongly encourage you to set asynchronous sleeping devices as end-devices using CE (Routing/Messaging Mode). This prevents the node from attempting to route data.

Asynchronous Pin Sleep mode (SM = 1)

Pin Sleep mode minimizes quiescent power (power consumed when in a state of rest or inactivity). In order to use Pin Sleep mode, configure SM (Sleep Mode) to **1** and configure D8 (DIO8/DTR/SLP_Request Configuration) (pin 9) for $\overline{\text{DTR}}$/SLEEP_RQ input (**D8** = **1**). This mode is voltage level-activated; when SLEEP_RQ is asserted, the device finishes any transmit or receive activities, enters Idle mode, and then enters a state of sleep. The device does not respond to either serial or RF activity while in pin sleep.

To wake a sleeping device operating in Pin Sleep mode, de-assert $\overline{\text{DTR}}$/SLEEP_RQ. The device wakes when SLEEP_RQ is de-asserted and is ready to transmit or receive when the $\overline{\text{CTS}}$ line is low.

Devices with SPI functionality can use the SPI_$\overline{\text{SSEL}}$ pin instead of **D8** for pin sleep control. If **D8** = **0** and **P7** = **1**, SPI_$\overline{\text{SSEL}}$ takes the place of $\overline{\text{DTR}}$/SLEEP_RQ and functions as described above. In order to use SPI_$\overline{\text{SSEL}}$ for sleep control while communicating on the UART, SPI pins–**P5**, **P6**, and **P8**–must not be set to 1 (peripheral). See Low power operation for information on using SPI_$\overline{\text{SSEL}}$ for sleep control while communicating over SPI.

Asynchronous Cyclic Sleep mode (SM = 4)

The Cyclic Sleep modes allow devices to periodically check for RF data. When the **SM** parameter is set to **4**, the XBee XR 900 RF Module is configured to sleep, then wakes once per cycle to check for data from a coordinator. The Cyclic Sleep Remote sends a poll request to the coordinator at a specific interval set by SP (Cyclic Sleep Period). The coordinator transmits any queued data addressed to that specific remote upon receiving the poll request.

If no data is queued for the remote, the messaging coordinator does not transmit and the remote returns to sleep for another cycle. If queued data is transmitted back to the remote, it stays awake to allow for back and forth communication until the ST (Cyclic Sleep Wake Time) timer expires. You can also set SO (Sleep Options) bit 8 to force the device to always wake for the full **ST** time.

To configure a node to act as a coordinator **CE** must be set to 1. A sleeping node also needs to be configured to know which node is its coordinator. This is done by setting the **DH** and **DL** of the sleeper to the **SH** and **SL** of the coordinator node. **CE** on the sleeper node must also be set to **4**. In order for the coordinator to queue transmissions meant for the sleeping node any transmissions sent to the sleeping node must be sent using 0x40 (point-to-point) **TO** options.

If configured, $\overline{\text{CTS}}$ goes low each time the remote wakes, allowing for communication initiated by the remote host if desired. If ON_$\overline{\text{SLEEP}}$ is configured it goes high (ON) after SN (Number of Sleep Periods) sleep periods. Change **SN** to allow external circuitry to sleep for longer periods if no data is received.

Asynchronous Cyclic Sleep with Pin Wake-up mode (SM = 5)

Use this mode to wake a sleeping remote device through either the RF interface or by asserting (low) $\overline{\text{DTR}}$/SLEEP_RQ for event-driven communications. The cyclic sleep mode works as described previously with the addition of a pin-controlled wake-up at the remote device.

The $\overline{\text{DTR}}$/SLEEP_RQ pin is level-triggered. The device wakes when a low is detected then sets $\overline{\text{CTS}}$ low as soon as it is ready to transmit or receive. The device stays awake as long as $\overline{\text{DTR}}$/SLEEP_RQ is low; once $\overline{\text{DTR}}$/SLEEP_RQ goes high the device returns to cyclic sleep operation. If $\overline{\text{DTR}}$/SLEEP_RQ is momentarily pulsed low, the minimum wake time is ST (Cyclic Sleep Wake Time) even if $\overline{\text{DTR}}$/SLEEP_RQ is low for less time.

Once awake, any activity resets the ST (Cyclic Sleep Wake Time) timer, so the device goes back to sleep only after there is no RF activity for the duration of the timer.

Synchronous sleep modes

Synchronous sleep makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time. This allows all or most devices in a network to use low power because, unlike Zigbee, low power devices do not need to be adjacent to mains powered devices.

Synchronous sleep forms a cyclic sleeping network with these features:

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

Synchronous sleep support mode (SM = 7)

Note Synchronous sleep is not available at the lowest baud rate–**BR 0**.

Note Sleep support nodes should be mains powered because they do not sleep.

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

A device in synchronous sleep support mode synchronizes itself with a sleeping network but will not itself sleep. At any time, the device responds to new devices that are attempting to join the sleeping network with a sync message. A sleep support device only transmits normal data when the other devices in the sleeping network are awake. You can use sleep support devices as sleep coordinator devices and as aids in adding new devices to a sleeping network.

Synchronous cyclic sleep mode (SM = 8)

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

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

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

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

If **D9** = 1 (ON_$\overline{\text{SLEEP}}$ enabled) on a cyclic sleep node, the ON_$\overline{\text{SLEEP}}$ line goes high when the device is awake and goes low when the device is asleep.

If **D7** = 1, the device de-asserts $\overline{\text{CTS}}$ while asleep. This allows hardware flow control to be used to avoid losing data that is sent while the radio is asleep.

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

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

Sleep parameters

The following AT commands are associated with the sleep modes. See the linked commands for the parameter's description, range and default values.

- SM (Sleep Mode)
- SN (Number of Sleep Periods)
- SO (Sleep Options)
- ST (Cyclic Sleep Wake Time)
- SP (Cyclic Sleep Period)
- WH (Wake Host Delay)

Sleep pins

The following table describes the five external device pins associated with sleep.

Pin name	Description
$\overline{\text{DTR}}$ /SLEEP_ RQ	For **SM** = **1**, high puts the device to sleep and low wakes it up. For **SM** = **5**, a high to low transition wakes the device up for **ST** time. The device ignores a low to high transition in **SM** = **5**.
SPI_$\overline{\text{SSEL}}$	This pin operates the same as SLEEP_RQ when **D8** is **0**.
$\overline{\text{CTS}}$	If **D7** = **1**, high indicates that the device is asleep and low indicates that it is awake and ready to receive serial data.
ON_ $\overline{\text{SLEEP}}$	Low indicates that the device is asleep and high indicates that it is awake and ready to receive serial data.
DIO0	When configured as a peripheral (1), this is a commissioning button. In this default condition, DIO0 can also awaken a device from pin sleep.

Sleep conditions

Since instructions stop executing while the device is sleeping, it is important to avoid sleeping when the device has work to do. For example, the device will not sleep if any of the following are true:

1. The device is operating in Command mode, or in the process of getting into Command mode with the **+++** sequence.
2. The device is processing AT commands from API mode
3. The device is processing remote AT commands
4. Something is queued to the serial port and that data is not blocked by $\overline{\text{RTS}}$ flow control

If each of the above conditions are false, then sleep may still be blocked in these cases:

1. Enough time has not expired since the device has awakened.
   a. If the device is operating in pin sleep, the amount of time needed for one character to be received on the UART is enough time.
   b. If the device is operating in cyclic sleep, enough time is defined by a timer. The duration of that timer is:
      i. defined by **ST** if in **SM** 5 mode and it is awakened by a pin
      ii. defined by ST if SO bit 8 is set
      iii. 30 ms if operating as a polling end device to allow enough time for a poll and a poll response
   c. In addition, the wake time is extended by an additional **ST** time when new OTA data or serial data is received.
2. Sleep Request pin is not asserted when operating in pin sleep mode
3. Data is waiting to be sent OTA.

The sleep timer

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

- The duration of that timer is defined by ST if SO bit 8 is set.

Indirect messaging and polling

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

Indirect messaging

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

Because it is possible for a polling device to be eliminated, a mechanism is in place to purge unrequested data packets. If the coordinator holds an indirect data packet for an indirect messaging poller for more than 2.5 times its **SP** value, then the packet is purged. We suggest setting the **SP** of the coordinator to the same value as the highest **SP** time that exists among the pollers in the network. If the coordinator is in API mode, a TxStatus message is generated for a purged data packet with a status of 0x75 (**INDIRECT_MESSAGE_UNREQUESTED**).

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

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

Polling

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

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

Sleep coordinator sleep modes in the network

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

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

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

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

Synchronization messages

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

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

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

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

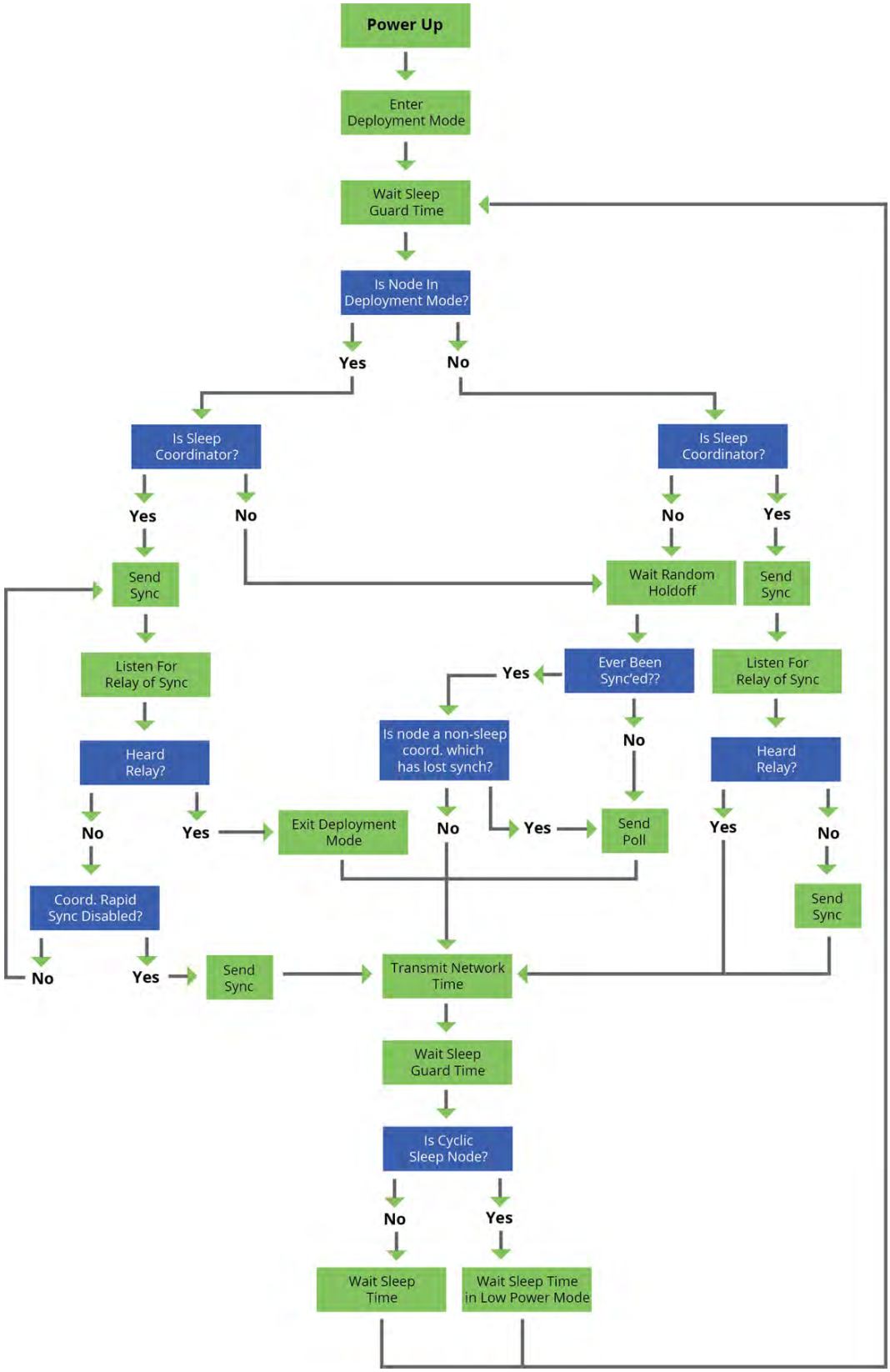
Use the **SO** (sleep options) command to disable deployment mode. This option is enabled by default.

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

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

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

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


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

Become a sleep coordinator

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

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

Set the sleep coordinator option

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

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

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

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

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

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

Resolution criteria and selection option

There is an automatic selection process with resolution criteria that occurs on a node if it loses contact with the network sleep coordinator.

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

- Misses three or more sync messages and it:
- Is not configured as a non-coordinator by setting bit 1 of **SO**.

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

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

1. Newer sleep parameters: the network considers a node using newer sleep parameters (**SP** and **ST**) as higher priority than a node using older sleep parameters. See Commissioning Pushbutton option. Note that when **SP** and/or **ST** is changed, it increments the sequence number such that it sends the newest sync message and it has priority to become the sleep coordinator.

2. Sleep coordinator: a node configured as the sleep coordinator is higher priority than other nodes.
3. Sleep support node: sleep support nodes are higher priority than cyclic sleep nodes. You can modify this behavior using the **SO** parameter.
4. Serial number: If the previous factors do not resolve the priority, the network considers the node with the higher serial number to be higher priority.

Commissioning Pushbutton option

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

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

Overriding syncs

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

An overriding sync occurs whenever **ST** or **SP** is changed to a value greater than 2% different than **OW** or **OS** respectively. For example no overriding sync will occur if **SP** is changed from 190 to C8 if the network was already operating with **OS** at C8. On the other hand, if **SP** is changed from 190 to 190 (meaning no change), and **OS** is C8, than an overriding sync will occur because the network parameters are being written.

Even parameters that seem unrelated to sleep can kick off an overriding sync. These are **NH**, **NN**, **RN**, and **MT**. When any of these parameters are changed, they can affect network traversal time. If such changes cause the configured value of **ST** to be smaller than the value needed for network traversal, then **ST** is increased and if that increased value is greater than 2% different than **OW**, then an overriding sync will occur.

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

Sleep guard times

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

Auto-early wake-up sleep option

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

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

Use bit 3 of the **SO** command to disable auto-early wake-up sleep.

Select sleep parameters

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

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

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

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

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

   This is the maximum amount of time it takes for a sleep synchronization message to propagate to every node in the network. You can estimate this number with the following formula:

   SMPT = **NH** * **%8** ms.

   Note The 4 msec constant applies to XBee DigiMesh, but it is different for every platform on which DigiMesh runs.

3. Select the duty cycle you want.
   The ratio of sleep time to wake time is the factor that has the greatest effect on the device's power consumption. Battery life can be estimated based on the following factors:
   - sleep period
   - wake time
   - sleep current
   - RX current
   - TX current
   - battery capacity
4. Choose the sleep period and wake time.

   The wake time must be long enough to transmit the desired data as well as the sync message. The **ST** parameter automatically adjusts upwards to its minimum value when you change other AT commands that affect it (**SP**, **NN**, and **NH**).

   Use a value larger than this minimum. If a device misses successive sync messages, it reduces its available transmit time to compensate for possible clock drift. Budget a large enough **ST** time to allow for the device to miss a few sync messages and still have time for normal data transmissions.

Sleep immediate

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

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

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

Start a sleeping synchronous network

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

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

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

1. If you are going to use a sleep coordinator in the network, deploy it first.
2. If more than one node can be the sleep coordinator, select a node for deployment, power it on and press the Commissioning Pushbutton twice. This causes the node to begin emitting sync messages.
3. Verify that the first node is emitting sync messages by watching its associate LED. A slow blink indicates that the node is acting as a sleep coordinator.

4. Power on nodes in range of the sleep coordinator or other nodes that have synchronized with the network. If the synchronized node is asleep, you can wake it by pressing the Commissioning Pushbutton once.
5. Wait a sleep cycle for the new node to sync itself.
6. Verify that the node syncs with the network. The associate LED blinks when the device is awake and synchronized.
7. Continue this process until you deploy all of the nodes.

Add a new node to an existing network

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

1. Power the new node on within range of a sleep support node. Sleep support nodes are always awake and able to respond to sync requests promptly.
2. You can wake a sleeping cyclic sleep node in the network using the Commissioning Pushbutton. Place the new node in range of the existing cyclic sleep node. Wake the existing node by pressing the Commissioning Pushbutton once. The existing node stays awake for 30 seconds and responds to sync requests while it is awake.

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

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

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

Change sleep parameters

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

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

When you make changes to a node's **SP** and/or **ST** parameters and that node does not have the non-sleep coordinator option set then:

- That node broadcasts an overriding sync to the network to advertise the new sleep cycle.
- That node nominates itself to become the sleep coordinator.
- That node will remain the sleep coordinator unless another node in the network designates

itself as the sleep coordinator.

- The network will apply the new sleep parameters at the beginning of the next wake cycle.

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

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

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

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

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

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

Rejoin nodes that lose sync

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

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

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

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

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

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

Diagnostics

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

Query sleep cycle

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

Sleep status

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

Missed sync messages command

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

Sleep status API messages

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

AT commands

Memory access commands

This section details the executable commands that provide memory access to the device.

AC (Apply Changes)

This command applies changes to all command parameters configured in Command mode.

Any of the following also applies changes the same as issuing an **AC** command:

- Exiting Command mode with a **CN** command.
- Exiting Command mode via timeout.
- Receiving a 0x08 API command frame.
- Issuing a 0x08 Local AT Command API frame.
- Issuing a remote 0x17 AT Command API frame with option bit 1 set.

Example: Altering the UART baud rate with the **BD** command does not change the operating baud rate until after an **AC** command is received; at this point, the interface immediately changes baud rates.

Parameter range

N/A

Default

N/A

FR (Software Reset)

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

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

Parameter range

N/A

Default

N/A

RE (Restore Defaults)

Parameter range

N/A

Default

N/A

WR (Write)

Immediately writes parameter values to non-volatile flash memory so they persist through a power cycle. Operating network parameters are persistent and do not require a **WR** command for the device to reattach to the network.

Note Once you issue a **WR** command, do not send any additional characters to the device until after you receive the **OK** response. Use the **WR** command sparingly; the device's flash only supports 10,000 erase/write cycles.

Parameter range

N/A

Default

N/A

MAC/PHY commands

The following commands are MAC/PHY commands.

PL (TX Power Level)

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

Parameter range

0 - 4

Setting	XBee Tx power level
0	0 dBm
1	+10 dBm
2	+13 dBm
3	+16 dBm
4	+19 dBm

Default

4

DR (Dynamic RSSI)

Sets or reads the mode of operation of the Low Noise Amplifier (LNA).

Note The dynamic LNA mode–option 2–is not implemented at this time and will keep the LNA enabled like option 1.

