XBee® 865/868LP RF Modules

XBee RF Modules by Digi International

Models: XBEE S8

Hardware: S8

Firmware: 8059

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

90002126_B
9/24/2012

Technical Support:	Phone:	(866) 765-9885 toll-free U.S.A. & Canada (801) 765-9885 Worldwide
		8:00 am - 5:00 pm [U.S. Mountain Time]
	Online Support:	http://www.digi.com/support/eservice/login.jsp
	Email:	rf-experts@digi.com

Contents

Contents

1. RF Module Hardware

This manual describes the operation of the XBee®/XBee® 865/868 Low Power SMT RF module, which consists of firmware loaded onto XBee S8 hardware.

XBee and XBee 865/868 Low Power SMT embedded RF modules provide wireless connectivity to end-point devices in mesh networks. Utilizing the XBee-PRO Feature Set, these modules are interoperable with other devices, including devices from other vendors. With the XBee, users can have their network up-and-running in a matter of minutes without configuration or additional development.

The XBee/XBee 865/868 modules are not compatible with all XBee products.

XBee S8 Hardware Description

The XBee S8 radio module hardware consists of an Energy Micro EFM32G230F128 microcontroller, and Analog Devices ADF7023 radio transceiver, and in the programmable version, a Freescale MC9S08QE32 microcontroller.

European Acceptance

The XBee 865/868 is manufactured under ISO 900:2000 registered standards.

XBee 865/868 SMT RF Modules are optimized for use in Europe. Please refer to Appendix A for more information.

Specifications

Specifications of the XBee® 865/868LP RF Module

Specification	XBee
Performance	
* Indoor/Urban Range	up to 370 ft (112 m) w/2.1 dBi antenna, up to 46 ft (14 m) w/ embedded antenna
* Outdoor RF line-of-sight Range	up to 5.2 miles (8.4 km) w/2.1dB dipole antenna, up to 0.4 mi (640 m) w/ embedded antenna
Transmit Power Output	12 dBm (16 mW)
RF Data Rate (High)	80 kbps
RF Data Rate (Low)	10 kbps
Serial UART interface	CMOS Serial UART, baud rate stability of <1%
Serial Interface Data Rate (software selectable)	9600-230400 baud
Receiver Sensitivity (typical)	-101 dBm, high data rate, -106 dBm, low data rate
Receiver Blocking (typical)	Frequency offset | Data rate --- | 10kbps 80kbps | --- +/- 400 kHz | 40 dB 35 dB | +/- 200 kHz | 35 dB 29 dB |
Power Requirements	
Supply Voltage	2.7 to 3.6 VDC
Transmit Current, high data rate	48mA, (45 mA typical)
Transmit Current, low data rate	47 mA (41 mA typical)
Idle / Receive Current (high data rate)	27mA (22 mA typical)
Idle / Receive Current (low data rate)	26 mA (24 mA typical)
Sleep Current	1.7 µA
General	
**Operating Frequency Band	863 to 870 MHz
Dimensions	0.866 in x 1.333 in x 1.2 in (2.119 cm x 3.4 cm x 0.305 cm)
Weight	.14 oz. (4 g)
Operating Temperature	-40º to 85º C (industrial)
Antenna Options	Pad on module edge, U. FL RF connector, embedded antenna
Digital I/O	13 I/O lines, 5 dedicated to SPI that can be used as digital outputs
ADC	6 10-bit analog inputs
Networking & Security	
Supported Network Topologies	Mesh, point-to-point, point-to-multipoint, peer-to-peer
**Number of Channels, user selectable channels	30 channels, LBT + AFA

Specifications of the XBee® 865/868LP RF Module

Specification	XBee
Addressing Options	PAN ID and 64-bit addresses
Encryption	128 bit AES
Agency Approvals	
**Europe (CE)	CE Marking

* To determine your range, perform a range test under your operating conditions.

**Please see the list of channel limits for countries in the European Community in this User Manual.

Serial Communications Specifications

XBee RF modules support both UART (Universal Asynchronous Receiver / Transmitter) and SPI (Serial Peripheral Interface) serial connections.

UART

UART Pin Assignments

UART Pins	Module Pin Number
DOUT	3
DIN / $\overline{\text{CONFIG}}$	4
$\overline{\text{CTS}}$ / DIO7	25
$\overline{\text{RTS}}$ / DIO6	29

More information on UART operation is found in the UART section in Chapter 2.

SPI

SPI Pin Assignments

SPI Pins	Module Pin Number
SPI_SCLK / DIO18	14
SPI_$\overline{\text{SSEL}}$ / DIO17	15
SPI_MOSI / DIO16	16
SPI_MISO / DIO15	17

For more information on SPI operation, see the SPI section in Chapter 2.

GPIO Specifications

XBee RF modules have 15 GPIO (General Purpose Input / Output) ports available. The exact list will depend on the module configuration, as some GPIO pads are used for purposes such as serial communication.

See GPIO section for more information on configuring and using GPIO ports.

Electrical Specifications for GPIO Pads

GPIO Electrical Specification	Value
Voltage - Supply	2.7- 3.6 V
Low Schmitt switching threshold	0.3 x Vdd
High Schmitt switching threshold	0.7 x Vdd
Input pull-up resistor value	40 kΩ
Input pull-down resistor value	40 kΩ
Output voltage for logic 0	0.05 x Vdd
Output voltage for logic 1	0.95 x Vdd
Output source current	6 mA

Electrical Specifications for GPIO Pads

GPIO Electrical Specification	Value
Output sink current for	6 mA
Total output current (for GPIO pads)	48 mA

Hardware Specs for Programmable Variant

If the module has the programmable secondary processor, add the following table values to the specifications listed on page 7. For example, if the secondary processor is running at 20 MHz and the primary processor is in receive mode then the new current value will be $I_{total} = I_{r2} + I_{rx}$ = 14 mA + 9 mA = 23 mA, where I_{r2} is the runtime current of the secondary processor and I_{rx} is the receive current of the primary.

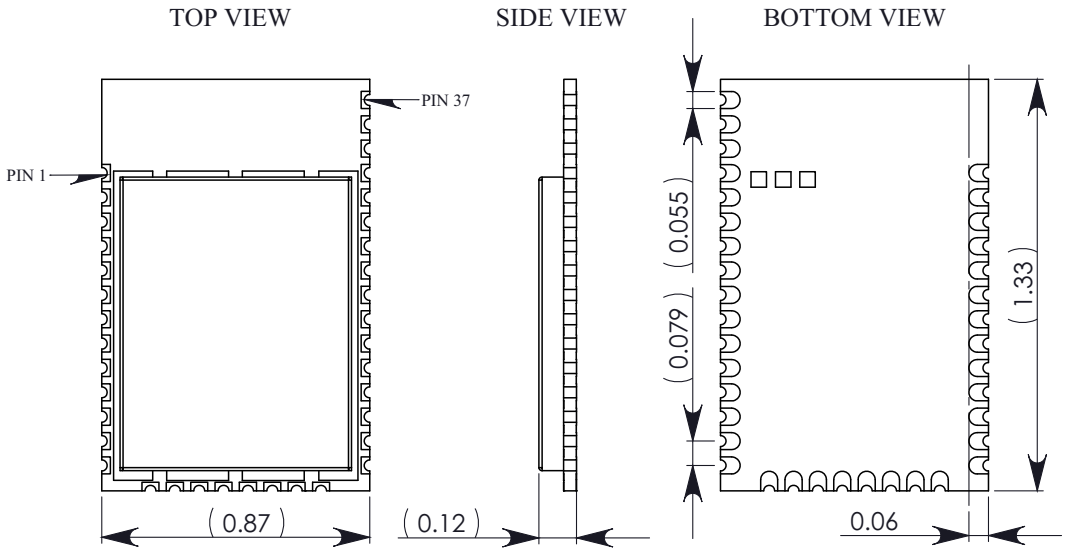
Specifications of the programmable secondary processor

Optional Secondary Processor Specification	These numbers add to specifications (Add to RX, TX, and sleep currents depending on mode of operation)
Runtime current for 32k running at 20MHz	+14mA
Runtime current for 32k running at 1MHz	+1mA
Sleep current	+0.5µA typical
For additional specifications see Freescale Datasheet and Manual	MC9SO8QE32
Minimum Reset Pulse for Programmable	100nS
Minimum Reset Pulse to Radio	50nS
VREF Range	1.8VDC to VCC

Mechanical Drawings

Mechanical drawings of the XBee® 865/868LP RF Modules (antenna options not shown). All dimentions are in inches.

.



Pin Signals

Pin Assignments for XBee Modules

(Low-asserted signals are distinguished with a horizontal line above signal name.)

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		Module Reset
7	RSSI PWM / DIO10	Both	Output	RX Signal Strength Indicator / GPIO
8	PWM1 / DIO11	Both	Disabled	Pulse Width Modulator / GPIO
9	[reserved]	-	Disabled	Do Not Connect
10	$\overline{\text{DTR}}$ / SLEEP_RQ / DIO8	Both	Input	Pin Sleep Control Line / GPIO
11	GND	-	-	Ground
12	SPI_$\overline{\text{ATTN}}$ / 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_$\overline{\text{SSEL}}$ / DIO 17	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	Module Status Indicator / GPIO
27	VREF	Input	-	Not used internally, used for programmable secondary processor. For compatibility with other XBee modules, we recommend connecting this pin to the voltage reference if Analog Sampling is desired. Otherwise, connect to GND.
28	ASSOCIATE / DIO5	Both	Output	Associate Indicator / GPIO
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	Both	Disabled	Analog Input / GPIO
33	AD0 / DIO0	Both	Input	Analog Input / GPIO
34	[reserved]	-	Disabled	Do Not Connect
35	GND	-	-	Ground
36	RF	Both	-	RF IO for RF Pad Variant
37	[reserved]	-	Disabled	Do Not Connect

- Signal Direction is specified with respect to the module
- See Design Notes section below for details on pin connections.
- * These pins are not available for customer use.

Internal Pin Mappings

Insert Table???

Design Notes

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

Power Supply Design

Poor power supply can lead to poor radio performance, especially if the supply voltage is not kept within tolerance or is excessively noisy. To help reduce noise, we recommend placing both a 1µF and 8.2pF capacitor as near to pin 2 on the PCB as possible. If using a switching regulator for your power supply, switching frequencies above 500kHz are preferred. Power supply ripple should be limited to a maximum 250mV peak to peak.

Note – For designs using the programmable modules, an additional 10µF decoupling cap is recommended near pin 2 of the module. The nearest proximity to pin 2 of the three caps should be in the following order: 8.2pf, 1µF followed by 10µF.

Recommended Pin Connections

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

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

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

Other pins may be connected to external circuitry for convenience of operation, including the Associate LED pad (pad 28) and the Commissioning pad (pad 33). The Associate LED pad will flash differently depending on the state of the module to the network, and a pushbutton attached to pad 33 can enable various join functions without having to send serial port commands. Please see the commissioning pushbutton and associate LED section in chapter 7 for more details. The source and sink capabilities are limited to 6mA on all I/O pads.

The VRef pad (pad 27) is only used on the programmable versions of these modules. For compatibility with other XBee modules, we recommend connecting this pin to a voltage reference if analog sampling is desired. Otherwise, connect to GND.

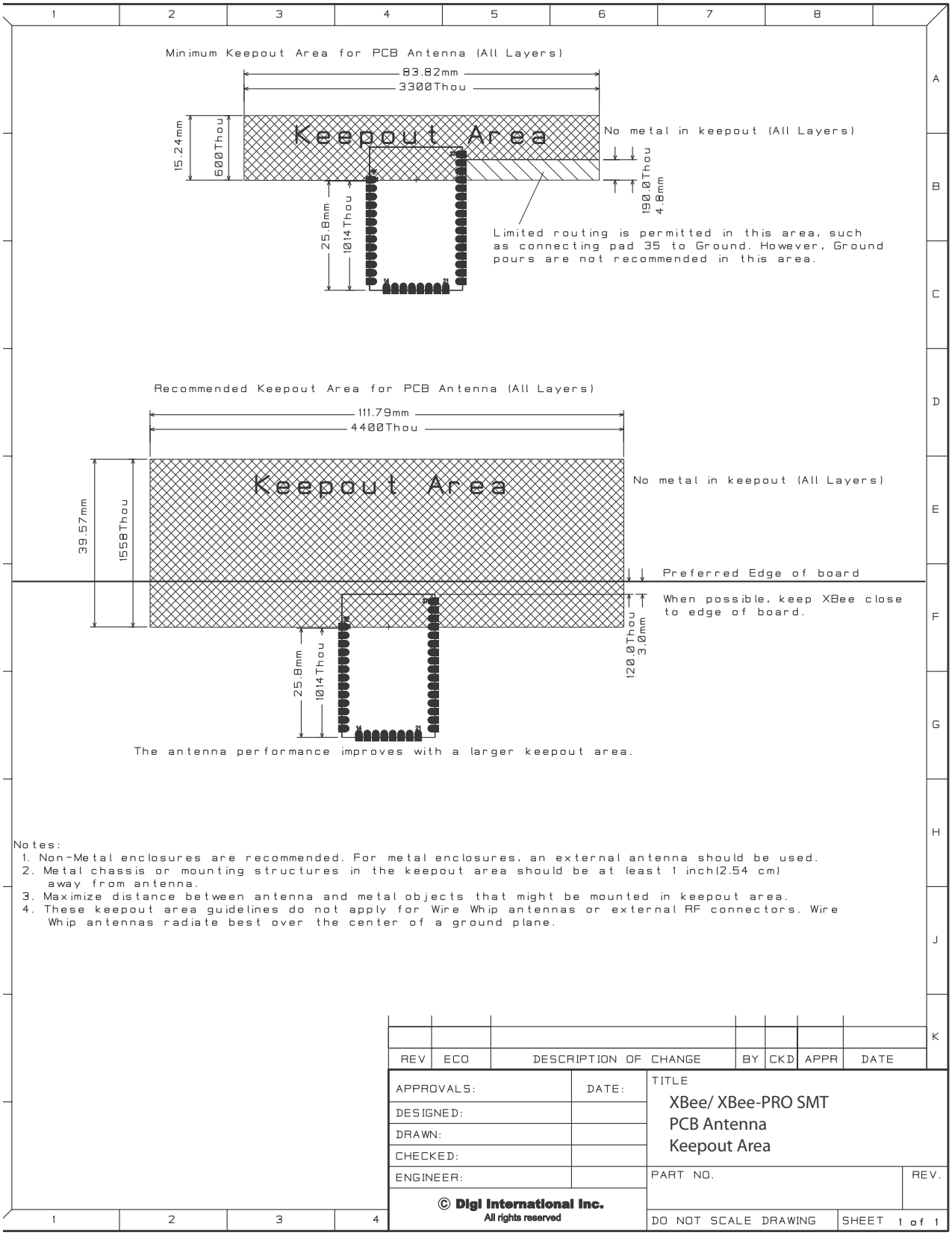
Board Layout

XBee modules are designed to be self sufficient and have minimal sensitivity to nearby processors, crystals or other PCB components. As with all PCB designs, Power and Ground traces should be thicker than signal traces and able to comfortably support the maximum current specifications. A recommended PCB footprint for the module can be found in Appendix C. No other special PCB design considerations are required for integrating XBee radios except in the antenna section.

The choice of antenna and antenna location is very important for correct performance. With the exception of the RF Pad variant, XBees do not require additional ground planes on the host PCB. In general, antenna elements radiate perpendicular to the direction they point. Thus a vertical antenna emits across the horizon. Metal objects near the antenna cause reflections and may reduce the ability for an antenna to radiate efficiently. Metal objects between the transmitter and receiver can also block the radiation path or reduce the transmission distance, so external antennas should be positioned away from them as much as possible. Some objects that are often overlooked are metal poles, metal studs or beams in structures, concrete (it is usually reinforced with metal rods), metal enclosures, vehicles, elevators, ventilation ducts, refrigerators, microwave ovens, batteries, and tall electrolytic capacitors.

Design Notes for PCB Antenna Modules

PCB Antenna modules should not have any ground planes or metal objects above or below the antenna. For best results, the module should not be placed in a metal enclosure, which may greatly reduce the range. The module should be placed at the edge of the PCB on which it is mounted. The ground, power and signal planes should be vacant immediately below the antenna section. The drawing on the following page illustrates important recommendations for designing with the PCB Antenna module. It should be noted that for optimal performance, this module should not be mounted on the RF Pad footprint described in the next section because the footprint requires a ground plane within the PCB Antenna keep out area.

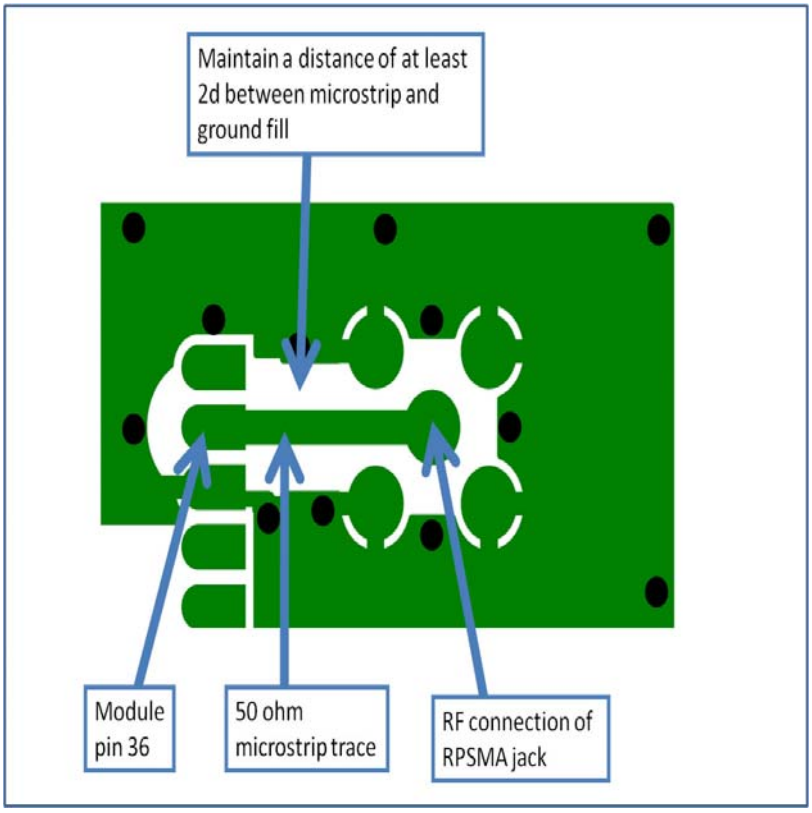

Minimum Keepout Area for PCB Antenna (All Layers)
83.82mm
3300Thou
15.24mm
600Thou
Keepout Area
No metal in keepout (All Layers)
190.0Thou
4.8mm
25.8mm
1014Thou
Limited routing is permitted in this area, such as connecting pad 35 to Ground. However, Ground pours are not recommended in this area.
Recommended Keepout Area for PCB Antenna (All Layers)
111.79mm
4400Thou
39.57mm
1558Thou
Keepout Area
No metal in keepout (All Layers)
Preferred Edge of board
When possible, keep XBee close to edge of board.
120.0Thou
3.0mm
25.8mm
1014Thou
The antenna performance improves with a larger keepout area.
Notes:
1. Non-Metal enclosures are recommended. For metal enclosures, an external antenna should be used.
2. Metal chassis or mounting structures in the keepout area should be at least 1 inch(2.54 cm) away from antenna.
3. Maximize distance between antenna and metal objects that might be mounted in keepout area.
4. These keepout area guidelines do not apply for Wire Whip antennas or external RF connectors. Wire Whip antennas radiate best over the center of a ground plane.
REV ECO DESCRIPTION OF CHANGE BY CKD APPR DATE
APPROVALS: DATE:
DESIGNED:
DRAWN:
CHECKED:
ENGINEER:
TITLE
XBee/ XBee-PRO SMT
PCB Antenna
Keepout Area
PART NO.
REV.
© Digi International Inc.
All rights reserved
DO NOT SCALE DRAWING
SHEET 1 of 1

Design Notes for RF Pad Modules

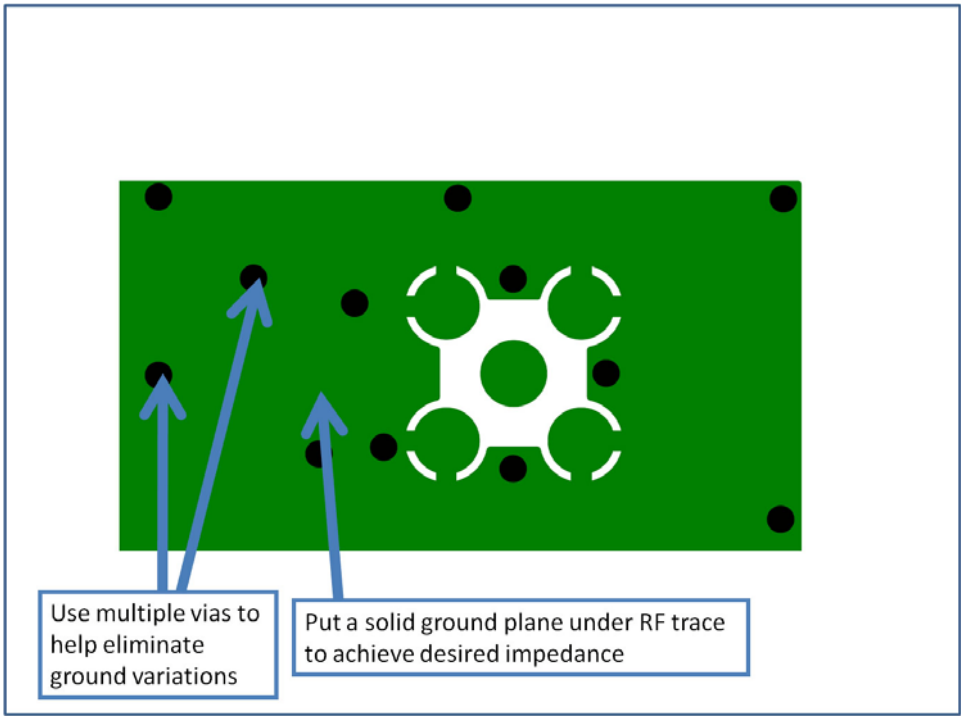
The RF Pad is a soldered antenna connection. The RF signal travels from pin 36 on the module to the antenna through an RF trace transmission line on the PCB. Please note that any additional components between the module and antenna will violate modular certification. The RF trace should have a controlled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

It is essential to follow good design practices when implementing the RF trace on a PCB. The following figures show a layout example of a host PCB that connects an RF Pad module to a right angle, through hole RPSMA jack. The top two layers of the PCB have a controlled thickness dielectric material in between. The second layer has a ground plane which runs underneath the entire RF Pad area. This ground plane is a distance d, the thickness of the dielectric, below the top layer. The top layer has an RF trace running from pin 36 of the module to the RF pin of the RPSMA connector. The RF trace's width determines the impedance of the transmission line with relation to the ground plane. Many online tools can estimate this value, although the PCB manufacturer should be consulted for the exact width. Assuming d=0.025", and that the dielectric has a relative permittivity of 4.4, the width in this example will be approximately 0.045" for a 50 ohm trace. This trace width is a good fit with the module footprint's 0.060" pad width. Using a trace wider than the pad width is not recommended, and using a very narrow trace (under 0.010") can cause unwanted RF loss. The length of the trace is minimized by placing the RPSMA jack close to the module. All of the grounds on the jack and the module are connected to the ground planes directly or through closely placed vias. Any ground fill on the top layer should be spaced at least twice the distance d (in this case, at least 0.050") from the microstrip to minimize their interaction.

