My Project

Generated by Doxygen 1.7.6.1

Sat Jun 2 2012 23:28:26

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

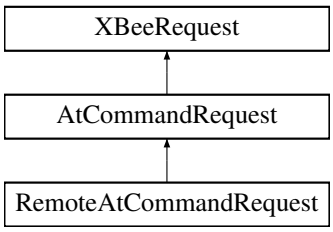
Chapter 3

Class Documentation

3.1 AtCommandRequest Class Reference

`#include <XBee.h>`

Inheritance diagram for AtCommandRequest:



Public Member Functions

- **AtCommandRequest** (uint8_t ∗command)
- **AtCommandRequest** (uint8_t ∗command, uint8_t ∗commandValue, uint8_t commandValueLength)
- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()
- uint8_t ∗ **getCommand** ()
- void **setCommand** (uint8_t ∗command)
- uint8_t ∗ **getCommandValue** ()
- void **setCommandValue** (uint8_t ∗command)
- uint8_t **getCommandValueLength** ()
- void **setCommandValueLength** (uint8_t length)
- void clearCommandValue ()

3.1.1 Detailed Description

Represents an AT Command TX packet The command is used to configure the serially connected XBee radio

3.1.2 Member Function Documentation

3.1.2.1 void AtCommandRequest::clearCommandValue ( )

Clears the optional commandValue and commandValueLength so that a query may be sent

3.1.2.2 uint8_t AtCommandRequest::getFrameData (uint8_t *pos*) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements XBeeRequest.

Reimplemented in RemoteAtCommandRequest.

3.1.2.3 uint8_t AtCommandRequest::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements XBeeRequest.

Reimplemented in RemoteAtCommandRequest.

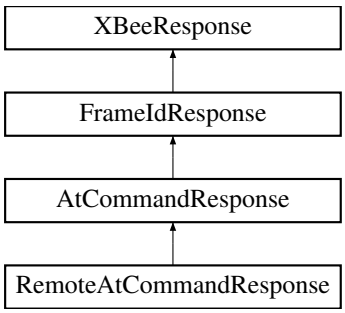
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.2 AtCommandResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for AtCommandResponse:



Public Member Functions

- uint8_t * getCommand ()
- uint8_t getStatus ()
- uint8_t * getValue ()
- uint8_t getValueLength ()
- bool isOk ()

3.2.1 Detailed Description

Represents an AT Command RX packet

3.2.2 Member Function Documentation

3.2.2.1 uint8_t * AtCommandResponse::getCommand ( )

Returns an array containing the two character command

Reimplemented in RemoteAtCommandResponse.

3.2.2.2 uint8_t AtCommandResponse::getStatus ( )

Returns the command status code. Zero represents a successful command

Reimplemented in RemoteAtCommandResponse.

3.2.2.3 uint8_t * AtCommandResponse::getValue ( )

Returns an array containing the command value. This is only applicable to query commands.

Reimplemented in RemoteAtCommandResponse.

3.2.2.4 uint8_t AtCommandResponse::getValueLength ( )

Returns the length of the command value array.

Reimplemented in RemoteAtCommandResponse.

3.2.2.5 bool AtCommandResponse::isOk ( )

Returns true if status equals AT_OK

Reimplemented in RemoteAtCommandResponse.

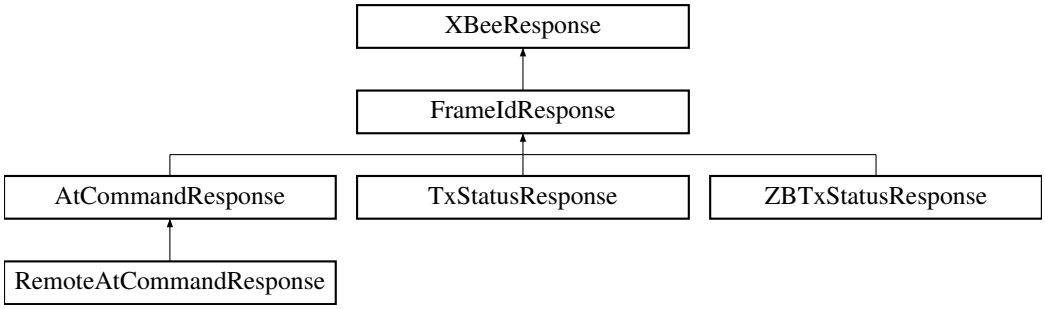
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.3 FrameIdResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for FrameIdResponse:



Public Member Functions

- uint8_t **getFrameId** ()

3.3.1 Detailed Description

This class is extended by all Responses that include a frame id

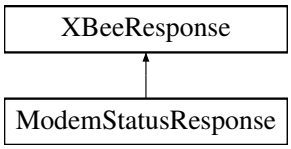
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.4 ModemStatusResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for ModemStatusResponse:



Public Member Functions

- uint8_t **getStatus** ()

3.4.1 Detailed Description

Represents a Modem Status RX packet

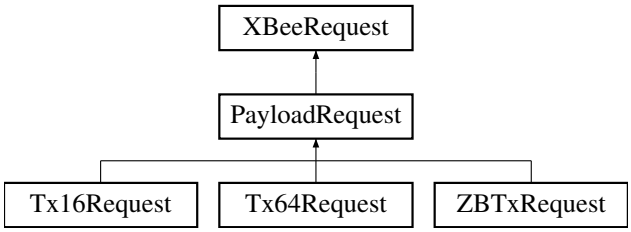
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.5 PayloadRequest Class Reference

`#include <XBee.h>`

Inheritance diagram for PayloadRequest:



Public Member Functions

- **PayloadRequest** (uint8_t apiId, uint8_t frameId, uint8_t *payload, uint8_t payloadLength)
- uint8_t * getPayload ()

- void setPayload (uint8_t *payloadPtr)
- uint8_t getPayloadLength ()
- void setPayloadLength (uint8_t payloadLength)

3.5.1 Detailed Description

All TX packets that support payloads extend this class

3.5.2 Member Function Documentation

3.5.2.1 uint8_t * PayloadRequest::getPayload ( )

Returns the payload of the packet, if not null

3.5.2.2 uint8_t PayloadRequest::getPayloadLength ( )

Returns the length of the payload array, as specified by the user.

3.5.2.3 void PayloadRequest::setPayload (uint8_t * *payloadPtr*)

Sets the payload array

3.5.2.4 void PayloadRequest::setPayloadLength (uint8_t *payloadLength*)

Sets the length of the payload to include in the request. For example if the payload array is 50 bytes and you only want the first 10 to be included in the packet, set the length to 10. Length must be <= to the array length.

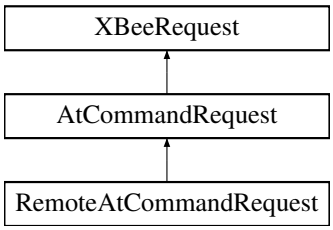
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.6 RemoteAtCommandRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RemoteAtCommandRequest:



Public Member Functions

- RemoteAtCommandRequest (uint16_t remoteAddress16, uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength)
- RemoteAtCommandRequest (uint16_t remoteAddress16, uint8_t *command)
- RemoteAtCommandRequest (XBeeAddress64 &remoteAddress64, uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength)
- RemoteAtCommandRequest (XBeeAddress64 &remoteAddress64, uint8_t *command)
- uint16_t **getRemoteAddress16** ()
- void **setRemoteAddress16** (uint16_t remoteAddress16)
- XBeeAddress64 & **getRemoteAddress64** ()
- void **setRemoteAddress64** (XBeeAddress64 &remoteAddress64)
- bool **getApplyChanges** ()
- void **setApplyChanges** (bool applyChanges)
- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()

Static Public Attributes

- static XBeeAddress64 **broadcastAddress64** = XBeeAddress64(0x0, BROADCAST_ADDRESS)

3.6.1 Detailed Description

Represents an Remote AT Command TX packet The command is used to configure a remote XBee radio

3.6.2 Constructor & Destructor Documentation

3.6.2.1 RemoteAtCommandRequest::RemoteAtCommandRequest (uint16_t *remoteAddress16,* uint8_t * *command,* uint8_t * *commandValue,* uint8_t *commandValueLength*)

Creates a RemoteAtCommandRequest with 16-bit address to set a command. 64-bit address defaults to broadcast and applyChanges is true.

3.6.2.2 RemoteAtCommandRequest::RemoteAtCommandRequest (uint16_t *remoteAddress16,* uint8_t * *command*)

Creates a RemoteAtCommandRequest with 16-bit address to query a command. 64-bit address defaults to broadcast and applyChanges is true.

3.6.2.3 RemoteAtCommandRequest::RemoteAtCommandRequest (XBeeAddress64 & *remoteAddress64,* uint8_t * *command,* uint8_t * *commandValue,* uint8_t *commandValueLength*)

Creates a RemoteAtCommandRequest with 64-bit address to set a command. 16-bit address defaults to broadcast and applyChanges is true.

3.6.2.4 RemoteAtCommandRequest::RemoteAtCommandRequest (XBeeAddress64 & *remoteAddress64,* uint8_t * *command*)

Creates a RemoteAtCommandRequest with 16-bit address to query a command. 16-bit address defaults to broadcast and applyChanges is true.

3.6.3 Member Function Documentation

3.6.3.1 uint8_t RemoteAtCommandRequest::getFrameData (uint8_t *pos*) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Reimplemented from AtCommandRequest.