Note DR changes must be applied to be effective, whether for receiving data, doing an energy detect, or doing LBT before a transmission. Whatever DR does to the LNA applies in all cases until it is changed again.

Parameter range

0 - 2

Parameter	Description
0	LNA disabled
1	LNA enabled
2	Let software enable/disable LNA automatically

Default

2

ED (Energy Detect)

Starts an energy detect scan. The device loops through all the available channels and returns the maximal energy on each channel, a comma follows each value, and the list ends with a carriage return. The values returned reflect the energy level that **ED** detects in -dBm units.

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

Note **DR** affects energy detect–for example if **DR** turns the LNA on, then **ED** gets energy levels with LNA on and vice-versa. Also, if **DR** is set in Command mode prior to the **ED** command, then an **AC** command is needed in between to apply the changes made with the **DR** command.

Parameter range

0 - 0xFF

Default

0xA

MF (Minimum Frequencies)

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

Parameter range

read-only

Default

United States/Canada: 0x32 (50 channels)

AF (Available Frequencies)

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

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

Channels for **BR0** and **BR1** data rates are spaced 250 kHz apart. But, channels for the **BR2** data rate are spaced 500 kHz apart.

Parameter range

0x7FF - 0x1F FFFF FFFF FFFF FFFF FFFF FFFF

Default

BR	Bitfield	Operating frequency range	Channel spacing
0/1	0x1F ffff FFFF ffff FFFF ffff FFFF	902,500 to 927,500 kHz	250 kHz
2	0x03 FFFF ffff FFFF	902,750 to 927,250 kHz	500 kHz

CM (Channel Mask)

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

When **CM** is queried, it returns the operating channel mask based on what value **BR** is set to.

When **BR** is set to **2**, a fixed channel mask is used–see the defaults below. A user-defined **CM** value is only used when **BR** is set to **0** or **1**.

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

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

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

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

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

Parameter range

For **BR2**, **CM** is read-only and returns 0x3 FFFF FFFF FFFF.

For **BR0** and **BR1**, exactly **MF** of the bits in 0x1F FFFF FFFF FFFF FFFF FFFF FFFF must be set. **MF** bits must be set because **MF** is the number of hopping channels.

Note It does not matter which bits are set as long as it is the correct number of bits and all the other nodes in the same network have the same bits set.

Default

Default CM when BR is 0 or 1	Default CM when BR is 2
0x05 5555 5555 5555 5555 5555 5555	0x00 0000 0000 0003 FFFF FFFF FFFF

BR (RF Data Rate)

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

When **BR** = **0** the RF data rate is 10 kb/s. This rate is too slow to support DigiMesh routing and synchronous sleep. Setting **BR** to **0** automatically disables these features and uses point to point communication for data routing.

All devices on the network must have the same **BR** value set in order to communicate. **BR** directly affects the range of the device. The higher the RF data rate, the lower the receive sensitivity.

Parameter range

0 - 2

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

Default

2

ID (Network ID)

Set or read the user network identifier.

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

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

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

Parameter range

0 - 0x7FFF

Default

0x7FFF

HP (Preamble ID)

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

Parameter range

0 - 9 (usually)

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

Default

0

RR (Unicast Mac Retries)

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

Parameter range

0 - 0xF

Default

0xA (10 retries)

MT (Broadcast Multi-Transmits)

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

Parameter range

0 - 5

Default

3

MAC diagnostics commands

The following commands provide Media Access Control diagnostic information.

BC (Bytes Transmitted)

The number of RF bytes transmitted. The firmware counts every byte of every packet, including MAC/PHY headers and trailers.

You can reset the counter to any 32-bit value by appending a hexadecimal parameter to the command. This value is volatile (the value does not persist in the device's memory after a power-up sequence).

Parameter range

0 - 0xFFFFFFFF

Default

N/A (0 after reset)

DB (Last Packet RSSI)

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

-dBm measurement.

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

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

If the XBee XR 900 RF Module has been reset and has not yet received a packet, **DB** reports **0**.

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

Parameter range

0 - 0xFF [read-only]

Default

GD (Good Packets Received)

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

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

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

Parameter range

0 - 0xFFFF

Default

N/A (0 after reset)

EA (MAC ACK Failure Count)

The number of unicast transmissions that time out awaiting a MAC ACK. This can be up to **RR** +1 timeouts per unicast when **RR** > 0.

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

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

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

Parameter range

0 - 0xFFFF

Default

0x0

TR (Transmission Failure Count)

This count increments whenever a MAC transmission attempt exhausts all MAC retries without ever receiving a MAC acknowledgment message from the destination node. Once the number reaches 0xFFFF, it does not count further events.

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

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

Parameter range

0 - 0xFFFF

Default

N/A (0 after reset)

UA (Unicasts Attempted Count)

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

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

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

Parameter range

0 - 0xFFFF

Default

0

%H (MAC Unicast One Hop Time)

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

The time to send a unicast between two nodes in the network should not exceed the product of the unicast one hop time (**%H**) and the number of hops between those two nodes.

Parameter range

[read-only]

Default

N/A

%8 (MAC Broadcast One Hop Time)

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

The time to send a broadcast between two nodes in the network should not exceed the product of the broadcast one hop time (**%8**) and the number of hops between those two nodes.

Parameter range

[read-only]

Default

N/A

Networking commands

CE (Routing/Messaging Mode)

The routing and messaging mode of the device.

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

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

Parameter range

0 - 6

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

Default

0

C8 (Compatibility Options)

Parameter range

Bit field:

Bit 2 (0x04) allows compatibility with SX versions prior to 9009 when encryption is enabled. In that case AES CBC mode is used. This bit may be set in all circumstances, even when backwards compatibility is not needed, but it is not recommended because CTR mode is considered to be more secure.

If bit 2 is not set, then AES CTR mode is used instead. This type of encryption is considered more secure than CBC mode and should be used when backwards compatibility is not needed.

All other bits should be 0 for future compatibility.

Bit	Meaning	Setting	Description
2	TX compatibility	0	When encryption is enabled, AES Counter mode is used with a 256-bit key.

Bit	Meaning	Setting	Description
		1	When encryption is enabled AES ECB mode is used with a 128-bit key. This is compatible with legacy versions of DigiMesh 2.4.
3	N/A	N/A	N/A

Default

0x00

BH (Broadcast Hops)

Parameter range

Default

0

NH (Network Hops)

Parameter range

Default

MR (Mesh Unicast Retries)

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

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

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

If you set this parameter to **0**, it disables network ACKs. Initially, the device can find routes, but a route will never be repaired if it fails.

Parameter range

0 - 7 mesh unicast retries

Default

1

NN (Network Delay Slots)

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

See %8 (MAC Broadcast One Hop Time) to get the timing for a network delay slot.

Parameter range

1 - 5

Default

3

AG (Aggregator Support)

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

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

Parameter range

Any 64-bit address

Default

N/A

SE (Source Endpoint)

Sets or displays the application layer source endpoint value used for data transmissions.

This command only affects outgoing transmissions in Transparent mode (**AP** = **0**).

Note Endpoints **0xDC** - **0xEE** are reserved for special use by Digi and should not be used in an application outside of the listed purpose.

The reserved Digi endpoints are:

- 0xE8 - Digi data endpoint
- 0xE6 - Digi device object endpoint
- 0xE5 - Secure Session Server endpoint
- 0xE4 - Secure Session Client endpoint
- 0xE3 - Secure Session SRP authentication endpoint

Parameter range

0 - 0xFF

Default

0xE8

DE (Destination Endpoint)

Sets or displays the application layer destination endpoint used for data transmissions.

This command only affects outgoing transmissions in Transparent mode (**AP** = **0**).

Note Endpoints **0xDC** - **0xEE** are reserved for special use and should not be used in an application outside of the listed purpose.

The reserved Digi endpoints are:

- 0xE8 - Digi data endpoint
- 0xE6 - Digi device object endpoint
- 0xE5 - Secure Session Server endpoint
- 0xE4 - Secure Session Client endpoint
- 0xE3 - Secure Session SRP authentication endpoint

Parameter range

0 - 0xFF

Default

0xE8

CI (Cluster ID)

Parameter range

0 - 0xFFFF

Default

0x11 (Transparent data cluster ID)

Addressing commands

SH (Serial Number High)

This value is read-only and it never changes.

Parameter range

0x0013A200 - 0x0013A2FF [read-only]

Default

Set in the factory

SL (Serial Number Low)

This value is read-only and it never changes.

Parameter range

0 - 0xFFFFFFFF [read-only]

Default

Set in the factory

DH (Destination Address High)

Set or read the upper 32 bits of the 64-bit destination address.

Parameter range

0 - 0xFFFFFFFF

Default

0

DL (Destination Address Low)

Set or read the lower 32 bits of the 64-bit destination address.

Parameter range

0 - 0xFFFFFFFF

Default

TO (Transmit Options)

The bitfield that configures the transmit options for Transparent mode.

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

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

Parameter range

0 - 0xFF

Bit field:

Unused bits must be set to **0**. These bits may be logically OR'ed together:

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

Default

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

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

NI (Node Identifier)

The node identifier is a user-defined name or description of the device. Use this string with network discovery commands in order to easily identify devices on the network.

Use the ND (Network Discover) command with this string as an argument to filter network discovery results.

Use the DN (Discover Node) command with this string as an argument to resolve the 64-bit address of a node with a matching **NI** string.

Parameter range

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

Default

0x20 (an ASCII space character)

NT (Network Discovery Back-off)

Sets the maximum amount of time that a node receiving an **ND**, **DN**, or **FN** command waits before transmitting the response. For example, if **NT** is 13 seconds, the response is sent in a random amount of time between 0 and 13 seconds.

When a discovery is performed, the broadcast transmission includes the **NT** value to provide all remote devices with a response timeout. Remote devices wait a random time, less than **NT**, before sending their response to avoid collisions.

The **N?** command should be used to determine how long the actual discovery timeout will be based on current device configuration.

Parameter range

Default

N? (Network Discovery Timeout)

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

Parameter range

0x20 - 0xFFFF (x 100 ms) [read-only]

Default

N/A

NO (Network Discovery Options)

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

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

Bit field:

0x0 - 0x7 (bit field)

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

Parameter range

0x0 - 0x7 (bit field)

Default

0x0

Discovery commands

Network Discovery and corresponding discovery options.

DN (Discover Node)

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

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

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

1. The requesting node sets **DL** and **DH** to the address of the device with the matching **NI** string.
2. The requesting node returns **OK** (or ERROR).
3. If the requesting node returns **OK** (node found), it exits Command mode immediately with **DH**/**DL** set to the node that is found so that the next serial input is sent to the node designated by the **DN** parameter.
4. If the requesting node returns **ERROR**, (node not found), it remains in Command mode, allowing you to enter further commands.

When **DN** is sent as a local Local AT Command Request - 0x08:

1. The requesting node returns 0xFFFE followed by its 64-bit extended addresses in an Local AT Command Response - 0x88.
2. If there is no response from a module within (**N?*** 100) milliseconds or you do not specify a parameter (by leaving it blank), the requesting node returns an ERROR message.

Parameter range

20-byte ASCII string

Default

N/A

ND (Network Discover)

Discovers and reports all of the devices it finds on a network. If you send **ND** through a local or remote API frame, each network node returns a separate AT Command Response (0x88) or Remote Command Response (0x97) frame, respectively.

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

SH<CR> (4 bytes)

SL<CR> (4 bytes)

DB<CR> (Contains the detected signal strength of the response in negative dBm units)

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

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

STATUS<CR> (1 byte: reserved)

PROFILE_ID<CR> (2 bytes)

MANUFACTURER_ID<CR> (2 bytes)

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

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

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

The **ND** command accepts an **NI** (Node Identifier) as an argument. For more details, see Directed node discovery.

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

The **NT** setting determines the range of the random offset delay.

For more information about options that affect the behavior of the **ND** command Refer to NO (Network Discovery Options) for options which affect the behavior of the **ND** command.

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

Parameter range

20-byte printable ASCII string (optional)

Default

N/A

FN (Find Neighbors)

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

MY<CR> (always 0xFFFE)

SH<CR>

SL<CR>

NI<CR> (Variable length)

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

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

STATUS<CR> (1 Byte: Reserved)

PROFILE_ID<CR> (2 Bytes)

MANUFACTURER_ID<CR> (2 Bytes)

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

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

<CR>

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

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

FN accepts a NI (Node Identifier) as an argument.

Parameter range

0 to 20 ASCII characters (optional)

Default

N/A

Security commands

EE (Encryption Enable)

Parameter range

0 - 1

Parameter	Description
0	Encryption Disabled
1	Encryption Enabled

Default

0

KY (AES Encryption Key)

This command is write-only and cannot be read. If you attempt to read **KY**, the device returns an **OK** status.

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

Parameter range

Default

0

Secure Session commands

These are the AT commands that enable Secure Session.

SA (Secure Access)

The Secure Access Options bit-field defines the feature set(s) intended to be secure against unauthorized access. The XBee XR 900 RF Module should establish a secure session in order to access functionality defined by the feature set(s) on the local device.

A password must be set using the Secure Session Salt and Verifier before access is secured.

Parameter range

0 - 0x1F (up to 0xFFFF)

Bit field

Unused bits must be set to **0**. These bits may be logically OR'ed together:

Bit	Description
0	Reserved
1	Remote AT Commands When set to **1** and if a password has been set, the device will not respond to insecure Remote AT Command requests (API Frame 0x17) but still can send insecure Remote AT Commands.
2	Serial Data When set to **1**, the device will not emit any serial data that was sent insecurely. This functionality applies to devices that are configured for Transparent mode, but in this instance, only the SRP server would be **AP** = **0**, the client would still have to send the Secure Session Control - 0x2E via API mode. The server will also not emit any 0x90 or 0x91 frames when this bit is set.

Default

0

*S (Secure Session Salt)

The Secure Remote Password (SRP) Salt is a 32-bit number used to create an encrypted password for the XBee XR 900 RF Module. The ***S** command contains the salt value in the salt/verifier pair used for secure session authentication.

Parameter range

0-FFFFFFFF

Default

0

*V, *W, *X, *Y (Secure Session Verifier)

The secure session verifier is a 128-byte value used together with *S (Secure Session Salt) for secure session authentication. The ***V**, ***W**, ***X**, and ***Y** commands each contain 32 bytes of the secure session verifier: ***V** contains bytes 0 - 31, ***W** bytes 32 - 63, ***X** bytes 54 - 95, and ***Y** bytes 96 - 127.

Parameter range

Each command can be any 32-byte value

Default

0

Sleep settings commands

The following commands enable and configure the low power sleep modes of the device.

SM (Sleep Mode)

Sets or displays the sleep mode of the device.

Parameter range

Parameter	Description
0	No sleep (always awake)
1	Pin sleep
2	Unused
3	Unused
4	Asynchronous cyclic sleep
5	Asynchronous cyclic sleep with pin wakeup
6	Unused
7	Synchronous sleep support mode
8	Synchronous cyclic sleep

Default

0

SO (Sleep Options)

A bitfield that contains advanced sleep options that do not have dedicated AT commands.

Parameter range

0 - 0x53E

Bit field:

Unused bits must be set to **0**. These bits may be logically OR'ed together:

Bit	Description
0	Sleep coordinator; setting this bit causes a sleep compatible device to always act as sleep coordinator.
1	Non-sleep coordinator; setting this bit causes a device to never act as a sleep coordinator.
2	Enable API sleep status messages.
3	Disable early wake-up for missed syncs.
4	Enable node type equality (disables seniority based on device type).
5	Disable coordinator rapid sync deployment mode.
8	Always wake for **ST** time.
10	Enable sync sleep randomize I/O sampling.

Default

0x2

SN (Number of Sleep Periods)

Parameter range

1 - 0xFFFF

Default

1

SP (Cyclic Sleep Period)

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

For a node operating as an Indirect Messaging Coordinator, this command defines the amount of time that it will hold an indirect message for an end device. The coordinator will hold the message for (2.5 * **SP**).

Parameter range

0x0 - 0x15F900 (x 10 ms) (4 hours)

Default

0x12C (3 seconds)

ST (Cyclic Sleep Wake Time)

Sets or displays the wake time of the device.

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

For devices in synchronous sleep, the minimum wake time is a function of **MT**, **RN**, **NH**, and **NN**. If you increase these values such that **ST** is no longer big enough to get a message through the network during a wake cycle, then **ST** will be increased appropriately.

Parameter range

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

Default

0xD08 (3.3 seconds)

WH (Wake Host Delay)

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

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

Parameter range

0 - 0xFFFF (x 1 ms)

Default

0

Diagnostic commands -sleep status/timing

The following AT commands provide timing and status information for a synchronized cyclically sleeping network (**SM** = **7** or **8**).

SS (Sleep Status)

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

Bit	Description
0	This bit is true when the network is awake and able to receive transmissions.
1	This bit is true if the node currently acts as a network sleep coordinator.
2	This bit is true if the node ever receives a valid sync message after it powers on.
3	This bit is true if the node receives a sync message in the current wake cycle.
4	This bit is true if you alter the sleep settings on the device so that the node nominates itself and sends a sync message with the new settings at the beginning of the next wake cycle.

Bit	Description
5	This bit is true if you request that the node nominate itself as the sleep coordinator using the Commissioning Pushbutton or the **CB**2 command.
6	This bit is true if the node is currently in deployment mode.
All other bits	Reserved. Ignore all non-documented bits.

Parameter range

N/A

Default

N/A

OS (Operating Sleep Time)

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

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

Parameter range

N/A

Default

N/A

OW (Operating Wake Time)

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

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

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

Parameter range

N/A

Default

N/A

MS (Missed Sync Messages)

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

Parameter range

N/A

Default

N/A

SQ (Missed Sleep Sync Count)

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

Set the value to 0 to reset this value.

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

Parameter range

0 - 0xFFFF

Default

N/A

UART interface commands

BD (Interface Data Rate)

This command configures the serial interface baud rate for communication between the UART port of the device and the host.

The device interprets any value between 0x12C and 0x0EC400 as a custom baud rate. Custom baud rates are not guaranteed and the device attempts to find the closest achievable baud rate. After setting a non-standard baud rate, query **BD** to find the actual operating baud rate before applying changes.

Parameter range

Standard baud rates: 0x0 - 0x0A

Non-standard baud rates: 0x12C - 0x0EC400

Default

NB (Parity)

Set or read the serial parity settings for UART communications.

The device does not actually calculate and check the parity. It only interfaces with devices at the configured parity and stop bit settings for serial error detection.

Parameter range

0 - 2

Parameter	Description
0	No parity
1	Even parity
2	Odd parity

Default

0

SB (Stop Bits)

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

Parameter range

0 - 1

Parameter	Description
0	One stop bit
1	Two stop bits

Default

0

RO (Packetization Timeout)

Set or read the number of character times of inter-character silence required before transmission begins when operating in Transparent mode. A “character time” is the amount of time it takes to send a single ASCII character at the operating baud rate (**BD**).