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



PCB Layer 1 of RF Layout Example



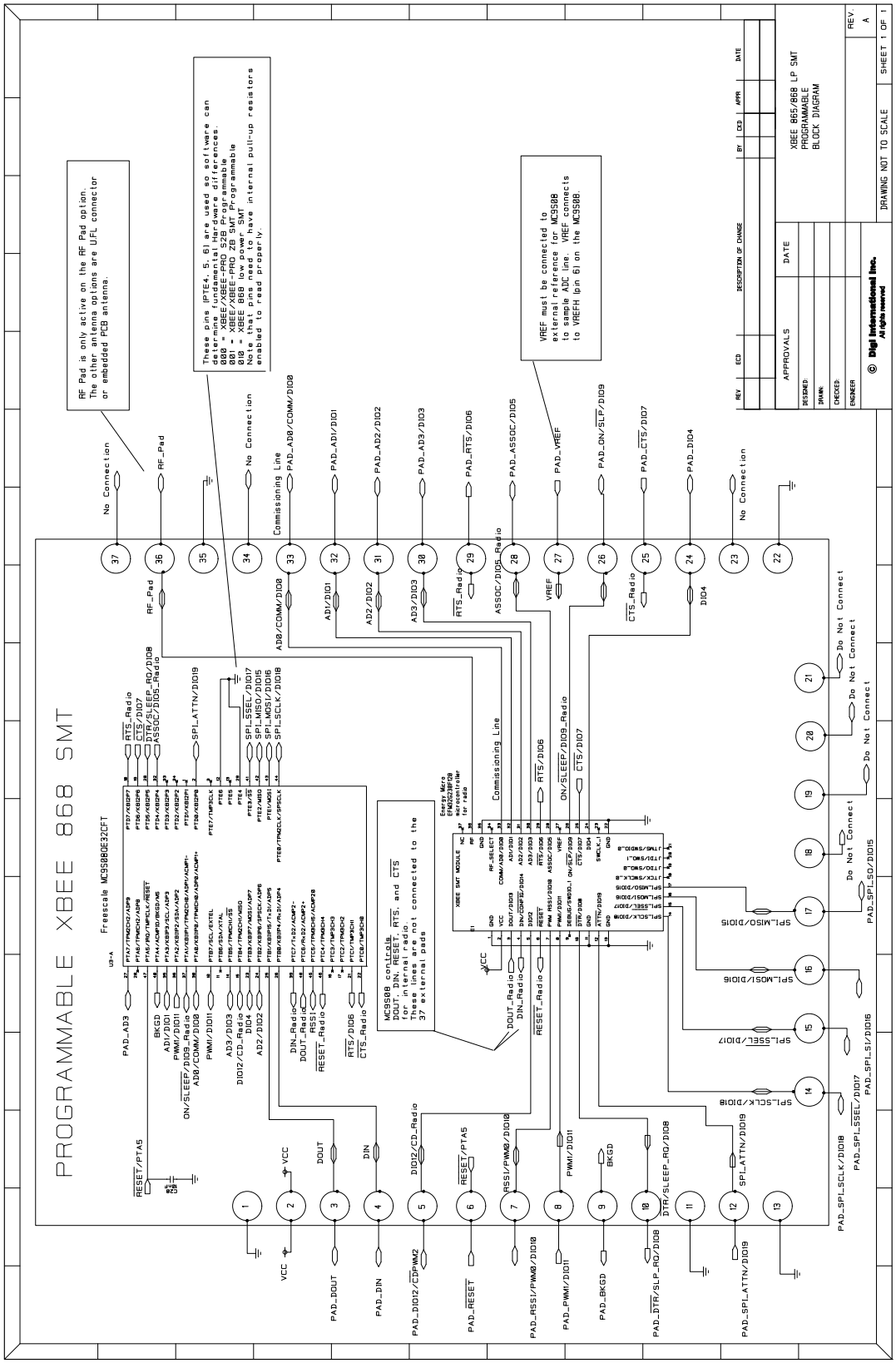
PCB Layer 2 of RF Layout Example

Module Operation for Programmable Variant

The modules with the programmable option have a secondary processor with 32k of flash and 2k of RAM. This allows module integrators to put custom code on the XBee module to fit their own unique needs. The DIN, DOUT, RTS, CTS, and RESET lines are intercepted by the secondary processor to allow it to be in control of the data transmitted and received. All other lines are in parallel and can be controlled by either the internal microcontroller or the MC9SO8QE micro (see Block Diagram for details). The internal microcontroller by default has control of certain lines. These lines can be released by the internal microcontroller by sending the proper command(s) to disable the desired DIO line(s) (see XBee Command Reference Tables).

In order for the secondary processor to sample with ADCs, the XBee pin 27 (VREF) must be connected to a reference voltage.

Digi provides a bootloader that can take care of programming the processor over the air or through the serial interface. This means that over the air updates can be supported through an XMODEM protocol. The processor can also be programmed and debugged through a one wire interface BKGD (Pin 9).


PROGRAMMABLE XBEE 868 SMT
Freescale MC9S08QE32CFT
RF Pad is only active on the RF Pad option.
The other antenna options are U.FL connector
or embedded PCB antenna.
These pins (PTE4, 5, 6) are used so software can
determine if certain hardware differences.
000 - XBEE/XBEE-PRO S2B Programmable
001 - XBEE/XBEE-PRO ZB SMT Programmable
010 - XBEE 868 low power SMT
Note that pins need to have internal pull-up resistors
enabled to read properly.
VREF must be connected to
external reference for MC9S08
to sample ADC line. VREF connects
to VREFH (pin 6) on the MC9S08.
MC9S08 controls
DOUT, DIN, RESET, RTS, and CTS
for internal radio.
These lines are not connected to the
37 external pads
XBEE SMT MODULE
Energy Micro
EFM32G230F128
Microcontroller
for radio
Commissioning Line
No Connection
RF_Pad
Commissioning Line
PAD_AD0/COMM/DIO0
PAD_AD1/DIO1
PAD_AD2/DIO2
PAD_AD3/DIO3
PAD_RTS/DIO6
PAD_ASSOC/DIO5
PAD_VREF
PAD_ON/SLP/DIO9
PAD_CTS/DIO7
PAD_DIO4
PAD_SPI_SO/DIO15
PAD_SPI_SI/DIO16
PAD_SPI_SCLK/DIO18
PAD_SPI_SSEL/DIO17
PAD_DOUT
PAD_DIN
PAD_DIO12/CDPWM2
PAD_RESET
PAD_RSSI/PWM0/DIO10
PAD_PWM1/DIO11
PAD_BKGD
PAD_DTR/SLP_RQ/DIO8
PAD_SPI_ATTN/DIO19
XBEE 865/868 LP SMT
PROGRAMMABLE
BLOCK DIAGRAM
DRAWING NOT TO SCALE
SHEET 1 OF 1
REV A
© Digi International Inc.
All rights reserved

XBee Programmable Bootloader

Overview

The XBee Programmable module is equipped with a Freescale MC9S08QE32 application processor. This application processor comes with a supplied bootloader. This section describes how to interface the customer's application code running on this processor to the XBee Programmable module's supplied bootloader.

The first section discusses how to initiate firmware updates using the supplied bootloader for wired and over-the-air updates.

Bootloader Software Specifics

Memory Layout

Figure 1 shows the memory map for the MC9S08QE32 application processor.

The supplied bootloader occupies the bottom pages of the flash from 0xF200 to 0xFFFF. Application code cannot write to this space.

The application code can exist in Flash from address 0x8400 to 0xF1BC. 1k of Flash from 0x8000 to 0x83FF is reserved for Non Volatile Application Data that will not be erased by the bootloader during a flash update.

A portion of RAM is accessible by both the application and the bootloader. Specifically, there is a shared data region used by both the application and the bootloader that is located at RAM address 0x200 to 0x215. Application code should not write anything to BLResetCause or AppResetCause unless informing the bootloader of the impending reset reason. The Application code should not clear BLResetCause unless it is handling the unexpected reset reason.

To prevent a malfunctioning application from running forever, the Bootloader increments BLResetCause after each watchdog or illegal instruction reset. If this register reaches above 0x10 the bootloader will stop running the application for a few minutes to allow an OTA or Local update to occur. If no update is initiated within the time period, BLResetCause is cleared and the application is started again. To prevent unexpected halting of the application, the application shall clear or decrement BLResetCause just before a pending reset. To disable this feature, the application shall clear BLResetCause at the start of the application.

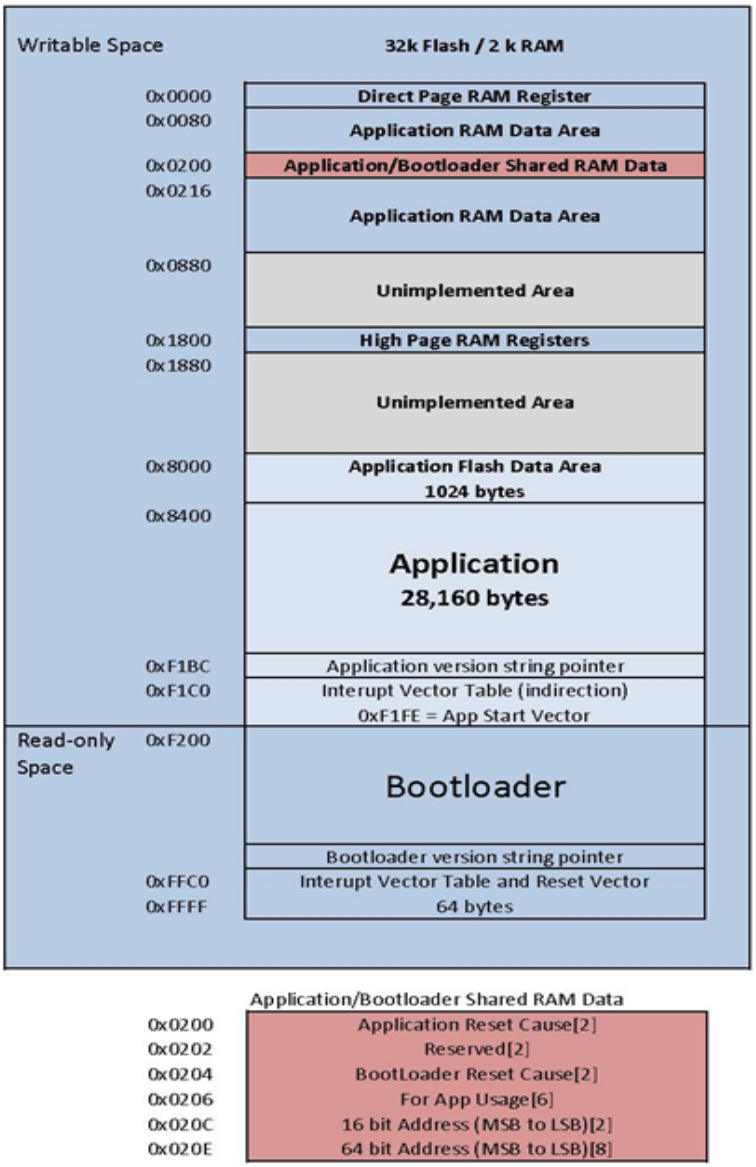


Figure 1: MC9S08QE32 Memory Map

Operation

Upon reset of any kind, the execution control begins with the bootloader.

If the reset cause is Power-On reset (POR), Pin reset (PIN), or Low Voltage Detect (LVD) reset (LVD) the bootloader will not jump to the application code if the override bits are set to RTS(D7)=1, DTR(D5)=0, and DIN(B0)=0. Otherwise, the bootloader writes the reset cause "NOTHING" to the shared data region, and jumps to the Application.

Reset causes are defined in the file *common. h* in an enumeration with the following definitions:

```
typedef enum {
  BL_CAUSE_NOTHING   = 0x0000,  //PIN, LVD, POR
  BL_CAUSE_NOTHING_COUNT   = 0x0001,//BL_Reset_Cause counter
// Bootloader increments cause every reset
  BL_CAUSE_BAD_APP   = 0x0010,//Bootloader considers APP invalid
} BL_RESET_CAUSES;
typedef enum {
   APP_CAUSE_NOTHING         = 0x0000,
   APP_CAUSE_USE001          = 0x0001,
// 0x0000 to 0x00FF are considered valid for APP use.
   APP_CAUSE_USE255          = 0x00FF,
   APP_CAUSE_FIRMWARE_UPDATE = 0x5981,
   APP_CAUSE_BYPASS_MODE     = 0x4682,
   APP_CAUSE_BOOTLOADER_MENU = 0x6A18,
} APP_RESET_CAUSES;
```

Otherwise, if the reset cause is a "watchdog" or other reset, the bootloader checks the shared memory region for the APP_RESET_CAUSE. If the reset cause is:

1. "APP_CAUSE_NOTHING" or 0x0000 to 0x00FF, the bootloader increments the BL_RESET_CAUSES, verifies that it is still less than BL_CAUSE_BAD_APP, and jumps back to the application. If the Application does not clear the BL_RESET_CAUSE, it can prevent an infinite loop of running a bad application that continues to perform illegal instructions or watchdog resets.
2. "APP_CAUSE_FIRMWARE_UPDATE", the bootloader has been instructed to update the application "over-the-air" from a specific 64-bit address. In this case, the bootloader will attempt to initiate an Xmodem transfer from the 64-bit address located in shared RAM.
3. "APP_CAUSE_BYPASS_MODE", the bootloader executes bypass mode. This mode passes the local UART data directly to the internal microcontroller allowing for direct communication with the internal microcontroller.
   The only way to exit bypass mode is to reset or power cycle the module.

If none of the above is true, the bootloader will enter "Command mode". In this mode, users can initiate firmware downloads both wired and over-the-air, check application/bootloader version strings, and enter Bypass mode.

Application version string

Figure 1 shows an "Application version string pointer" area in application flash which holds the pointer to where the application version string resides. The application's linker command file ultimately determines where this string is placed in application flash.

It is preferable that the application version string be located at address 0x8400 for MC9S08QE32 parts. The application string can be any characters terminated by the NULL character (0x00). There is not a strict limit on the number of characters in the string, but for practical purposes should be kept under 100 bytes including the terminating NULL character. During an update the bootloader erases the entire application from 0x8400 on. The last page has the vector table specifically the redirected reset vector. The version string pointer and reset vector are used to determine if the application is valid.

Application Interrupt Vector table and Linker Command File

Since the bootloader flash region is read-only, the interrupt vector table is redirected to the region 0xF1C0 to 0xF1FD so that application developers can use hardware interrupts. Note that in order for Application interrupts to function properly, the Application's linker command file (*.prm extension) must be modified appropriately to allow the linker to place the developers code in the correct place in memory. For example, the developer desires to use the serial communications port SCI1 receive interrupt. The developer would add the following line to the Codewarrior linker command file for the project:

VECTOR ADDRESS 0x0000F1E0 vSci1Rx

This will inform the linker that the interrupt function "vSci1Rx()" should be placed at address 0x0000F1E0. Next, the developer should add a file to their project "vector_table.c" that creates an array of function pointers to the ISR routines used by the application.

```
extern void _Startup(void);/* _Startup located in Start08.c */
extern void vSci1Rx(void);/* sci1 rx isr */
extern short iWriteToSci1(unsigned char *);
void vDummyIsr(void);
#pragma CONST_SEG VECTORS
void (* const vector_table[])(void) = /* Relocated Interrupt vector table */{
    vDummyIsr,/* Int.no. 0 Vtpm3ovf (at F1C0)Unassigned */
    vDummyIsr, /* Int.no. 1 Vtpm3ch5 (at F1C2) Unassigned */
    vDummyIsr, /* Int.no. 2 Vtpm3ch4 (at F1C4) Unassigned */
    vDummyIsr, /* Int.no. 3 Vtpm3ch3 (at F1C6) Unassigned */
    vDummyIsr, /* Int.no. 4 Vtpm3ch2 (at F1C8) Unassigned */
    vDummyIsr, /* Int.no. 5 Vtpm3ch1 (at F1CA) Unassigned */
    vDummyIsr, /* Int.no. 6 Vtpm3ch0 (at F1CC) Unassigned */
    vDummyIsr, /* Int.no. 7 Vrtc (at F1CE) Unassigned */
    vDummyIsr, /* Int.no. 8 Vsci2tx (at F1D0) Unassigned */
    vDummyIsr, /* Int.no. 9 Vsci2rx (at F1D2) Unassigned */
    vDummyIsr, /* Int.no. 10 Vsci2err (at F1D4) Unassigned */
    vDummyIsr, /* Int.no. 11 Vacmpx (at F1D6) Unassigned */
    vDummyIsr, /* Int.no. 12 Vadc (at F1D8) Unassigned */
    vDummyIsr, /* Int.no. 13 Vkeyboard (at F1DA) Unassigned */
    vDummyIsr, /* Int.no. 14 Viic (at F1DC) Unassigned */
    vDummyIsr, /* Int.no. 15 Vsci1tx (at F1DE) Unassigned */
    vSci1Rx, /* Int.no. 16 Vsci1rx (at F1E0) SCI1RX */
    vDummyIsr, /* Int.no. 17 Vsci1err (at F1E2) Unassigned */
    vDummyIsr, /* Int.no. 18 Vspi (at F1E4) Unassigned */
    vDummyIsr, /* Int.no. 19 VReserved12 (at F1E6) Unassigned */
    vDummyIsr, /* Int.no. 20 Vtpm2ovf (at F1E8) Unassigned */
    vDummyIsr, /* Int.no. 21 Vtpm2ch2 (at F1EA) Unassigned */
    vDummyIsr, /* Int.no. 22 Vtpm2ch1 (at F1EC) Unassigned */
    vDummyIsr, /* Int.no. 23 Vtpm2ch0 (at F1EE) Unassigned */
    vDummyIsr, /* Int.no. 24 Vtpm1ovf (at F1F0) Unassigned */
    vDummyIsr, /* Int.no. 25 Vtpm1ch2 (at F1F2) Unassigned */
    vDummyIsr, /* Int.no. 26 Vtpm1ch1 (at F1F4) Unassigned */
    vDummyIsr, /* Int.no. 27 Vtpm1ch0 (at F1F6) Unassigned */
```

```
                    vDummyIsr,  /* Int.no. 28 Vlvd (at F1F8)       Unassigned */
                    vDummyIsr,  /* Int.no. 29 Virq (at F1FA)       Unassigned */
                    vDummyIsr,  /* Int.no. 30 Vswi (at F1FC)       Unassigned */
                    _Startup    /* Int.no. 31 Vreset (at F1FE)     Reset vector */
};
void vDummyIsr(void){
 for(;;){
                  if(iWriteToSci1("STUCK IN UNASSIGNED ISR\n\r>"));
                 }
}
```

The interrupt routines themselves can be defined in separate files. The "vDummyIsr" function is used in conjunction with "iWritetoSci1" for debugging purposes.

Bootloader Menu Commands

The bootloader accepts commands from both the local UART and OTA. All OTA commands sent must be Unicast with only 1 byte in the payload for each command. A response will be returned to the sender. All Broadcast and multiple byte OTA packets are dropped to help prevent general OTA traffic from being interpreted as a command to the bootloader while in the menu.

Bypass Mode - "B"

The bootloader provides a “bypass” mode of operation that essentially connects the freescale mcu to the internal microcontroller’s serial UART. This allows direct communication to the internal microcontroller’s radio for the purpose of firmware and radio configuration changes. Once in bypass mode, the X-CTU utility can change modem configuration and/or update module’s firmware. Bypass mode automatically handles any baud rate up to 115.2kbps. Note that this command is unavailable when module is accessed remotely.

Update Firmware - "F"

The "F" command initiates a firmware download for both wired and over-the-air configurations. Depending on the source of the command (received via Over the Air or local UART), the download will proceed via wired or over-the-air respectively.

Adjust Timeout for Update Firmware - "T"

The "T" command changes the timeout before sending a NAK by Base-Time*2^(T). The Base-Time for the local UART is different than the Base-Time for Over the Air. During a firmware update, the bootloader will automatically increase the Timeout if repeat packets are received or multiple NAKs for the same packet without success occur.

Application Version String - "A"

The "A" command provides the version of the currently loaded application. If no application is present, "Unkown" will be returned.

Bootloader Version String - "V"

The "V" command provides the version of the currently loaded bootloader. The version will return a string in the format BLFFF-HHH-XYZ_DDD where FFF represents the Flash size in kilo bytes, HHH is the hardware, XYZ is the version, and DDD is the preferred XMODEM packet size for updates. Double the preferred packet size is also possible, but not guaranteed. For example "BL032-2B0-023_064" will take 64 byte CRC XMODEM payloads and may take 128 byte CRC XMODEM payloads also. In this case, both 64 and 128 payloads are handled, but the 64 byte payload is preferred for better Over the Air reliability.

Bootloader Version BL032-2x0-025_064 only operates at 9600 baud on the local UART as well as communications to the internal microcontroller. A newer version of the Bootloader BL032-2x0-033_064 or newer BL032-2B0-XXX_064 has changed the baud rate to 115200 between the Programmable and

the internal microcontroller. The internal module is also set to 115200 as the default baud rate. The default rate of the programmable local UART is also set to 115200, however, the local UART has an auto baud feature added to detect if the UART is at the wrong baud rate. If a single character is sent, it will automatically switch to 115200 or 9600 baud.