3.6.3.2 uint8_t RemoteAtCommandRequest::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Reimplemented from AtCommandRequest.

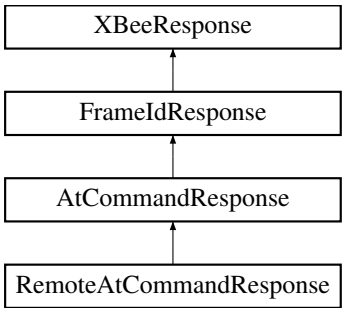
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.7 RemoteAtCommandResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RemoteAtCommandResponse:



Public Member Functions

- uint8_t * getCommand ()
- uint8_t getStatus ()
- uint8_t * getValue ()
- uint8_t getValueLength ()
- uint16_t getRemoteAddress16 ()
- XBeeAddress64 & getRemoteAddress64 ()
- bool isOk ()

3.7.1 Detailed Description

Represents a Remote AT Command RX packet

3.7.2 Member Function Documentation

3.7.2.1 uint8_t * RemoteAtCommandResponse::getCommand ( )

Returns an array containing the two character command

Reimplemented from AtCommandResponse.

3.7.2.2 uint16_t RemoteAtCommandResponse::getRemoteAddress16 ( )

Returns the 16-bit address of the remote radio

3.7.2.3 XBeeAddress64 & RemoteAtCommandResponse::getRemoteAddress64 ( )

Returns the 64-bit address of the remote radio

3.7.2.4 uint8_t RemoteAtCommandResponse::getStatus ( )

Returns the command status code. Zero represents a successful command

Reimplemented from AtCommandResponse.

3.7.2.5 uint8_t * RemoteAtCommandResponse::getValue ( )

Returns an array containing the command value. This is only applicable to query commands.

Reimplemented from AtCommandResponse.

3.7.2.6 uint8_t RemoteAtCommandResponse::getValueLength ( )

Returns the length of the command value array.

Reimplemented from AtCommandResponse.

3.7.2.7 bool RemoteAtCommandResponse::isOk ( )

Returns true if command was successful

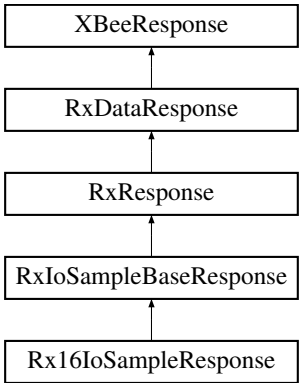
Reimplemented from AtCommandResponse.

The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.8 Rx16IoSampleResponse Class Reference

Inheritance diagram for Rx16IoSampleResponse:



Public Member Functions

- uint16_t **getRemoteAddress16** ()
- uint8_t **getRssiOffset** ()

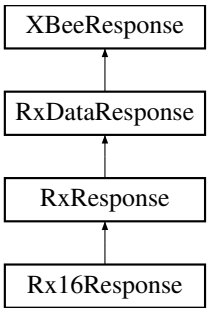
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.9 Rx16Response Class Reference

`#include <XBee.h>`

Inheritance diagram for Rx16Response:



Public Member Functions

- uint8_t **getRssiOffset** ()
- uint16_t **getRemoteAddress16** ()

Protected Attributes

- uint16_t **_remoteAddress**

3.9.1 Detailed Description

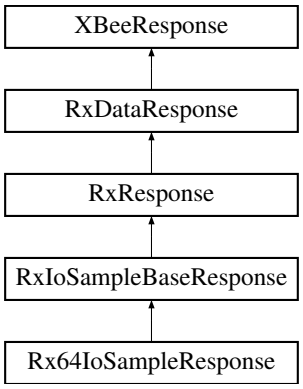
Represents a Series 1 16-bit address RX packet

The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.10 Rx64IoSampleResponse Class Reference

Inheritance diagram for Rx64IoSampleResponse:



Public Member Functions

- XBeeAddress64 & **getRemoteAddress64** ()
- uint8_t **getRssiOffset** ()

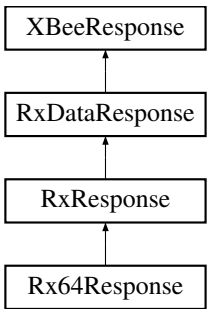
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.11 Rx64Response Class Reference

`#include <XBee.h>`

Inheritance diagram for Rx64Response:



Public Member Functions

- uint8_t **getRssiOffset** ()
- XBeeAddress64 & **getRemoteAddress64** ()