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

The **RO** command only applies to Transparent mode, it does not apply to API mode.

Parameter range

0 - 0xFF (x character times)

Default

3

FT (Flow Control Threshold)

Set or display the flow control threshold.

The device de-asserts $\overline{\text{CTS}}$ when **FT** bytes are in the UART receive buffer. It re-asserts $\overline{\text{CTS}}$ when somewhat less than **FT** bytes are in the UART receive buffer. "Somewhat less than" allows for hysteresis so that $\overline{\text{CTS}}$ is not toggling rapidly when close to **FT** bytes are in the UART receive buffer.

Parameter range

0x47 - 0x3B9 bytes

Default

0x2F6

AP (API Enable)

Parameter range

0 - 2

Parameter	Description
0	API disabled (operate in Transparent mode)
1	API enabled
2	API enabled (with escaped control characters)

Default

0

AO (API Options)

Configure the serial output options for received API frames. This parameter is only applicable when the device is operating in API mode (**AP** = **1** or **2**).

Parameter range

0 - 2

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

Default

0

AZ (Extended API Options)

Optionally output additional ZCL messages that would normally be masked by the XBee application.

Use this when debugging FOTA updates by enabling client-side messages to be sent out of the serial port.

Parameter range

0x00 - 0x0A (bitfield)

Unused bits must be set to **0**. These bits may be logically OR'ed together:

Bit	Description
0	Reserved
1	Reserved
2	Reserved
3	Output Extended Modem Status (0x98) frames instead of Modem Status (0x8A) frames when a Secure Session status change occurs

Default

0

AT Command options

The following commands affect how Command mode operates.

CC (Command Character)

Sets or displays the character value used to break from data mode to Command mode. The command character must be sent three times in succession while observing the minimum guard time (**GT**) of silence before and after this sequence.

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

Parameter range

0 - 0xFF

Recommended: 0x20 - 0x7F (ASCII)

Default

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

CT (Command Mode Timeout)

Sets or displays the Command mode timeout parameter. If the local device enters Command mode and does not receive any valid AT commands within this time period, Command mode silently exits.

Parameter range

Default

0x64 (10 seconds)

CN (Exit Command Mode)

Executable command. **CN** immediately exits Command mode and applies pending changes.

Parameter range

N/A

Default

N/A

GT (Guard Times)

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

Command mode if a data stream in Transparent mode includes the **CC** character. For more information, see Enter Command mode.

Parameter range

0x2 - 0x68D (x 1 ms)

Default

0x3E8 (one second)

UART pin configuration commands

The following commands are related to pin configuration for the UART interface.

D6 (DIO6/RTS Configuration)

Sets or displays the DIO6/$\overline{\text{RTS}}$ configuration.

Parameter range

0, 1, 3 - 5

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

Default

0

D7 (DIO7/CTS Configuration)

Sets or displays the DIO7/$\overline{\text{CTS}}$ configuration.

Parameter range

0, 1, 3 - 7

Parameter	Description
0	Disabled
1	$\overline{\text{CTS}}$ flow control
2	N/A

Parameter	Description
3	Digital input
4	Digital output, low
5	Digital output, high
6	RS-485 Tx enable, low Tx (0 V on transmit, high when idle)
7	RS-485 Tx enable, high Tx (high on transmit, 0 V when idle)

Default

1

P3 (DIO13/DOUT Configuration)

Sets or displays the DIO13/UART_DOUT configuration.

Parameter range

0, 1, 3 - 5

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

Default

1

P4 (DIO14/DIN Configuration)

Sets or displays the DIO14/UART_DIN configuration.

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	UART DIN
2	N/A

Parameter	Description
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

SMT/MMT SPI interface commands

P5 (DIO15/SPI_MISO Configuration)

Sets or displays the DIO15/SPI_MISO configuration.

Parameter range

0, 1, 4, 5

Default

1

P6 (DIO16/SPI_MOSI Configuration)

Sets or displays the DIO16/SPI_MOSI configuration.

Parameter range

0, 1, 4, 5

Default

1

P7 (DIO17/SPI_SSEL Configuration)

Sets or displays the DIO17/SPI_$\overline{\text{SSEL}}$ configuration.

Parameter range

0, 1, 4, 5

Default

1

P8 (DIO18/SPI_CLK Configuration)

Sets or displays the DIO18/SPI_CLK configuration.

Parameter range

0, 1, 4, 5

Default

1

P9 (DIO19/SPI_ATTN Configuration)

Sets or displays the DIO19/SPI_ATTN configuration.

Parameter range

0, 1, 4, 5

Default

1

I/O settings commands

The following commands configure the various I/O lines available on the XBee XR 900 RF Module.

Note See Digital I/O support for physical I/O pin mapping for the supported module form factors.

D0 (DIO0/AD0/Commissioning Button Configuration)

Sets or displays the DIO0/AD0 configuration.

Parameter range

0 - 5

Default

1

Use **CB** to simulate Commissioning Pushbutton presses in software.

You can enable a physical commissioning pushbutton with D0 (DIO0/AD0/Commissioning Button Configuration).

Set the parameter value to the number of button presses that you want to simulate. For example, send **CB1** to perform the action of pressing the Commissioning Pushbutton once.

Parameter range

1, 2, 4

Parameter	Description
1	Keeps device awake for 30 seconds.
2	Nominate the node as the sleep coordinator for synchronous sleep networks.
4	Restore defaults (equivalent to sending an RE (Restore Defaults)).

Default

N/A

D1 (AD1/DIO1/TH_SPI_ATTN Configuration)

Sets or displays the DIO1/ADC1/TH_SPI_$\overline{\text{ATTN}}$ configuration.

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	SPI_$\overline{\text{ATTN}}$ for the through-hole device N/A for the surface-mount device
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D2 (DIO2/AD2/TH_SPI_CLK Configuration)

Sets or displays the DIO2/ADC2/TH_SPI_CLK configuration.

Parameter range

0, 2 - 6

Parameter	Description
0	Disabled
1	SPI_CLK for through-hole devices N/A for surface-mount devices
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high
6	Command mode indicator (high while in Command mode)

Default

0

D3 (DIO3/AD3/TH_SPI_SSEL Configuration)

Sets or displays the DIO3/ADC3/TH_SPI_$\overline{SSEL}$ configuration.

Parameter range

SMT/MMT: 0, 2 - 5

TH: 0 - 5

0, 2 - 5

Parameter	Description
0	Disabled
1	SPI_$\overline{SSEL}$ for the through-hole device N/A for surface-mount device
2	ADC
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D4 (DIO4/TH_SPI_MOSI Configuration)

Sets or displays the DIO4/TH_SPI_MOSI configuration.

Parameter range

0, 3 - 5

Parameter	Description
0	Disabled
1	SPI_MOSI for the through-hole device N/A for the surface-mount and micro device
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

D5 (DIO5/Associate Configuration)

Sets or displays the DIO5/ASSOCIATED_INDICATOR configuration.

Parameter range

0, 1, 3 - 5

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

Default

1

D8 (DIO8/DTR/SLP_Request Configuration)

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

Note If **D8** is configured as DTR/Sleep_Request (**1**), the line will be left floating while the device sleeps. Leaving **D8** set to **1** and the corresponding pin not connected to anything external to the device may result in higher sleep current draw.

Parameter range

0, 1, 3 - 5

Parameter	Description
0	Disabled
1	$\overline{\text{DTR}}$/Sleep_Request (used with pin sleep and cyclic sleep with pin wake)
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

D9 (DIO9/ON_SLEEP Configuration)

Sets or displays the DIO9/ON_$\overline{\text{SLEEP}}$ configuration.

Parameter range

0, 1, 3 - 6

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

Default

1

P0 (DIO10/RSSI/PWM0 Configuration)

Sets or displays the DIO10/RSSI/PWM0 configuration (pin 7).

When configured as RSSI PWM output, the device outputs a PWM signal with a duty cycle equivalent to the dBm of the received packet.

Use RP (RSSI PWM Timer) to configure the timeout.

When configured as PWM output (**2**): you can use **M0** to explicitly control the PWM0 output. When used with Analog I/O support, PWM0 corresponds with ADC0.

PWM output frequency

1 kHz

Parameter range

0 - 5

Parameter	Description
0	Disabled
1	RSSI PWM output
2	PWM0 output. M0 (PWM0 Duty Cycle) or I/O line passing control the value.
3	Digital input
4	Digital output, low
5	Digital output, high

Default

1

P1 (DIO11/PWM1 Configuration)

Sets or displays the DIO11 configuration (pin 8).

PWM output frequency

1 kHz

Parameter range

0, 2 - 5

Parameter	Description
0	Disabled
1	N/A
2	PWM1 output. M1 (PWM1 Duty Cycle) or I/O line passing control the value.
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

P2 (DIO12/TH_SPI_MISO Configuration)

Sets or displays the DIO12/TH_SPI_MISO configuration.

Parameter range

0, 3 - 5

Parameter	Description
0	Disabled
1	SPI_MISO for the through-hole device N/A for the surface-mount and micro device
2	N/A
3	Digital input
4	Digital output, low
5	Digital output, high

Default

0

PR (Pull-up/Down Resistor Enable)

The bit field that configures the internal pull-up resistor status for the I/O lines.

- If you set a **PR** bit to 1, it enables the pull-up/down resistor
- If you set a **PR** bit to 0, it specifies no internal pull-up/down resistor.

PR and **PD** only affect lines that are configured as digital inputs (**3**) or disabled (**0**).
The following table defines the bit-field map for **PR** and **PD** commands.

Bit	I/O line
0	DIO4
1	DIO3
2	DIO2
3	DIO1
4	DIO0
5	DIO6
6	DIO8
7	DIO14
8	DIO5
9	DIO9
10	DIO12
11	DIO10
12	DIO11
13	DIO7
14	DIO13
15	DIO15
16	DIO16
17	DIO17
18	DIO18
19	DIO19

Parameter range

0 - 0xFFFFF

Default

0xFFFF

PD (Pull Up/Down Direction)

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

See PR (Pull-up/Down Resistor Enable) for the bit mappings.

Parameter range

0 - 0xFFFFF

Default

0xFFFF

M0 (PWM0 Duty Cycle)

The duty cycle of the PWM0 line.

1. Enable PWM0 output (**P0** = **2**).
2. Change **M0** to the desired value.
3. Apply settings (use **CN** or **AC**).

The PWM period is 1 ms and there are 0x03FF (1023 decimal) steps within this period. When **M0** = **0** (0% PWM), 0x01FF (50% PWM), 0x03FF (100% PWM), and so forth.

Parameter range

0 - 0x3FF

Default

0

M1 (PWM1 Duty Cycle)

1. Enable PWM1 output (**P1** = 2).
2. Change **M1** to the desired value.
3. Apply settings (use **CN** or **AC**).

The PWM period is 1 ms and there are 0x03FF (1023 decimal) steps within this period. When **M0** = **0** (0% PWM), 0x01FF (50% PWM), 0x03FF (100% PWM), and so forth.

Parameter range

0 - 0x3FF

Default

0

LT (Associate LED Blink Time)

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

Parameter range

0, 0x14 - 0xFF (x 10 ms)

Default

0

RP (RSSI PWM Timer)

Parameter range

0 - 0xFF (x 100 ms)

Default

0x28 (four seconds)

I/O sampling commands

The following commands configure I/O sampling on an originating device. Any I/O sample generated by this device is sent to the address specified by **DH** and **DL**. You must configure at least one I/O line as an input or output for a sample to be generated.

AV (Analog Voltage Reference)

The analog voltage reference used for A/D sampling.

Parameter range

0 - 2

Parameter	Description
0	1.25 V reference
1	2.5 V reference
2	VDD reference

Default

0

IC (DIO Change Detect)

Bit field

Bit	I/O line	Micro pin
0	DIO0	31
1	DIO1	30

Bit	I/O line	Micro pin
2	DIO2	29
3	DIO3	28
4	DIO4	23
5	DIO5	26
6	DIO6	27
7	DIO7	24
8	DIO8	9
9	DIO9	25
10	DIO10	7
11	DIO11	8
12	DIO12	5
13	DIO13	3
14	DIO14	4
15	N/A	N/A

Parameter range

0 - 0x7FFF

Default

0

IF (Sleep Sample Rate)

Set or read the number of sleep cycles that must elapse between periodic I/O samples. This allows the firmware to take I/O samples only during some wake cycles. During those cycles, the firmware takes I/O samples at the rate specified by IR (Sample Rate). In addition, setting **IF** to zero allows I/O samples to occur before the device goes to sleep and occur thereafter every wake cycle specified by **IR**.

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

For more information, see the following commands:

- D0 (DIO0/AD0/Commissioning Button Configuration) through D9 (DIO9/ON_SLEEP Configuration)
- P0 (DIO10/RSSI/PWM0 Configuration) through P4 (DIO14/DIN Configuration)

Parameter range

0 - 0xFF (x 1 ms)

Default

1

IR (Sample Rate)

Parameter range

Default

0

IS (Immediate Sample)

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

Parameter range

N/A

Default

N/A

I/O line passing commands

The following AT commands allow I/O line passing to be enabled and configure the timeout that will be used for each I/O line. Line Passing requires the device to receive an I/O sample from the address specified by **IA** and have an I/O lines configured as outputs that corresponds to inputs in the received I/O sample.

IU (I/O Output Enable)

Enables or disables the display of I/O samples over the serial port (UART or SPI) if I/O passing is enabled (if IA is not the default value of all FFs).

Parameter range

0 - 1

Parameter	Description
0	Disables output of I/O samples over the serial port
1	Enables output of I/O samples over the serial port

Default

1

IA (I/O Input Address)

The source address of the device to which outputs are bound.

To disable I/O line passing, set all bytes to **0xFF**.

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

Parameter range

0 - 0xFFFF FFFF FFFF FFFF

Default

0xFFFFFFFFFFFFFFFF (I/O line passing disabled)

T0 (D0 Timeout)

Specifies how long pin D0 (DIO0/AD0/Commissioning Button Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T1 (D1 Output Timeout)

Specifies how long pin D1 (AD1/DIO1/TH_SPI_ATTN Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T2 (D2 Output Timeout)

Specifies how long pin D2 (DIO2/AD2/TH_SPI_CLK Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T3 (D3 Output Timeout)

Specifies how long pin D3 (DIO3/AD3/TH_SPI_SSEL Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T4 (D4 Output Timeout)

Specifies how long pin D4 (DIO4/TH_SPI_MOSI Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T5 (D5 Output Timeout)

Specifies how long pin D5 (DIO5/Associate Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T6 (D6 Output Timeout)

Specifies how long pin D6 (DIO6/RTS Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T7 (D7 Output Timeout)

Specifies how long pin D7 (DIO7/CTS Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T8 (D8 Timeout)

Specifies how long pin D8 (DIO8/DTR/SLP_Request Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

T9 (D9 Timeout)

Specifies how long pin D9 (DIO9/ON_SLEEP Configuration) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q0 (P0 Timeout)

Specifies how long P0 (DIO10/RSSI/PWM0 Configuration) (pin 7) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q1 (P1 Timeout)

Specifies how long P1 (DIO11/PWM1 Configuration) (pin 8) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q2 (P2 Timeout)

Specifies how long pin P2 (DIO12/TH_SPI_MISO Configuration) (pin 5) holds a given value before it reverts to configured value. If set to **0**, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q3 (P3 Timeout)

Specifies how long pin P3 (DIO13/DOUT Configuration) (pin 3) holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

Q4 (P4 Timeout)

Specifies how long pin P4 (DIO14/DIN Configuration) (pin 4) holds a given value before it reverts to configured value. If set to 0, there is no timeout.

Parameter range

0 - 0x1770 (x 100 ms)

Default

0

PT (PWM Output Timeout)

Specifies how long both PWM outputs (**P0**, **P1**) output a given PWM signal before it reverts to the configured value (**M0**/**M1**). If set to **0**, there is no timeout. This timeout only affects these pins when they are configured as PWM output and an analog sample for AD0/AD1 is received.

Parameter range

Default

0xFF

Diagnostic commands – firmware/hardware Information

The following read-only commands are diagnostics that provide more information about the device.

VR (Firmware Version)

Reads the firmware version on a device.

Parameter range

0xA000 - 0xA0FF

Default

Set in the firmware

Shows detailed version information including the application build date and time.

Parameter range

N/A

Default

N/A

VH (Bootloader Version)

Reads the bootloader version of the device.

Parameter range

N/A

Default

N/A

HV (Hardware Version)

Display the hardware version number and revision number of the device. The upper byte is the Hardware version and the lower byte is the hardware revision.

The hardware version distinguishes one radio type from another.

The hardware revision for a particular module can change for a variety of reasons and should not be used as the sole determination that a module's functionality has changed from previous revisions. The revision may change for various reasons including a new software version, a minor hardware modification, or even due to a label update. Furthermore, the firmware on a module may be upgraded or downgraded by a user thus making it different from the firmware version it was manufactured with. Thus the revision number is not a reliable indicator of the firmware version on the module. If an explanation for the revision number is not found in the release notes and it is a concern, contact Digi Support. In most cases the revision number does not relay any useful information to the consumer and it can be ignored.

Parameter range

0 - 0xFFFF [read-only]

Pre-defined **HV** values for XBee XR 900 RF Modules:

- 0x50 - XR 900 Micro (MMT) and Surface Mount (SMT)

Default

Set in the factory

%C (Hardware/Software Compatibility)

Specifies what firmware is compatible with this device's hardware. **%C** is compared to the to the "compatibility_number" field of the firmware configuration xml file. Firmware with a compatibility number lower than the value returned by **%C** cannot be loaded onto the board. If an invalid firmware is loaded, the device will not boot until a valid firmware is reloaded.

Parameter range

[read-only]

Default

N/A

%V (Voltage Supply Monitoring)

Reads the voltage on the Vcc pin in mV.

Parameter range

0 - 0xFFFF (in mV) [read only]

Default

N/A

DD (Device Type Identifier)

Stores the Digi device type identifier value. Use this value to differentiate between multiple types of devices (for example, sensors or lights).

This command can optionally be included in network discovery responses by setting bit 1 of **NO**.

Parameter range

0 - 0xFFFFFFFF

Default

0x160000

NP (Maximum Packet Payload Bytes)

Parameter range

0 - 0xFFFF [read-only]

Default

N/A

CK (Configuration CRC)

Reads the cyclic redundancy check (CRC) of the current AT command configuration settings to determine if the configuration has changed.

After a firmware update this command may return a different value.

Parameter range

0 - 0xFFFF [read-only]

Default

N/A

%P (Invoke Bootloader)

Forces the device to reset into the bootloader menu.

This command can only be issued locally.

Parameter range

N/A

Default

N/A

R? (Region Code)

Specifies the region of the module. A module can only run firmware for its intended region.