Firmware Updates

Wired Updates

A user can update their application using the bootloader in a wired configuration with the following steps:

a. Plug XBee programmable module into a suitable serial port on a PC.

b. Open a hyperterminal (or similar dumb terminal application) session with 115200 baud, no parity, and 8 data bits with one stop bit.

c. Hit Enter to display the bootloader menu.

d. Hit the "F" key to initiate a wired firmware update.

e. A series of "C" characters Will be displayed within the hyperterminal window. At this point, select the "transfer->send file" menu item. Select the desired flat binary output file.

f. Select "Xmodem" as the protocol.

g. Click "Send" on the "Send File" dialog. The file will be downloaded to the XBee Programmable module. Upon a successful update, the bootloader will jump to the newly loaded application.

Over-The-Air updates

A user can update their application using the bootloader in an “over-the-air” configuration with the following steps…(This procedure assumes that the bootloader is running and not the application. The internal microcontroller baud rate of the programmable module must be set to 115200 baud. The

bootloader only operates at 115200 baud between the Radio and programmable bootloader. The application must be programmed with some way to support returning to the bootloader in order to support Over the Air (OTA) updates without local intervention.)

a. Open a hyperterminal session to the host module with no parity, no hardwareflow control, 8 data bits and 1 stop bit. (The host module does not have to operate at the same baud rate as the remote module.) For faster updates and less latency due to the UART, set the host module to a faster baud rate. (i.e. 115200)

b. Enter 3 pluses "+++" to place the module in command mode. (or XCTU’s “Modem Configuration” tab can be used to set the correct parameters)

c. Set the Host Module destination address to the target module’s 64 bit address that the host module will update (ATDH aabbccdd, ATDL eeffgghh, ATCN, where aabbccddeeffgghh is the hexadecimal 64 bit address of the target module).

d. Hit Enter and the bootloader command menu will be displayed from the remote module. (Note that the option "B" doesn't exist for OTA)

e. Hit the "F" key to cause the remote module to request the new firmware file over-the-air.

f. The host module will begin receiving "C" characters indicating that the remote module is requesting an Xmodem CRC transfer. Using XCTU or another terminal program, Select "XMODEM" file transfer. Select the Binary file to upload/transfer. Click Send to start the transfer. At the conclusion of a successful transfer, the bootloader will jump to the newly loaded application.

Output File Configuration

BKGD Programming

P&E Micro provides a background debug tool that allows flashing applications on the MC9S08QE parts through their background debug mode port. By default, the Codewarrior tool produces an "ABS" output file for use in programming parts through the background debug interface. The programmable XBee from the factory has the BKGD debugging capability disabled. In order to debug, a bootloader

with the debug interface enabled needs to be loaded on the secondary processor or a stand-alone app needs to be loaded.

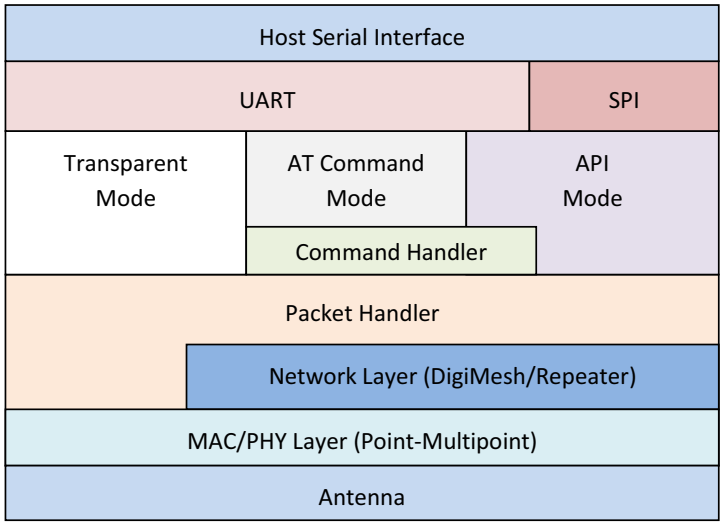
Bootloader updates

The supplied bootloader requires files in a "flat binary" format which differs from the default ABS file produced. The Codewarrior tool also produces a S19 output file. In order to successfully flash new applications, the S19 file must be converted into the flat binary format. Utilities are available on the web that will convert S19 output to "BIN" outputs. Often times, the "BIN" file conversion will pad the addresses from 0x0000 to the code space with the same number. (Often 0x00 or 0xFF) These extra bytes before the APP code starts will need to be deleted from the bin file before the file can be transferred to the bootloader.

2. RF Module Operation

Basic Operational Design

The XBee® 865/868LP RF Module uses a multi-layered firmware base to order the flow of data, dependent on the hardware and software configuration chosen by the user. This configuration block diagram is shown below, with the host serial interface as the physical starting point, and the antenna as the physical endpoint for the transferred data. As long as a block is able to touch another block, the two interfaces can interact. For example, if the module is using SPI mode, Transparent Mode is not available. See below:



The command handler is the code that processes commands from AT Command Mode or API Mode (see AT Commands section). The command handler can also process commands from remote radios (see Remote AT Commands section).

Listen Before Talk + Automatic Frequency Agility (LBT+AFA): g Band Mode

This radio implements LBT (Listen Before Talk) and AFA (Automatic Frequency Agility). The advantage of LBT+AFA is that the radio can bypass the Duty Cycle requirement imposed by ETSI. LBT+AFA requires that at least two frequencies be used for transmission. The g band mode contains several sub-bands. Refer to Chapter 1 for a full list of channels and frequencies.

The advantage of this feature is that it gives a level of fairness to the radios in a given area. Before this radio transmits, it will sense a channel to determine if there is activity by taking an RSSI measurement for 5ms. If the measurement is below the threshold then the radio will transmit on that channel. If there is activity, then that channel will not be used, and the radio will listen for at least 5ms to allow transmissions to be received.

After the radio transmits on a channel, it will not transmit on that channel again until the minimum TX off time has been met, which is greater than 100ms. For this reason it is useful to have many channels, so transmissions are not delayed.

There is also an ETSI requirement that only 100 seconds of transmission may occur over the period of an hour on 200kHz of spectrum. This method simplifies and optimizes the calculations of spectrum use over the period of one hour. As the ETSI specification states, the more channels you have, the more transmission time you have in a one hour period. The effective duty cycle can be calculated based on the number of available channels enabled as follows: Effective Duty Cycle = (number of channels * 100) / 3600

For example, if you enabled 2 channels you would have an effective duty cycle of 5.6%.

The XBee radio uses a sliding bucket algorithm to calculate usage over the period of 1 hour for each channel. Each bucket accumulates for 6 minutes.

This radio has a maximum of 30 AFA channels that it can choose from, and channels can be excluded by setting the channel mask (CM) to reduce them. Since not all countries allow for all of these channels, the set may be dramatically smaller for some countries. For a complete list, refer to www.digi.com.

g4 Band Mode

When the channel mask is set to 0x200000000, the radio will be in g4 band mode. In this mode:

- LBT+AFA mode is disabled
- Module assumes no duty cycle requirement (or 100% duty cycle)
- The PL setting must be set to 5 mW to comply with g4 band regulations

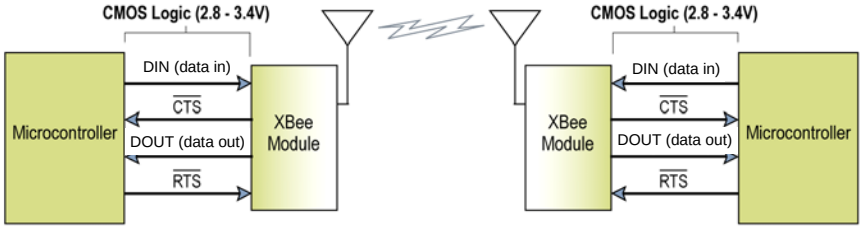
Serial Communications

XBee RF Modules interface to a host device through a serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART, through a level translator to any serial device (for example, through a RS-232 or USB interface board), or through a Serial Peripheral Interface (SPI), which is a synchronous interface to be described later.

UART Data Flow

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

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

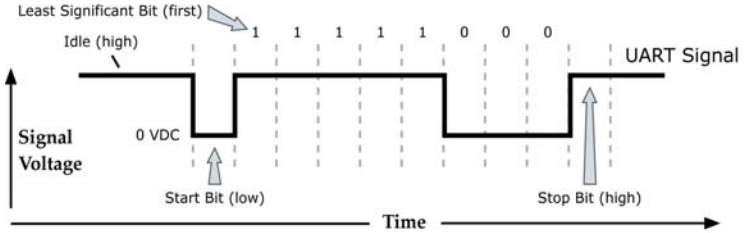


Serial Data

Data enters the module UART through the DIN (pin 4) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

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

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



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

The UART baud rate, parity, and stop bits settings on the XBee module can be configured with the BD, NB, and SB commands respectively. See the command table in chapter 10 for details.

SPI Communications

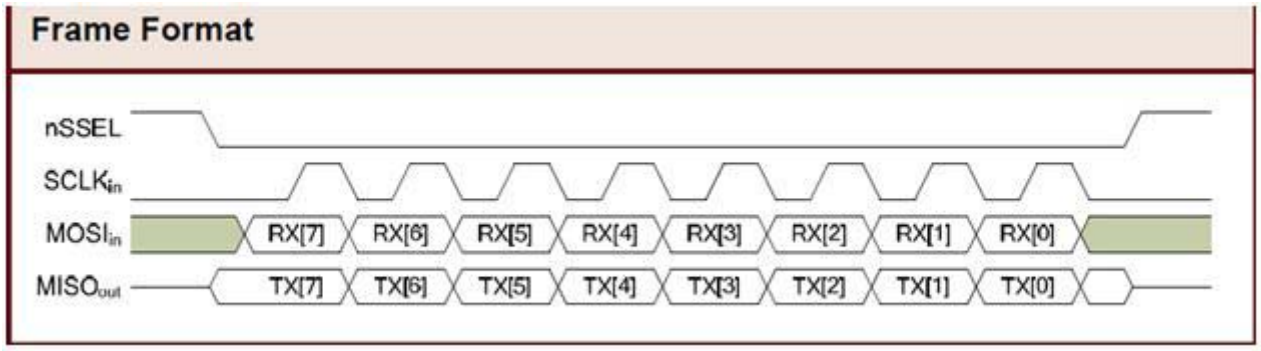
The XBee modules support SPI communications in slave mode. Slave mode receives the clock signal and data from the master and returns data to the master. The SPI port uses the following signals on the XBee:

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

In this mode, the following apply:

- SPI Clock rates up to 3.5 MHz are possible.
- Data is MSB first
- Frame Format mode 0 is used. This means CPOL=0 (idle clock is low) and CPHA=0 (data is sampled on the clock's leading edge). Mode 0 is diagramed below.
- SPI port is setup for API mode and is equivalent to AP=1.

Frame Format for SPI Communications



SPI Operation

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

XBee Implementation of SPI

The module operates as a SPI slave only. This means that an external master will provide the clock and will decide when to send. The 865/868LP supports an external clock rate of up to 3.5 Mbps.

Data is transmitted and received with most significant bit first using SPI mode 0. This means the CPOL and CPHA are both 0. Mode 0 was chosen because it's the typical default for most microcontrollers and would simplify configuration of the master. Further information on Mode 0 is not included in this manual, but is well-documented on the internet.

SPI Signals

The official specification for SPI includes the four signals **SPI_MISO**, **SPI_MOSI**, **SPI_CLK**, and **SPI_$\overline{\text{SSEL}}$**. Using only these four signals, the master cannot know when the slave needs to send and the SPI slave cannot transmit unless enabled by the master. For this reason, the **SPI_$\overline{\text{ATTN}}$** signal is available in the design. This allows the module to alert the SPI master that it has data to send. In turn, the SPI master is expected to assert **SPI_$\overline{\text{SSEL}}$** and start **SPI_CLK**, unless these signals are already asserted and active respectively. This, in turn, allows the XBee module to send data to the master.

The table below names the SPI signals and specifies their pinouts. It also describes the operation of each pin:

Signal Name	Pin Number	Applicable AT Command	Description
SPI_MISO (Master In, Slave out)	17	ATP5	When SPI_SSEL is asserted (low) and SPI_CLK is active, the module outputs the data on this line at the SPI_CLK rate. When SPI_SSEL is de-asserted (high), this output should be tri-stated such that another slave device can drive the line.
SPI_MOSI (Master out, Slave in)	16	ATP6	The SPI master outputs data on this line at the SPI_CLK rate after it selects the desired slave. When the module is configured for SPI operations, this pin is an input.
SPI_SSEL (Slave Select) (Master out, Slave in)	15	ATP7	The SPI master outputs a low signal on this line to select the desired slave. When the module is configured for SPI operations, this pin is an input.
SPI_CLK (Clock) (Master out, Slave in)	14	ATP8	The SPI master outputs a clock on this pin, and the rate must not exceed the maximum allowed, 3.5 Mbps. When the module is configured for SPI operations, this pin is an input.
SPI_ATTN (Attention) (Master in, Slave out)	12	ATP9	The module asserts this pin low when it has data to send to the SPI master. When this pin is configured for SPI operations, it is an output (not tri-stated).

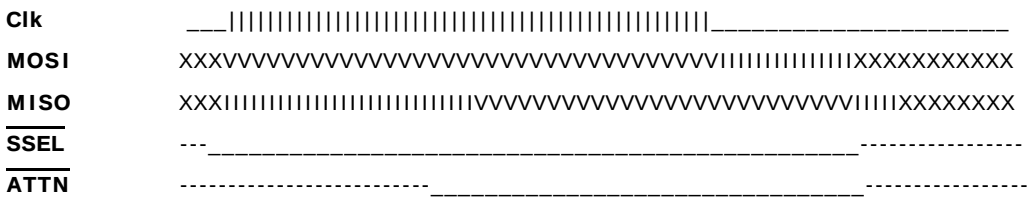
Note: By default, the inputs have pull-up resistors enabled. See the PR command to disable the pull-up resistors. When the SPI pins are not connected but the pins are configured for SPI operation, then the pull-ups are needed for proper UART operation.

Full Duplex Operation

SPI on XBee requires usage of API mode (without escaping) to packetize data. However, by design, SPI is a full duplex protocol, even when data is only available in one direction. This means that whenever data is received, it will also transmit, and that data will normally be invalid. Likewise, whenever data is transmitted, invalid data will probably be received. The means of determining whether or not received data is invalid is by packetizing the data with 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 is sending data to the slave and the slave has valid data to send in the middle of receiving data from the master, this allows a true full duplex operation 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 as specified.

An example follows to more fully illustrate the SPI interface while valid data is being sent in both directions.



In the above (character oriented) timing diagram the notations mean the following:

___	indicates a low voltage signal
-----	indicates a high voltage signal
|||	indicates an oscillating clock
XXX	indicates don't care data that will not be processed
VVV	indicates valid data that should be processed
IIIII	indicates invalid data that should be discarded

Low Power Operation

In general, sleep modes work the same on SPI as they do on UART. However, due to the addition of SPI mode, there is the option of another sleep pin, as described below:

By default, DIO8 (SLEEP_REQUEST) is configured as a peripheral and is used for pin sleep to awaken and to sleep the radio. This applies regardless of the selected serial interface (UART or SPI).

However, if SLEEP_REQUEST is not configured as a peripheral and SPI_$\overline{\text{SSEL}}$ is configured as a peripheral, then pin sleep is controlled by SPI_$\overline{\text{SSEL}}$ rather than by SLEEP_REQUEST. Asserting SPI_$\overline{\text{SSEL}}$ by driving it low either awakens the radio or keeps it awake. Negating SPI_$\overline{\text{SSEL}}$ by driving it high puts the radio to sleep.

Using SPI_$\overline{\text{SSEL}}$ for two purposes (to control sleep and to indicate that the SPI master has selected a particular slave device) has the advantage of requiring one less physical pin connection to implement pin sleep on SPI. It has the disadvantage of putting the radio to sleep whenever the SPI master negates SPI_$\overline{\text{SSEL}}$ (meaning time will be lost waiting for the device to wake), even if that wasn't the intent. Therefore, if the user has full control of SPI_$\overline{\text{SSEL}}$ so that it can control pin sleep, whether or not data needs to be transmitted, then sharing the pin may be a good option in order to make the SLEEP_REQUEST pin available for another purpose. If the radio is one of multiple slaves on the SPI, then the radio would sleep while the SPI master talks to the other slave, but this is acceptable in most cases.

If neither pin is configured as a peripheral, then the radio stays awake, being unable to sleep in SM1 mode.

Configuration

The three considerations for configuration are:

- How is the serial port selected? (I.e. Should the UART or the SPI port be used?)
- If the SPI port is used, what should be the format of the data in order to avoid processing invalid characters while transmitting?
- What SPI options need to be configured?

Serial Port Selection

In the default configuration the UART and SPI ports will both be configured for serial port operation.

If both interfaces are configured, serial data will go out the UART until the SPI_$\overline{\text{SSEL}}$ signal is asserted. Thereafter, all serial communications will operate on the SPI interface.

If only the UART is enabled, then only the UART will be used, and SPI_$\overline{\text{SSEL}}$ will be ignored. If only the SPI is enabled, then only the SPI will be used.

If neither serial port is enabled, the module will not support serial operations and all communications must occur over the air. All data that would normally go to the serial port is discarded.

Forcing UART Operation

In the rare case that a module has been configured with only the SPI enabled and no SPI master is available to access the SPI slave port, the module may be recovered to UART operation by holding DIN/$\overline{\text{CONFIG}}$ low at reset time. As always, DIN/$\overline{\text{CONFIG}}$ forces a default configuration on the UART at 9600 baud and it will bring up the module in command mode on the UART port. Appropriate commands can then be sent to the module to configure it for UART operation. If those parameters are written, then the module will come up with the UART enabled, as desired on the next reset.

Data Format

The SPI will only operate in API mode 1. Neither transparent mode nor API mode 2 (which escapes control characters) will be supported. This means that the AP configuration only applies to the UART and will be ignored while using the SPI.

SPI Parameters

Most host processors with SPI hardware allow the bit order, clock phase and polarity to be set. For communication with all XBee radios the host processor must set these options as follows:

- Bit Order - send MSB first
- Clock Phase (CPHA) - sample data on first (leading) edge
- Clock Polarity (CPOL) - first (leading) edge rises

This is SPI Mode 0 and MSB first for all XBee radios. Mode 0 means that data is sampled on the leading edge and that the leading edge rises. MSB first means that bit 7 is the first bit of a byte sent over the interface.

Serial Buffers

To enable the UART port, DIN and DOUT must be configured as peripherals. To enable the SPI port, SPI_MISO, SPI_MOSI, SPI_$\overline{\text{SSEL}}$, and SPI_CLK must be enabled as peripherals. If both ports are enabled then output will go to the UART until the first input on SPI.

In the default configuration, the UART and the SPI ports will both be configured for I/O. Initially, all serial data will go out the UART. But, as soon as input occurs on either port, that port is selected as the active port and no input or output will be allowed on the other port until the next reset of the module.

If the configuration is changed so that only one port is configured, then that port will be the only one enabled or used. If the parameters are written with only one port enabled, then the port that is not enabled will not even be used temporarily after the next reset.

If both ports are disabled on reset, the UART will be used in spite of the wrong configuration so that at least one serial port will be operational.

Serial Receive Buffer

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

If the SPI is the serial port, no hardware flow control is available. It is the user's responsibility to ensure that receive buffer is not overflowed. One reliable strategy is to wait for a TX_STATUS response after each frame sent to ensure that the module has had time to process it.

Serial Transmit Buffer

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

UART Flow Control

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

CTS Flow Control

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

RTS Flow Control

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

The UART Data Present Indicator is a useful feature when using RTS flow control. When enabled, the DIO19 line asserts when UART data is queued to be transmitted from the module. See the P9 command in the Command Reference Tables for more information.

Note: If the XBee is sending data out the UART when $\overline{\text{RTS}}$ is de-asserted (set high), the XBee could send up to 5 characters out the UART or SPI port after $\overline{\text{RTS}}$ is de-asserted.

Serial Interface Protocols

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

Transparent Operation - UART

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

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

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

API Operation

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

Transmit Data Frames (received through the serial port) include:

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

Receive Data Frames (sent out the serial port) include:

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

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

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

-> Transmitting data to multiple destinations without entering Command Mode

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

-> Identify the source address of each received packet

A Comparison of Transparent and API Operation

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

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

Transparent Operation Features	
Advanced networking diagnostics	API frames can provide indication of IO samples from remote devices, and node identification messages.
Remote Configuration	Set / read configuration commands can be sent to remote devices to configure them as needed using the API.

As a general rule of thumb, API mode is recommended when a device:

- sends RF data to multiple destinations
- sends remote configuration commands to manage devices in the network
- receives RF data packets from multiple devices, and the application needs to know which device sent which packet
- must support multiple endpoints, cluster IDs, and/or profile IDs
- uses the Device Profile services.

API mode is required when:

- receiving I/O samples from remote devices
- using SPI for the serial port

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

Modes of Operation

Description of Modes

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

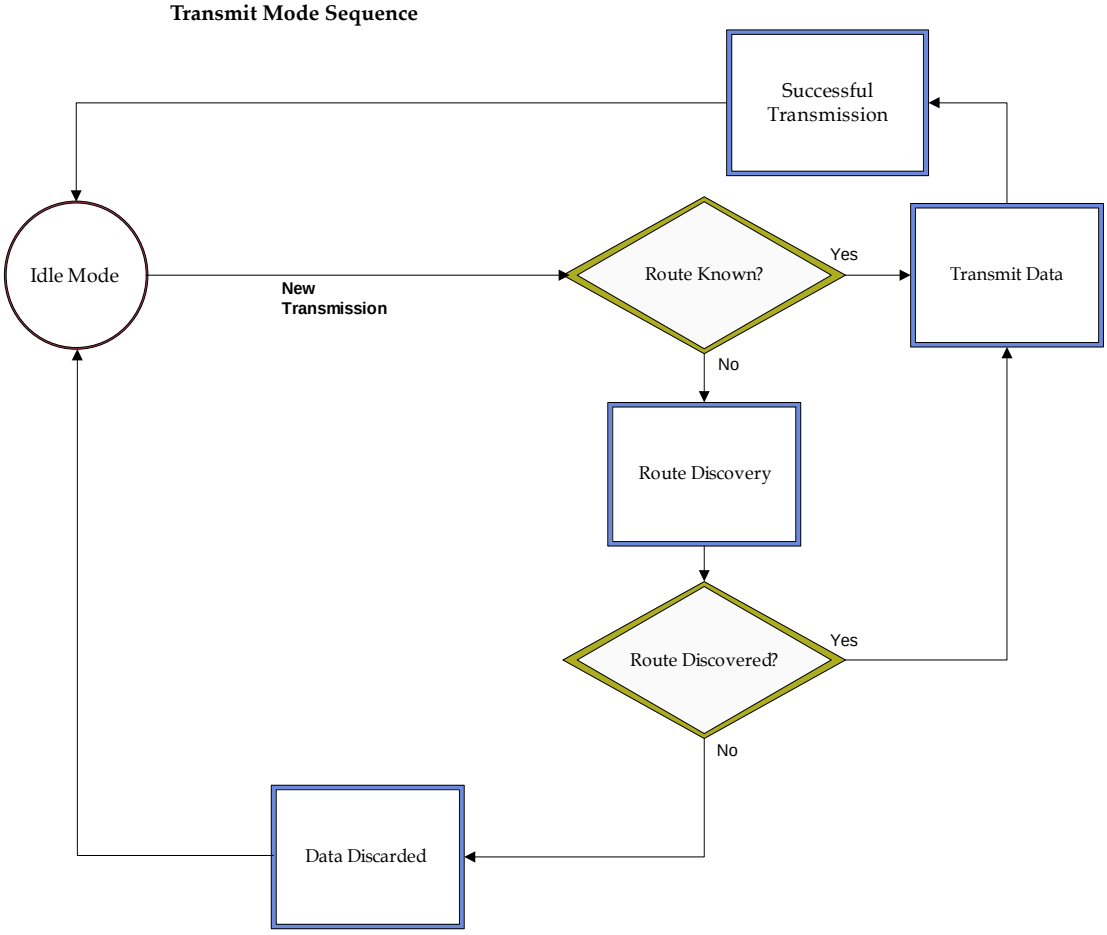
- Transmit Mode (Serial data in the serial receive buffer is ready to be packetized)
- Sleep Mode
- Command Mode (Command Mode Sequence is issued, not available when using the SPI port)

Transmit Mode

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

In the diagram below, route discovery applies only to DigiMesh transmissions. The data will be transmitted once a route is established. If route discovery fails to establish a route, the packet will be discarded.