3.11.1 Detailed Description

Represents a Series 1 64-bit address RX packet

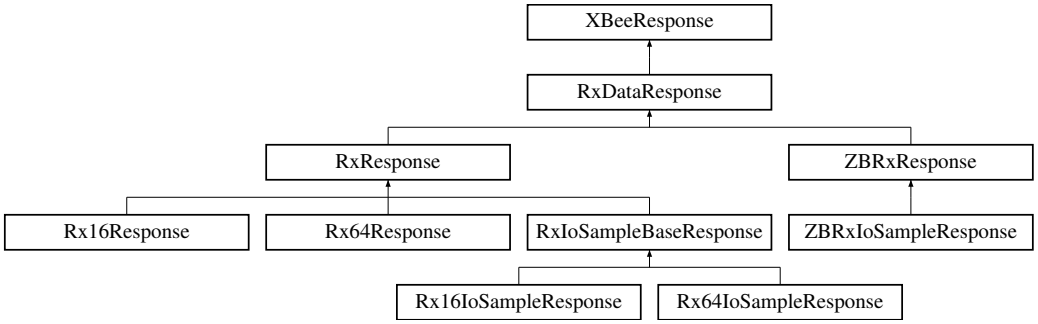
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.12 RxDataResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for RxDataResponse:



Public Member Functions

- uint8_t getData (int index)
- uint8_t * getData ()
- virtual uint8_t getDataLength ()=0
- virtual uint8_t getDataOffset ()=0

3.12.1 Detailed Description

Common functionality for both Series 1 and 2 data RX data packets

3.12.2 Member Function Documentation

3.12.2.1 uint8_t RxDataResponse::getData (int *index*)

Returns the specified index of the payload. The index may be 0 to getDataLength() - 1
This method is deprecated; use uint8_t* getData()

3.12.2.2 uint8_t * RxDataResponse::getData ( )

Returns the payload array. This may be accessed from index 0 to getDataLength() - 1

3.12.2.3 virtual uint8_t RxDataResponse::getDataLength ( ) [pure virtual]

Returns the length of the payload

Implemented in RxResponse, and ZBRxResponse.

3.12.2.4 virtual uint8_t RxDataResponse::getDataOffset ( ) [pure virtual]

Returns the position in the frame data where the data begins

Implemented in RxResponse, and ZBRxResponse.

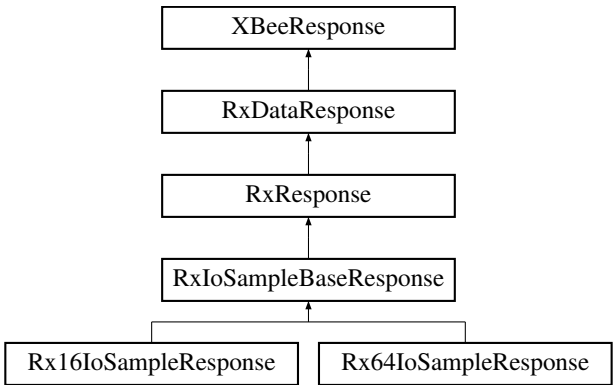
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.13 RxIoSampleBaseResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for RxIoSampleBaseResponse:



Public Member Functions

- uint8_t getSampleSize ()
- bool **containsAnalog** ()
- bool **containsDigital** ()
- bool isAnalogEnabled (uint8_t pin)
- bool isDigitalEnabled (uint8_t pin)
- uint16_t getAnalog (uint8_t pin, uint8_t sample)
- bool isDigitalOn (uint8_t pin, uint8_t sample)
- uint8_t **getSampleOffset** ()

3.13.1 Detailed Description

Represents a Series 1 RX I/O Sample packet

3.13.2 Member Function Documentation

3.13.2.1 uint16_t RxIoSampleBaseResponse::getAnalog (uint8_t *pin,* uint8_t *sample*)

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:0-5. Sample index starts at 0

3.13.2.2 uint8_t RxIoSampleBaseResponse::getSampleSize ( )

Returns the number of samples in this packet

3.13.2.3 bool RxIoSampleBaseResponse::isAnalogEnabled (uint8_t *pin*)

Returns true if the specified analog pin is enabled

3.13.2.4 bool RxIoSampleBaseResponse::isDigitalEnabled (uint8_t *pin*)

Returns true if the specified digital pin is enabled

3.13.2.5 bool RxIoSampleBaseResponse::isDigitalOn (uint8_t *pin,* uint8_t *sample*)

Returns true if the specified pin is high/on. Valid pins include DIO:0-8. Sample index starts at 0