Parameter values

- 1: USA (900 MHz)

Parameter range

0-0xFFFF (read-only)

Default

N/A

TP (Temperature)

The current module temperature in degrees Celsius. The temperature is represented in two's complement, as shown in the following example:

1 °C = 0x0001 and -1°C = 0xFFFF

Parameter range

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

Default

N/A

D% (Manufacturing Date)

Reads the manufacturing date of the module.

The format of the value given for ATD% is 16 hex characters, i.e. ATD%DDDDDDHH000FFFFF, where DDDDDD represents the manufacturing date as the number of days since 1/1/1900: 1/1/2000=0x008EAC, etc. HH represents the hour based on a 24-hour clock. 000 is three empty hex digits. FFFFF represents the test fixture serial number as a decimal (this number is not converted to hex).

Parameter range

0 - 0xFFFFFFFFFFFFFFFF [read-only]

Default

N/A

Custom Default commands

The following commands are used to assign custom defaults to the device. Send RE (Restore Defaults) to restore custom defaults. You must send these commands as local AT commands, they cannot be set using Remote AT Command Request - 0x17.

%F (Set Custom Default)

When **%F** is received, the XBee XR 900 RF Module takes the next command received and applies it to both the current configuration and the custom defaults, so that when defaults are restored with

RE (Restore Defaults) the custom value is used.

Parameter range

N/A

Default

N/A

!C (Clear Custom Defaults)

Clears all custom defaults. This command does not change the current settings, but only changes the defaults so that RE (Restore Defaults) restores settings to the factory values.

Parameter range

N/A

Default

N/A

R1 (Restore Factory Defaults)

Restores factory defaults, ignoring any custom defaults set using %F (Set Custom Default).

Parameter range

N/A

Default

N/A

Operate in API mode

API mode overview

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

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

Use the AP command to set the operation mode

Use AP (API Enable) to specify the operation mode:

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

The API data frame structure differs depending on what mode you choose.

API frame format

An API frame consists of the following:

- Start delimeter
- Length
- Frame data
- Checksum

API operation (AP parameter = 1)

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

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

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

API operation with escaped characters (AP parameter = 2)

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

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

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

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

Start delimiter field

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

Escaped characters in API frames

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

The following data bytes need to be escaped:

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

Note Since software flow control is not implemented on this device, having hex values of 0x11 and 0x13 in the API frame isn't a reason to use AP = 2.
Since 0x7D is the escape character itself, the only value of AP = 2 is to distinguish a 0x7E in the data compared to the start delimiter 0x7E.

To escape a character:

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

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

Example: escape an API frame

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

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

You must escape the 0x13 byte:

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

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

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

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

Length field

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

Frame data

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

Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Data							Single byte

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

Multi-byte values are sent big-endian.

Calculate and verify checksums

To calculate the checksum of an API frame:

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

To verify the checksum of an API frame:

1. Add all bytes including the checksum; do not include the delimiter and length.
2. If the checksum is correct, the last two digits on the far right of the sum equal 0xFF.

Example

Consider the following sample data packet: **7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8**+

Byte(s)	Description
7E	Start delimiter
00 0A	Length bytes
01	API identifier
01	API frame ID
50 01	Destination address low
00	Option byte
48 65 6C 6C 6F	Data packet
B8	Checksum

To calculate the check sum you add all bytes of the packet, excluding the frame delimiter **7E** and the length (the second and third bytes):

7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8

Add these hex bytes:

01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F = 247

Now take the result of 0x247 and keep only the lowest 8 bits which in this example is 0xC4 (the two far right digits). Subtract 0x47 from 0xFF and you get 0x3B (0xFF - 0xC4 = 0x3B). 0x3B is the checksum for this data packet.

If an API data packet is composed with an incorrect checksum, the XBee XR 900 RF Module will consider the packet invalid and will ignore the data.

To verify the check sum of an API packet add all bytes including the checksum (do not include the delimiter and length) and if correct, the last two far right digits of the sum will equal FF.

01 + 01 + 50 + 01 + 00 + 48 + 65 + 6C + 6C + 6F + B8 = 2FF

Frame descriptions

The following sections describe the API frames.

64-bit Transmit Request - 0x00

Response frame: Transmit Status - 0x89

Description

This frame type is used to send serial payload data as an RF packet to a remote device with a corresponding 64-bit IEEE address.

Note This frame format is deprecated and should only be used by customers who require compatibility with legacy Digi RF products. For new designs, we encourage you to use Transmit Request - 0x10 to initiate API transmissions.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	64-bit Transmit Request - **0x00**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**Destination address**	Set to the 64-bit IEEE address of the destination device. If set to **0x000000000000FFFF**, the broadcast address is used.
13	8-bit	**Options**	A bit field of options that affect the outgoing transmission: ■ **Bit 0**: Disable MAC ACK [**0x01**] ■ Bit 1: Reserved (set to 0) ■ **Bit 2**: Send packet with Broadcast PAN ID [**0x04**] • 802.15.4 firmwares only **Note** Option values may be combined. Set all unused bits to 0.
14-n	variable	**RF data**	The serial data to be sent to the destination. Use **NP** to query the maximum payload size that can be supported based on current settings.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to a device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**".

The corresponding Transmit Status - 0x89 response with a matching Frame ID will indicate whether the transmission succeeded.

7E 00 11 **00 52 00 13 A2 00 12 34 56 78 00 54 78 44 61 74 61** 9E

Frame type	Frame ID	64-bit dest address	Tx options	RF data
0x00	0x52	0x0013A200 12345678	0x00	0x547844617461
Input	*Matches response*			*"TxData"*

64-bit broadcast

Sending a broadcast transmission of the serial data "**Broadcast**" and suppressing the corresponding response by setting Frame ID to **0**.

7E 00 14 **00 00 00 00 00 00 00 00 FF FF 00 42 72 6F 61 64 63 61 73 74** 6E

Frame type	Frame ID	64-bit dest address	Tx options	RF data
0x00	0x00	0x00000000 0000FFFF	0x00	0x42726F616463617374
Input	*Suppress response*	*Broadcast address*		*"Broadcast"*

Local AT Command Request - 0x08

Response frame: Local AT Command Response - 0x88

Description

This frame type is used to query or set command parameters on the local device. Any parameter that is set with this frame type will apply the change immediately. If you wish to queue multiple parameter changes and apply them later, use the Queue Local AT Command Request - 0x09 instead.

When querying parameter values, this frame behaves identically to Queue Local AT Command Request - 0x09: You can query parameter values by sending this frame with a command but no parameter value field–the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Local AT Command Response - 0x88 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x88 response is the same one set by the command in the 0x08 request frame.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Local AT Command Request - **0x08**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set the local command parameter

Set the **NI** string of the radio to "**End Device**".

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will indicate whether the parameter change succeeded.

7E 00 0E **08 A1 4E 49 45 6E 64 20 44 65 76 69 63 65** 38

Frame type	Frame ID	AT command	Parameter value
0x08	0xA1	0x4E49	0x456E6420446576696365
Request	*Matches response*	*"NI"*	*"End Device"*

Query local command parameter

Query the temperature of the module—**TP** command.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will return the temperature value.

7E 00 04 **08 17 54 50** 3C

Frame type	Frame ID	AT command	Parameter value
0x08	0x17	0x5450	(omitted)
Request	*Matches response*	*"TP"*	*Query the parameter*

Queue Local AT Command Request - 0x09

Response frame: Local AT Command Response - 0x88

Description

This frame type is used to query or set queued command parameters on the local device. In contrast to Local AT Command Request - 0x08, this frame queues new parameter values and does not apply them until you either:

- Issue a Local AT Command using the 0x08 frame
- Issue an **AC** command–queued or otherwise

When querying parameter values, this frame behaves identically to Local AT Command Request - 0x08: You can query parameter values by sending this frame with a command but no parameter value field–the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Local AT Command Response - 0x88 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x88 response is the same one set by the command in the 0x09 request frame.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Queue Local AT Command Request - **0x09**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register at a later time. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Queue setting local command parameter

Set the UART baud rate to 115200, but do not apply changes immediately.

The device will continue to operate at the current baud rate until the change is applied with a subsequent **AC** command.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID will indicate whether the parameter change succeeded.

7E 00 05 **09 53 42 44 07** 16

Frame type	Frame ID	AT command	Parameter value
0x09	0x53	0x4244	0x07
Request	*Matches response*	*"BD"*	*7 = 115200 baud*

Query local command parameter

Query the temperature of the module (**TP** command).

The corresponding 0x88 - Local AT Command Response frame with a matching Frame ID will return the temperature value.

7E 00 04 **09 17 54 50** 3B

Frame type	Frame ID	AT command	Parameter value
0x09	0x17	0x5450	(omitted)
Request	*Matches response*	*"TP"*	*Query the parameter*

Transmit Request - 0x10

Response frame: Extended Transmit Status - 0x8B

Description

This frame type is used to send payload data as an RF packet to a specific destination. This frame type is typically used for transmitting serial data to one or more remote devices.

The endpoints used for these data transmissions are defined by the **SE** and **DE** commands and the cluster ID defined by the **CI** command–excluding 802.15.4. To define the application-layer addressing fields on a per-packet basis, use the Explicit Addressing Command Request - 0x11 instead.

Query the **NP** command to read the maximum number of payload bytes that can be sent.

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Request - **0x10**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response frame. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**.
15	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to **0**–recommended–the value of **NH**specifies the broadcast radius.
17-n	variable	**Payload data**	Data to be sent to the destination device. Up to **NP** bytes per packet.

Offset	Size	Frame Field	Description
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

The available transmit options vary depending on the protocol being used. Bitfield options can be combined. Set all unused bits to **0**.

Sending a unicast message with MAC ACKs disabled is not intended to be a reliable form of communication, as no ACKs are produced by recipients.

Examples

Each example is written without escapes (**AP**=**1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to a device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**". Transmit options are set to **0**, which means the transmission will send using the options set by the **TO** command.

The corresponding Transmit Status - 0x89 response with a matching Frame ID will indicate whether the transmission succeeded.

7E 00 14 **10 52 00 13 A2 00 12 34 56 78 FF FE 00 00 54 78 44 61 74 61** 91

Frame type	Frame ID	64-bit dest	Bcast radius	Options	RF data
0x10	0x52	0x0013A200 12345678	0x00	0x00	0x547844617461
Request	*Matches response*	*Destination*	*N/A*	*Will use **TO***	*"TxData"*

64-bit broadcast

Sending a broadcast transmission of the serial data "**Broadcast**" to neighboring devices and suppressing the corresponding response by setting Frame ID to **0**.

7E 00 17 **10 00 00 00 00 00 00 00 FF FF FF FE 01 00 42 72 6F 61 64 63 61 73 74** 60

Frame type	Frame ID	64-bit dest	Bcast radius	Tx Options	RF data
0x10	0x00	0x00000000 0000FFFF	0x01	0x00	0x42726F616463617374
Request	*Suppress response*	*Broadcast address*	*Single hop broadcast*	*Will use **TO***	*"Broadcast"*

Explicit Addressing Command Request - 0x11

Response frame: Extended Transmit Status - 0x8B

Description

This frame type is used to send payload data as an RF packet to a specific destination using application-layer addressing fields. The behavior of this frame is similar to Transmit Request - 0x10, but with additional fields available for user-defined endpoints, cluster ID, and profile ID.

Query NP (Maximum Packet Payload Bytes) to read the maximum number of payload bytes that can be sent in.

64-bit addressing

- For broadcast transmissions, set the 64-bit destination address to **0x000000000000FFFF**
- For unicast transmissions, set the 64-bit address field to the address of the desired destination node

Reserved endpoints

For serial data transmissions, the **0xE8** endpoint should be used for both source and destination endpoints.

The active Digi endpoints are:

- **0xE8** - Digi Data endpoint
- **0xE6** - Digi Device Object (DDO) endpoint
- **0xE5** - XBee - Secure Session Server endpoint
- **0xE4** - XBee - Secure Session Client endpoint
- **0xE3** - XBee - Secure Session SRP authentication endpoint

Reserved cluster IDs

For serial data transmissions, the **0x0011** cluster ID should be used.

The following cluster IDs can be used on the **0xE8** data endpoint:

- **0x0011**- Transparent data cluster ID
- **0x0012** - Loopback cluster ID: The destination node echoes any transmitted packet back to the source device.
- **0x0023** - General Purpose Memory cluster ID: allows for reading and writing flash on the device.

Reserved profile IDs

The Digi profile ID of **0xC105** should be used when sending serial data between XBee devices.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Addressing Command Request - **0x11**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Broadcast address is **0x000000000000FFFF**.
15	8-bit	**Source Endpoint**	Source endpoint for the transmission. Serial data transmissions should use **0xE8**.
16	8-bit	**Destination Endpoint**	Destination endpoint for the transmission. Serial data transmissions should use **0xE8**.
17	16-bit	**Cluster ID**	The Cluster ID that the host uses in the transmission. Serial data transmissions should use **0x11**.
19	16-bit	**Profile ID**	The Profile ID that the host uses in the transmission. Serial data transmissions between XBee devices should use **0xC105**.
21	8-bit	**Broadcast radius**	Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions. If set to **0** (recommended), the value of **NH** specifies the broadcast radius.
23-n	variable	**Command data**	Data to be sent to the destination device. Up to **NP** bytes per packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Transmit options bit field

The available transmit options vary depending on the protocol being used. Bitfield options can be combined. Set all unused bits to **0**.

Sending a unicast message with MAC ACKs disabled is not intended to be a reliable form of communication, as no ACKs are produced by recipients.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

Sending a unicast transmission to an XBee device with the 64-bit address of **0013A20012345678** with the serial data "**TxData**". Transmit options are set to **0**, which means the transmission will send using the options set by the **TO** command. This transmission is identical to a Transmit Request - 0x10 using default settings.

The corresponding Extended Transmit Status - 0x8B response with a matching Frame ID will indicate whether the transmission succeeded.

7E 00 1A **11 87 00 13 A2 00 12 34 56 78 FF FE E8 E8 00 11 C1 05 00 00 54 78 44 61 74 61** B4

Frame type	Frame ID	64-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0x87	0x0013A200 12345678	0xE8	0xE8	0x0011	0xC105	0x00	0x00	0x5478446174 61
Explicit request	*Matches response*	*Destination*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Use **TO***	*"TxData"*

Loopback Packet

Sending a loopback transmission to an device with the 64-bit address of **0013A20012345678** using Cluster ID **0x0012**. To better understand the raw performance, retries and acknowledgements are disabled.

The corresponding Extended Transmit Status - 0x8B response with a matching Frame ID can be used to verify that the transmission was sent.

The destination will not emit a receive frame, instead it will return the transmission back to the sender. The source device will emit the receive frame–the frame type is determined by the value of **AO**–if the packet looped back successfully.

7E 00 1A **11 F8 00 13 A2 00 12 34 56 78 FF FE E8 E8 00 12 C1 05 00 01 54 78 44 61 74 61** 41

Frame type	Frame ID	64-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
0x11	0xF8	0x0013A200 12345678	0xE8	0xE8	0x0012	0xC105	0x00	0x01	0x5478446174 61

Frame type	Frame ID	64-bit dest	Source EP	Dest EP	Cluster	Profile	Bcast radius	Tx options	Command data
Explicit request	*Matches response*	*Destination*	*Digi data*	*Digi data*	*Data*	*Digi profile*	*N/A*	*Disable retries*	*"TxData"*

Remote AT Command Request - 0x17

Response frame: 0x97 - Remote AT Command Response

Description

This frame type is used to query or set AT command parameters on a remote device.

For parameter changes on the remote device to take effect, you must apply changes, either by setting the **Apply Changes** options bit, or by sending an **AC** command to the remote.

When querying parameter values you can query parameter values by sending this framewith a command but no parameter value field–the two-byte AT command is immediately followed by the frame checksum. When an AT command is queried, a Remote AT Command Response-0x97 frame is populated with the parameter value that is currently set on the device. The Frame ID of the 0x97 response is the same one set by the command in the 0x17 request frame.

XBee XR 900 RF Module firmwares support secured remote configuration through a Secure Session. Refer to Secured remote AT commands for information on how to secure your devices against unauthorized remote configuration.

Note Remote AT Command Requests should only be issued as unicast transmissions to avoid potential network disruption. Broadcasts are not acknowledged, so there is no guarantee all devices will receive the request. Responses are returned immediately by all receiving devices, which can cause congestion on a large network.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Request - **0x17**.
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to **0**, the device will not emit a response frame.
5	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. When using 16-bit addressing, set this field to **0xFFFFFFFFFFFFFFFF**.
15	8-bit	**Remote command options**	Bit field of options that apply to the remote AT command request: ■ **Bit 0**: Disable ACK [**0x01**] ■ **Bit 1**: Apply changes on remote [**0x02**]

Offset	Size	Frame Field	Description
			• If not set, changes will not applied until the device receives an **AC** command or a subsequent command change is received with this bit set ■ Bit 2: Reserved (set to 0) ■ Bit 3: Reserved (set to 0) ■ **Bit 4**: Send the remote command securely [**0x10**] • Requires a secure session be established with the destination Note Option values may be combined. Set all unused bits to 0.
16	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
18-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes–**AP** = **1**–and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set remote command parameter

Set the **NI** string of a device with the 64-bit address of **0013A20012345678** to "**Remote**" and apply the change immediately.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will indicate success.

7E 00 15 **17 27 00 13 A2 00 12 34 56 78 FF FE 02 4E 49 52 65 6D 6F 74 65** F6

Frame type	Frame ID	64-bit dest	Command options	AT command	Parameter value
0x17	0x27	0x0013A200 12345678	0x02	0x4E49	0x52656D6F7465
Request	*Matches response*		*Apply Change*	*"NI"*	*"Remote"*

Queue remote command parameter change

Change the PAN ID of a remote device so it can migrate to a new PAN, since this change would cause network disruption, the change is queued so that it can be made active later with a

subsequent **AC** command or written to flash with a queued **WR** command so the change will be active after a power cycle.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will indicate success.

7E 00 11 **17 68 00 13 A2 00 12 34 56 78 FF FE 00 49 44 04 51** D8

Frame type	Frame ID	64-bit dest	Command options	AT command	Parameter value
0x17	0x68	0x0013A200 12345678	0x00	0x4944	0x0451
Request	*Matches response*		*Queue Change*	*"ID"*	

Query remote command parameter

Query the temperature of a remote device–**TP** command.

The corresponding Remote AT Command Response- 0x97 with a matching Frame ID will return the temperature value.

7E 00 0F **17 FA 00 13 A2 00 12 34 56 78 FF FE 00 54 50** 84

Frame type	Frame ID	64-bit dest	Command options	AT command	Parameter value
0x17	0xFA	0x0013A200 12345678	0x00	0x5450	(omitted)
Request	*Matches response*		*N/A*	*"TP"*	*Query the parameter*

Secure Session Control - 0x2E

Response frame: 0xAE - Secure Session Response

Description

This frame type is used to control a secure session between a client and a server. If the remote node has a password set and you set the frame to login, this will establish a secure session that will allow secured messages to be passed between the server and client.