Transmit Mode Sequence



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

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

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

Receive Mode

If a valid RF packet is received, the data is transferred to the serial transmit buffer. This is the default mode for the XBee radio.

Command Mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming serial characters are interpreted as commands. The API Mode section in Chapter 7 describes an alternate means for configuring modules which is available with the SPI, as well as over the UART with code.

AT Command Mode

To Enter AT Command Mode:

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

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

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

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

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

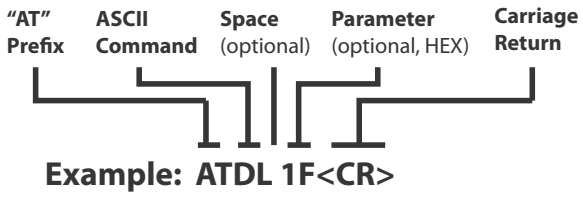
All of the parameter values in the sequence can be modified to reflect user preferences.

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

To Send AT Commands:

Send AT commands and parameters using the syntax shown below.

Syntax for sending AT Commands



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

The preceding example would change the RF module Destination Address (Low) to "0x1F". To store the new value to non-volatile (long term) memory, send the WR (Write) command. This allows modified parameter values to persist in the module's registry after a reset. Otherwise, parameters are restored to previously saved values after the module is reset.

Command Response

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

Applying Command Changes

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

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

To Exit AT Command Mode:

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

   [OR]

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

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, please see the Command Reference Table chapter.

Sleep Mode

Sleep modes allow the RF module to enter states of low power consumption when not in use. XBee RF modules support both pin sleep (sleep mode entered on pin transition) and cyclic sleep (module sleeps for a fixed time). XBee sleep modes are discussed in detail in chapter 5.

3. Advanced Application Features

Remote Configuration Commands

A module in API mode has provisions to send configuration commands to remote devices using the Remote Command Request API frame (See API Operations chapter.) This API frame can be used to send commands to a remote module to read or set command parameters.

Sending a Remote Command

To send a remote command, the Remote Command Request frame should be populated with the 64-bit address of the remote device, the correct command options value, and the command and parameter data (optional). If a command response is desired, the Frame ID should be set to a non-zero value. Only unicasts of remote commands are supported. Remote commands cannot be broadcast.

Applying Changes on Remote Devices

When remote commands are used to change command parameter settings on a remote device, parameter changes do not take effect until the changes are applied. For example, changing the BD parameter will not change the actual serial interface rate on the remote until the changes are applied. Changes can be applied using remote commands in one of three ways:

- Set the apply changes option bit in the API frame
- Issue an AC command to the remote device
- Issue a WR + FR command to the remote device to save changes and reset the device.

Remote Command Responses

If the remote device receives a remote command request transmission, and the API frame ID is non-zero, the remote will send a remote command response transmission back to the device that sent the remote command. When a remote command response transmission is received, a device sends a remote command response API frame out its UART. The remote command response indicates the status of the command (success, or reason for failure), and in the case of a command query, it will include the register value. The device that sends a remote command will not receive a remote command response frame if:

- The destination device could not be reached
- The frame ID in the remote command request is set to 0.

Network Commissioning and Diagnostics

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

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

Device Configuration

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

Network Link Establishment and Maintenance

Building Aggregate Routes

In many applications it is necessary for many or all of the nodes in the network to transmit data to a central aggregator node. In a new DigiMesh network the overhead of these nodes discovering routes to the aggregator node can be extensive and taxing on the network. To eliminate this overhead the AG command can be used to automatically build routes to an aggregate node in a DigiMesh network.

To send a unicast, modules configured for transparent mode (AP=0) must set their DH/DL registers to the MAC address of the node to which they need to transmit to. In networks of transparent mode modules which transmit to an aggregator node it is necessary to set every module's DH/DL registers to the MAC address of the aggregator node. This can be a tedious process. The AR command can be used to set the DH/DL registers of all the nodes in a DigiMesh network to that of the aggregator node in a simple and effective method.

Upon deploying a DigiMesh network the AG command can be issued on the desired aggregator node to cause all nodes in the network to build routes to the aggregator node. The command can optionally be used to automatically update the DH/DL registers to match the MAC address of the aggregator node. The AG command requires a 64-bit parameter. The parameter indicates the current value of the DH/DL registers on a module which should be replaced by the 64-bit address of the node sending the AG broadcast. If it is not desirable to update the DH/DL of the module receiving the AG broadcast then the invalid address of 0xFFFE can be used. API enabled modules will output an Aggregator Update API frame if they update their DH/DL address (see the API section of this manual for a description of the frame). All modules which receive an AG broadcast will update their routing table information to build a route to the sending module, regardless of whether or not their DH/DL address is updated. This routing information will be used for future transmissions of DigiMesh unicasts.

Example 1: To update the DH/DL registers of all modules in the network to be equal to the MAC address of an aggregator node with a MAC address of 0x0013a2004052c507 after network deployment the following technique could be employed:

1. Deploy all modules in the network with the default DH/DL of 0xFFFF.
2. Issue an ATAGFFFF command on the aggregator node.

Following the preceding sequence would result in all of the nodes in the network which received the AG broadcast to have a DH of 0x0013a200 and a DL of 0x4052c507. These nodes would have automatically built a route to the aggregator.

Example 2: To cause all nodes in the network to build routes to an aggregator node with a MAC address of 0x0013a2004052c507 without affecting the DH/DL of any nodes in the network the ATAGFFFE command should be issued on the aggregator node. This will cause an AG broadcast to be sent to all nodes in the network. All of the nodes will update their internal routing table information to contain a route to the aggregator node. None of the nodes will update their DH/DL registers (because none of the registers are set to an address of 0xFFFE).

Node Replacement

The AG command can also be used to update the routing table and DH/DL registers in the network after a module is replaced. The DH/DL registers of nodes in the network can also be updated. To update only the routing table information without affecting the DH/DL registers then the process of Example 2 above can be used. To update the DH/DL registers of the network then the method of Example 3 below can be used.

Example 3: The module with serial number 0x0013a2004052c507 was being used as a network aggregator. It was replaced with a module with serial number 0x0013a200f5e4d3b2. The AG0013a2004052c507 command should be issued on the new module. This will cause all modules which had a DH/DL register setting of 0x0013a2004052c507 to update their DH/DL register setting to the MAC address of the sending module (0x0013a200f5e4d3b2).

Device Placement

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

Link Testing

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

The configuration steps to send data to the loopback cluster ID depend on the AP setting:

AT Configuration (AP=0)

To send data to the loopback cluster ID on the data endpoint of a remote device, set the CI command value to 0x12. The SE and DE commands should be set to 0xE8 (default value). The DH and DL commands should be set to the address of the remote (0 for the coordinator, or the 64-bit address of the remote). After exiting command mode, any received serial characters will be transmitted to the remote device, and returned to the sender.

API Configuration (AP=1 or AP=2)

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

RSSI Indicators

It is possible to measure the received signal strength on a device using the DB command. DB returns the RSSI value (measured in -dBm) of the last received packet. However, this number can be misleading in 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 and should be used sparingly.

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

Device Discovery

Network Discovery

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

When a device receives the network discovery command, it waits a random time before sending its own response. The maximum time delay is set on the ND sender with the NT command. The ND originator includes its NT setting in the transmission to provide a delay window for all devices in the network. Large networks may need to increase NT to improve network discovery reliability. The default NT value is 0x82 (13 seconds).

Neighbor Polling

The neighbor poll command can be used to discover the modules which are immediate neighbors (within RF range) of a particular node. This command is useful in determining network topology and determining possible routes. The command is issued using the FN command. The FN command can be initiated locally on a node using AT command mode or by using a local AT

command request frame. The command can also be initiated remotely by sending the target node an FN command using a remote AT command request API frame.

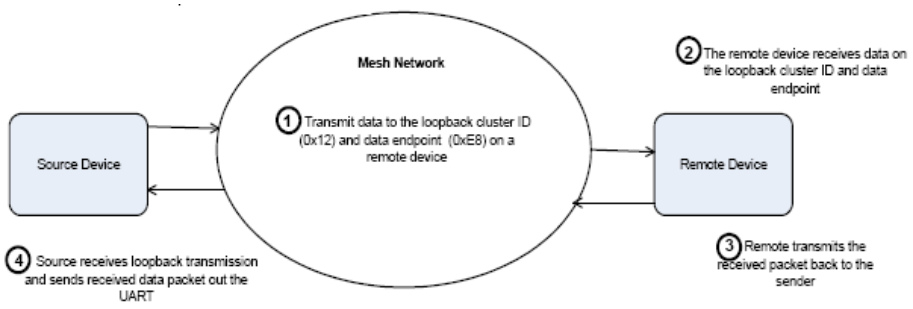
A node which executes an FN command will send a broadcast to all of its immediate neighbors. All radios which receive this broadcast will send an RF packet to the node that initiated the FN command. In the case where the command is initiated remotely this means that the responses are sent directly to the node which sent the FN command to the target node. The response packet is output on the initiating radio in the same format as a network discovery frame.

Link Reliability

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

Network Link Testing

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



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

The configuration steps to send data to the loopback cluster ID depend on the AP setting:

AT Configuration (AP=0)

To send data to the loopback cluster ID on the data endpoint of a remote device, set the CI command value to 0x12. The SE and DE commands should be set to 0xE8 (default value). The DH and DL commands should be set to the address of the remote. After exiting command mode, any received serial characters will be transmitted to the remote device, and returned to the sender.

API Configuration (AP=1 or AP=2)

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

Link Testing Between Adjacent Devices

It is often advantageous to test the quality of a link between two adjacent nodes in a network. The Test Link Request Cluster ID can be used to send a number of test packets between any two nodes in a network.

A link test can be initiated using an Explicit API command frame. The command frame should be addressed to the Test Link Request Cluster ID (0x0014) on destination endpoint 0xE6 on the radio which should execute the test link. The Explicit API command frame should contain a 12 byte payload with the following format:

Number of Bytes	Field Name	Description
8	Destination address	The address with which the radio should test its link
2	Payload size	The size of the test packet. Maximum payload size of 256 bytes
2	Iterations	The number of packets which should be sent. This should be a number between 1 and 4000.

After completing the transmissions of the test link packets the executing radio will send the following data packet to the requesting radio's Test Link Result Cluster (0x0094) on endpoint (0xE6). If the requesting radio is configured to operate in API mode then the following information will be output as an API Explicit RX Indicator Frame:

Number of Bytes	Field Name	Description
8	Destination address	The address with which the radio tested its link
2	Payload size	The size of the test packet that was sent to test the link.
2	Iterations	The number of packets which were sent.
2	Success	The number of packets successfully acknowledged.
2	Retries	The total number of MAC retries used to transfer all the packets.
1	Result	0x00 - 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 the link between radio A (SH/SL = 0x0013a20040521234) and radio B (SH/SL=0x0013a2004052abcd) is to be tested by transmitting 1000 40 byte packets. The following API packet should be sent to the UART of the radio on which the results should be output, radio C. Note that radio C can be the same radio as radio A or B (whitespace used to delineate 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

And the following is a possible packet that could be returned:

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

(999 out of 1000 packets successful, 100 retries used, RR=10, maxRSSI=-80dBm, minRSSI=-83dBm, avgRSSI=-82dBm)

If the result field is not equal to zero then an error has occurred. The other fields in the packet should be ignored. If the Success field is equal to zero then the RSSI fields should be ignored.

Trace Routing

In many applications it is useful to determine the route which a DigiMesh unicast takes to its destination. This information is especially useful when setting up a network or diagnosing problems within a network. The Trace Route API option of Tx Request Packets (see the API section of this manual for a description of the API frames) causes routing information packets to be transmitted to the originator of a DigiMesh unicast by the intermediate nodes.

When a unicast is sent with the Trace Route API option enabled, the unicast is sent to its destination radios which forward the unicast to its eventual destination will transmit a Route Information (RI) packet back along the route to the unicast originator. A full description of Route Information API packets can be found in the API section of this manual. In general they contain addressing information for the unicast and the intermediate hop for which the trace route packet was generated, RSSI information, and other link quality information.

Example:

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

- After the successful MAC transmission of the data packet from A to B, A would output a RI Packet indicating that the transmission of the data packet from A to E was successfully forwarded one hop from A to B.
- After the successful MAC transmission of the data packet from B to C, B would transmit a RI Packet to A. A would output this RI packet out its UART upon reception.
- After the successful MAC transmission of the data packet from C to D, C would transmit a RI Packet to A (through B). A would output this RI packet out its UART upon reception.
- After the successful MAC transmission of the data packet from D to E, D would transmit a RI Packet to A (through C and B). A would output this RI packet out its UART upon reception.

It is important to note that Route Information packets are not guaranteed to arrive in the same order as the unicast packet took. It is also possible for the transmission of Route Information packets on a weak route to fail before arriving at the unicast originator.

Because of the large number of Route Information packets which can be generated by a unicast with Trace Route enabled it is suggested that the Trace Route option only be used for occasional diagnostic purposes and not for normal operations.

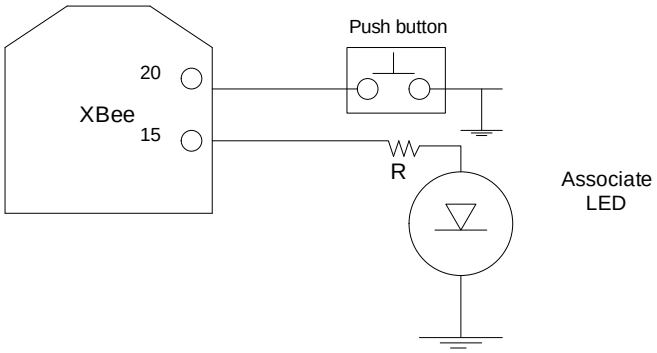
NACK Messages

The NACK API option of Tx Request Packets (see the API section of this manual for a description of the API frames) provides the option to have a Route Information packet generated and sent to the originator of a unicast when a MAC acknowledgment failure occurs on one of the hops to the destination. This information is useful because it allows marginal links to be identified and repaired.

Commissioning Pushbutton and Associate LED

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

Commissioning Pushbutton and Associate LED Functionalities



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

Commissioning Pushbutton

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

Button Presses	Sleep Configuration and Sync Status	Action
1	Not configured for sleep	Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission will blink their Associate LED rapidly for 1 second. All API devices that receive this transmission will send a Node Identification frame out their UART (API ID 0x95)
1	Configured for asynchronous sleep	Wakes the module for 30 seconds. Immediately sends a Node Identification broadcast transmission. All devices that receive this transmission will blink their Associate LED rapidly for 1 second. All API devices that receive this transmission will send a Node Identification frame out their UART (API ID 0x95).
1	Configured for synchronous sleep	Wakes the module for 30 seconds (or until the entire module goes to sleep). Queues a Node Identification broadcast transmission to be sent at the beginning of the next network wake cycle. All devices that receive this transmission will blink their Associate LEDs rapidly for 1 second. All API devices that receive this transmission will send a Node Identification frame out their UART (API ID 0x95).
2	Not configured for synchronous sleep	No effect.
2	Configured for synchronous sleep	Causes a node which is configured with sleeping router nomination enabled (see description of the ATSO – sleep options command in the XBee module's Product Manual) to immediately nominate itself as the network sleep coordinator.
4	Any	Issues an ATRE to restore module parameters to default values.

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

The node identification frame is similar to the node discovery response frame – it contains the device's address, node identifier string (NI command), and other relevant data. All API devices

that receive the node identification frame send it out their UART as an API Node Identification Indicator frame (0x95).

Associate LED

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

The Associate pin indicates the synchronization status of a sleep compatible node. On a non-sleep compatible node the pin functions as a power indicator. The following table describes this functionality.

The LT command can be used to override the blink rate of the Associate pin. When set to 0, the device uses the default blink time (500ms for sleep coordinator, 250ms otherwise).

Sleep mode	LED Status	Meaning
0	On, blinking	The device is powered and operating properly.
1, 4, 5	Off	The device is in a low power mode.
1, 4, 5	On, blinking	The device is powered, awake and is operating properly.
7	On, solid	The network is asleep or the device has not synchronized with the network or has lost synchronization with the network.
7, 8	On, slow blinking (500 ms blink time)	The device is acting as the network sleep coordinator and is operating properly.
7, 8	On, fast blinking (250 ms blink time)	The device is properly synchronized with the network.
8	Off	The device is in a low power mode.
8	On, solid	The deviced has not synchronized or has lost synchronization with the network.

Diagnostics Support

The Associate pin works with the commissioning pushbutton to provide additional diagnostic behaviors to aid in deploying and testing a network. If the commissioning push button is pressed once the device transmits a broadcast node identification packet at the beginning of the next wake cycle if sleep compatible, or immediately if not sleep compatible. If the Associate LED functionality is enabled (D5 command), a device that receive this transmission will blink its Associate pin rapidly for 1 second.

I/O Line Monitoring

I/O Samples

The XBee modules support both analog input and digital IO line modes on several configurable pins.

Queried Sampling

Parameters for the pin configuration commands typically include the following:

Pin Command Parameter	Description
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionalities.

Pin Command Parameter	Description
2	Analog input (A/D pins) or PWM output (PWM pins)
3	Digital input, monitored.
4	Digital output, low.
5	Digital output, high.
7	Alternate functionalities, where applicable.

Setting the configuration command that corresponds to a particular pin will configure the pin:

Module Pin Names	Module Pin Number	Configuration Command
CD / DIO12	4	P2
PWM0 / RSSI / DIO10	6	P0
PWM1 / DIO11	7	P1
DTR / SLEEP_RQ / DIO8	9	D8
AD4 / DIO4	11	D4
CTS / DIO7	12	D7
ON_SLEEP / DIO9	13	D9
ASSOC / AD5 / DIO5	15	D5
RTS / DIO6	16	D6
AD3 / DIO3	17	D3
AD2 / DIO2	18	D2
AD1 / DIO1	19	D1
AD0 / DIO0 / CommissioningButton	20	D0

See the command table for more information. Pullup resistors for each digital input can be enabled using the PR command.

1	Sample Sets	Number of sample sets in the packet. (Always set to 1.)
2	Digital Channel Mask	Indicates which digital IO lines have sampling enabled. Each bit corresponds to one digital IO line on the module. • bit 0 = AD0/DIO0 • bit 1 = AD1/DIO1 • bit 2 = AD2/DIO2 • bit 3 = AD3/DIO3 • bit 4 = DIO4 • bit 5 = ASSOC/DIO5 • bit 6 = RTS/DIO6 • bit 7 = CTS/GPIO7 • bit 8 = DTR / SLEEP_RQ / DIO8 • bit 9 = ON_SLEEP / DIO9 • bit 10 = RSSI/DIO10 • bit 11 = PWM/DIO11 • bit 12 = CD/DIO12 For example, a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital IO.

1	Analog Channel Mask	Indicates which lines have analog inputs enabled for sampling. Each bit in the analog channel mask corresponds to one analog input channel. • bit 0 = AD0/DIO0 • bit 1 = AD1/DIO1 • bit 2 = AD2/DIO2 • bit 3 = AD3/DIO3 • bit 4 = AD4/DIO4 • bit 5 = ASSOC/AD5/DIO5
Variable	Sampled Data Set	If any digital IO lines are enabled, the first two bytes of the data set indicate the state of all enabled digital IO. Only digital channels that are enabled in the Digital Channel Mask bytes have any meaning in the sample set. If no digital IO are enabled on the device, these 2 bytes will be omitted. Following the digital IO data (if any), each enabled analog channel will return 2 bytes. The data starts with AIN0 and continues sequentially for each enabled analog input channel up to AIN5.

If the IS command is issued from AT command mode then a carriage return delimited list will be returned containing the above-listed fields. If the command is issued via an API frame then the module will return an AT command response API frame with the IO data included in the command data portion of the packet.

Example	Sample AT Response
0x01\r	[1 sample set]
0x0C0C\r	[Digital Inputs: DIO 2, 3, 10, 11 enabled]
0x03\r	[Analog Inputs: A/D 0, 1 enabled]
0x0408\r	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0\r	[Analog input ADIO 0= 0x3D0]
0x0124\r	[Analog input ADIO 1=0x120]

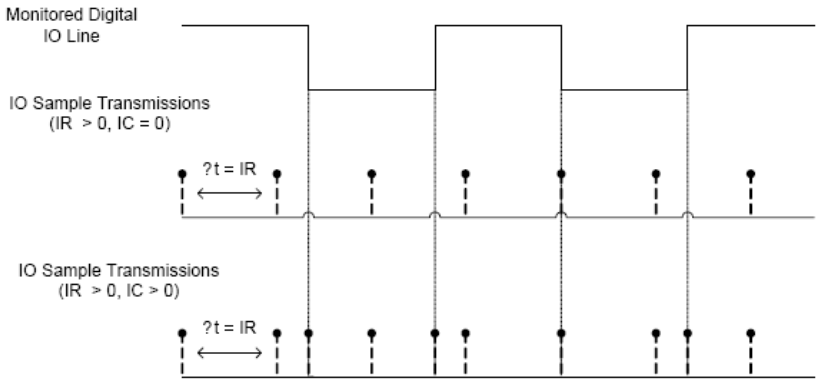
Periodic I/O Sampling

Periodic sampling allows an XBee-PRO module to take an I/O sample and transmit it to a remote device at a periodic rate. The periodic sample rate is set by the IR command. If IR is set to 0, periodic sampling is disabled. For all other values of IR, data will be sampled after IR milliseconds have elapsed and transmitted to a remote device. The DH and DL commands determine the destination address of the IO samples. Only devices with API mode enabled will send IO data samples out their UART. Devices not in API mode will discard received IO data samples.