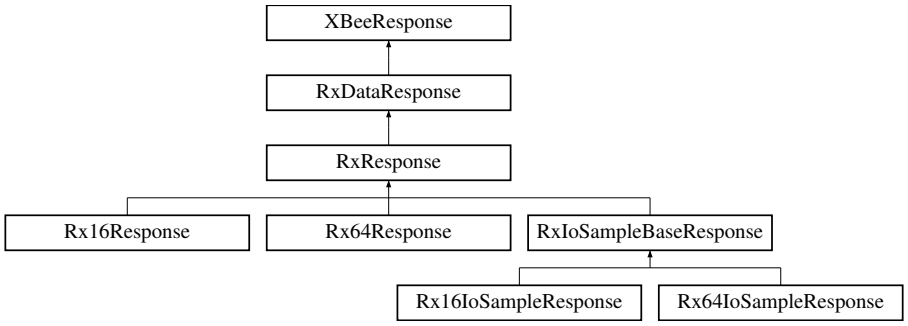
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.14 RxResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for RxResponse:



Public Member Functions

- uint8_t **getRssi** ()
- uint8_t **getOption** ()
- bool **isAddressBroadcast** ()
- bool **isPanBroadcast** ()
- uint8_t getDataLength ()
- uint8_t getDataOffset ()
- virtual uint8_t **getRssiOffset** ()=0

3.14.1 Detailed Description

Represents a Series 1 RX packet

3.14.2 Member Function Documentation

3.14.2.1 uint8_t RxResponse::getDataLength ( ) [virtual]

Returns the length of the payload

Implements RxDataResponse.

3.14.2.2 uint8_t RxResponse::getDataOffset ( ) [virtual]

Returns the position in the frame data where the data begins

Implements RxDataResponse.

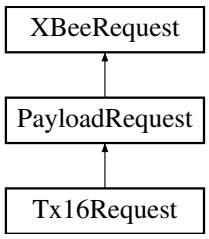
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.15 Tx16Request Class Reference

`#include <XBee.h>`

Inheritance diagram for Tx16Request:



Public Member Functions

- **Tx16Request** (uint16_t addr16, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameId)
- Tx16Request (uint16_t addr16, uint8_t *payload, uint8_t payloadLength)
- Tx16Request ()
- uint16_t **getAddress16** ()
- void **setAddress16** (uint16_t addr16)
- uint8_t **getOption** ()
- void **setOption** (uint8_t option)
- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()

3.15.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX_16_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Tx16Request::Tx16Request (uint16_t *addr16*, uint8_t * *payload*, uint8_t *payloadLength*)

Creates a Unicast Tx16Request with the ACK option and DEFAULT_FRAME_ID

3.15.2.2 Tx16Request::Tx16Request ( )

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

3.15.3 Member Function Documentation

3.15.3.1 uint8_t Tx16Request::getFrameData (uint8_t *pos*) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements XBeeRequest.

3.15.3.2 uint8_t Tx16Request::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements XBeeRequest.

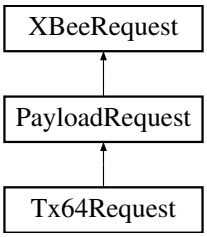
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.16 Tx64Request Class Reference

`#include <XBee.h>`

Inheritance diagram for Tx64Request:



Public Member Functions

- **Tx64Request** (XBeeAddress64 &addr64, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameId)

- Tx64Request (XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)
- Tx64Request ()
- XBeeAddress64 & **getAddress64** ()
- void **setAddress64** (XBeeAddress64 &addr64)
- uint8_t **getOption** ()
- void **setOption** (uint8_t option)
- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()

3.16.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX_64_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 Tx64Request::Tx64Request (XBeeAddress64 & *addr64*, uint8_t * *payload*, uint8_t *payloadLength*)

Creates a unicast Tx64Request with the ACK option and DEFAULT_FRAME_ID

3.16.2.2 Tx64Request::Tx64Request ( )

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

3.16.3 Member Function Documentation

3.16.3.1 uint8_t Tx64Request::getFrameData (uint8_t *pos*) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements XBeeRequest.

3.16.3.2 uint8_t Tx64Request::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements XBeeRequest.

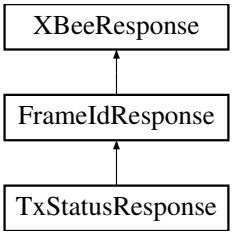
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.17 TxStatusResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for TxStatusResponse:



Public Member Functions

- uint8_t **getStatus** ()
- bool **isSuccess** ()

3.17.1 Detailed Description

Represents a Series 1 TX Status packet

The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.18 XBee Class Reference

`#include <XBee.h>`

Public Member Functions

- void readPacket ()
- bool readPacket (int timeout)
- void readPacketUntilAvailable ()

- void begin (long baud)
- void **getResponse** (XBeeResponse &response)
- XBeeResponse & getResponse ()
- void send (XBeeRequest &request)
- uint8_t getNextFrameId ()
- void setSerial (HardwareSerial &serial)

3.18.1 Detailed Description

Primary interface for communicating with an XBee Radio. This class provides methods for sending and receiving packets with an XBee radio via the serial port. The XBee radio must be configured in API (packet) mode (AP=2) in order to use this software.