This frame is also used for clients to log out of an existing secure session.

Secure Sessions are end-to-end connections. If a login attempt is addressed to a broadcast address, the attempt will fail with an invalid value–status **0xA**–error.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Secure Session Control - **0x2E**
4	64-bit	**64-bit destination address**	Set to the 64-bit IEEE address of the destination device. Set to a broadcast address (**0x000000000000FFFF**) to affect all active incoming sessions.
12	8-bit	**Secure Session options**	Bit field of options that alter the session behavior: ■ **Bit 0**: Client-side control: • [**0x00**] = **Login** - Log in to a server as a client. ◦ If this bit is clear, the local device will act as a client and initiate SRP authentication with the target server. • [**0x01**] = **Logout** - Log out of an existing session as a client. ◦ If this bit is set, the local device will attempt to end an existing client-side session with the target server. ◦ When set, all other options, the timeout field, and password will be ignored. ■ **Bit 1**: Server-side control: • [**0x02**] = **Terminate Session** - If this bit is set, the server will end active incoming session(s).

Offset	Size	Frame Field	Description
			◦ The address field can be set to a specific node or the broadcast address can be used to end all incoming sessions. ◦ Use Extended Modem Status - 0x98 frames to manage multiple incoming sessions. ■ **Bit 2**: Timeout type: • [**0x00**] = **Fixed timeout** - The session terminates after the timeout period has elapsed. • [**0x04**] = **Inter-packet timeout** - The timeout is refreshed every time a secure transmission occurs between client and server. Note Option values may be combined. Set all unused bits to 0.
13	16-bit	**Timeout**	Timeout value for the secure session in units of ⅒th second. Accepts up to **0x4650** (30 minutes). A session with a timeout of **0x0000** is considered a yielding session. Yielding sessions will never time out, but if a server receives a request to start a session when it has the maximum incoming sessions, the oldest yielding session will be ended by the server to make room for the new session. Sessions with non-zero timeouts will never be ended in this way.
15-n	variable	**Password**	The password set on the remote node–up to 64 ASCII characters. Will be ignored if this frame is a logout or server termination frame.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte–between length and checksum.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Secure Session Client - Login with fixed timeout

A change is needed to be made on a device that is secured against unauthorized configuration changes. A gateway that is authorized to make the change logs into the remote node for 5 minutes as a client using the following frame:

The corresponding Secure Session Response - 0xAE will indicate whether the login attempt succeeded.

7E 00 14 **2E 00 13 A2 00 12 34 56 78 00 0B B8 50 41 53 53 57 4F 52 44** D2

Frame type	64-bit dest	Session options	Timeout	Password
0x2E	0x0013A200 12345678	0x00	0x02B8	0x50415353574F5244D2
Request		*Login Fixed*	*5 minutes*	*"PASSWORD"*

Secure Session Client - Login for streaming data

A large stream of data needs to be sent to a gateway that is secured against receiving unauthorized data. Because the data stream, and the gateway's ability to process the data is unknown, a Secure Session using a 60 second inter-packet timeout is established. The sending device logs into the gateway as a client using the following frame:

The corresponding Secure Session Response - 0xAE will indicate whether the login attempt succeeded.

7E 00 13 **2E 00 00 00 00 00 00 00 00 04 02 58 52 6F 73 33 62 75 64** D1

Frame type	64-bit dest	Session options	Timeout	Password
0x2E	0x00000000 00000000	0x04	0x0258	0x526F7333627564
Request	*Zigbee coordinator*	*Login Inter-packet*	*60 seconds*	*"Ros3bud"*

64-bit Receive Packet - 0x80

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11
- 64-bit Transmit Request - 0x00

Description

This frame type is emitted when a device configured with legacy API output–AO (API Options) = **2**–receives an RF data packet from a device configured to use 64-bit source addressing–**MY** = **0xFFFE**.

Note This frame format is deprecated and should only be used by customers who require compatibility with legacy Digi RF products. For new designs, we encourage you to use Receive Packet - 0x90 for reception of API transmissions.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	64-bit Receive Packet - **0x80**
4	64-bit	**64-bit source address**	The sender's 64-bit IEEE address.
12	8-bit	**RSSI**	Received Signal Strength Indicator. The Hexadecimal equivalent of (-dBm) value. For example if RX signal strength is -40 dBm, then 0x28 (40 decimal) is returned.
13	8-bit	**Options**	Bit field of options that apply to the received message: ■ Bit 0: Reserved ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] ■ **Bit 2**: 802.15.4 only - Packet was broadcast across all PANs [**0x04**] Note Option values may be combined.

Offset	Size	Frame Field	Description
14-n	variable	**RF data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

64-bit unicast

A device with the 64-bit address of **0013A20087654321** sent a unicast transmission to a specific device with the payload of "**TxData**". The following frame is emitted if the destination is configured with **AO** = **2**.

```
7E 00 11 80 00 13 A2 00 12 34 56 78 5E 01 54 78 44 61 74 61 11
```

Frame type	64-bit source	RSSI	Rx options	Received data
0x80	0x0013A200 87654321	0x5E	0x01	0x547844617461
Output		*-94 dBm*	*ACK was sent*	*"TxData"*

Local AT Command Response - 0x88

Request frames:

- Local AT Command Request - 0x08
- Queue Local AT Command Request - 0x09

Description

This frame type is emitted in response to a local AT Command request. Some commands send back multiple response frames; for example, ND (Network Discover). Refer to individual AT command descriptions for details on API response behavior.

This frame is only emitted if the Frame ID in the request is non-zero.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.

Offset	Size	Frame Field	Description
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Local AT Command Response - **0x88**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7	8-bit	**Command status**	Status code for the host's request: **0** = OK **1** = ERROR **2** = Invalid command **3** = Invalid parameter
8-n	variable	**Command data (optional)**	If the host requested a command parameter change, this field will be omitted. If the host queried a command by omitting the parameter value in the request, this field will return the value currently set on the device.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set local command parameter

Host set the NI string of the local device to "**End Device**" using a 0x08 request frame.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID is emitted as a response:

```
7E 00 05 88 01 4E 49 00 DF
```

Frame type	Frame ID	AT command	Command Status	Command data
0x88	0xA1	0x4E49	0x00	(omitted)
Response	*Matches request*	*"NI"*	*Success*	*Parameter changes return no data*

Query local command parameter

Host queries the temperature of the local device–TP command–using a 0x08 request frame.

The corresponding Local AT Command Response - 0x88 with a matching Frame ID is emitted with the temperature value as a response:

```
7E 00 07 88 01 54 50 00 FF FE D5
```

Frame type	Frame ID	AT command	Command Status	Command data
0x88	0x17	0x5450	0x00	0xFFFE
Response	*Matches request*	*"TP"*	*Success*	*-2 °C*

Transmit Status - 0x89

Request frames:

- 64-bit Transmit Request - 0x00

Description

This frame type is emitted when a transmit request completes. The status field of this frame indicates whether the request succeeded or failed and the reason.

This frame is only emitted if the Frame ID in the request is non-zero.

Note This frame format is deprecated and should only be used by customers who require compatibility with legacy Digi RF products.

Note Broadcast transmissions are not acknowledged and always return a status of **0x00**, even if the delivery failed.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x89**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = MAC ACK failure **0x02** = LBT failure (couldn't find a clear channel for transmission) **0x03** = Message purged (no spectrum available) **0x20** = Invalid cluster ID specified **0x21** = Network ACK failure **0x25** = Route not found **0x31** = Internal resource error

Offset	Size	Frame Field	Description
			0x32 = Resource error lack of free buffers, timers, etc. **0x34** = No Secure Session connection **0x35** = Secure Session Encryption failure **0x74** = Data payload too large **0x75** = Indirect message unrequested
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Example

This example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, only the highlighted bytes of the frame are documented.

Successful transmission

Host sent a unicast transmission to a remote device using a 64-bit Transmit Request - 0x00 frame.

The corresponding 0x89 Transmit Status with a matching Frame ID is emitted as a response to the request:

7E 00 03 **89 52 00** 24

Frame type	Frame ID	Delivery status
0x89	0x52	0x00
Response	*Matches request*	*Success*

Modem Status - 0x8A

Description

This frame type is emitted in response to specific conditions. The status field of this frame indicates the device behavior.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Modem Status - **0x8A**
4	8-bit	**Modem status**	Complete list of modem statuses: **0x00** = Hardware reset or power up **0x01** = Watchdog timer reset **0x0B** = Network woke up **0x0C** = Network went to sleep **0x0D** = Voltage supply limit exceeded–see **Over-voltage detection** in the *XBee 3 RF Module Hardware Reference Manual*. **0x3B** = Secure session successfully established **0x3C** = Secure session ended **0x3D** = Secure session authentication failed **0x33** = BLE Disconnect
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Modem status codes

Statuses for specific modem types are listed here.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Boot status

When a device powers up, it returns the following API frame:

```
7E 00 02 8A 00 75
```

Frame type	Modem Status
0x8A	0x00
Status	*Hardware Reset*

Extended Transmit Status - 0x8B

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a network transmission request completes. The status field of this frame indicates whether the request succeeded or failed and the reason. This frame type provides additional networking details about the transmission.

This frame is only emitted if the Frame ID in the request is non-zero.

Note Broadcast transmissions are not acknowledged and always return a status of **0x00**, even if the delivery failed.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Transmit Status - **0x8B**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
7	8-bit	**Transmit retry count**	The number of application transmission retries that occur.
7	8-bit	**Reserved**	Unused.
8	8-bit	**Delivery status**	Complete list of delivery statuses: **0x00** = Success **0x01** = MAC ACK failure **0x02** = LBT failure (could not find a clear channel for transmission) **0x03** = Message purged (no spectrum available) **0x20** = Invalid cluster ID specified **0x21** = Network ACK failure **0x25** = Route not found **0x31** = Internal resource error **0x32** = Resource error lack of free buffers, timers, etc. **0x34** = No Secure Session connection **0x35** = Secure Session Encryption failure

Offset	Size	Frame Field	Description
			0x74 = Data payload too large **0x75** = Indirect message unrequested Refer to the tables below for a filtered list of status codes that are appropriate for specific devices.
9	8-bit	**Discovery status**	Complete list of delivery statuses: **0x00** = No discovery overhead
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Route Information - 0x8D

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type contains the DigiMesh routing information for a remote device on the network. This route information can be used to diagnose marginal links between devices across multiple hops.

This frame type is emitted in response to a DigiMesh unicast transmission request which has Trace Routing or NACK enabled. See Trace route option and NACK messages for more information.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Route Information - **0x8D**
4	8-bit	**Source event**	Event that caused the route information to be generated: **0x11** = NACK **0x12** = Trace route
5	8-bit	**Data length**	The number of bytes that follow, excluding the checksum. If the length increases, new items have been added to the end of the list for future revisions.
6	32-bit	**Timestamp**	System timer value on the node generating the Route Information Packet.The timestamp is in microseconds. Only use this value for relative time measurements because the time stamp count restarts approximately every hour.
10	8-bit	**ACK timeout count**	The number of MAC ACK timeouts that occur.
11	8-bit	**TX blocked count**	The number of times the transmission was blocked due to reception in progress.
12	8-bit	**Reserved**	Not used.
14	64-bit	**Destination address**	The 64-bit IEEE address of the final destination node of this network-level transmission.
21	64-bit	**Source**	The 64-bit IEEE address of the source node of this network-

Offset	Size	Frame Field	Description
		address	level transmission.
29	64-bit	**Responder address**	The 64-bit IEEE address of the node that generates this Route Information packet after it sends (or attempts to send) the data packet to the next hop (the Receiver node).
37	64-bit	**Receiver address**	The 64-bit IEEE address of the node that the device sends (or attempts to send) the data packet.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Routing information

The following example represents a possible Route Information Packet. A device emits this frame when it performs a trace route enabled transmission from one device–serial number 0x0013A200 4052AAAA–to another–serial number 0x0013A200 4052DDDD–across a DigiMesh network.

This particular frame indicates that the network successfully forwards the transmission from one device–serial number 0x0013A200 4052BBBB–to another device–serial number 0x0013A200 4052CCCC.

7E 00 2A **8D 12 27 6B EB CA 93 00 00 00 00 13 A2 00 40 52 DD DD 00 13 A2 00 40 52 AA AA 00 13 A2 00 40 52 BB BB 00 13 A2 00 40 52 CC CC** 4E

Frame type	Source event	Data length	Timestamp	ACK timeout	TX Blocked	Reserved	Dest address	Source address	Responder address	Receiver address
0x8D	0x12	0x27	0x6BEBCA93	0x00	0x00	0x00	0x0013A2004052DDDD	0x0013A2004052AAAA	0x0013A2004052BBBB	0x0013A2004052CCCC
Route	*Trace Route*		*~30 minutes*	*No retries this hop*	*No error*	*N/A*	*Destination*	*Source*	*Node that sent this information*	*Next hop*

Aggregate Addressing Update - 0x8E

Description

This frame type is emitted on devices that update it addressing information in response to a network aggregator issuing an addressing update. A network aggregator is defined by a device on the network who has had the AG (Aggregator Support) command issued. A device on the network who's current **DH** and **DL** matches the address provided in the **AG** command request will update **DH** and **DL** and emit this frame.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Aggregate Addressing Update - **0x8E**
4	8-bit	**Reserved**	Reserved for future functionality. This field returns 0.
5	64-bit	**New address**	Address to which **DH** and **DL** are being set.
13	64-bit	**Old address**	Address to which **DH** and **DL** were previously set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Aggregate address update

In the following example, a device with destination address (**DH**/**DL**) of 0x0013A200 4052AAAA updates its destination address to 0x0013A200 4052BBBB.

7E 00 12 **8E 00 00 13 A2 00 40 52 BB BB 00 13 A2 00 40 52 AA AA** 19

Frame type	Reserved	New address	Old address
0x8E	0x00	0x0013A200 4052BBBB	0x0013A200 4052AAAA
Update	*N/A*	*What **DH/DL** is now set to*	*What **DH/DL** was set to*

Receive Packet - 0x90

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a device configured with standard API output–AO (API Options) = **0**–receives an RF data packet.

Typically this frame is emitted as a result of a device on the network sending serial data using the Transmit Request - 0x10 or Explicit Addressing Command Request - 0x11 addressed either as a broadcast or unicast transmission.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Receive Packet - **0x90**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
15-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Explicit Receive Indicator - 0x91

Request frames:

- Transmit Request - 0x10
- Explicit Addressing Command Request - 0x11

Description

This frame type is emitted when a device configured with explicit API output–AO (API Options) bit1 set–receives a packet.

Typically this frame is emitted as a result of a device on the network sending serial data using the Transmit Request - 0x10 or Explicit Addressing Command Request - 0x11 addressed either as a broadcast or unicast transmission.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Explicit Receive Indicator - **0x91**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
14	8-bit	**Source endpoint**	Endpoint of the source that initiated transmission.
15	8-bit	**Destination endpoint**	Endpoint of the destination that the message is addressed to.
16	16-bit	**Cluster ID**	The Cluster ID that the frame is addressed to.
18	16-bit	**Profile ID**	The Profile ID that the fame is addressed to.
21-n	variable	**Received data**	The RF payload data that the device receives.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

I/O Sample Indicator - 0x92

Description

This frame type is emitted when a device configured with standard API output–AO (API Options) = **0**–receives an I/O sample frame from a remote device. Only devices running in API mode will send I/O samples out the serial port.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	I/O Sample Indicator - **0x92**
4	64-bit	**64-bit source address**	The sender's 64-bit IEEE address.
12	16-bit	**Reserved**	Unused, but typically **0XFFFE**.
14	8-bit	**Receive options**	Bit field of options that apply to the received message: ■ **Bit 0**: Packet was Acknowledged [**0x01**] ■ **Bit 1**: Packet was sent as a broadcast [**0x02**] Note Option values may be combined.
15	8-bit	**Number of samples**	The number of sample sets included in the payload. This field typically reports 1 sample.
16	16-bit	**Digital sample mask**	Bit field that indicates which I/O lines on the remote are configured as digital inputs or outputs, if any: **bit 0**: DIO0 **bit 1**: DIO1 **bit 2**: DIO2 **bit 3**: DIO3 **bit 4**: DIO4 **bit 5**: DIO5 **bit 6**: DIO6 **bit 7**: DIO7 **bit 8**: DIO8 **bit 9**: DIO9 **bit 10**: DIO10 **bit 11**: DIO11

Offset	Size	Frame Field	Description
			bit 12: DIO12 **bit 13**: **bit 14**: **bit 15**: N/A For example, a digital channel mask of **0x002F** means DIO **0**, **1**, **2**, **3**, and **5** are enabled as digital I/O.
18	8-bit	**Analog sample mask**	Bit field that indicates which I/O lines on the remote are configured as analog input, if any: **bit 0**: AD0 **bit 1**: AD1 **bit 2**: AD2 **bit 3**: AD3 **bit 7**: Supply Voltage (enabled with **V+** command)
19	16-bit	**Digital samples (if included)**	If the sample set includes any digital I/O lines (**Digital channel mask** > **0**), this field contain samples for all enabled digital I/O lines. If no digital lines are configured as inputs or outputs, this field will be omitted. DIO lines that do not have sampling enabled return 0. Bits in this field are arranged the same as they are in the Digital channel mask field.
22	16-bit variable	**Analog samples (if included)**	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 16-bit value indicating the ADC measurement of that input. Analog samples are ordered sequentially from AD0 to AD3.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

I/O sample

A device with the 64-bit address of **0013A20012345678** is configured to periodically send I/O sample data to a particular device. The device is configured with DIO3, DIO4, and DIO5 configured as digital I/O, and AD1 and AD2 configured as an analog input.

7E 00 16 **92 00 13 A2 00 12 34 56 78 FF FE C1 01 00 38 06 00 28 02 25 00 F8** E8

Frame type	64-bit source	Reserved	Rx options	Num samples	Digital channel mask	Analog channel mask	Digital samples	Analog sample 1	Analog sample 2
0x92	0x0013A2	0x87AC	0xC1	0x01	0x0038	0x06	0x0028	0x0225	0x00F8

Frame type	64-bit source	Reserved	Rx options	Num samples	Digital channel mask	Analog channel mask	Digital samples	Analog sample 1	Analog sample 2
	00 12345678								
Sample		*Unused*	*ACK was sent in mesh network*	*Single sample (typical)*	*b'00 **111** 000 DIO3, DIO4, and DIO5 enabled*	*b'0**11**0 AD1 and AD2 enabled*	*b'00 **101**000 DIO3 and DIO5 are HIGH; DI04 is LOW*	*AD1 data*	*AD2 data*

Node Identification Indicator - 0x95

Description

This frame type is emitted when a node identification broadcast is received. The node identification indicator contains information about the identifying device, such as address, identifier string (**NI**), and other relevant data.

A node identifies itself to the network under these conditions:

- The commissioning button is pressed once.
- A **CB 1** command is issued.