A module with sleep enabled will transmit periodic I/O samples at the IR rate until the ST time expires and the device can resume sleeping. See the sleep section for more information on sleep.

Digital I/O Change Detection

Modules can be configured to transmit a data sample immediately whenever a monitored digital I/O pin changes state. The IC command is a bitmask that can be used to set which digital I/O lines should be monitored for a state change. If one or more bits in IC is set, an I/O sample will be transmitted as soon as a state change is observed in one of the monitored digital I/O lines. The figure below shows how edge detection can work with periodic sampling.



Enabling Edge Detection will force an immediate sample of all monitored digital IO lines if any digital IO lines change state.

General Purpose Flash Memory

XBee 868LP modules provide 119 512-byte blocks of flash memory which can be read and written by the user application. This memory provides a non-volatile data storage area which can be used for a multitude of purposes. Some common uses of this data storage include: storing logged sensor data, buffering firmware upgrade data for a host microcontroller, or storing and retrieving data tables needed for calculations performed by a host microcontroller. The General Purpose Memory (GPM) is also used to store a firmware upgrade file for over-the-air firmware upgrades of the XBee module itself.

Accessing General Purpose Flash Memory

The GPM of a target node can be accessed locally or over-the-air by sending commands to the MEMORY_ACCESS cluster ID (0x23) on the DIGI_DATA endpoint (0xE6) of the target node using explicit API frames. (Explicit API frames are described in section XXXX)

To issue a GPM command the payload of an explicit API frame should be formatted in the following way:

Byte Offset in Payload	Number of Bytes	Field Name	General Field Description
0	1	GPM_CMD_ID	Specific GPM commands are described below
1	1	GPM_OPTIONS	Command-specific 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
7	varies	GPM_DATA	

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

When a GPM command is sent to a radio via a unicast the receiving radio will unicast a response back to the requesting radio's source endpoint specified in the request packet. No response is sent for broadcast requests. If the source endpoint is set to the DIGI_DATA endpoint (0xE6) or

explicit API mode is enabled on the requesting radio then a GPM response will be output as an explicit API RX indicator frame on the requesting node (assuming API mode is enabled.)

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

Byte Offset in Payload	Number of Bytes	Field Name	General Field Description
0	1	GPM_CMD_ID	This field will be 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
7	varies	GPM_DATA	

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

The following commands exist for interacting with GPM:

PLATFORM_INFO_REQUEST (0x00):

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

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to PLATFORM_INFO_REQUEST (0x00)
GPM_OPTIONS	This field is unused for this command. Set to 0.
GPM_BLOCK_NUM	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 will send a response in the following format to the source endpoint specified in the requesting frame.

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

Example:

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

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

ERASE (0x01):

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

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 will send a response in the following format to the source endpoint specified in the requesting frame.

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to ERASE_RESPONSE (0x81)
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	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 an ERASE packet should be formatted as follows (spaces added to delineate fields):

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

WRITE (0x02) and **ERASE_THEN_WRITE (0x03):**

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

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to WRITE (0x02) or ERASE_THEN_WRITE (0x03)
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be written.
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. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. It is also important to remember that the number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the radio. The maximum payload size can be queried with the NP AT command.
GPM_DATA	The data to be written.

WRITE _RESPONSE (0x82) and **ERASE_THEN_WRITE_RESPONSE(0x83):**

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

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to WRITE_RESPONSE (0x82) or ERASE_THEN_WRITE_RESPONSE (0x83)
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
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 UART:

7E 0007 8B 01 FFFE 00 00 00 76

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

READ (0x04):

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

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to READ (0x04)
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	Set to the index of the GPM block that should be read.
GPM_START_INDEX	Set to the byte index within the GPM block where the given data should be read.
GPM_NUM_BYTES	Set to the number of data bytes to be read. Only one GPM block can be operated on per command. For this reason, GPM_START_INDEX + GPM_NUM_BYTES cannot be greater than the GPM block size. It is also important to remember that the number of bytes sent in an explicit API frame (including the GPM command fields) cannot exceed the maximum payload size of the radio. The maximum payload size can be queried with the NP AT command.
GPM_DATA	No data bytes should be specified for this command.

READ _RESPONSE (0x84):

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

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to READ_RESPONSE (0x84)
GPM_STATUS	A 1 in the least significant bit indicates an error occurred. All other bits are reserved at this time.
GPM_BLOCK_NUM	Matches the parameter passed in the request frame.
GPM_START_INDEX	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 UART:

7E 0007 8B 01 FFFE 00 00 00 76

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

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

The FIRMWARE_VERIFY and FIRMWARE_VERIFY_AND_INSTALL commands are used when remotely updating firmware on a module. Remote firmware upgrades are covered in detail in section XXXXX. These commands check if the General Purpose Memory contains a valid over-the-

air update file. For the FIRMWARE_VERIFY_AND_INSTALL command, if the GPM contains a valid firmware image then the module will reset and begin using the new firmware.

Field Name	Command-Specific Description
GPM_CMD_ID	Should be set to FIRMWARE_VERIFY (0x05) or FIRMWARE_VERIFY_AND_INSTALL (0x06)
GPM_OPTIONS	There are currently no options defined for this command. Set this field to 0.
GPM_BLOCK_NUM	This field is unused for this command. Set to 0.
GPM_START_INDEX	This field is unused for this command. Set to 0.
GPM_NUM_BYTES	This field is unused for this command. Set to 0.
GPM_DATA	This field is unused for this command.

FIRMWARE_VERIFY _RESPONSE (0x85):

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

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

FIRMWARE_VERIFY _AND_INSTALL_RESPONSE (0x86):

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

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

Example:

To verify a firmware image previously loaded into the GPM on a target radio with serial number of 0x0013a200407402ac a FIRMWARE_VERIFY packet should be formatted as follows (spaces added to delineate fields):

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

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

7E 0007 8B 01 FFFE 00 00 00 76

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

Working with Flash Memory

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

- Flash memory write operations are only capable of changing binary 1's to binary 0's. Only the erase operation can change binary 0's to binary 1's. For this reason it is usually necessary to erase a flash block before performing a write operation.
- A flash memory block must be erased in its entirety when performing an erase operation. A block cannot be partially erased.
- Flash memory has a limited lifetime. The flash memory on which the GPM is based is rated at 20,000 erase cycles before failure. Care must be taken to ensure that the frequency of erase/write operations allows for the desired product lifetime. Digi's warranty will not cover products whose number of erase cycles has been exceeded.
- Over-the-Air firmware upgrades (described in the next section) require the entire GPM be erased. Any user data stored in the GPM will be lost during an over-the-air upgrade.

Over-the-Air Firmware Upgrades

XBee DigiMesh 868 modules provide two methods of updating the firmware on the module. Firmware can be updated locally via X-CTU (a free testing and configuration utility provided by Digi) using the radio's serial port interface. Firmware can also be updated using the radios' RF interface (Over-the-Air Updating.)

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

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

Distributing the New Application

The first phase of performing an over-the-air upgrade on a module is transferring the new firmware file to the target node. The new firmware image should be loaded in the target node's GPM prior to installation. XBee DigiMesh 868 modules use an encrypted binary (.ebin) file for both serial and over-the-air firmware upgrades. These firmware files are available on the Digi Support website.

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

Example:

XBee DigiMesh 868 firmware version 80xx has an .ebin file of 55,141 bytes in length. Based on network traffic it was determined that sending a 128 byte packet every 30 seconds minimized network disruption. For this reason the .ebin should be divided and addressed as follows:

GPM_BLOCK_NUM	GPM_START_INDEX	GPM_NUM_BYTES	.ebin bytes
0	0	128	0 to 127
0	128	128	128 to 255
0	256	128	256 to 383
0	384	128	384 to 511
1	0	128	512 to 639
1	128	128	640 to 767
–	-	-	-
–	-	-	-
–	-	-	-
107	0		54784 to 54911
107	128		54912 to 55039
107	256	101	55040 to 55140

Verifying the New Application

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

Installing the Application

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

Things to Remember

- The firmware upgrade process requires that the module resets itself. Because of this reset parameters which have not been written to flash will be lost after the reset. To avoid this, write all parameters with the WR command before doing a firmware upgrade. Packet routing information will also be lost after this reset. Route discoveries will be necessary for DigiMesh unicasts involving the upgraded 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 module in either case.

4. Networking Methods

Directed Broadcast / Repeater Mode

In this broadcast mode, the exact transmission method is determined by the data rate of your module. In the 10k version, the network is set in a repeater mode, where there is no route discovery. The transmission is simply sent out to the network and each radio will repeat the message to its neighboring radios. There is no route discovery in this method. On the 80k version of the module, the transmission is directed to a specific IP address, using a route discovered by a router module. In both methods, all transmissions are broadcasts, not unicast messages.

Point to Point / Multipoint

In this mode, there is a permanent link between two endpoints. Switched point-to-point topologies are the basic model of conventional telephony. The value of a permanent point-to-point network is unimpeded communications between the two endpoints. The value of an on-demand point-to-point connection is proportional to the number of potential pairs of subscribers.

Permanent (dedicated)

One of the variations of point-to-point topology is a point-to-point communications channel that appears, to the user, to be permanently associated with the two endpoints. Within many switched telecommunications systems, it is possible to establish a permanent circuit. One example might be a telephone in the lobby of a public building, which is programmed to ring only the number of a telephone dispatcher. "Nailing down" a switched connection saves the cost of running a physical circuit between the two points. The resources in such a connection can be released when no longer needed.

Switched:

Using circuit-switching or packet-switching technologies, a point-to-point circuit can be set up dynamically, and dropped when no longer needed.

DigiMesh Networking

Mesh networking allows messages to be routed through several different nodes to a final destination. DigiMesh firmware allows manufacturers and system integrators to bolster their networks with the self-healing attributes of mesh networking. In the event that one RF connection between nodes is lost (due to power-loss, environmental obstructions, etc.) critical data can still reach its destination due to the mesh networking capabilities embedded inside the modules.

DigiMesh Feature Set

DigiMesh contains the following features

- **Self-healing**
  Any node may enter or leave the network at any time without causing the network as a whole to fail.
- **Peer-to-peer architecture**
  No hierarchy and no parent-child relationships are needed.
- **Quiet Protocol**
  Routing overhead will be reduced by using a reactive protocol similar to AODV.
- **Route Discovery**
  Rather than maintaining a network map, routes will be discovered and created only when needed.
- **Selective acknowledgements**
  Only the destination node will reply to route requests.
- **Reliable delivery**
  Reliable delivery of data is accomplished by means of acknowledgements.
- **Sleep Modes**
  Low power sleep modes with synchronized wake are supported with variable sleep and wake times.

Networking Concepts

Device Configuration

DigiMesh modules can be configured to act as routers or end devices with the CE command. By default all modules in a DigiMesh network act as routers. Modules configured as routers will actively relay network unicast and broadcast traffic as described below.

Network ID

DigiMesh networks are defined with a unique network identifier. This identifier is set with the ID command. For modules to communicate they must be configured with the same network identifier. The ID parameter allows multiple DigiMesh networks to co-exist on the same physical channel.

Data Transmission and Routing

Unicast Addressing

When transmitting while using Unicast communications, reliable delivery of data is accomplished using retries and acknowledgements. The number of retries is determined by the NR (Network Retries) parameter. RF data packets are sent up to NR + 1 times and ACKs (acknowledgements) are transmitted by the receiving node upon receipt. If a network ACK is not received within the time it would take for a packet to traverse the network twice, a retransmission occurs.

To send Unicast messages, set the DH and DL on the transmitting module to match the corresponding SH and SL parameter values on the receiving module.

Broadcast Addressing

Broadcast transmissions will be received and repeated by all routers in the network. Because ACKs are not used the originating node will send the broadcast multiple times. By default a broadcast transmission is sent four times. Essentially the extra transmissions become automatic retries without acknowledgments. This will result in all nodes repeating the transmission four times as well. In order to avoid RF packet collisions, a random delay is inserted before each router relays the broadcast message. (See NN parameter for details on changing this random delay time.) Sending frequent broadcast transmissions can quickly reduce the available network bandwidth and as such should be used 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 and DL to 0xFFFF. In API mode the destination address would be set to 0x000000000000FFFF.

Routing

A module within a mesh network is able to determine reliable routes using a routing algorithm and table. The routing algorithm uses a reactive method derived from AODV (Ad-hoc On-demand Distance Vector). An associative routing table is used to map a destination node address with its next hop. By sending a message to the next hop address, either the message will reach its destination or be forwarded to an intermediate router which will route the message on to its destination. A message with a broadcast address is broadcast to all neighbors. All routers receiving the message will rebroadcast the message MT+1 times and eventually the message will reach all corners of the network. Packet tracking prevents a node from resending a broadcast message more than MT+1 times.

Route Discovery

If the source node doesn't have a route to the requested destination, the packet is queued to await a route discovery (RD) process. This process is also used when a route fails. A route fails when the source node uses up its network retries without ever receiving an ACK. This results in the source node initiating RD.

RD begins by the source node broadcasting a route request (RREQ). Any router that receives the RREQ that is not the ultimate destination is called 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, information from the RREQ is saved and the RREQ is updated and broadcast. When the ultimate destination receives the RREQ, it unicasts a route reply (RREP) back to the source node along the path of the RREQ. This is done regardless of route quality and regardless of how many times an RREQ has been seen 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 will use for the queued packet and for subsequent packets with the same destination address.

Throughput

Throughput in a DigiMesh network can vary by a number of variables, including: number of hops, encryption enabled/disabled, sleeping end devices, failures/route discoveries. Our empirical testing showed the following throughput performance in a robust operating environment (low interference).

80 kbps version, 115.2 kbps serial data rate, 100 KB

Configuration	Data Throughput
Mesh unicast, 1 hop, Encryption Disabled	35.6 kbps
Mesh unicast, 3 hop, Encryption Disabled	11.9 kbps
Mesh unicast, 6 hop, Encryption Disabled	7.1 kbps
Mesh unicast, 1 hop, Encryption Enabled	35.3 kbps
Mesh unicast, 3 hop, Encryption Enabled	11.8 kbps
Mesh unicast, 6 hop, Encryption Enabled	7.0 kbps
Point to point unicast, Encryption Disabled	54.7 kbps
Point to point unicast, Encryption Enabled	53.9 kbps

10 kbps version, 115.2 kbps serial data rate, 100 KB

Configuration	Data Throughput
Point to point unicast, Encryption Disabled	8.4 kbps
Point to point unicast, Encryption Enabled	8.3 kbps

Note: Data throughput measurements were made setting the serial interface rate to 115200 bps, and measuring the time to send 100,000 bytes from source to destination. During the test, no route discoveries or failures occurred.

Transmission Timeouts

When a node receives an API TX Request (API configured modules) or an RO timeout occurs (modules configured for Transparent Mode) the time required to route the data to its destination depends on a number of configured parameters, whether the transmission is a unicast or a broadcast, and if the route to the destination address is known. Timeouts or timing information is provided for the following transmission types:

- Transmitting a broadcast
- Transmitting a unicast with a known route
- Transmitting a unicast with an unknown route
- Transmitting a unicast with a broken route.

Note: The timeouts in this section are theoretical timeouts and not precisely accurate. The application should pad the calculated maximum timeouts by a few hundred milliseconds. When using API mode, Tx Status API packets should be the primary method of determining if a transmission has completed.

Unicast One Hop Time

A building block of many of the calculations presented below is the unicastOneHopTime. As its name indicates, it represents the amount of time it takes to send a unicast transmission between two adjacent nodes. It is dependent upon the %H setting. It is defined as follows:

unicastOneHopTime = %H

Transmitting a broadcast

A broadcast transmission must be relayed by all routers in the network. The maximum delay would be when the sender and receiver are on the opposite ends of the network. The NH and %H parameters define the maximum broadcast delay as follows:

BroadcastTxTime = NH * %8

Transmitting a unicast with a known route

When a route to a destination node is known the transmission time is largely a function of the number of hops and retries. The timeout associated with a unicast assumes the maximum number of hops is necessary (as specified by NH). The timeout can be estimated in the following manner:

knownRouteUnicast=2*NH*MR*unicastOneHopTime

Transmitting a unicast with an unknown route

If the route to the destination is not known the transmitting module will begin by sending a route discovery. If the route discovery is successful and a route is found then the data is transmitted. The timeout associated with the entire operation can be estimated as follows:

unknownRouteUnicast=BroadcastTxTime+NH*unicastOneHopTime +knownRouteUnicast

Transmitting a unicast with a broken route

If the route to a destination node has changed since the last time a route discovery was completed a node will begin by attempting to send the data along the previous route. After it fails a route discovery will be initiated and, upon completion of the route discovery, the data will be transmitted along the new route. The timeout associated with the entire operation can be estimated as follows:

brokenRouteUnicast=BroadcastTxTime+NH*unicastOneHopTime +2*knownRouteUnicast

5. Sleep Mode

A number of low-power modes exist to enable modules to operate for extended periods of time on battery power. These sleep modes are enabled with the SM command. The sleep modes are characterized as either asynchronous (SM = 1, 4, 5) or synchronous (SM = 7,8). Asynchronous sleeping modes should not be used in a synchronous sleeping network, and vice versa.

Asynchronous sleep modes can be used to control the sleep state on a module by module basis. Modules operating in an asynchronous sleep mode should not be used to route data. Digi strongly encourages users to set asynchronous sleeping modules as end-devices using the CE command. This will prevent the node from attempting to route data.

The synchronous sleep feature of DigiMesh makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time. This forms a cyclic sleeping network. Nodes synchronize by receiving a special RF packet called a sync message which is sent by a node acting as a sleep coordinator. A node in the network can become a coordinator through a process called nomination. The sleep coordinator will send one sync message at the beginning of each wake period. The sync message is sent as a broadcast and repeated by every node in the network. The sleep and wake times for the entire network can be changed by locally changing the settings on an individual node. The network will use the most recently set sleep settings.

Sleep Modes

Normal Mode (SM=0)

Normal mode is the default for a newly powered-on node. In this mode, a node will not sleep. Normal mode nodes should be mains-powered.

A normal mode module will synchronize to a sleeping network, but will not observe synchronization data routing rules (it will route data at any time, regardless of the wake state of the network). When synchronized, a normal node will relay sync messages generated by sleep-compatible nodes but will not generate sync messages. Once a normal node has synchronized with a sleeping network, it can be put into a sleep-compatible sleep mode at any time.

Asynchronous Pin Sleep Mode (SM=1)

Pin sleep allows the module to sleep and wake according to the state of the **Sleep_RQ** pin (pin 9). Pin sleep mode is enabled by setting the SM command to 1. When **Sleep_RQ** is asserted (high), the module will finish any transmit or receive operations and enter a low-power state. The module will wake from pin sleep when the **Sleep_RQ** pin is de-asserted (low).

Asynchronous Cyclic Sleep Mode (SM=4)

Cyclic sleep allows the module to sleep for a specified time and wake for a short time to poll. Cyclic sleep mode is enabled by setting the SM command to 4. In cyclic sleep, the module sleeps for a specified time. If the XBee receives serial or RF data while awake, it will then extend the time before it returns to sleep by the amount specified by the ST command. Otherwise, it will enter sleep mode immediately. The **On_SLEEP** line is asserted (high) when the module wakes, and is de-asserted (low) when the module sleeps. If hardware flow control is enabled (D7 command), the **CTS** pin will assert (low) when the module wakes and can receive serial data, and de-assert (high) when the module sleeps.

Asynchronous Cyclic Sleep with Pin Wake Up Mode (SM=5)

(SM=5) is a slight variation on (SM=4) that allows the module to be woken prematurely by asserting the **Sleep_RQ** pin (pin 9). In (SM=5), the XBee can wake after the sleep period expires, or if a high-to-low transition occurs on the **Sleep_RQ** pin.

Synchronous Sleep Support Mode (SM=7)

A node in synchronous sleep support mode will synchronize itself with a sleeping network but will not itself sleep. At any time, the node will respond to new nodes which are attempting to join the sleeping network with a sync message. A sleep support node will only transmit normal data when the other nodes in the sleeping network are awake. Sleep support nodes are especially useful when used as preferred sleep coordinator nodes and as aids in adding new nodes to a sleeping network.

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

Synchronous Cyclic Sleep Mode (SM=8)

A node 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 read commands fromt the UART port. Generally, sleep and wake times are specified by the SP and ST respectively of the network's sleep coordinator. These parameters are only used at start up until the node is synchronized with the network. When a module has synchronized with the network, its sleep and wake times can be queried with the OS and OW commands respectively. If D9 = 1 (**ON_SLEEP** enabled) on a cyclic sleep node, the **ON_SLEEP** line will assert when the module is awake and de-assert when the module is asleep. **CTS** is also de-asserted while asleep (D7 = 1). A newly-powered unsynchronized sleeping node will poll for a synchronized message and then sleep for the period specified by SP, repeating this cycle until it becomes synchronized by receiving a sync message. Once a sync message is received, the node will synchronize itself with the network.

Note: All nodes in a synchronous sleep network should be configured to operate in either Synchronous Sleep Support Mode or Synchronous Cyclic Sleep Mode. Asynchronous sleeping nodes are not compatible with synchronous sleep nodes.

Asynchronous Sleep Operation

Wake Timer

In cyclic sleep mode (SM=4 or SM=5), if serial or RF data is received, the module will start a sleep timer (time until sleep). Any data received serially or by RF link will reset the timer. The timer duration can be set using the ST command. The module returns to sleep when the sleep timer expires.

Sleeping Routers

The Sleeping Router feature of DigiMesh makes it possible for all nodes in the network to synchronize their sleep and wake times. All synchronized cyclic sleep nodes enter and exit a low power state at the same time. This forms a cyclic sleeping network. Nodes synchronize by receiving a special RF packet called a sync message which is sent by a node acting as a sleep coordinator. A node in the network can become a sleep coordinator through a process called nomination. The sleep coordinator will send one sync message at the beginning of each wake period. The sync message is sent as a broadcast and repeated by every node in the network. The sleep and wake times for the entire network can be changed by locally changing the settings on an individual node. The network will use the most recently set sleep settings.