Since this code is designed to run on a microcontroller, with only one thread, you are responsible for reading the data off the serial buffer in a timely manner. This involves a call to a variant of readPacket(...). If your serial port is receiving data faster than you are reading, you can expect to lose packets. Arduino only has a 128 byte serial buffer so it can easily overflow if two or more packets arrive without a call to readPacket(...)

In order to conserve resources, this class only supports storing one response packet in memory at a time. This means that you must fully consume the packet prior to calling readPacket(...), because calling readPacket(...) overwrites the previous response.

This class creates an array of size MAX_FRAME_DATA_SIZE for storing the response packet. You may want to adjust this value to conserve memory.

Author

Andrew Rapp

3.18.2 Member Function Documentation

3.18.2.1 void XBee::begin (long *baud*)

Starts the serial connection at the supplied baud rate

3.18.2.2 uint8_t XBee::getNextFrameId ( )

Returns a sequential frame id between 1 and 255

3.18.2.3 XBeeResponse & XBee::getResponse ( )

Returns a reference to the current response Note: once readPacket is called again this response will be overwritten!

3.18.2.4 void XBee::readPacket ( )

Reads all available serial bytes until a packet is parsed, an error occurs, or the buffer is empty. You may call *xbee*.getResponse().isAvailable() after calling this method to determine if a packet is ready, or *xbee*.getResponse().isError() to determine if a error occurred.

This method should always return quickly since it does not wait for serial data to arrive. You will want to use this method if you are doing other timely stuff in your loop, where a delay would cause problems. NOTE: calling this method resets the current response, so make sure you first consume the current response

3.18.2.5 bool XBee::readPacket (int *timeout*)

Waits a maximum of *timeout* milliseconds for a response packet before timing out; returns true if packet is read. Returns false if timeout or error occurs.

3.18.2.6 void XBee::readPacketUntilAvailable ( )

Reads until a packet is received or an error occurs. Caution: use this carefully since if you don't get a response, your Arduino code will hang on this call forever!! often it's better to use a timeout: readPacket(int)

3.18.2.7 void XBee::send (XBeeRequest & *request*)

Sends a XBeeRequest (TX packet) out the serial port

3.18.2.8 void XBee::setSerial (HardwareSerial & *serial*)

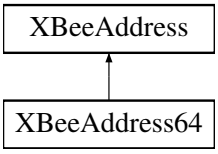
Specify the serial port. Only relevant for Arduinos that support multiple serial ports (e.g. Mega)

The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.19 XBeeAddress Class Reference

Inheritance diagram for XBeeAddress:



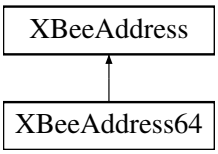
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.20 XBeeAddress64 Class Reference

`#include <XBee.h>`

Inheritance diagram for XBeeAddress64:



Public Member Functions

- **XBeeAddress64** (uint32_t msb, uint32_t lsb)
- uint32_t **getMsb** ()
- uint32_t **getLsb** ()
- void **setMsb** (uint32_t msb)
- void **setLsb** (uint32_t lsb)

3.20.1 Detailed Description

Represents a 64-bit XBee Address

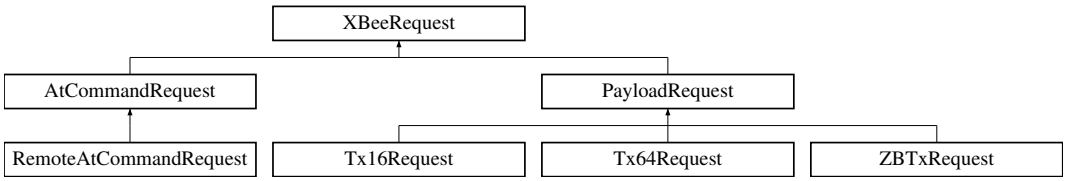
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.21 XBeeRequest Class Reference

`#include <XBee.h>`

Inheritance diagram for XBeeRequest:



Public Member Functions

- XBeeRequest (uint8_t apiId, uint8_t frameId)
- void setFrameId (uint8_t frameId)
- uint8_t getFrameId ()
- uint8_t getApiId ()
- virtual uint8_t getFrameData (uint8_t pos)=0
- virtual uint8_t getFrameDataLength ()=0

Protected Member Functions

- void **setApiId** (uint8_t apiId)

3.21.1 Detailed Description

Super class of all XBee requests (TX packets) Users should never create an instance of this class; instead use an subclass of this class It is recommended to reuse Subclasses of the class to conserve memory

This class allocates a buffer to

3.21.2 Constructor & Destructor Documentation

3.21.2.1 XBeeRequest::XBeeRequest (uint8_t *apiId,* uint8_t *frameId*)

Constructor TODO make protected

3.21.3 Member Function Documentation

3.21.3.1 uint8_t XBeeRequest::getApiId ( )

Returns the API id

3.21.3.2 virtual uint8_t XBeeRequest::getFrameData (uint8_t *pos*) [pure virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implemented in RemoteAtCommandRequest, AtCommandRequest, ZBTxRequest, -Tx64Request, and Tx16Request.