See ND (Network Discover) for information on the payload formatting.

See NO (Network Discovery Options) for configuration options that modify the output of this frame.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Node Identification Indicator - **0x95**
4	64-bit	**64-bit source address**	The sender's 64-bit address.
17	64-bit	**64-bit remote address**	The 64-bit address of the device that sent the Node Identification.
25	variable (2-byte minimum)	**Node identification string**	Node identification string on the remote device set by NI (Node Identifier). The identification string is terminated with a NULL byte (0x00).
29+NI	8-bit	**Network device type**	What type of network device the remote identifies as: 0 = Coordinator 1 = Router 2 = End Device
31+NI	16-bit	**Digi Profile ID**	The Digi application Profile ID–**0xC105**.
33+NI	16-bit	**Digi Manufacturer ID**	The Digi Manufacturer ID–**0x101E**.
35+NI	32-bit	**Device type identifier (optional)**	The user-defined device type on the remote device set by DD (Device Type Identifier). Only included if the receiving device has the

Offset	Size	Frame Field	Description
			appropriate NO (Network Discovery Options) bit set.
EOF-1	8-bit	**RSSI (optional)**	The RSSI of the last hop that relayed the message. Only included if the receiving device has the appropriate NO (Network Discovery Options) bit set.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte—between length and checksum.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Identify remote device

A technician is replacing a device in the field and needs to have the its entry removed from a cloud server's database. The technician pushes the commissioning button on the old device once to send an identification broadcast. The server can use the broadcast to identify which device is being replaced and perform the necessary action.

When the node identification broadcast is sent, every device that receives the message will flash the association LED and emit the following information frame:

7E 00 27 **95 00 13 A2 00 12 34 56 78 FF FE C2 FF FE 00 13 A2 00 12 34 56 78 4C 48 37 35 00 FF FE 01 01 C1 05 10 1E 00 14 00 08** 0D

Frame type	64-bit source	16-bit source	Options	16-bit remote	64-bit remote	NI String	Parent	Device type	Event	Profile ID	MFG ID
0x95	0x0013A200 12345678	0xFFFE	0xC2	0xFFFE	0x0013A200 12345678	0x4C483735 00	0xFFFE	0x01	0x01	0xC105	0x101E
Identification		*Unknown*	*DigiMesh broadcast*	*Unknown*		*"LH75" + null*	*Unknown*	*Router*	*Button press*	*Digi*	*Digi*

Remote AT Command Response- 0x97

Request frame: Remote AT Command Request - 0x17

Description

This frame type is emitted in response to a Remote AT Command Request - 0x17. Some commands send back multiple response frames; for example, the **ND** command. Refer to individual AT command descriptions for details on API response behavior.

This frame is only emitted if the Frame ID in the request is non-zero.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Remote AT Command Response - **0x97**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a prior request.
5	64-bit	**64-bit source address**	The sender's 64-bit address.
15	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
17	8-bit	**Command status**	Status code for the host's request: **0x00** = OK **0x01** = ERROR **0x02** = Invalid command **0x03** = Invalid parameter **0x04** = Transmission failure **0x0C** = Encryption error
18-n	variable	**Parameter value (optional)**	If the host requested a command parameter change, this field will be omitted. If the host queried a command by omitting the parameter value in the request, this field will return the value currently set on the device.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Set remote command parameter

Host set the **NI** string of a remote device to "**Remote**" using a Remote AT Command Request - 0x17.

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted as a response:

```
7E 00 0F 97 27 00 13 A2 00 12 34 56 78 12 7E 4E 49 00 51
```

Frame type	Frame ID	64-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x4E49	0x00	(omitted)
Response	*Matches request*		*"NI"*	*Success*	*Parameter changes return no data*

Transmission failure

Host queued the the PAN ID change of a remote device using a Remote AT Command Request - 0x17. Due to existing network congestion, the host will retry any failed attempts.

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted as a response:

```
7E 00 0F 97 27 00 13 A2 00 12 34 56 78 FF FE 49 44 04 EA
```

Frame type	Frame ID	64-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x4944	0x04	(omitted)
Response	*Matches request*		*"ID"*	*Transmission failure*	*Parameter changes return no data*

Query remote command parameter

Query the temperature of a remote device–TP (Temperature).

The corresponding 0x97 Remote AT Command Response with a matching Frame ID is emitted with the temperature value as a response:

```
7E 00 11 97 27 00 13 A2 00 12 34 56 78 FF FE 54 50 00 00 2F A8
```

Frame type	Frame ID	64-bit source	AT command	Command Status	Command data
0x97	0x27	0x0013A200 12345678	0x4944	0x00	0x002F
Response	*Matches request*		*"TP"*	*Success*	*+47 °C*

Extended Modem Status - 0x98

Description

This frame type can be used to manage and troubleshoot Secure Session connections. To enable extended modes statuses set AZ (Extended API Options) bit 3.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Extended Modem Status - **0x98**
4	8-bit	**Status code**	Refer to the tables below for appropriate status codes
n	variable	**Status data (optional)**	Additional fields that provide information about the status
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Secure Session status codes

When AZ (Extended API Options) is configured to output extended secure session statuses, whenever Secure Session API Frames are emitted, the extended modem status will provide additional details about the event.

Status code	Description	Status data	Size	Description
0x3B	A Secure Session was established with this node	Address	64-bit	The address of the client in the session.
		Options	8-bit	Session options set by the client.
		Timeout	16-bit	Session timeout set by the client.
0x3C	A Secure Session ended	Address	64-bit	The address of the other node in this session.
		Reason	8-bit	The reason the session was ended: **0x00** - Session was terminated by the other node **0x01** - Session Timed out **0x02** - Received a transmission with

Status code	Description	Status data	Size	Description
				an invalid encryption counter **0x03** - Encryption counter overflow - the maximum number of transmissions for a single session has been reached **0x04** - Remote node out of memory
0x3D	A Secure Session authentication attempt failed	Address	64-bit	Address of the client node.
		Error	8-bit	Error that caused the authentication to fail. See Secure Session Response - 0xAE for a list of error statuses.

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Secure Session established

A device has established a secure session with the local node that has AZ (Extended API Options) configured to output extended secure session information. The following frame is emitted that announces the secure session establishment.

7E 00 0D **98 3B 00 13 A2 00 12 34 56 78 00 46 50** CD

Frame type	Status code	Status data
0x98	0x3B	■ 0x0013A20012345678 ■ 0x00 ■ 0x4650
Extended status	*Secure Session established*	■ Address ■ Options ■ Timeout (30 min)

Secure Session Response - 0xAE

Request frame: Secure Session Control - 0x2E

Description

This frame type is output as a response to a Secure Session Control - 0x2E attempt. It indicates whether the Secure Session operation was successful or not.

Format

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

Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Secure Session Response - **0xAE**
4	8-bit	Response type	The type of response to correlate with the preceding request: **0x00** - Login response **0x01** - Logout response **0x02** - Server Termination
5	64-bit	**64-bit source address**	The 64-bit IEEE address of the responding device.
13	8-bit	**Status**	Typical statuses: **0x00** - SRP operation was successful **0x01** - Invalid Password - SRP verification failed due to mismatched **M1** and **M2** values **0x02** - Session request was rejected as there are too many active sessions on the server already **0x03** - Session options or timeout are invalid **0x05** - Timed out waiting for the other node to respond **0x06** - Could not allocate memory needed for authentication **0x07** - A request to terminate a session in progress has been made **0x08** - There is no password set on the server **0x09** - There was no initial response from the server **0x0A** - Data within the frame is not valid or formatted incorrectly Atypical statuses: **0x80** - Server received a packet that was intended for a client or vice-versa **0x81** - Received an SRP packet we were not expecting **0x82** - Offset for a split value (A/B) came out of order **0x83** - Unrecognized or invalid SRP frame type

Offset	Size	Frame Field	Description
			0x84 - Authentication protocol version is not supported **0xFF** - An undefined error occurred
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Examples

Each example is written without escapes (**AP** = **1**) and all bytes are represented in hex format. For brevity, the start delimiter, length, and checksum fields have been excluded.

Secure Session Login attempt

A client attempted to log into a Secure Session server.

The following Secure Session Response - 0xAE is emitted as a response:

7E 00 0B **AE 00 00 13 A2 00 12 34 56 78 00** 88

Frame type	Response type	64-bit source	Status
0x2E	0x00	0x0013A200 12345678	0x00
Response	*Login*		*success*

General Purpose Flash Memory

General Purpose Flash Memory

XBee XR 900 RF Module provides blocks of flash memory that an application can read and write to. This memory provides a non-volatile data storage area that an application uses for many purposes. Some common uses of this data storage include:

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

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

The usage of sleep during a GPM update is unsupported. Users are recommended to turn off sleep on the target device, perform the update, and then switch sleep back on to avoid data loss and increase the update speed.

Access General Purpose Flash Memory

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

To issue a GPM command, format the payload of an explicit API frame as follows:

Byte offset in payload	Number of bytes	Field name	General field description
0	1	GPM_CMD_ID	Specific GPM commands are described in detail in the topics that follow.
1	1	GPM_OPTIONS	Command-specific options.
2	2*	GPM_BLOCK_NUM	The block number addressed in the GPM.
4	2*	GPM_START_INDEX	The byte index within the addressed GPM block.
6	2*	GPM_NUM_BYTES	The number of bytes in the GPM_DATA field, or in the case of a READ, the number of bytes requested.
8	varies	GPM_DATA	
* Specify multi-byte parameters with big-endian byte ordering.			

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

requesting node outputs a GPM response as an explicit API RX indicator frame (assuming it has API mode enabled).

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

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

General Purpose Flash Memory commands

This section provides information about commands that interact with GPM:

PLATFORM_INFO_REQUEST (0x00)

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

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

PLATFORM_INFO (0x80)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to PLATFORM_INFO (0x80).
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	Indicates the number of GPM blocks available.
GPM_START_INDEX	Indicates the size, in bytes, of a GPM block.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

ERASE (0x01)

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

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

ERASE_RESPONSE (0x81)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to ERASE_RESPONSE (0x81).
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0.
GPM_DATA	No data bytes are specified for this command.

Example

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

WRITE (0x02) and ERASE_THEN_WRITE (0x03)

The WRITE command writes the specified bytes to the GPM location specified. Before writing bytes to a GPM block it is important that the bytes have been erased previously. The ERASE_THEN_WRITE command performs an ERASE of the entire GPM block specified with the GPM_BLOCK_NUM field prior to doing a WRITE. WRITE commands cannot index past the end of a GPM block boundary.

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE (0x02) or ERASE_THEN_WRITE (0x03).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be written.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be written.
GPM_NUM_BYTES	Set to the number of bytes specified in the GPM_DATA field.

Field name	Command-specific description
	Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. The maximum payload size can be queried with the **NP** command.
GPM_DATA	The data to be written.

WRITE _RESPONSE (0x82) and ERASE_THEN_WRITE_ RESPONSE (0x83)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to WRITE_RESPONSE (0x82) or ERASE_THEN_ WRITE_RESPONSE (0x83)
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	Matches the parameter passed in the request frame
GPM_START_INDEX	Matches the parameter passed in the request frame
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field. For this command, this field will be set to 0
GPM_DATA	No data bytes are specified for these commands

Example

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

READ (0x04)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ (0x04).
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be read.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be read.
GPM_NUM_BYTES	Set to the number of data bytes to be read. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. The number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the device. You can query the maximum payload size with the **NP** AT command.
GPM_DATA	No data bytes should be specified for this command.

READ_RESPONSE (0x84)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to READ_RESPONSE (0x84).
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	Matches the parameter passed in the request frame.
GPM_NUM_BYTES	The number of bytes in the GPM_DATA field.
GPM_DATA	The bytes read from the GPM block specified.

Example

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

7E 0029 91 0013A200407402AC FFFE E6 E6 0023 C105 C1 84 00 0016 0000 000F 010203040506070809 0A0B0C0D0E0F C3

FIRMWARE_VERIFY (0x05) and FIRMWARE_VERIFY_AND_INSTALL (0x06)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY (0x05) or FIRMWARE_VERIFY_AND_INSTALL (0x06)
GPM_OPTIONS	Reserved. Set to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

Note The target device will be unable to receive RF packets for a short period of time (around half a second) while verifying the firmware after receiving either of these commands.

FIRMWARE_VERIFY_RESPONSE (0x85)

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

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_RESPONSE (0x85)
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command

FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86)

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

Note If the firmware image is valid, after that node sends the response the device will reset and begin using the new firmware.

Field name	Command-specific description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY_AND_INSTALL_ RESPONSE (0x86).
GPM_STATUS	Indicates success if 0. Otherwise, an error occurred (see Possible Errors Returned from GPM Commands).
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command.

Example

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

Possible Errors Returned from GPM Commands

Below are listed possible errors that may return from sending a GPM command:

Return Code	Description
0x00	Success
General command failures	
0x01	General failure
0x02	Bad payload length
0x03	Tried to access memory block beyond the max available
0x04	Attempted to read/write across a block boundary
0x05	Attempted to read/write with a valid file system mounted
0x06	Unrecognized GPM command
0x07	GPM is currently busy executing another GPM command

Return Code	Description
Erase command failures	
0x10	Flash erase operation failed
Write command failures	
0x20	Flash write operation failed
0x21	Flash write would have created a valid FS header. Writing a file system into GPM is disallowed due to security concerns.
Read command failures	
0x30	Flash read operation failed
0x31	Tried to read more than can be transmitted in a single packet
0x32	Couldn't get a buffer to send read updates
Verify and install failures	
0x40	Firmware verify operation failed
0x41	The given image is not compatible with this device
0x42	The given image appears corrupted or invalid
0x50	Firmware install operation failed

Update the firmware over-the-air

The XBee XR 900 RF Module supports firmware over-the-air (FOTA) updates. To perform an FOTA update, the device to be updated must be associated and communicable with a network. In this section, the node performing the update is considered the server and the node being updated is the client.

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

Over-the-air firmware updates

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

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

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

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

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

Distribute the new application

The first phase of performing an over-the-air update on a device is transferring the new firmware file to the target node. Load the new firmware image in the target node's GPM prior to installation. XBee XR 900 RF Modules use a a Gecko Bootloader (.gbl) file for both serial and over-the-air firmware updates. These firmware files are available on the Digi Support website and via XCTU.

Note The firmware files contain two images: one with the .gbl extension, and one with the .apponly.gbl extension. The file with the .gbl extension should be used for most updates. The .apponly.gbl file does not contain bootloader update information. It can be used to perform OTA updates slightly faster, but only after manually verifying that a bootloader update is not needed.

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

Example

The example firmware version has an .gbl file of 55,141 bytes in length. Based on network traffic, we determine that sending a 128 byte packet every 30 seconds minimizes network disruption. For this reason, you would divide and address the .gbl as follows:

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.gbl bytes
0	0	128	0 to 127
0	128	128	128 to 255
0	256	128	256 to 383
--	--	--	--
0	1920	128	1920 to 2047

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.gbl bytes
1	0	128	2048 to 2175
1	128	128	2176 to 2303
--	--	--	--
--	--	--	--
26	1536	128	54784 to 54911
26	1664	128	54912 to 55039
26	1792	101	55040 to 55140

Verify the new application

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

Install the application

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

Important considerations

Write all parameters with the **WR** command before performing a firmware update. Packet routing information is also lost after a reset. Route discoveries are necessary for DigiMesh unicasts involving the updated node as a source, destination, or intermediate node.

Because explicit API Tx frames can be addressed to a local node (accessible via the SPI or UART) or a remote node (accessible over the RF port) the same process can be used to update firmware on a device in either case.

Regulatory information

United States (FCC)

XBee XR 900 RF Modules comply with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

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

1. The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.
2. RF Modules may only be used with antennas that have been tested and approved for use with the modules.

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

OEM labeling requirements

WARNING! As an Original Equipment Manufacturer (OEM) you must ensure that FCC labeling requirements are met. You must include a clearly visible label on the outside of the final product enclosure that displays the following content:

Required FCC Label for OEM products containing the XBee XR 900 RF Module

Contains FCC ID: MCQ-XB9XR

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

FCC notices

IMPORTANT: XBee XR 900 RF Modules have been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by Digi could void the user's authority to operate the equipment.

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

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

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

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect equipment and receiver to outlets on different circuits.
- Consult the dealer or an experienced radio/TV technician for help.

FCC-approved antennas

The XBee XR 900 RF Module can be installed using antennas and cables constructed with non-standard connectors (RPSMA, RPTNC, etc.) An adapter cable may be necessary to attach the XBee connector to the antenna connector.

The modules are FCC approved for fixed base station and mobile applications for the channels indicated in the tables below. If the antenna is mounted at least 20 cm (8 in) from nearby persons, the application is considered a mobile application or a base station. Antennas not listed in the table must be tested to comply with FCC Section 15.203 (Unique Antenna Connectors) and Section 15.247 (Emissions).

The antennas in the tables below have been approved for use with this module. Correct cable loss or power reduction is required when using gain antennas as shown in the tables.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

1. If using the RF module in a portable application (for example, if the module is used in a hand-held device and the antenna is less than 20 cm from the human body when the device is in operation), the integrator is responsible for passing additional Specific Absorption Rate (SAR) testing based on FCC rules 2.1093 and FCC Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields, OET Bulletin and Supplement C. The testing results will be submitted to the FCC for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person.

FCC-approved antennas

The following tables cover the antennas that are approved for use with the XBee XR 900 RF Module. If applicable, the tables show the required cable loss between the device and the antenna.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

Note An antenna not listed in the equipment authorization that is the same type with equal or lower gain (same physical arrangement; generates the same in-band and out-of-band characteristics in all spatial directions) may be marketed and used with a part 15 transmitter.

Dipole antennas

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

Part number	Type	Connector	Gain	Required antenna cable loss	Application
A09-HASM-675 (MPD: SATAC-915NF)	Articulated half-wave	RPSMA	2.96 dBi	0.3	Fixed/mobile

Yagi antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-Y15NF*	13 element Yagi	15.1 dBi	N	0.62	Fixed/mobile

Omni-directional base station antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F8NF-M* (TE Connectivity: FG9026)	Fiberglass Base Station	8.0 dBi	N	0.62	Fixed

Dome antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-D3NF* (Laird: TRA8903)	Phandom (Dome) Antenna	3.0 dBi	NMO	0.62	Fixed/mobile

Note The best performance of the TRA8903 "dome" antenna is given when mounted on a ground plane.

Ceramic chip antenna

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

This antenna type is mounted on the surface mount carrier.