Operation

One node in a sleeping network acts as the sleeping coordinator. The process by which a node becomes a sleep coordinator is described later in this document. During normal operations, at the beginning of a wake cycle the sleep coordinator will send 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 receiving a sync message will remain awake for the wake time and then sleep for the sleep period specified.

The sleep coordinator will send one sync message at the beginning of each cycle with the currently configured wake and sleep times. All router nodes which receive this sync message will relay the message to the rest of the network. If the sleep coordinator does not hear a re-broadcast of the sync message by one of its immediate neighbors then it will re-send the message one additional time. It should be noted that if SP or ST are changed, the network will not apply the new settings until the beginning of the next wake time. See the Changing Sleep Parameters section below for more information.

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 in the wake time is reduced to maintain synchronization accuracy. By default, a module will also reduce its active sleep time progressively as sync messages are missed.

Synchronization Messages

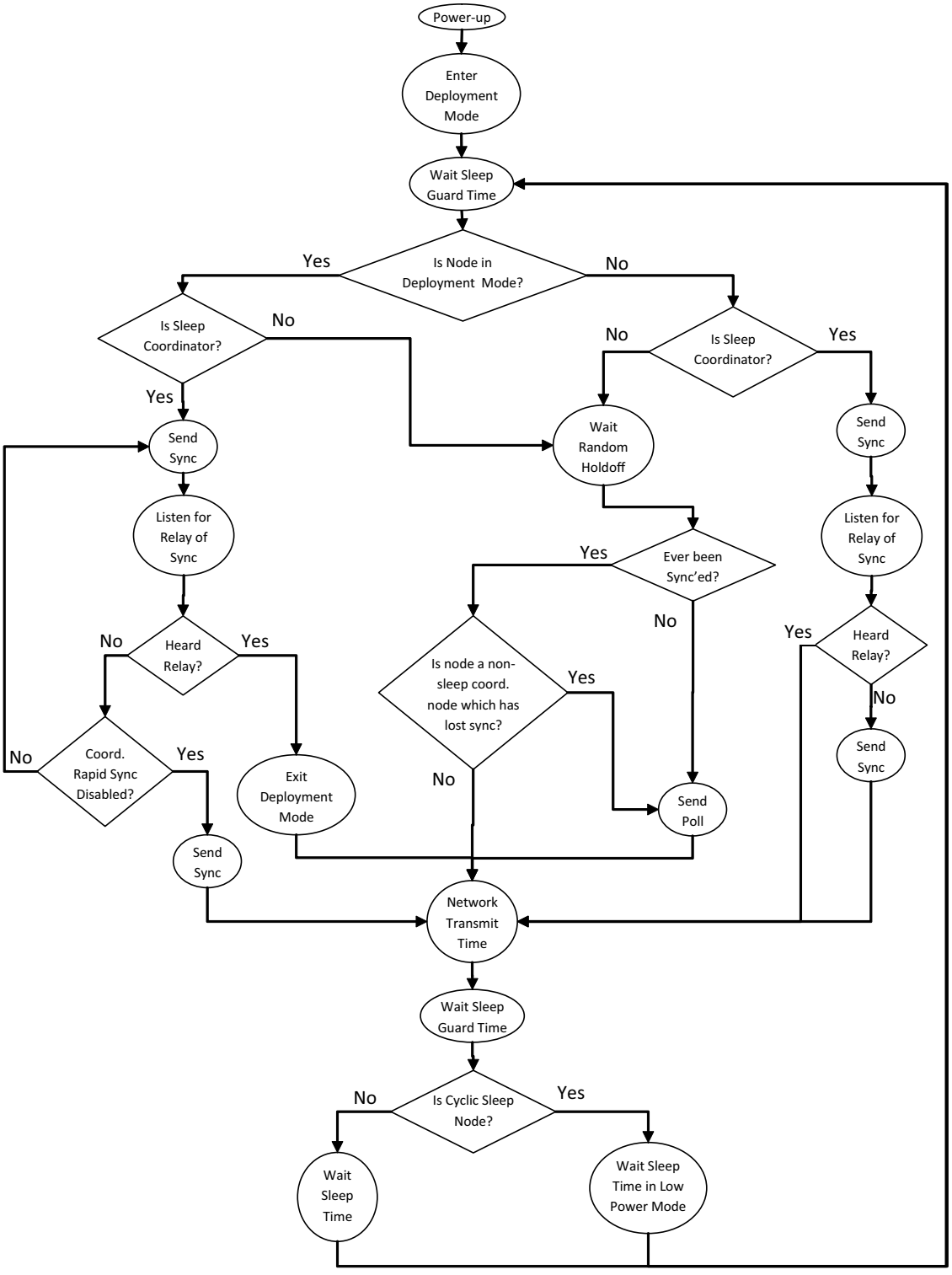
A sleep coordinator will regularly send sync messages to keep the network in sync. Nodes which have not been synchronized or, in some cases, which have lost sync will also send messages requesting sync information.

Deployment mode is used by sleep compatible nodes when they are first powered up and the sync message has not been relayed. A sleep coordinator in deployment mode will rapidly send sync messages until it receives a relay of one of those messages. This allows a network to be deployed more effectively and allows a sleep coordinator which is accidentally or intentionally reset to rapidly re-synchronize with the rest of the network. If a node which has exited deployment mode receives a sync message from a sleep coordinator which is in deployment mode, the sync will be rejected and a corrective sync will be sent to the sleep coordinator. Deployment mode can be disabled using the sleep options command (SO).

A sleep coordinator which is not in deployment mode or which has had deployment mode disabled will send a sync message at the beginning of the wake cycle. The sleep coordinator will then listen for a neighboring node to relay the sync. If the relay is not heard, the sync coordinator will send the sync one additional time.

A node which is not acting as a sleep coordinator which has never been synchronized will send a message requesting sync information at the beginning of its wake cycle. Synchronized nodes which receive one of these messages will respond with a synchronization packet. Nodes which are configured as non-sleep coordinators (using the SO command) which have gone six or more cycles without hearing a sync will also send a message requesting sync at the beginning of their wake period.

The following diagram illustrates the synchronization behavior of sleep compatible modules:


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

Becoming a Sleep Coordinator

A node can become a sleep coordinator in one of four ways:

Preferred Sleep Coordinator Option

A node can be specified to always act as a sleep coordinator. This is done by setting the preferred sleep coordinator bit (bit 0) in the sleep operations parameter (SO) to 1. A node with the sleep coordinator bit set will always send a sync message at the beginning of a wake cycle. For this reason, it is imperative that no more than one node in the network has this bit set. Although it is not necessary to specify a preferred sleep coordinator, it is often useful to select a node for this purpose to improve network performance. A node which is centrally located in the network can serve as a good sleep coordinator to minimize the number of hops a sync message must take to get across the network. A sleep support node and/or a node which is mains powered may be a good candidate.

The preferred sleep coordinator bit should be used with caution. The advantages of using the option become weaknesses when used on a node that is not positioned or configured properly. The preferred sleep coordinator option can also be used when setting up a network for the first time. When starting a network, a node can be configured as a sleep coordinator so it will begin sending sleep messages. After the network is set up, the preferred sleep coordinator bit can be disabled.

Nomination and Election

Nomination is an optional process that can occur on a node in the event that contact with the network sleep coordinator is lost. By default, this behavior is disabled. This behavior can be enabled with the sleep options command (SO). This process will automatically occur in the event that contact with the previous sleep coordinator is lost. Any sleep compatible node which has this behavior enabled is eligible to become the sleep coordinator for the network. If a sleep compatible node has missed three or more sync messages and is not configured as a non-sleep coordinator (presumably because the sleep coordinator has been disabled) it may become a sleep coordinator. Depending on the platform and other configured options, such a node will eventually nominate itself after a number of cycles without a sync. A nominated node will begin acting as the new network sleep coordinator. It is possible for multiple nodes to nominate themselves as the sleep coordinator. If this occurs, an election will take place to establish seniority among the multiple sleep coordinators. Seniority is determined by four factors (in order of priority):

1. Newer sleep parameters: a node using newer sleep parameters (SP/ST) is considered senior to a node using older sleep parameters. (See the Changing Sleep Parameters section below.)

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

3. Sleep Support Node: sleep support nodes are senior to cyclic sleep nodes. (This behavior can be modified using the SO parameter.)

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

Commissioning Button

The commissioning button can be used to select a module to act as the sleep coordinator. If the commissioning button functionality has been enabled, a node can be immediately nominated as a sleep coordinator by pressing the commissioning button twice or by issuing the CB2 command. A node nominated in this manner is still subject to the election process described above. A node configured as a non-sleep coordinator will ignore commissioning button nomination requests.

Changing Sleep Parameters

Any sleep compatible node in the network which does not have the non-sleep coordinator sleep option set can be used to make changes to the network's sleep and wake times. If a node's SP and/or ST are changed to values different from those that the network is using, that node will become the sleep coordinator. That node will begin sending sync messages with the new sleep parameters at the beginning of the next wake cycle.

Note #1: For normal operations, a module will use the sleep and wake parameters it gets from the sleep sync message, not the ones specified in its SP and ST parameters. The SP and ST parameters are not updated with the values of the sync message. The current network sleep and wake times used by the node can be queried using the OS and OW commands.

Note #2: Changing network parameters can cause a node to become a sleep coordinator and change the sleep settings of the network. The following commands can cause this to occur: NH, NN, NQ, and MR. In most applications, these network parameters should only be configured during deployment.

Sleep Guard Times

To compensate for variations in the timekeeping hardware of the various modules in a sleeping router network, sleep guard times are allocated at the beginning and end of the wake time. The size of the sleep guard time varies based on the sleep and wake times selected and the number of cycles that have elapsed since the last sync message was received. The sleep guard time guarantees that a destination radio will be awake when a transmission is sent. As more and more consecutive sync messages are missed, the sleep guard time increases in duration and decreases the available transmission time.

Auto-Early Wake-Up Sleep Option

Similarly to the sleep guard time, the auto early wake-up option decreases the sleep period based on the number of sync messages missed. This option comes at the expense of battery life. Auto-early wake-up sleep can be disabled using the sleep options (SO) command.

Configuration

Selecting Sleep Parameters

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

1. **Choose NH.** Based on the placement of the nodes in your network, select appropriate values for the Network Hops (NH) parameter.

Note: the default value of NH has been optimized to work for the majority of deployments. In most cases, we suggest that the parameter not be modified from its default value. Decreasing its parameters for small networks can improve battery life, but care should be taken so that the value is not made 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. This number can be estimated with the following formula:

SMPT = NH * (MT + 1) 18ms

3. **Select desired duty cycle.** The ratio of sleep time to wake time is the factor that has the greatest effect on the RF module's power consumption. Battery life can be estimated based on the following factors: sleep period, wake time, sleep current, RX current, TX current, and battery capacity.

4. **Choose sleep period and wake time.** The wake time needs to be long enough to transmit the desired data as well as the sync message. The ST parameter will automatically adjust upwards to its minimum value when other AT commands are changed that will affect it (SP, and NH). Use a value larger than this minimum. If a module 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 a few sync messages to be missed and still have time for normal data transmissions.

Starting a Sleeping Network

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

1. Enable the preferred sleep coordinator option on one of the nodes, and set its SM to a 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 commands to be sent quickly through the network during commissioning.

2. Next, power on the new nodes within range of the sleep coordinator. The nodes will quickly receive a sync message and synchronize themselves to the short cycle SP and ST.

3. Configure the new nodes in their desired sleep mode as cyclic sleeping nodes or sleep support nodes.

4. Set the SP and ST values on the sleep coordinator to the desired values for the deployed network.

5. Wait a cycle for the sleeping nodes to sync themselves to the new SP and ST values.

6. Disable the preferred sleep coordinator option bit on the sleep coordinator (unless a preferred sleep coordinator is desired).

7. Deploy the nodes to their positions.

Alternatively, nodes can be set up with their sleep pre-configured and written to flash (using the WR command) prior to deployment. If this is the case, the commissioning button and associate LED can be used to aid in deployment:

1. If a preferred sleep coordinator is going to be used in the network, deploy it first. If there will be no preferred sleep coordinator, select a node for deployment, power it on and press the commissioning button twice. This will cause the node to begin emitting sync messages.

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.

2. Next, power on nodes in range of the sleep coordinator or other nodes which have synchronized with the network. If the synchronized node is asleep, it can be woken by pressing the commissioning button once.

3. Wait a cycle for the new node to sync itself.

4. Verify that the node syncs with the network. The associate LED will blink when the module is awake and synchronized.

5. Continue this process until all nodes have been deployed.

Adding 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 will periodically send a broadcast requesting a sync message and then sleep for its SP period. Any node in the network that receives this message will respond with a sync. Because the network can be asleep for extended periods of time, and as such cannot respond to requests for sync messages, there are methods that can be used 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 will be able to respond to sync requests promptly.

2. A sleeping cyclic sleep node in the network can be woken by the commissioning button. Place the new node in range of the existing cyclic sleep node and wake the existing node by holding down the commissioning button for 2 seconds, or until the node wakes. The existing node stays awake for 30 seconds and will respond 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. The new node should be placed in range of the network with a sleep/wake cycle that is shorter than the wake period of the network. The new node will periodically send sync requests until the network wakes up and it receives a sync message.

Changing Sleep Parameters

Changes to the sleep and wake cycle of the network can be made by selecting any node in the network and changing the SP and/or ST of the node to values different than those the network is currently using. If using a preferred sleep coordinator or if it is known which node is acting as the sleep coordinator, it is suggested that this node be used to make changes to network settings. If

the network sleep coordinator is not known, any node that does not have the non-sleep coordinator sleep option bit set (see the SO command) can be used.

When changes are made to a node's sleep parameters, that node will become the network's sleep coordinator (unless it has the non-sleep coordinator option selected) and will send a sync message with the new sleep settings to the entire network at the beginning of the next wake cycle. The network will immediately begin using the new sleep parameters after this sync is sent.

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 will continue 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, the following precautions can be taken:

1. Whenever possible, avoid changing sleep parameters.

2. Enable the missed sync early wake up sleep option (SO). This command is used to tell a node to wake up progressively earlier based on the number of cycles it has gone without receiving a sync. This will increase 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 which miss sync messages will have an increased wake time and decreased sleep time during cycles in which the sync message is missed. This will reduce battery conservation.

3. When changing between two sets of sleep settings, choose settings so that the wake periods of the two sleep settings will happen at the same time. In other words, try to satisfy the following equation: (SP1 + ST1) = N * (SP2 + ST2), where SP1/ST1 and SP2/ST2 are the desired sleep settings and N is an integer.

Rejoining Nodes Which Have Lost Sync

Mesh networks get their robustness from taking advantage of routing redundancies which may be available in a network. It is recommended to architect the network with redundant mesh nodes to increase robustness. If a scenario exists such that 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 (catastrophic failure condition), then multiple subnets may arise while 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 modules to fix the problem and prevent it from occurring in the future.

When the default DigiMesh sleep parameters are used, separated subnets will not drift out of phase with each other. Subnets can drift out of phase with each other if the network is configured in one of the following ways:

- If multiple modules in the network have had the non-sleep coordinator sleep option bit disabled and are thus eligible to be nominated as a sleep coordinator.
- If the modules in the network are not using the auto early wake-up sleep option.

If a network has multiple subnets that have drifted 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 up with. Use this node to slightly change the sleep cycle settings of the network (increment ST, for example).

3. Wait for the subnet's next wake cycle. During this cycle, the node selected to change the sleep cycle parameters will send the new settings to the entire subnet it is in range of, including the sleep support node which is in range of the other subnet.

4. Wait for the out of sync subnet to wake up and send a sync. When the sleep support node receives this sync, it will reject it and send a sync to the subnet with the new sleep settings.

5. The subnets will now be in sync. The sleep support node can be removed. If desired, the sleep cycle settings can be changed back to what they were.

In the case that only a few nodes need to be replaced, this method can also be used:

1. Reset the out of sync node and set its sleep mode to cyclic sleep (SM = 8). Set it up to have a short sleep cycle.

2. Place the node in range of a sleep support node or wake a sleeping node with the commissioning button.

3. The out of sync node will receive a sync from the node which is synchronized to the network and sync to the network sleep settings.

Diagnostics

The following are useful in some applications when managing a sleeping router network:

Query current sleep cycle: the OS and OW command can be used to query the current operational sleep and wake times a module is currently using.

Sleep Status: the SS command can be used to query useful information regarding the sleep status of the module. This command can be used to query if the node is currently acting as a network sleep coordinator, as well as other useful diagnostics.

Missed Sync Messages Command: the MS command can be used to query the number of cycles that have elapsed since the module last received a sync message.

Sleep Status API messages: when enabled with the SO command, a module configured in API mode will output modem status frames immediately after a module wakes up and just prior to a module going to sleep.

6. Command Reference Tables

Special

Special Commands

AT Command	Name and Description	Parameter Range	Default
WR	Write. Write parameter values to non-volatile memory so that parameter modifications persist through subsequent resets. Note: Once WR is issued, no additional characters should be sent to the module until after the "OK\r" response is received.	--	--
RE	Restore Defaults. Restore module parameters to factory defaults.	--	--
CN	Exit Command Mode. Explicitly exit the module from AT Command Mode.	--	--
FR	Software Reset. Reset module. Responds immediately with an “OK” then performs a reset 100ms later.	--	--
AC	Apply Changes. Immediately applies new settings without exiting command mode.	--	--

Addressing

Addressing Commands

AT Command	Name and Description	Parameter Range	Default
DH	Destination Address High. Set/Get the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the destination address used for transmission.	0 to 0xFFFFFFFF	0
DL	Destination Address Low. Set/Get the lower 32 bits of the 64-bit destination address. When combined with DH, DL defines the destination address used for transmission.	0 to 0xFFFFFFFF	0x0000FFFF
DD	Device Type Identifier. Stores a device type value. This value can be used to differentiate multiple XBee-based products.	0-0xFFFFFFFF [read only]	0x40000
SH	Serial Number High. Read high 32 bits of the RF module's unique IEEE 64-bit address. 64-bit source address is always enabled. This value is read-only and it never changes	0-0xFFFFFFFF	Factory
SL	Serial Number Low. Read low 32 bits of the RF module's unique IEEE 64-bit address. 64-bit source address is always enabled. This is read only and it is also the serial number of the node..	0-0xFFFFFFFF	Factory
SE	Source Endpoint. Set/read the application layer source endpoint value. This value will be used as the source endpoint for all data transmissions. The default value 0xE8 (Data endpoint) is the Digi data endpoint	0-0xFF	0xE8
DE	Destination Endpoint. Set/read application layer destination ID value. This value will be used as the destination endpoint for all data transmissions. The default value (0xE8) is the Digi data endpoint.	0-0xFF	0xE8
CI	Cluster Identifier. Set/read application layer cluster ID value. This value will be used as the cluster ID for all data transmissions. The default value 0x11 (Transparent data cluster ID)	0-0xFFFF	0x11
NP	Maximum RF Payload Bytes. This value returns the maximum number of RF payload bytes that can be sent in a unicast transmission based on the current configurations.	0-0xFFFF	n/a
CE	Node Messaging Options Set/read options related to node type and messaging modes. This command is a bitfield with the bits defined as follows: bit 0 - Indirect messaging coordinator enable. All point-multipoint unicasts will be held until requested by a polling end device. bit 1 - Disable DigiMesh routing on this node. When set, this node will not propagate broadcasts or become and intermediate node in a route. bit 2 - Indirect messaging polling enable. Periodically send requests for messages held by the node's coordinator. Supported in both firmware variants.	0-0x07	0

Addressing Commands

AT Command	Name and Description	Parameter Range	Default
AG	Aggregator Support. The AG command sends a broadcast through the network that has the following affects on nodes which receive the broadcast: • The receiving node will establish a DigiMesh route back to the originating node, provided there is space in the routing table. • The DH and the DL of the receiving node will be updated to the address of the originating node if the AG parameter matches the current DH/DL of the receiving node. • For API enabled modules on which DH and DL are updated, an Aggregate Addressing Update API frame will be sent out the serial port. Note that the AG command is only available on products which support DigiMesh.	Any 64-bit number	n/a
TO	Transmit Options. This command defines transmission options for all packets originating from this radio. These options can be overridden on a packet-by-packet basis by using the TxOptions field of the API TxRequest frames. 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 on 10k product) 5 — Reserved — <set this bit to 0> 4 — Reserved — <set this bit to 0> 3 — Trace Route — Enable a Trace Route on all DigiMesh API packets 2 — NACK — Enable a NACK messages on all DigiMesh API packets 1 — Disable RD — Disable Route Discovery on all DigiMesh unicasts 0 — Disable ACK — Disable acknowledgments on all unicasts Example #1: Setting TO to 0x80 would cause all transmissions to be sent using repeater mode. Example #2: Setting TO to 0xC1 would cause all transmissions to be sent using DigiMesh, with acknowledgments disabled.	Bits 6 & 7 cannot be set to DigiMesh on the 10k build. Bits 4 & 5 must be set to 0 Bits 1, 2, & 3 cannot be set on the 10k build	0x40 (10k product) 0xC0 (80k product)

Serial Interfacing (I/O)

Serial Interfacing Commands

AT Command	Name and Description	Parameter Range	Default
AP	API mode. Set or read the API mode of the radio. The following settings are allowed: 0 API mode is off. All UART input and output is raw data and packets are delineated using the RO parameter. 1 API mode is on. All UART input and output data is packetized in the API format, without escape sequences. 2 API mode is on with escaped sequences inserted to allow for control characters (XON, XOFF, escape, and the 0x7e delimiter to be passed as data.)	0, 1, or 2	0
AO	API Output Format. Enables different API output frames. Options include: 0 Standard Data Frames (0x90 for RF RX) 1 Explicit Addressing Data Frames (0x91 for RF RX)	0, 1	0
BD	Baud rate. Set or read serial interface rate (speed for data transfer between radio modem and host). Values from 0-8 select preset standard rates. Values at 0x39 and above select the actual baud rate. Providing the host supports it. Baud rates can go as high as 1.875Mbps. The values from 0 to 8 are interpreted as follows: 0 - 1,200bps 3 - 9,600bps 6 - 57,600bps 1 - 2,400bps 4 - 19,200bps 7 - 115,200bps 2 - 4,800bps 5 - 38,400bps 8 - 230,400bps	0 to 8, and 0x39 to 0x1c9c38	0x03 (9600 bps)
RO	Packetization Timeout. Set/Read number of character times of inter-character silence required before packetization. Set (RO=0) to transmit characters as they arrive instead of buffering them into one RF packet.	0 - 0xFF [x character times]	3
FT	Flow Control Threshhold. Set or read flow control threshhold. De-assert CTS and/or send XOFF when FT bytes are in the UART receive buffer. Re-assert CTS when less than FT - 16 bytes are in the UART receive buffer.	0x11 - 0xEE	0xBE=190d
NB	Parity. Set or read parity settings for UART communications. The values from 0 to 4 are interpreted as follows: 0 No parity 3 Forced high parity 1 Even parity 4 Forced low parity 2 Odd parity	0 to 4	0 (No parity)
D7	DIO7 Configuration. Configure options for the DIO7 line of the module. Options include: 0 = Input, unmonitored 1 = CTS flow control 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high 6 = RS-485 Tx enable, low TX (0V on transmit, high when idle) 7 = RS-485 Tx enable, high TX (high on transmit, 0V when idle)	0-1, 3-7	0
D6	DIO6 Configuration. Configure options for the DIO6 line of the module. Options include: 0 = Input, unmonitored 1 = RTS flow control 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0-1, 3-5	0