3.21.3.3 virtual uint8_t XBeeRequest::getFrameDataLength ( ) [pure virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implemented in RemoteAtCommandRequest, AtCommandRequest, ZBTxRequest, -Tx64Request, and Tx16Request.

3.21.3.4 uint8_t XBeeRequest::getFrameId ( )

Returns the frame id

3.21.3.5 void XBeeRequest::setFrameId (uint8_t *frameId*)

Sets the frame id. Must be between 1 and 255 inclusive to get a TX status response.

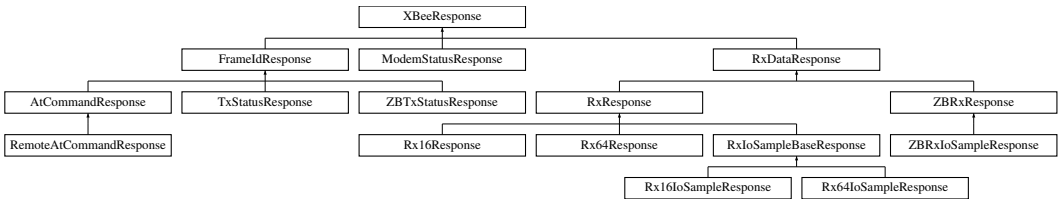
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.22 XBeeResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for XBeeResponse:



Public Member Functions

- XBeeResponse ()
- uint8_t getApiId ()
- void **setApiId** (uint8_t apiId)
- uint8_t getMsbLength ()
- void **setMsbLength** (uint8_t msbLength)
- uint8_t getLsbLength ()
- void **setLsbLength** (uint8_t lsbLength)
- uint8_t getChecksum ()
- void **setChecksum** (uint8_t checksum)
- uint8_t getFrameDataLength ()
- void **setFrameData** (uint8_t *frameDataPtr)
- uint8_t * getFrameData ()
- void **setFrameLength** (uint8_t frameLength)
- uint16_t getPacketLength ()
- void reset ()
- void init ()
- void getZBTxStatusResponse (XBeeResponse &response)
- void getZBRxResponse (XBeeResponse &response)
- void getZBRxIoSampleResponse (XBeeResponse &response)
- void getTxStatusResponse (XBeeResponse &response)
- void getRx16Response (XBeeResponse &response)
- void getRx64Response (XBeeResponse &response)
- void getRx16IoSampleResponse (XBeeResponse &response)
- void getRx64IoSampleResponse (XBeeResponse &response)
- void getAtCommandResponse (XBeeResponse &responses)
- void getRemoteAtCommandResponse (XBeeResponse &response)
- void getModemStatusResponse (XBeeResponse &response)
- bool isAvailable ()
- void **setAvailable** (bool complete)
- bool isError ()
- uint8_t getErrorCode ()
- void **setErrorCode** (uint8_t errorCode)

Protected Attributes

- uint8_t * **_frameDataPtr**

3.22.1 Detailed Description

The super class of all XBee responses (RX packets) Users should never attempt to create an instance of this class; instead create an instance of a subclass It is recommend to reuse subclasses to conserve memory

3.22.2 Constructor & Destructor Documentation

3.22.2.1 XBeeResponse::XBeeResponse ( )

Default constructor

3.22.3 Member Function Documentation

3.22.3.1 uint8_t XBeeResponse::getApiId ( )

Returns Api Id of the response

3.22.3.2 void XBeeResponse::getAtCommandResponse (XBeeResponse & *responses*)

Call with instance of AtCommandResponse only if getApiId() == AT_COMMAND_RESPONSE

3.22.3.3 uint8_t XBeeResponse::getChecksum ( )

Returns the packet checksum

3.22.3.4 uint8_t XBeeResponse::getErrorCode ( )

Returns an error code, or zero, if successful. Error codes include: CHECKSUM_FAILURE, PACKET_EXCEEDS_BYTE_ARRAY_LENGTH, UNEXPECTED_START_BYTE

3.22.3.5 uint8_t * XBeeResponse::getFrameData ( )

Returns the buffer that contains the response. Starts with byte that follows API ID and includes all bytes prior to the checksum Length is specified by getFrameDataLength() Note: Unlike Digi's definition of the frame data, this does not start with the API ID.. The

reason for this is all responses include an API ID, whereas my frame data includes only the API specific data.