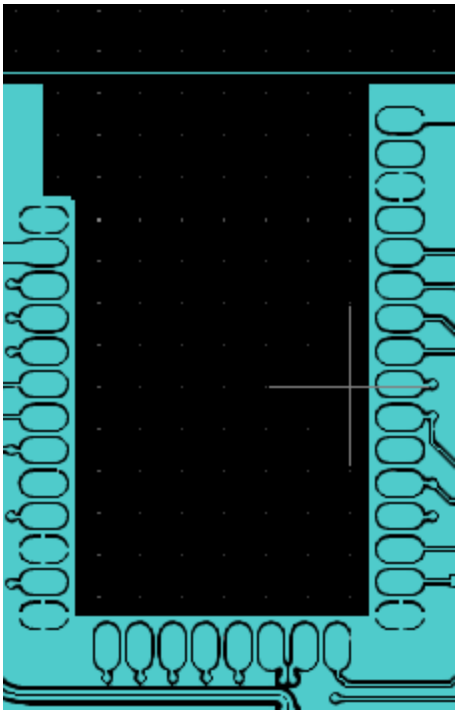
Follow all directions for use of this product.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
29001056*	Ceramic chip antenna (SMT module only)	0.8 dBi	N/A	0	Fixed/mobile

Recommended use of the XB9XR module with ceramic chip antenna

The XB9XR module (SMT, with ceramic antenna) should be mounted on a host board with the following ground plane cutouts (dimensions and other information requested):

- Recommended ground cutout:



- Minimum ground cutout:

The SMT module with the ceramic chip antenna should be placed flat, or horizontally, on the surface it's mounted to. The module can be placed vertically, as long as the module's antenna is mounted towards other radios in the network.

Flexible (flat) antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A07W27-PU18-7x* (Taoglas:	Flexible multiband	2.0 dBi	RPSMA	0	Fixed/mobile

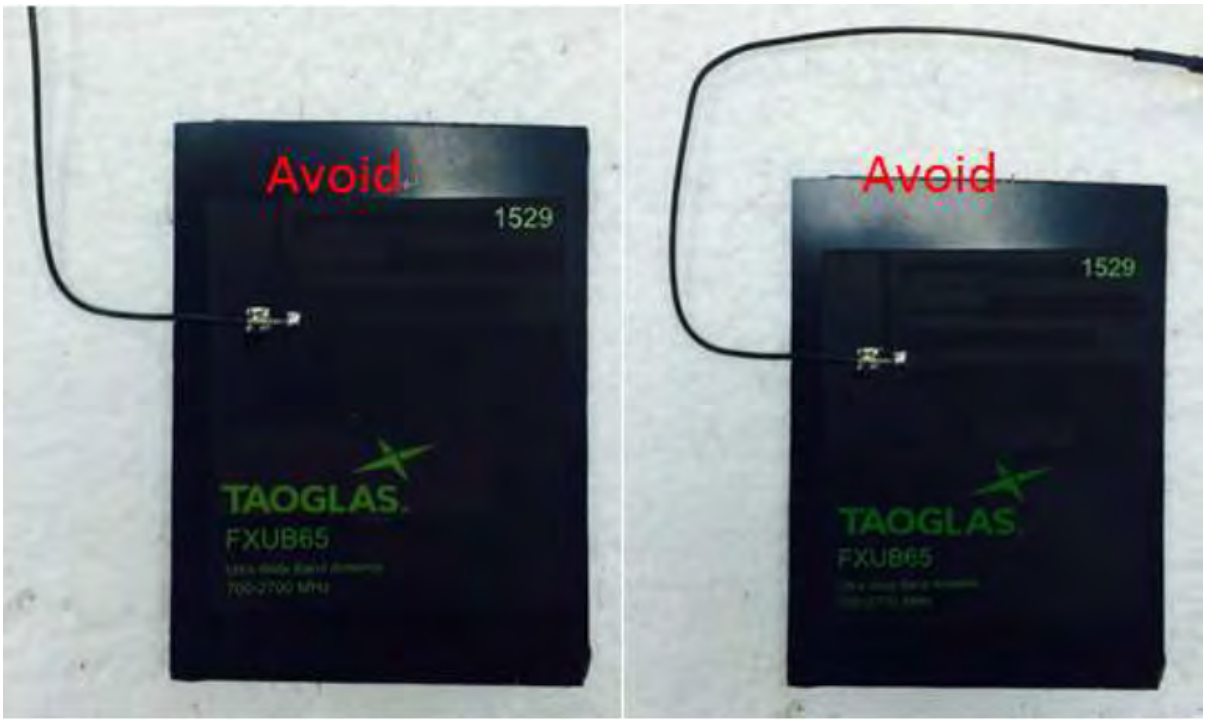
Part number	Type	Gain	Connector	Required antenna cable loss	Application
FXUB65.07.0180C)	antenna				
HXT-211140-010* (Molex: 211140-0100)	Flexible monopole	1.0 dBi	RPSMA	0	Fixed/mobile

Recommended usage of Taoglas

Taoglas (PN: FXUB65.07.0180C) performs best when it's positioned flat and horizontally. The Taoglas antenna can be fastened to glass or plastic surfaces.

The Taoglas FXUB65 datasheet provides these preferred orientations. Place the cable as shown:

The Taoglas datasheet does not recommend the following cable orientations:



Recommended usage of Molex antenna

Orient this antenna upright. This gives an antenna pattern similar to a dipole. The Molex antenna can be fastened to a glass or plastic surface.

The cable should be bent as shown in the first two figures below. Do not place the cable to the side of the antenna, as shown in the third figure.

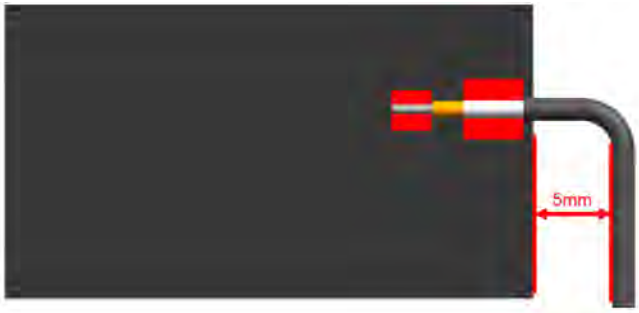


FIGURE 5.2.1 CABLE BENDING

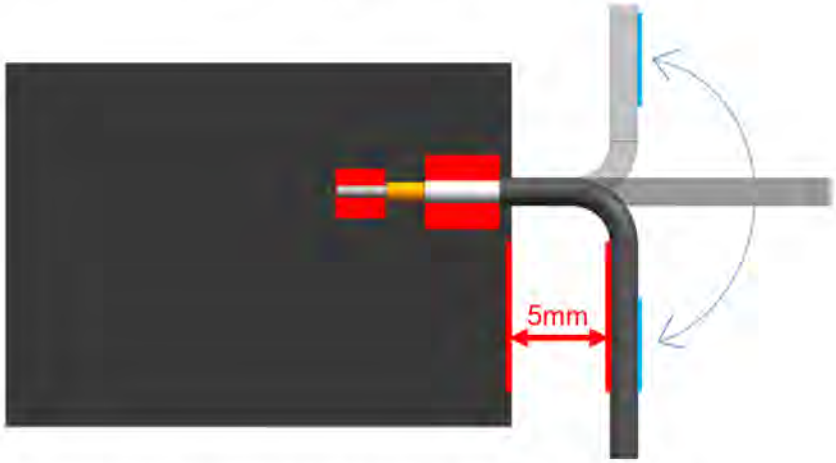


FIGURE 5.2.2 CABLE ACTIVITY RANGE

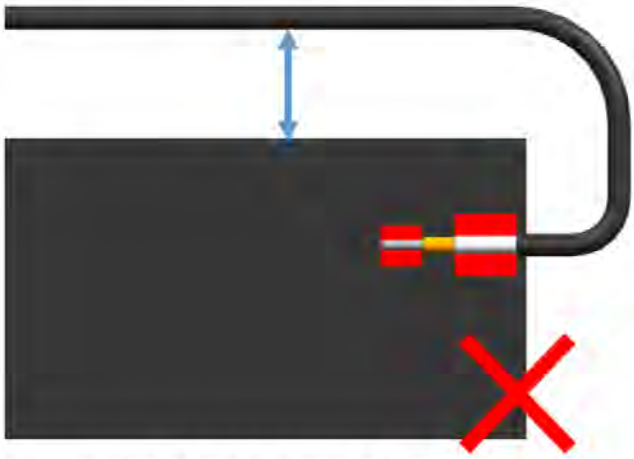


FIGURE 5.2.3 CABLE BENDING

RF exposure

If you are an integrating the XBee into another product, you must include the following Caution statement in OEM product manuals to alert users of FCC RF exposure compliance:

CAUTION! To satisfy FCC RF exposure requirements for mobile transmitting devices, a separation distance of 26 cm or more should be maintained between the antenna of this device and persons during device operation. To ensure compliance, operations at closer than this distance are not recommended. The antenna used for this transmitter must not be co-located in conjunction with any other antenna or transmitter.

FCC publication 996369 related information

In Publication 996369 section D03, the FCC requires information concerning a module to be presented by OEM manufacturers. This section assists in answering or fulfilling these requirements.

2.1 General

No requirements are associated with this section.

2.2 List of applicable FCC rules

This module conforms to FCC Part 15.247.

2.3 Summarize the specific operational use conditions

Certain approved antennas require attenuation for operation. For the XBee XR 900 RF Module, see FCC-approved antennas.

Host product user guides should include the antenna table if end customers are permitted to select antennas.

2.4 Limited module procedures

Not applicable.

2.5 Trace antenna designs

While it is possible to build a trace antenna into the host PCB, this requires at least a Class II permissive change to the FCC grant which includes significant extra testing and cost. If an embedded trace antenna is desired, refer to literature for trace antenna designs.

For the FCC requirements to use trace antennas, note the following:

- Modules certified with trace antennas or trace to an antenna or RF connector design shall follow the requirements of Question 11 of 996369 D02 Module Q&A. See Question and Answer 11.
- If you decide to use a trace antenna, you must request assistance from Digi to obtain a “change in FCC ID”. You will then work with a lab of your choice to obtain the testing and documentation needed to file for your product with the trace antenna in a Class II Permissive filing.

2.6 RF exposure considerations

For RF exposure considerations see RF exposure and FCC-approved antennas.

Host product manufacturers need to provide end-users a copy of the “RF Exposure” section of the manual: RF exposure.

The FCC Grant for the XBee product only allows only “MOBILE” and “FIXED” applications at distances greater than 26 cm from a person’s body.

No “PORTABLE” use is allowed by the FCC Grant.

Please request assistance from Digi to obtain a “change in FCC ID”. You will then work with a lab of your choice to obtain the testing and documentation needed to file for your product for the specific “PORTABLE” use using a Class II Permissive filing.

Host manufacturers must:

- Integrate the XBee module per the instructions in this user guide
- Follow the directions in “FCC publication KDB 996369 related information” of this guide.
- If your product is to be used in a “PORTABLE” application, please request assistance from Digi to obtain the request for “change in FCC ID”.
- You are responsible for having the integrated host product tested according to the instructions in the most recent revision of FCC KDB 996369 “TRANSMITTER MODULE EQUIPMENT AUTHORIZATION GUIDE”. This FCC KDB also directs you to obtain proof of compliance with RF Exposure as directed in the FCC KDB “447498 D04 Interim General RF Exposure Guidance v01”.
- You must direct your customer in the use of your product. Include text similar to this:
  - “This product must be used according to the user guide. The product can (or cannot) be used within 26 cm (10.2 in) from any human, pet, plant, or agricultural animal.”

2.7 Antennas

A list of approved antennas is provided for the XBee XR 900 RF Module product. See FCC-approved antennas.

2.8 Label and compliance information

Host product manufacturers need to follow the sticker guidelines outlined in OEM labeling requirements.

OEM labeling requirements

WARNING! As an Original Equipment Manufacturer (OEM) you must ensure that FCC labeling requirements are met. You must include a clearly visible label on the outside of the final product enclosure that displays the following content:

Required FCC Label for OEM products containing the XBee XB9XR RF Modules

- Contains FCCID: MCQ-XB9XR

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

CAUTION! Changes or modifications not expressly approved by Digi International Inc. could void the user's authority to operate the equipment.

2.9 Information on test modes and additional testing requirements

Contact a Digi sales representative for information on how to configure test modes for the XBee XR 900 RF Module product.

2.10 Additional testing, Part 15 Subpart B disclaimer

All final host products must be tested to be compliant to FCC Part 15 Subpart B standards. While the XBee XR 900 RF Module unit was tested to be compliant to FCC unintentional radiator standards, FCC Part 15 Subpart B compliance testing is still required for the final host product. This testing is required for all end products, and XBee XR 900 RF Module Part 15 Subpart B compliance does not affirm the end product's compliance.

See FCC notices for more details.

Over-voltage detection

Over-voltage detection sends out a modem status of **0x0D** indicating that the voltage supply limit has been exceeded. The device will still operate but limits the RF power level **PL** setting to a value of **3** when the operating voltage reaches 3.7 volts or higher to meet regulatory RF power requirements. While the device is in this mode of operation it will be forced into API mode for the over-voltage modem status to be sent out the serial port every 15 seconds when API mode is set to 1 or 2.

ISED (Innovation, Science and Economic Development Canada)

This device contains licence-exempt transmitter(s)/receiver(s) that comply with Innovation, Science and Economic Development Canada's licence-exempt RSS(s). Operation is subject to the following two conditions:

1. This device may not cause interference.
2. This device must accept any interference, including interference that may cause undesired operation of the device.

This radio transmitter IC: 1846A-XB9XR has been approved by Innovation, Science and Economic Development Canada to operate with the antenna types listed below, with the maximum permissible gain indicated. Antenna types not included in this list that have a gain greater than the maximum gain indicated for any type listed are strictly prohibited for use with this device.

L'émetteur/récepteur exempt de licence contenu dans le présent appareil est conforme aux CNR d'Innovation, Sciences et Développement économique Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

1. *L'appareil ne doit pas produire de brouillage;*
2. *L'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.*

Le présent émetteur radio IC: 1846A-XB9XR a été approuvé par Innovation, Sciences et Développement économique Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal. Les types d'antenne non inclus dans cette liste, et dont le gain est supérieur au gain maximal indiqué pour tout type figurant sur la liste, sont strictement interdits pour l'exploitation de l'émetteur.

ISED-approved antennas

The XBee XR 900 RF Module can be installed using antennas and cables constructed with nonstandard connectors (RPSMA, RPTNC, etc.) An adapter cable may be necessary to attach the XBee connector to the antenna connector.

The antennas in the tables below have been approved for use with this module. Correct cable loss or power reduction is required when using gain antennas as shown in the tables.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

1. If using the RF module in a portable application (for example, if the module is used in a hand-held device and the antenna is less than 20 cm from the human body when the device is in operation), the integrator is responsible for passing additional Specific Absorption Rate (SAR) testing based on ISED rule RSS-Gen and ISED Guidelines for Human Exposure to Radio Frequency Electromagnetic Fields. The testing results will be submitted to the ISED for approval prior to selling the integrated unit. The required SAR testing measures emissions from the module and how they affect the person.

ISED-approved antennas

The following tables cover the antennas that are approved for use with the XBee XR 900 RF Module. If applicable, the tables show the required cable loss between the device and the antenna.

Digi does not carry all of these antenna variants. Contact Digi Sales for available antennas.

Note An antenna not listed in the equipment authorization that is the same type with equal or lower gain (same physical arrangement; generates the same in-band and out-of-band characteristics in all spatial directions) may be marketed and used with a part 15 transmitter.

Dipole antennas

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

Part number	Type	Connector	Gain	Required antenna cable loss	Application
A09-HASM-675 (MPD: SATAC-915NF)	Articulated half-wave	RPSMA	2.96 dBi	0.3	Fixed/mobile

Yagi antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-Y15NF*	13 element Yagi	15.1 dBi	N	0.62	Fixed/mobile

Omni-directional base station antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-F8NF-M* (TE Connectivity: FG9026)	Fiberglass Base Station	8.0 dBi	N	0.62	Fixed

Dome antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A09-D3NF* (Laird: TRA8903)	Phandom (Dome) Antenna	3.0 dBi	NMO	0.62	Fixed/mobile

Note The best performance of the TRA8903 "dome" antenna is given when mounted on a ground plane.

Ceramic chip antenna

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

This antenna type is mounted on the surface mount carrier.

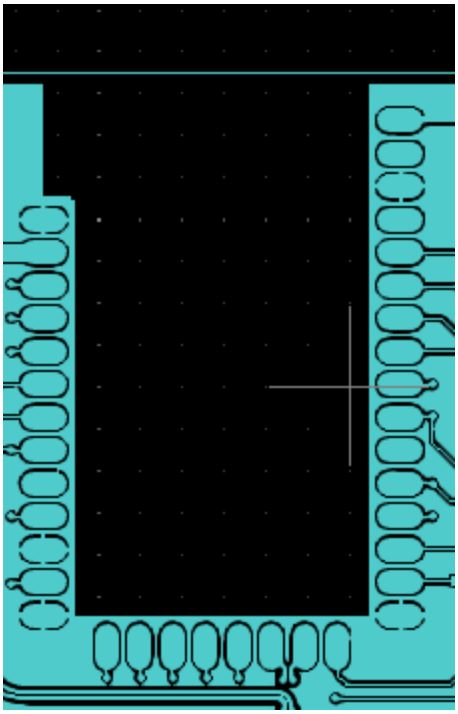
Follow all directions for use of this product.

Part number	Type	Gain	Connector	Required antenna cable loss	Application
29001056*	Ceramic chip antenna (SMT module only)	0.8 dBi	N/A	0	Fixed/mobile

Recommended use of the XB9XR module with ceramic chip antenna

The XB9XR module (SMT, with ceramic antenna) should be mounted on a host board with the following ground plane cutouts (dimensions and other information requested):

- Recommended ground cutout:



- Minimum ground cutout:

The SMT module with the ceramic chip antenna should be placed flat, or horizontally, on the surface it's mounted to. The module can be placed vertically, as long as the module's antenna is mounted towards other radios in the network.