I/O Commands

I/O Commands

AT Command	Name and Description	Parameter Range	Default
P0	DIO10/PWM0 Configuration. Configure options for the DIO10/PWM0 line of the module. Options include: 0 = Input, unmonitored 1 = RSSI PWM 2 = PWM0 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0-5	1
P1	DIO11/PWM1 Configuration. Configure options for the DIO11/PWM1 line of the module. Options include: 0 = Input, unmonitored 2 = PWM1 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 2-5	0

I/O Commands

AT Command	Name and Description	Parameter Range	Default
P2	DIO12 Configuration. Configure options for the DIO12 line of the module. Options include: 0 = Input, unmonitored 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 3-5	0
RP	RSSI PWM Timer. Time RSSI signal will be output after last transmission. When RP = 0xFF, output will always be on.	0 - 0xFF [x 100 ms]	0x28 (4 seconds)
D0	AD0/DIO0 Configuration. Configure options for the AD0/DIO0 line of the module. Options include: 0 = Input, unmonitored 1 = Commissioning button enable 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0 - 5	1
D1	AD1/DIO1 Configuration. Configure options for the AD1/DIO1 line of the module. Options include: 0 = Input, unmonitored 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 2-5	0
D2	AD2/DIO2 Configuration. Configure options for the AD2/DIO2 line of the module. Options include: 0 = Input, unmonitored 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 2-5	0
D3	AD3/DIO3 Configuration. Configure options for the AD3/DIO3 line of the module. Options include: 0 = Input, unmonitored 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 2-5	0
D4	AD4/DIO4 Configuration. Configure options for the AD4/DIO4 line of the module. Options include: 0 = Input, unmonitored 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0, 2-5	0
D5	AD5/DIO5 Configuration. Configure options for the AD5/DIO5 line of the module. Options include: 0 = Input, unmonitored 1 = Associate LED 2 = Analog Input 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0-5	1
D8	DIO8/SLEEP_RQ Configuration. Configure options for the DIO8/SLEEP_RQ line of the module. Options include: 0 = Input, unmonitored 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high When used as SLEEP_RQ, the D8 parameter should be configured in mode 0 or 3.	0,3-5	0

I/O Commands

AT Command	Name and Description	Parameter Range	Default
D9	DIO9 / ON/SLEEP. Configuration. Configure options for the DIO9/ON/SLEEP line of the module. Options include: 0 = Input, unmonitored 1 = ON/SLEEP 3 = Digital input, monitored 4 = Digital output low 5 = Digital output high	0,1,3-5	
PR	Pull-up Resistor. Set/read the bit field that configures the internal pull-up resistor status for the I/O lines. "1" specifies the pull-up resistor is enabled. "0" specifies no pullup. Bits: 0 - DIO4/AD4 (Pin 11) 1 - AD3 / DIO3 (Pin 17) 2 - AD2 / DIO2 (Pin 18) 3 - AD1 / DIO1 (Pin 19) 4 - AD0 / DIO0 (Pin 20) 5 - RTS / DIO6 (Pin 16) 6 - DTR / SLEEP_RQ/DI08 / DIO8 (Pin 9) 7 - DIN / Config (Pin 3) 8 - Associate / DIO5 (Pin 15) 9 - On/Sleep / DIO9 (Pin 13) 10 - DIO12 (Pin 4) 11 - PWM0 / RSSI / DIO10 (Pin 6) 12 - PWM1 / DIO11 (Pin 7) 13 - DIO7/CTS (Pin 12) 14 - DOUT (Pin 2)	0 - 0x1FFF	0x1FFF
M0	PWM0 Output Level. Set/read the output level of the PWM0 line. The line should be configured as a PWM output using the P0 command.	0-0x03FF	0
M1	PWM1 Output Level. Set/read the output level of the PWM1 line. The line should be configured as a PWM output using the P1 command.	0-0x03FF	0
LT	Assoc LED Blink Time. Set/Read the Associate LED blink time. If the Associate LED functionality is enabled (D5 command), this value determines the on and off blink times for the LED. If LT=0, the default blink rate will be used (500ms sleep coordinator, 250ms otherwise). For all other LT values, LT is measured in 10ms	0x14-0xFF (x10ms)	0
IS	Force Sample. Forces a read of all enabled digital and analog input lines.	n/a	n/a
IC	I/O Digital Change Detection. Set/Read the digital I/O pins to monitor for changes in the I/O state. IC works with the individual pin configuration commands (D0-D9, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate I/O sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0. Bit (I/O pin): 0 (DIO0) 1 (DIO1) 2 (DIO2) 3 (DIO3) 4 (DIO4) 5 (DIO5) 6 (DIO6) 7 (DIO7) 8 (DIO8) 9 (DIO9) 10 (DIO10) 11 (DIO11) 12 (DIO12)	0-0xFFFF	0
IR	IO Sample Rate. Set/Read the I/O sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital I/O functionality enabled (see D0-D9, P0-P2 commands). The sample rate is measured in milliseconds.	0 - 0xFFFF (ms)	0

I/O Commands

AT Command	Name and Description	Parameter Range	Default
IF	Sleep Sample Rate. Set/read the number of sleep cycles that must elapse between periodic I/O samples. This allows I/O samples to be taken only during some wake cycles. During those cycles I/O samples are taken at the rate specified by IR.	1-0xFF	1
CB	Commissioning Pushbutton. This command can be used to simulate commissioning button presses in software. The parameter value should be set to the number of button presses to be simulated. For example, sending the ATCB1 command will execute the action associated with 1 commissioning button press.	0-4	n/a

Diagnostics

Diagnostics Commands

AT Command	Name and Description	Parameter Range	Default
VR	Firmware Version. Read firmware version of the module.	0 - 0xFFFFFFFF [read-only]	Firmware-set
VL	Version Long. Shows detailed version information including application build date and time.	--	--
HV	Hardware Version. Read hardware version of the module.	0 - 0xFFFF [read-only]	Factory-set
CK	Configuration Code. Read the configuration code associated with the current AT command configuration.The code returned can be used as a quick check to determine if a node has been configured as desired.	0-0xFFFFFFFF	n/a
ER	RF Errors. This count is incremented whenever a packet is recieved which contained integrity errors of some sort. Once the number reaches 0xFFFF, further events will not be counted. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0
GD	Good Packets. This count is incremented whenever a good frame with a valid MAC header is recieved on the RF interface. Once the number reaches 0xFFFF, further events will not be counted. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0
TR	Transmission Errors. This count is incremented whenever a MAC transmission attempt exhausts all MAC retries without ever receiving a MAC acknowledgement message from the destination node. Once the number reaches 0xFFFF, further events will not be counted. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0
TP	Temperature. Read module temperature in Celsius. Negatives temperatures can be returned.	0xff74 to 0x0258	n/a
DB	Received Signal Strength. This command reports the received signal strength of the last received RF data packet. The DB command only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link. The DB command value is measured in -dBm. For example if DB returns 0x60, then the RSSI of the last packet received was -96dBm.	n/a	n/a
EA	MAC ACK Timeouts. This count is incremented whenever a MAC ACK timeout occurs on a MAC level unicast. Once the number reaches 0xFFFF further events will not be counted. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0
UA	MAC Unicast Transmission Count. This count is incremented whenever a MAC unicast transmission occurs for which an ACK is requested. Once the number reaches 0xFFFF further events will not be counted. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0
BC	RF Byte Count. This count is incremented for every PHY level byte transmitted. The purpose of this count is to estimate battery life by tracking time doing transmissions. This number rolls over to zero from 0xFFFF. The counter can be reset to any 16-bit value by appending a hexadecimal parameter to the command.	0-0xFFFF	0

AT Command Options

AT Command Options Commands

AT Command	Name and Description	Parameter Range	Default
CT	Command Mode Timeout. Set/Read the period of inactivity (no valid commands received) after which the RF module automatically exits AT Command Mode and returns to Idle Mode.	2-0x1770	0x64 (100d)
GT	Guard Times. Set required period of silence before and after the Command Sequence Characters of the AT Command Mode Sequence (GT + CC + GT). The period of silence is used to prevent inadvertent entrance into AT Command Mode.	0 to 0xFFFF	0x3E8 (1000d)
CC	Command Character. Set or read the character to be used between guard times of the AT Command Mode Sequence. The AT Command Mode Sequence causes the radio modem to enter Command Mode (from Idle Mode).	0 - 0xFF	0x2B

Node Identification

Node Identification Commands

AT Command	Name and Description	Parameter Range	Default
FN	Find Neighbors. Discovers and reports all modules found within immediate RF range. The following information is reported for each module discovered: 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). See the NO Command 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) <CR> If the FN command is issued in command mode, after (NT * 100) ms + overhead time, the command ends by returning a <CR>. If the FN command is issued via a Local or Remote Command Request API frame, each response is returned as a separate Local or Remote AT Command Response API packet, respectively. The data consists of the above listed bytes without the carriage return delimiters. The NI string will end in in a 0x00 null character.	n/a	n/a
NT	Node Discover Timeout. Set/Read the amount of time a node will spend discovering other nodes when ND or DN is issued.	0 - 0xFC [x 100 msec]	0x82 (130d)
NI	Node Identifier. Stores a string identifier. The string accepts only printable ASCII data In AT Command Mode, the string can not start with a space. A carriage return or comma ends the command. Command will automatically end when maximum bytes for the string have been entered. This string is returned as part of the ATND (Network Discover) command. This identifier is also used with the ATDN (Destination Node) command.	up to 20 byte ASCII string	a space character
DN	Discover Node - Destination Node. Resolves an NI (Node Identifier) string to a physical address (case sensitive). The following events occur after the destination node is discovered: <AT Firmware> 1. DL & DH are set to the extended (64-bit) address of the module with the matching NI (Node Identifier) string. 2. OK (or ERROR)\r is returned. 3. Command Mode is exited to allow immediate communication <API Firmware> 0xFFFE and 64-bit extended addresses are returned in an API Command Response frame. If there is no response from a module within (NT * 100) milliseconds or a parameter is not specified (left blank), the command is terminated and an “ERROR” message is returned. In the case of an ERROR, Command Mode is not exited.	20 byte ascii string	

Node Identification Commands

AT Command	Name and Description	Parameter Range	Default
ND	Network Discover. Discovers and reports all RF modules found. The following information is reported for each module discovered. MY<CR> (always 0xFFFE) SH<CR> SL<CR> NI<CR> (Variable length) PARENT_NETWORK ADDRESS (2 Bytes)<CR> (always 0xFFFE) DEVICE_TYPE<CR> (1 Byte: 0=Coord, 1=Router, 2=End Device). See the NO Command STATUS<CR> (1 Byte: Reserved) PROFILE_ID<CR> (2 Bytes) MANUFACTURER_ID<CR> (2 Bytes) DIGI DEVICE TYPE<CR> (4 Bytes. Optionally include based on NO settings.) RSSI<CR> (1 Byte. Optionally include based on NO settings.) <CR> If the ND command is issued in command mode, after (NT * 100) ms + overhead time, the command ends by returning a <CR>. If the ND command is issued via a Local Command Request API frame, each response is returned as a separate Local AT Command Response API packet. The data consists of the above listed bytes without the carriage return delimiters. The NI string will end in a "0x00" null character.		
NO	Network Discovery Options. Set/Read the options value for the network discovery command. The options bitfield value can change the behavior of the ND (network discovery) command and/or change what optional values are returned in any received ND responses or API node identification frames. Options include: 0x01 = Append DD value (to ND responses or API node identification frames) 0x02 = Local device sends ND or FN response frame when ND is issued. 0x04 = Append RSSI (of the last hop for DigiMesh networks) to ND or FN responses or API node identification frames. 0x08 = Enable Verbose Device Type field. When enabled the device type field of ND and FN responses will contain the CE value of the responding node. If the responding node is acting as a synchronized sleep coordinator then bit 7 will be set.	0-0x07 [bitfield]	0

Security

Security Commands

AT Command	Name and Description	Parameter Range	Default
EE	Security Enable Enables or disables 128-bit AES encryption. This command parameter should be set the same on all devices.	0 to 1	0
KY	Security Key Sets the 16 byte network security key value. This command is write-only. Attempts to read KY will return an OK status. This command parameter should be set the same on all devices.	128-bit value	n/a

MAC Level

MAC-level Commands

AT Command	Name and Description	Parameter Range	Default
MT	Broadcast Multi-Transmit. Set/Read the number of additional MAC-level broadcast transmissions. All broadcast packets are transmitted MT+1 times to ensure it is received.	0-0xF	3
RR	Unicast Mac Retries. Set/Read the maximum number of MAC level packet delivery attempts for unicasts. If RR is non-zero packets sent from the radio will request an acknowledgement, and can be resent up to RR times if no acknowledgements are received.	0-0xF	10
ID	Network ID. Set or read the user network identifier. Nodes must have the same network identifier to communicate. Changes to ID can be written to non-volatile memory using the WR command.	0x0000 to 0x7FFF	0x7FFF
PL	Power Level. Set/Read the power level at which the RF module transmits conducted power.	0 = 0 dBm 1 = 5 dBm 2 = 8 dBm 3 = 10 dBm 4 = 12 dBm	4

DigiMesh

Mesh Commands: Network Level Commands (Some of these commands are supported on both the 10k and 80k variants)

AT Command	Name and Description	Parameter Range	Default
NH	Network Hops Set or read the maximum number of hops expected to be seen in a network route. This value doesn't limit the number of hops allowed, but it is used to calculate timeouts waiting for network acknowledgements. Supported in both variants.	1 to 0xff	7
MR	Mesh Network Retries Set or read the maximum number of network packet delivery attempts. If MR is non-zero, packets sent will request a network acknowledgement, and can be resent up to MR+1 times if no acknowledgements are received. Supported in the 80k variant only.	0 to 7	1
BH	Broadcast Radius. Set/read the transmission radius for broadcast data transmissions. Set to 0 for maximum radius. If BH is set greater than NH then the value of NH is used. Supported in both variants.	0-0x20	0

Sleep

Sleep Commands

AT Command	Name and Description	Parameter Range	Default
SM	Sleep Mode. Set/read the sleep mode of the module. 0 - No sleep mode enabled 1 - Pin sleep. In this mode, the sleep/wake state of the module is controlled by the SLEEP_RQ line. 4 - Asynchronous cyclic sleep. In this mode, the module periodically sleeps and wakes based on the SP and ST commands. 5 - Asynchronous cyclic sleep with pin wake-up. In this mode, the module acts in the same way as asynchronous cyclic sleep with the exception that the module will prematurely terminate a sleep period when a falling edge of the SLEEP_RQ line is detected. 7 - Sleep support mode. 8 - Synchronous cyclic sleep mode.	0, 1, 4, 5, 7, 8	0
SO	Sleep Options. Set/read the sleep options of the module. This command is a bitmask. For synchronous sleep modules, the following sleep options are defined: bit 0 = Preferred sleep coordinator bit 1 = Non-sleep coordinator bit 2 = Enable API sleep status messages bit 3 = Disable early wake-up bit 4 = Enable node type equality bit 5 = Disable lone coordinator sync repeat For ansynchronous sleep modules, the following sleep options are defined: bit 8 = Always wake for ST time	Any of the available sleep option bits can be set or cleared. Bit 0 and bit 1 cannot be set at the same time.	0x02
ST	Wake Time. Set/read the wake period of the module. For asynchronous sleep modules, this command defines the amount of time that the module will stay awake after receiving RF or serial data. For synchronous sleep modules, this command defines the amount of time that the module will stay awake when operating in cyclic sleep mode. This value will be adjusted upwards automatically if it is too small to function properly based on other settings.	0x45-0x36EE80	0x7D0 (2 seconds)
SP	Sleep Period. Set/read the sleep period of the module. This command defines the amount of time the module will sleep per cycle.	1 - 1440000 (x 10 ms)	2 seconds
MS	Number of Missed Syncs. Read the number of wake cycles that have elapsed since the last sync message was received. Supported in the 80k firmware variant only.		
SN	Number of Sleep Periods. Set/read the number of sleep periods value. This command controls the number of sleep periods that must elapse between assertions of the ON_SLEEP line during the wake time of asynchronous cyclic sleep. During cycles when the ON_SLEEP line is not asserted, the module will wake up and check for any serial or RF data. If any such data is recieved, then the ON_SLEEP line will be asserted and the module will fully wake up. Otherwise, the module will return to sleep after checking. This command does not work with synchronous sleep.	1 - 0xFFFF	1
SQ	Missed Sync Count. Count of the number of syncs that have been missed. This value can be reset by setting ATSQ to 0. When the value reaches 0xFFFF it will not be incremented anymore.	n/a	n/a

AT Command	Name and Description	Parameter Range	Default
SS	Sleep Status The SS command can be used to query a number of Boolean values describing the status of the module. Bit 0: This bit will be true when the network is in its wake state. Bit 1: This bit will be true if the node is currently acting as a network sleep coordinator. Bit 2: This bit will be true if the node has ever received a valid sync message since the time it was powered on. Bit 3: This bit will be true if the node has received a sync message in the current wake cycle. Bit 4: This bit will be true if the user has altered the sleep settings on the module so that the node will nominate itself and send a sync message with the new settings at the beginning of the next wake cycle. Bit 5: This bit will be true if the user has requested that the node nominate itself as the sleep coordinator (using the commissioning button or the CB2 command). All other bits: Reserved - All non-documented bits can be any value and should be ignored. bit 6 = This bit will be true if the node is currently in deployment mode. All other bits: Reserved - All non-documented bits can be any value and should be ignored.	n/a	n/a
OS	Operational Sleep Period. Read the sleep period that the node is currently using. This number will oftentimes be different from the SP parameter if the node has synchronized with a sleeping router network. Units of 10mSec	n/a	n/a
OW	Operational Wake Period. Read the wake time that the node is currently using. This number will oftentimes be different from the ST parameter if the node has synchronized with a sleeping router network. Units of 1 ms	n/a	n/a
WH	Wake Host. Set/Read the wake host timer value. If the wake host timer is set to a non-zero value, this timer specifies a time (in millisecond units) that the device should allow after waking from sleep before sending data out the UART or transmitting an I/O sample. If serial characters are received, the WH timer is stopped immediately. When in synchronous sleep, the device will shorten its sleep period by the value specified by the WH command to ensure that it is prepared to communicate when the network wakes up. When in this this sleep mode, the device will always stay awake for the WH time plus the amount of time it takes to transmit a one-hop unicast to another node.	0-0xFFFF (x 1ms)	0

7. API Operation

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

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

API Frame Specifications

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

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

API Operation (AP parameter = 1)

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

Figure 7-01. UART Data Frame Structure:

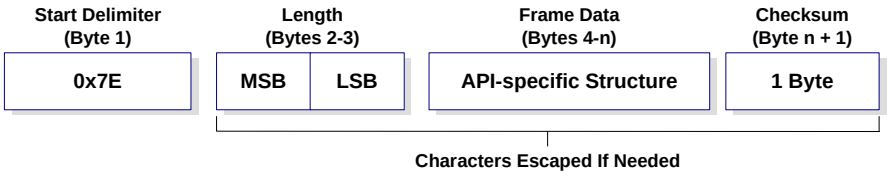
MSB = Most Significant Byte, LSB = Least Significant Byte

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

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

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

Figure 7-02. UART Data Frame Structure - with escape control characters:



MSB = Most Significant Byte, LSB = Least Significant Byte

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

Data bytes that need to be escaped:

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

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

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

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

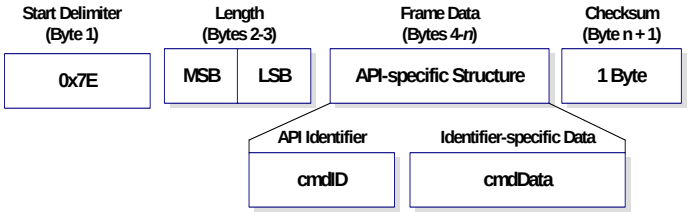
Length

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

Frame Data

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

Figure 7-03. UART Data Frame & API-specific Structure:



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

API Frame Names and Values

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
Transmit Request	0x10
Explicit Addressing Command Frame	0x11
Remote Command Request	0x17
AT Command Response	0x88
Modem Status	0x8A
Transmit Status	0x8B
Receive Packet (AO=0)	0x90
Explicit Rx Indicator (AO=1)	0x91
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97

Checksum

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

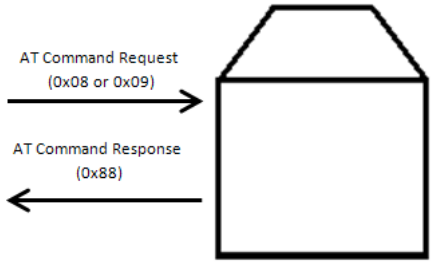
To calculate: Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract the result from 0xFF.

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

API UART Exchanges

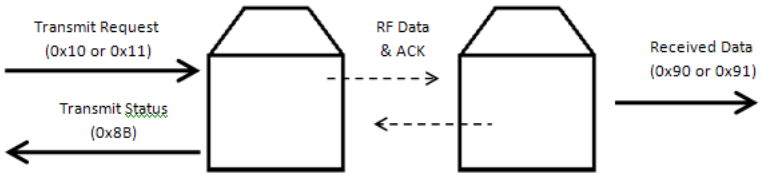
AT Commands

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



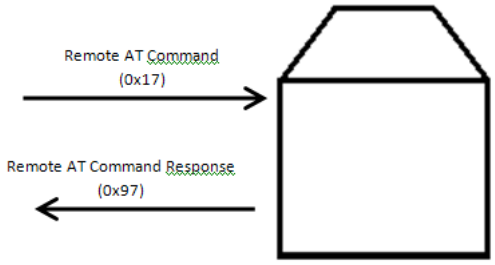
Transmitting and Receiving RF Data

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



Remote AT Commands

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



Supporting the API

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

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

      case RX_IO_SAMPLE_FRAME:
         //process IO sample frame
         break;

      case NODE_IDENTIFICATION_FRAME:
         //process node identification frame
         break;

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

Frame Data

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

AT Command

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x04	
	Frame-specific Data	Frame Type	3	0x08	
		Frame ID	4	0x52 (R)	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		AT Command	5	0x4E (N)	Command Name - Two ASCII characters that identify the AT Command.
			6	0x48 (H)	
		Parameter Value (optional)			If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
	Checksum		8	0x0F	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

AT Command - Queue Parameter Value

Frame Type: 0x09

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

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

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

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

Transmit Request

Frame Type: 0x10

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

The 64-bit destination address should be set to 0x000000000000FFFF for a broadcast transmission (to all devices). For unicast transmissions the 64 bit address field should be set to the address of the desired destination node. The reserved field should be set to 0xFFFE.

This example shows if escaping is disabled (AP=1).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following address is also supported: 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		Reserved	13	0xFF	Set to 0xFFFE.
			14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Transmit Options	16	0x00	Bitfield: bit 0: Disable ACK bit 1: Disable RD bit 2: Enable Unicast NACK messages. bit 3: Enable Unicast Trace Route messages. bits 6,7: b'01 - Point-Multipoint b'10 - Repeater mode (directed broadcast) b'11 - DigiMesh (not available on 10k product) All other bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: The example above shows how to send a transmission to a module where escaping is disabled (AP=1) with destination address 0x0013A200 40014011, payload "TxData0A". If escaping is enabled (AP=2), the frame should look like:

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

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

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

Explicit Addressing Command Frame

Frame Type: 0x11

Allows application layer fields (endpoint and cluster ID) to be specified for a data transmission. Similar to the Transmit Request, but also requires application layer addressing fields to be specified (endpoints, cluster ID, profile ID). An Explicit Addressing Request API frame causes the

module to send data as an RF packet to the specified destination, using the specified source and destination endpoints, cluster ID, and profile ID.

The 64-bit destination address should be set to 0x000000000000FFFF for a broadcast transmission (to all devices). For unicast transmissions the 64 bit address field should be set to the address of the desired destination node. The reserved field should be set to 0xFFFE.

The broadcast radius can be set from 0 up to NH to 0xFF. If the broadcast radius exceeds the value of NH then the value of NH will be used as the radius. This parameter is only used for broadcast transmissions.

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x1A	
	Frame-specific Data	Frame Type	3	0x11	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following address is also supported: 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x01	
			10	0x23	
			11	0x84	
			LSB12	0x00	
		Reserved	13	0xFF	Set to 0xFFFE.
			14	0xFE	
		Source Endpoint	15	0xA0	Source endpoint for the transmission.
		Destination Endpoint	16	0xA1	Destination endpoint for the transmission.
		Cluster ID	17	0x15	Cluster ID used in the transmission
			18	0x54	
		Profile ID	19	0xC1	Profile ID used in the transmission
			20	0x05	
		Broadcast Radius	21	0x00	Sets the maximum number of hops a broadcast transmission can traverse. If set to 0, the transmission radius will be set to the network maximum hops value.
		Transmit Options	22	0x00	Bitfield: bit 0: Disable ACK bit 1: Don't attempt route Discovery. bit 2: Enable Unicast NACK messages. bit 3: Enable Unicast Trace Route messages. All other bits must be set to 0.
		Data Payload	23	0x54	
			24	0x78	
			25	0x44	
			26	0x61	
			27	0x74	
			28	0x61	
	Checksum		29	0xDD	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: The above example sends a data transmission to a radio with a 64 bit address of 0x0013A20001238400 using a source endpoint of 0xA0, destination endpoint 0xA1, cluster ID =0x1554, and profile ID 0xC105. Payload will be "TxData".

Remote AT Command Request

Frame Type: 0x17

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

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

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

AT Command Response

Frame Type: 0x88

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x05	
	Frame-specific Data	Frame Type	3	0x88	
		Frame ID	4	0x01	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
		AT Command	5	'B' = 0x42	Command Name - Two ASCII characters that identify the AT Command.
			6	'D' = 0x44	
		Command Status	7	0x00	The least significan nibble indicates the command status: 0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter The most significant nibble is a bitfield as follows: 0x40 = The RSSI field is invalid and should be ignored. Software prior to version 8x60 did not include RSSI information 0x80 = Response is a remote command.
		Command Data			Register data in binary format. If the register was set, then this field is not returned, as in this example.
	Checksum		8	0xF0	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Modem Status

Frame Type: (0x8A)

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

Example: The following API frame is returned when an API device powers up.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x02	
	Frame-specific Data	Frame Type	3	0x8A	
		Status	4	0x00	0x00 = Hardware reset 0x01= Watchdog timer reset 0x0B = Network Woke Up 0x0C = Network Went To Sleep
	Checksum		5	0x75	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Transmit Status

Frame Type: 0x8B

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x07	
	Frame-specific Data	Frame Type	3	0x8B	
		Frame ID	4	0x47	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
		Reserved	5	0xFF	Reserved.
			6	0xFE	
		Transmit Retry Count	7	0x00	The number of application transmission retries that took place.
		Delivery Status	8	0x00	0x00 = Success 0x01 = MAC ACK Failure 0x15 = Invalid destination endpoint 0x21 = Network ACK Failure 0x25 = Route Not Found
		Discovery Status	9	0x02	0x00 = No Discovery Overhead 0x02 = Route Discovery
	Checksum		10	0x2E	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: In the above example, a unicast data transmission was sent successfully to a destination device using a frame ID of 0x47.)

Route Information Packet

Frame type: 0x8D

A Route Information Packet can be output for DigiMesh unicast transmissions on which the NACK enable or the Trace Route enable TX option is enabled.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x2A	
	Frame-specific Data	Frame Type	3	0x8D	
		Source Evemt	4	0x12	0x11 = NACK, 0x12 = Trace Route
		Length	5	0x2B	Number of bytes that follow (excluding checksum). If length increases, then new items have been added to the end of the list (for future revisions).
		Timestamp	MSB 6	0x9C	System timer value on the node generating the Route Information Packet.
			7	0x93	
			8	0x81	
			LSB 9	0x7F	
		ACK Timout Count	10	0x00	The number of MAC ACK timeouts that occurred.
		Reserved	11	0x00	Reserved
		Reserved	12	0x00	Reserved
		Destination Address	MSB 13	0x00	Address of the final destination node of this network level transmission.
			14	0x13	
			15	0xA2	
			16	0x00	
			17	0x40	
			18	0x52	
			19	0xAA	
			LSB 20	0xAA	
		Source Address	MSB 21	0x00	Address of the source node of this network level transmission.
			22	0x13	
			23	0xA2	
			24	0x00	
			25	0x40	
			26	0x52	
			27	0xDD	
			LSB 28	0xDD	
		Responder Address	MSB 29	0x00	Address of the node that generated this Route Information Packet after sending (or attempting to send) the packet to the next hop (the Reciever Node)
			30	0x13	
			31	0xA2	
			32	0x00	
			33	0x40	
			34	0x52	
			35	0xBB	
			LSB 36	0xBB	
		Receiver Address	MSB 37	0x00	Address of the node to which the data packet was just sent (or attempted to be sent to)
			38	0x13	
			39	0xA2	
			40	0x00	
			41	0x40	
			42	0x52	
			43	0xCC	
			LSB 44	0xCC	
	Checksum		45	0xCE	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: The above example represents a possible Route Information Frame that could be received when doing a trace route on a transmission from a radio with serial number 0x0013a2004052AAAA to a radio with serial number 0x0013a2004052DDDD. This particular

frame indicates that the transmission was successfully forwarded from the radio with serial number 0x0013a2004052BBBB to the radio with serial number 0x0013a2004052CCCC.

Aggregate Addressing Update

Frame type: 0x8E

An Aggregate Addressing Update frame is output on an API-enabled node when an address update frame (generated by the AG command being issued on a node in the network) causes the node to update its DH and DL registers.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x12	
	Frame-specific Data	Frame Type	3	0x8E	
		Format ID	4	0x00	Byte reserved to indicate format of additional packet information which may be added in future firmware revisions. In the current firmware revision, 0x00 is returned in this field.
		New Address	MSB 5	0x00	Address to which DH and DL are being set
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x52	
			11	0xBB	
			LSB 12	0xBB	
		Old Address	13	0x00	Address to which DH and DL were previously set
			14	0x13	
			15	0xA2	
			16	0x00	
			17	0x40	
			18	0x52	
			19	0xAA	
			20	0xAA	
	Checksum		21	0x2E	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: In the above example a radio which had a destination address (DH/DL) of 0x0013A2004052AAAA updated its destination address to 0x0013A2004052BBBB.

Receive Packet

Frame Type: (0x90)

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x12	
	Frame-specific Data	Frame Type	3	0x90	
		Frame ID	4	0x00	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Source Address	MSB 5	0x13	64-bit address of sender
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		Reserved	12	0xFF	Reserved
			13	0xFE	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet
		Received Data	15	0x52	Received RF data
			16	0x78	
			17	0x44	
			18	0x61	
			19	0x74	
			20	0x61	
	Checksum		21	0x11	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Explicit Rx Indicator

Frame Type:0x91

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

	Frame Fields		Offset	Example	Description
APL Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x18	
	Frame-specific Data	Frame Type	3	0x91	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		Reserved	12	0xFF	Reserved.
			13	0xFE	
		Source Endpoint	14	0xE0	Endpoint of the source that initiated the transmission
		Destination Endpoint	15	0xE0	Endpoint of the destination the message is addressed to.
		Cluster ID	16	0x22	Cluster ID the packet was addressed to.
			17	0x11	
		Profile ID	18	0xC1	Profile ID the packet was addressed to.
			19	0x05	
		Receive Options	20	0x02	0x01 – Packet Acknowledged 0x02 – Packet was a broadcast packet
		Received Data	21	0x52	Received RF data
			22	0x78	
			23	0x44	
			24	0x61	
			25	0x74	
			26	0x61	
	Checksum		27	0x56	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Node Identification Indicator

Frame Type:0x95

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x25	
	Frame-specific Data	Frame Type	3	0x95	
		64-bit Source Address	MSB 4	0x00	64-bit address of sender
			5	0x13	
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		Reserved	12	0xFF	Reserved.
			13	0xFE	
		Receive Options	14	0xC2	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet 0x40 - Point-multipoint packet 0x80 - Directed broadcast packet 0xC0 - DigiMesh packet
		Reserved	15	0xFF	Reserved
		Destination Endpoint	15	0xE0	Endpoint of the destination the message is addressed to.
		Cluster ID	16	0x22	Cluster ID the packet was addressed to.
			17	0x11	
		Profile ID	18	0xC1	Profile ID the packet was addressed to.
			19	0x05	
		Receive Options	20	0x02	0x01 – Packet Acknowledged 0x02 – Packet was a broadcast packet
		Received Data	21	0x52	Received RF data
			22	0x78	
			23	0x44	
			24	0x61	
			25	0x74	
			26	0x61	
	Checksum		27	0x56	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: If the commissioning push button is pressed on a remote router device with 64-bit address 0x0013a200407402ac and default NI string, the following node identification indicator would be received: 0x7e 0025 9500 13a2 0040 7402 acff fec2 fffe 0013 a200 4074 02ac 2000 fffe 0101 c105 101e 000c 0000 2e33

Remote Command Response

Frame Type: 0x97

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

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x13	
	Frame-specific Data	Frame Type	3	0x97	
		Frame ID	4	0x55	This is the same value passed in to the request.
		64-bit Source (remote) Address	MSB 5	0x00	The address of the remote radio returning this response.
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x52	
			11	0x2B	
			LSB 12	0xAA	
		Reserved	13	0xFF	Reserved
			14	0xFE	
		AT Commands	15	0x53	Name of the command
			16	0x4C	
		Command Status	17	0x00	The least significan nibble indicates the command status: 0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter The most significant nibble is a bitfield as follows: 0x40 = The RSSI field is invalid and should be ignored. Software prior to version 8x60 did not include RSSI information 0x80 = Response is a remote command.
		Command Data	18	0x40	The value of the required register
			19	0x52	
			20	0x2B	
			21	0xAA	
	Checksum		22	0xF4	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

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

Appendix A: Agency Certifications

Europe (ETSI)

The XBee RF Modules (excluding the PRO) have been certified for use in several European countries. For a complete list, refer to www.digi.com

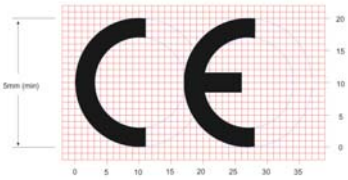
If the XBee RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product to the European harmonized EMC and low-voltage/safety standards. A Declaration of Conformity must be issued for each of these standards and kept on file as described in Annex II of the R&TTE Directive.

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

OEM Labeling Requirements

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

CE Labeling Requirements



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

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

Restrictions

According to REC70-03E, the following restrictions for radio operation apply:

Austria: Not implemented Planned

Georgia: Not implemented

Greece: Limited implementation to 863-865 MHz

Norway: Not implemented

Russian Federation:

- Limited implementation 864-865 MHz with max e.r.p. 25 mW, duty cycle 0.1% or LBT.
- Forbidden to use at the airports (aerodromes)

Spain: Limited implementation to the band 863-865 MHz

Sweden: Not implemented

The Netherlands: Not implemented Under study

Ukraine: Limited implementation 863-865 / 868-868.6 / 868.6-868.7 / 869.2-869.25 MHz

Declarations of Conformity

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

Important Note:

Digi does not list the entire set of standards that must be met for each country. Digi customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. For more information relating to European compliance of an OEM product incorporating the XBee RF Module, contact Digi, or refer to the following web site: www.cept.org. Search for “short range device regulations”.

XBee RF Module

The following antennas have been tested and approved for use with the embedded XBee RF Module:

- dipole (2.1 dBi, digi PN A08-HABUF-P5I
- PCB antenna (-9 dBi), which is included in the module

Appendix B: Migrating from XBee to SMT Modules

The XBee 865/868 LP modules are designed to be compatible with the XBee through-hole modules. The SMT modules have all the features of the through-hole modules, and offer the increased feature set described in this user's guide.

Pin Mapping

Mapping of the XBee SMT module pins to the XBee through-hole pins is shown in the following table.

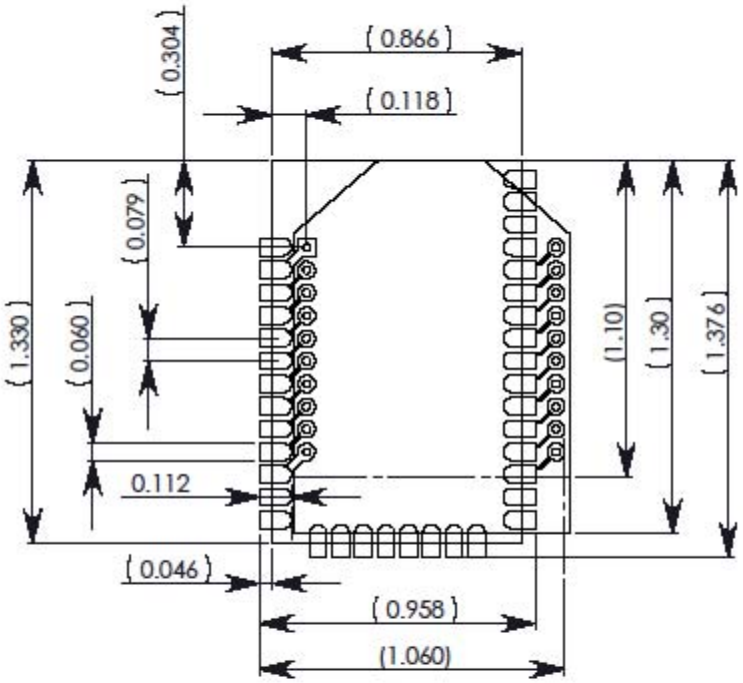
SMT Pin #	Name	Through Hole Pin #
1	GND	
2	VCC	1
3	DOUT / DIO13	2
4	DIN / $\overline{\text{CONFIG}}$ / DIO14	3
5	DIO12	4
6	$\overline{\text{RESET}}$	5
7	RSSI PWM / DIO10	6
8	PWM1 / DIO11	7
9	[reserved]	8
10	$\overline{\text{DTR}}$ / SLEEP_RQ / DIO8	9
11	GND	10
12	SPI_$\overline{\text{ATTN}}$ / DIO19	
13	GND	
14	SPI_CLK / DIO18	
15	SPI_$\overline{\text{SSEL}}$ / DIO17	
16	SPI_MOSI / DIO16	
17	SPI_MISO / DIO15	
18	[reserved]	
19	[reserved]	
20	[reserved]	
21	[reserved]	
22	GND	
23	[reserved]	
24	DIO4	11
25	$\overline{\text{CTS}}$ / DIO7	12
26	ON / $\overline{\text{SLEEP}}$ / DIO9	13
27	VREF	14
28	ASSOCIATE / DIO5	15
29	$\overline{\text{RTS}}$ / DIO6	16

SMT Pin #	Name	Through Hole Pin #
30	AD3 / DIO3	17
31	AD2 / DIO2	18
32	AD1 / DIO1	19
33	AD0 / DIO0	20
34	[reserved]	
35	GND	
36	RF	
37	[reserved]	

Mounting

One of the important differences between the SMT and the through hole modules is the way they mount to the PCB. The XBee through hole module is designed with through-hole pins, while the SMT module is designed with Surface Mount Technology (SMT). As such, different mounting techniques are required.

Digi International has designed a footprint which will allow either module to be attached to a PCB. The layout is shown below. All dimensions are in inches.



The round holes in the diagram are for the XBee through-hole design, and the semi-oval pads are for the XBee SMT design. Pin 1 of the through-hole design is lined up with pin 1 of the SMT design, but the pins are actually offset by one pad (see Pin Mapping above). By using diagonal traces to connect the appropriate pins, the layout will work for both modules.

Information on attaching the XBee SMT module is included in Appendix C below.

Appendix C: Manufacturing Information

The XBee SMT module is designed for surface mount on the OEM PCB. It has castellated pads to allow for easy solder attach inspection. The pads are all located on the edge of the module, so that there are no hidden solder joints on these modules.

Recommended Solder Reflow Cycle

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

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

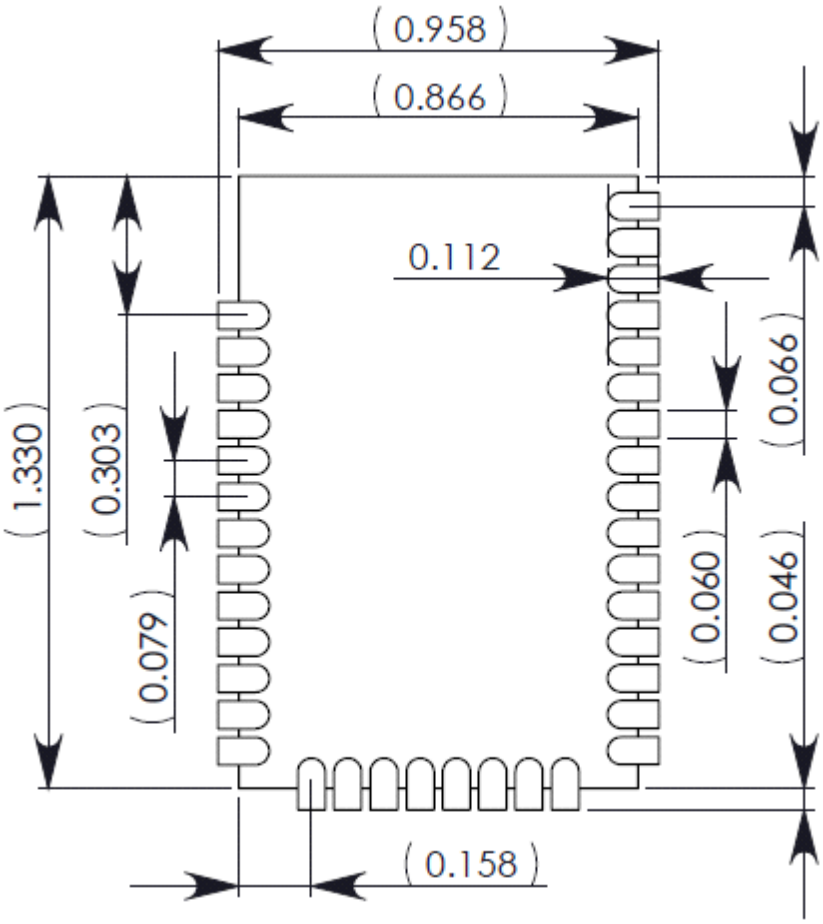
The maximum temperature should not exceed 260 degrees Celsius.

The module will reflow during this cycle, and therefore must not be reflowed upside down. Care should be taken not to jar the module while the solder is molten, as parts inside the module can be removed from their required locations.

Hand soldering is possible and should be done in accordance with approved standards. This module has a Moisture Sensitivity Level (MSL) of 3.

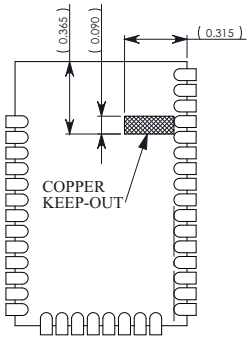
Recommended Footprint

It is recommended that you use the PCB footprint shown below for surface mounting. Dimensions are in inches.



The solder footprint should be matched to the copper pads, but may need to be adjusted depending on the specific needs of assembly and product standards.

While the underside of the module is mostly coated with solder resist, it is recommended that the copper layer directly below the module 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 module (see below). Furthermore, these modules have a ground plane in the middle on the back side for shielding purposes, which can be affected by copper traces directly below the module.



Flux and Cleaning

It is recommended that a “no clean” solder paste be used in assembling these modules. This will eliminate the clean step and ensure unwanted residual flux is not left under the module where it is difficult to remove. In addition:

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

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

Reworking

Rework should never be performed on the module itself. The module has been optimized to give the best possible performance, and reworking the module itself will void warranty coverage and certifications. We recognize that some customers will choose to rework and void the warranty; the following information is given as a guideline in such cases to increase the chances of success during rework, though the warranty is still voided.

The module may be removed from the OEM PCB by the use of a hot air rework station, or hot plate. Care should be taken not to overheat the module. During rework, the module temperature may rise above its internal solder melting point and care should be taken not to dislodge internal components from their intended positions.