Flexible (flat) antennas

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

Part number	Type	Gain	Connector	Required antenna cable loss	Application
A07W27-PU18-7x* (Taoglas: FXUB65.07.0180C)	Flexible multiband antenna	2.0 dBi	RPSMA	0	Fixed/mobile

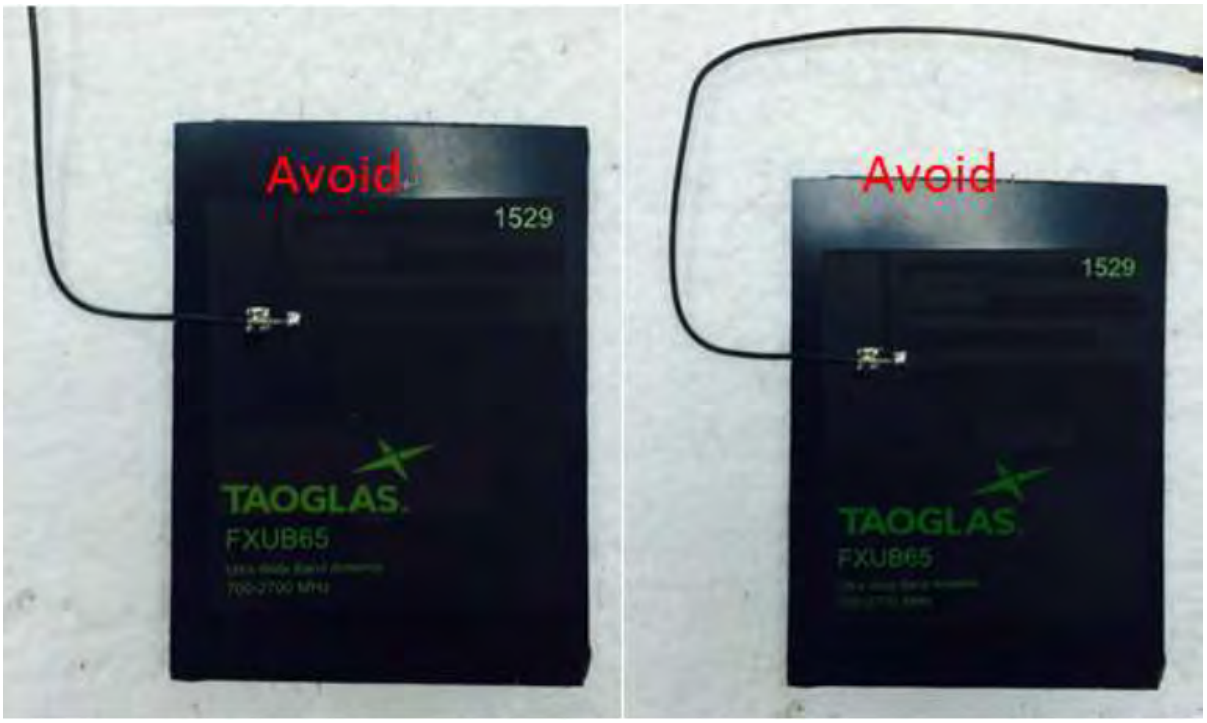
Part number	Type	Gain	Connector	Required antenna cable loss	Application
HXT-211140-010* (Molex: 211140-0100)	Flexible monopole	1.0 dBi	RPSMA	0	Fixed/mobile

Recommended usage of Taoglas

Taoglas (PN: FXUB65.07.0180C) performs best when it's positioned flat and horizontally. The Taoglas antenna can be fastened to glass or plastic surfaces.

The Taoglas FXUB65 datasheet provides these preferred orientations. Place the cable as shown:

The Taoglas datasheet does not recommend the following cable orientations:



Recommended usage of Molex antenna

Orient this antenna upright. This gives an antenna pattern similar to a dipole. The Molex antenna can be fastened to a glass or plastic surface.

The cable should be bent as shown in the first two figures below. Do not place the cable to the side of the antenna, as shown in the third figure.

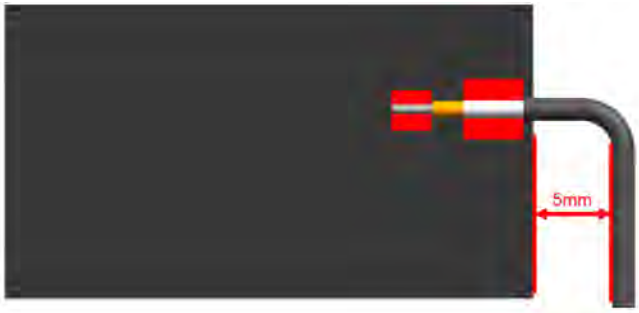


FIGURE 5.2.1 CABLE BENDING

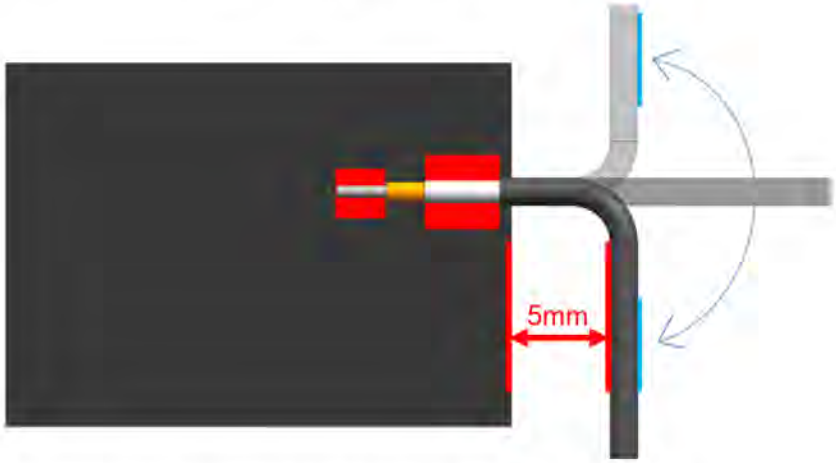


FIGURE 5.2.2 CABLE ACTIVITY RANGE

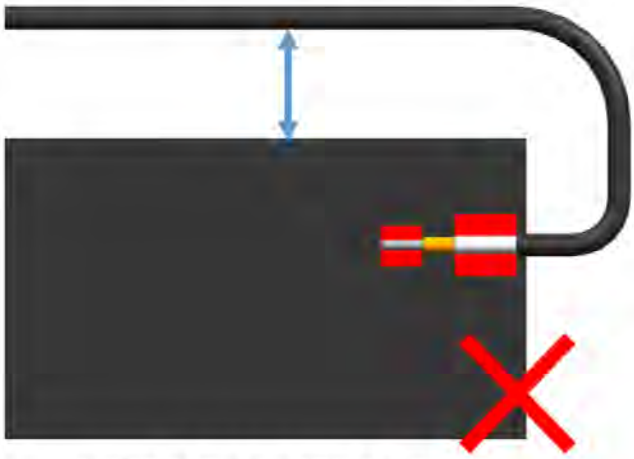


FIGURE 5.2.3 CABLE BENDING

Labeling requirements

Digi XBee XR 900

Labeling requirements for Innovation, Science and Economic Development Canada are similar to those of the ISED. A clearly visible label on the outside of the final product must display the following text:

Contains IC: 1846A-XB9XR

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

Transmitters for detachable antennas

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the tables in FCC-approved antennas with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device. The required antenna impedance is 50 ohms.

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

Detachable antennas

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

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

RF exposure

If you are an integrating the XBee into another product, you must include the following Caution statement in OEM product manuals to alert users of ISED RF exposure compliance:

CAUTION! To satisfy ISED RF exposure requirements for mobile transmitting devices, a separation distance of 26 cm or more should be maintained between the antenna of this device and persons during device operation. To ensure compliance, operations at closer than this distance are not recommended. The antenna used for this transmitter must not be co-located in conjunction with any other antenna or transmitter.

ACMA (Australia)

Power requirements

Regulations in Australia stipulate a maximum of 30 dBm EIRP (Effective Isotropic Radiated Power). The EIRP equals the sum (in dBm) of power output, antenna gain and cable loss and cannot not exceed 30 dBm.

The EIRP formula for Australia is:

power output + antenna gain - cable loss <= 30 dBm

Note The maximum RF power allowed in Australia is 30 dBm. The radio has RF power output at 19 dBm. For example, if you add a Yagi antenna with 15 dBi of gain, then you will need to add additional loss of 6 dB inline to the antenna.

Note The maximum EIRP for the FCC (United States) and IC (Canada) is 36 dBm.

These modules comply with requirements to be used in end products in Australia. All products with EMC and radio communications must have a registered RCM mark. Registration to use the compliance mark will only be accepted from Australian manufacturers or importers, or their agent, in Australia. In order to have an RCM mark on an end product, a company must comply with a or b below:

a. have a company presence in Australia.
b. have a company/distributor/agent in Australia that will sponsor the import of the end product.

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

RSM (New Zealand)

Power requirements

No antenna with gain greater that 2.1 dBi (dipole) can be used with this radio in New Zealand. These modules comply with requirements to be used in end products in New Zealand. All products with EMC and radio communications must have a registered R-NZ mark. Registration to use the compliance mark will only be accepted from manufacturers or importers, or their agent, in New Zealand. In order to have a R-NZ mark on an end product, a company must comply with a or b below:

a. have a company presence in New Zealand.
b. have a company/distributor/agent in New Zealand that will sponsor the import of the end product.

Module support

Custom defaults

Custom defaults allow you to preserve a subset of the device configuration parameters even after returning to default settings using RE (Restore Defaults). This can be useful for settings that identify the device—such as NI (Node Identifier).

Set custom defaults

Use %F (Set Custom Default) to set custom defaults. When the XBee XR 900 RF Module receives **%F** it takes the next command it receives and applies it to both the current configuration and the custom defaults.

To set custom defaults for multiple commands, send a **%F** before each command.

Restore factory defaults

!C (Clear Custom Defaults) clears all custom defaults, so that RE (Restore Defaults) will restore the device to factory defaults. Alternatively, R1 (Restore Factory Defaults) restores all parameters to factory defaults without erasing their custom default values.

Limitations

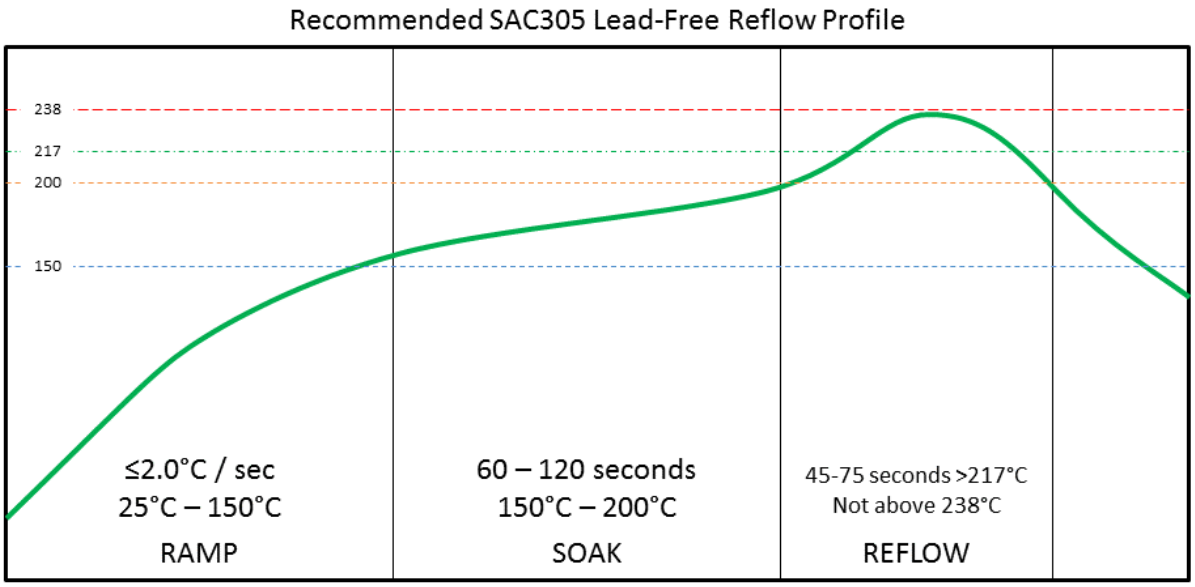
There is a limitation on the number of custom defaults that can be set on a device. The number of defaults that can be set depends on the size of the saved parameters and the devices' firmware version. When there is no more room for custom defaults to be saved, any command sent immediately after a **%F** returns an error.

Manufacturing information

The micro XBee XR 900 RF Module is designed for surface-mounting on the OEM PCB. It has castellated pads to allow for easy solder attaching and inspection. The pads are all located on the edge of the device so there are no hidden solder joints on these devices.

Recommended solder reflow cycle

The following diagram shows the recommended solder reflow cycle.



Recommended reflow profile only
Modifications to profile may be required to fit specific application, process or design

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

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

Handling and storage

The XBee XR 900 RF Modules are level 3 Moisture Sensitive Devices. When using this kind of device, consider the relative requirements in accordance with standard IPC/JEDEC J-STD-020.

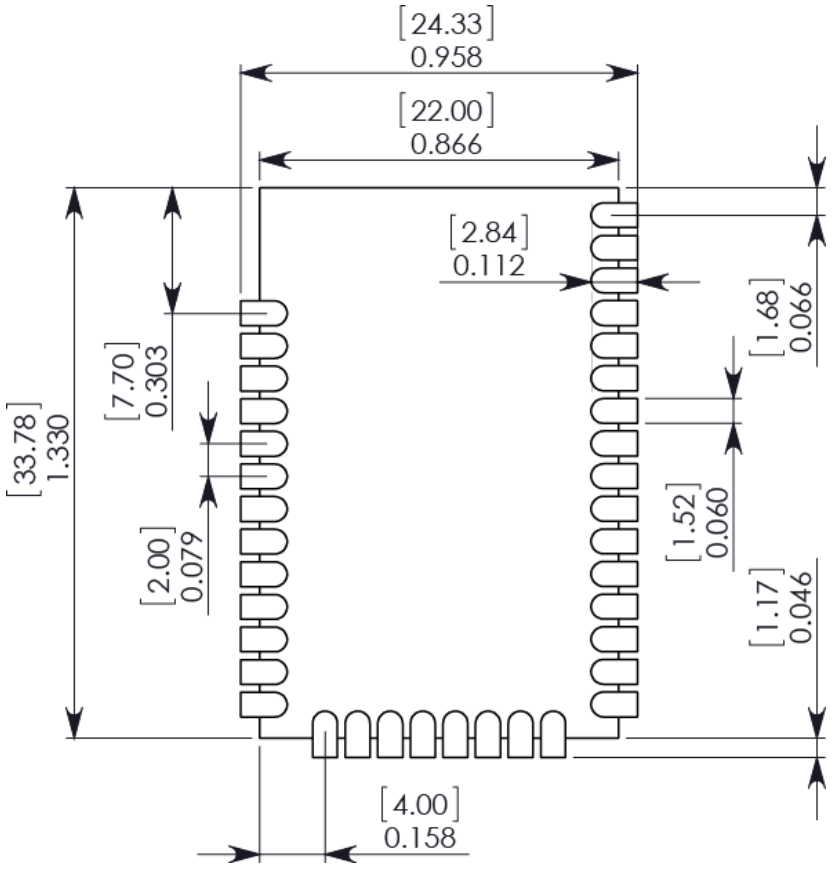
In addition, note the following conditions:

a. Calculated shelf life in sealed bag: 12 months at <40 °C and <90% relative humidity (RH).
b. Environmental condition during the production: 30 °C /60% RH according to IPC/JEDEC J-STD -033C, paragraphs 5 through 7.
c. The time between the opening of the sealed bag and the start of the reflow process cannot exceed 168 hours if condition b) is met.
d. Baking is required if conditions b) or c) are not met.
e. Baking is required if the humidity indicator inside the bag indicates a RH of 10% more.
f. If baking is required, bake modules in trays stacked no more than 10 high for 4-6 hours at 125 °C.

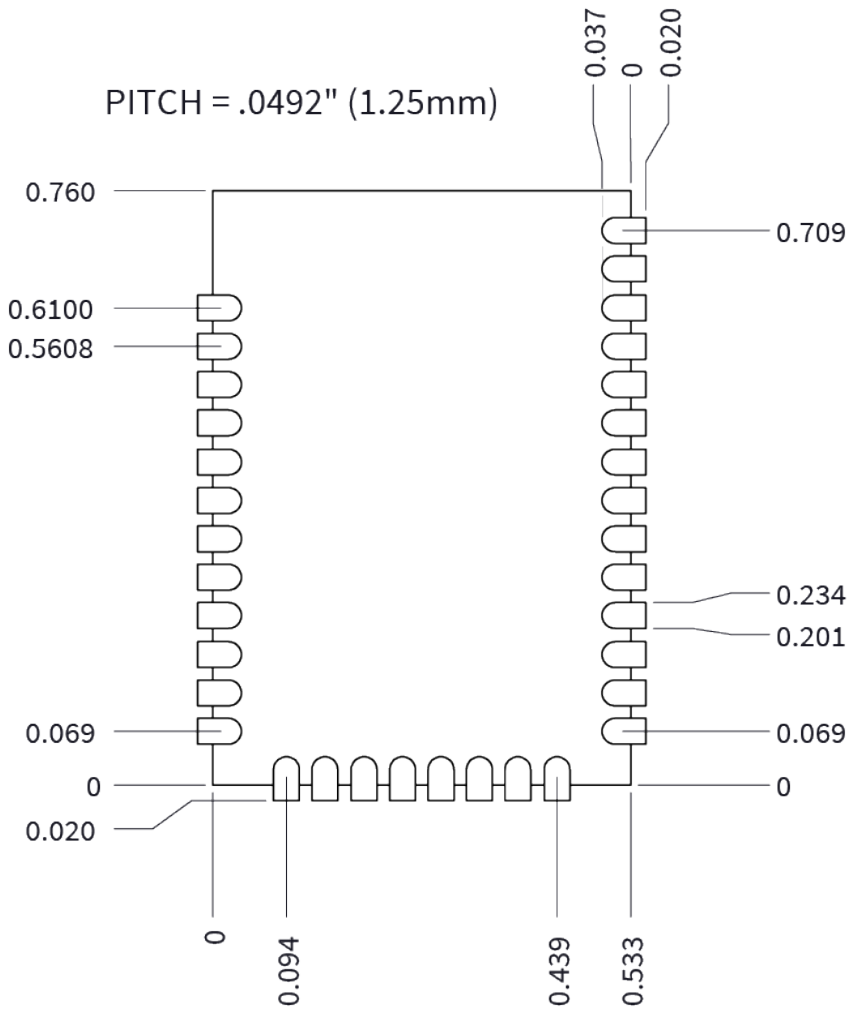
Recommended footprint

We recommend that you use the following PCB footprints for surface-mounting. The dimensions without brackets are in inches, and those in brackets are in millimeters.

Surface-mount recommended footprint



XBee XR Micro recommended footprint



Match the solder footprint to the copper pads, but it may need to be adjusted depending on the specific needs of assembly and product standards. Recommended stencil thickness is 0.15 mm/0.005”. Place the component last and set the placement speed to the slowest setting.

Flux and cleaning

Digi recommends that a “no clean” solder paste be used in assembling these devices. This eliminates the clean step and ensures unwanted residual flux is not left under the device where it is difficult to remove.

In addition the following issues can occur:

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

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

Reworking

Never perform rework on the device itself. The device has been optimized to give the best possible performance, and reworking the device itself will void warranty coverage and certifications. We recognize that some customers choose to rework and void the warranty. The following information serves as a guideline in such cases to increase the chances of success during rework, though the warranty is still voided.

The device may be removed from the OEM PCB by the use of a hot air rework station, or hot plate. Be careful not to overheat the device. During rework, the device temperature may rise above its internal solder melting point and care should be taken not to dislodge internal components from their intended positions.