3.22.3.6 uint8_t XBeeResponse::getFrameDataLength ( )

Returns the length of the frame data: all bytes after the api id, and prior to the checksum Note up to release 0.1.2, this was incorrectly including the checksum in the length.

3.22.3.7 uint8_t XBeeResponse::getLsbLength ( )

Returns the LSB length of the packet

3.22.3.8 void XBeeResponse::getModemStatusResponse (XBeeResponse & *response*)

Call with instance of ModemStatusResponse only if getApiId() == MODEM_STATUS_-RESPONSE

3.22.3.9 uint8_t XBeeResponse::getMsbLength ( )

Returns the MSB length of the packet

3.22.3.10 uint16_t XBeeResponse::getPacketLength ( )

Returns the length of the packet

3.22.3.11 void XBeeResponse::getRemoteAtCommandResponse (XBeeResponse & *response*)

Call with instance of RemoteAtCommandResponse only if getApiId() == REMOTE_AT-_COMMAND_RESPONSE

3.22.3.12 void XBeeResponse::getRx16IoSampleResponse (XBeeResponse & *response*)

Call with instance of Rx16IoSampleResponse only if getApiId() == RX_16_IO_RESPO-NSE

3.22.3.13 void XBeeResponse::getRx16Response (XBeeResponse & *response*)

Call with instance of Rx16Response only if getApiId() == RX_16_RESPONSE

3.22.3.14 void XBeeResponse::getRx64IoSampleResponse (XBeeResponse & *response*)

Call with instance of Rx64IoSampleResponse only if getApiId() == RX_64_IO_RESPONSE

3.22.3.15 void XBeeResponse::getRx64Response (XBeeResponse & *response*)

Call with instance of Rx64Response only if getApiId() == RX_64_RESPONSE

3.22.3.16 void XBeeResponse::getTxStatusResponse (XBeeResponse & *response*)

Call with instance of TxStatusResponse only if getApiId() == TX_STATUS_RESPONSE

3.22.3.17 void XBeeResponse::getZBRxIoSampleResponse (XBeeResponse & *response*)

Call with instance of ZBRxIoSampleResponse class only if getApiId() == ZB_IO_SAMPLE_RESPONSE to populate response

3.22.3.18 void XBeeResponse::getZBRxResponse (XBeeResponse & *response*)

Call with instance of ZBRxResponse class only if getApiId() == ZB_RX_RESPONSE to populate response

3.22.3.19 void XBeeResponse::getZBTxStatusResponse (XBeeResponse & *response*)

Call with instance of ZBTxStatusResponse class only if getApiId() == ZB_TX_STATUS_RESPONSE to populate response

3.22.3.20 void XBeeResponse::init ( )

Initializes the response

3.22.3.21 bool XBeeResponse::isAvailable ( )

Returns true if the response has been successfully parsed and is complete and ready for use

3.22.3.22 bool XBeeResponse::isError ( )

Returns true if the response contains errors

3.22.3.23 void XBeeResponse::reset ( )

Resets the response to default values

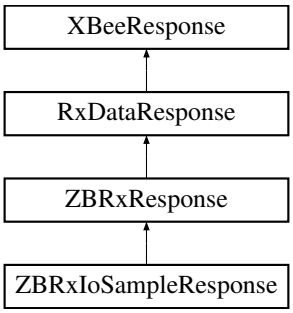
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.23 ZBRxIoSampleResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for ZBRxIoSampleResponse:



Public Member Functions

- bool **containsAnalog** ()
- bool **containsDigital** ()
- bool isAnalogEnabled (uint8_t pin)
- bool isDigitalEnabled (uint8_t pin)
- uint16_t getAnalog (uint8_t pin)
- bool isDigitalOn (uint8_t pin)
- uint8_t **getDigitalMaskMsb** ()
- uint8_t **getDigitalMaskLsb** ()
- uint8_t **getAnalogMask** ()

3.23.1 Detailed Description

Represents a Series 2 RX I/O Sample packet

3.23.2 Member Function Documentation

3.23.2.1 uint16_t ZBRxIoSampleResponse::getAnalog (uint8_t *pin*)

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:xxx.

3.23.2.2 bool ZBRxIoSampleResponse::isAnalogEnabled (uint8_t *pin*)

Returns true if the pin is enabled

3.23.2.3 bool ZBRxIoSampleResponse::isDigitalEnabled (uint8_t *pin*)

Returns true if the pin is enabled

3.23.2.4 bool ZBRxIoSampleResponse::isDigitalOn (uint8_t *pin*)

Returns true if the specified pin is high/on. Valid pins include DIO:xxx.

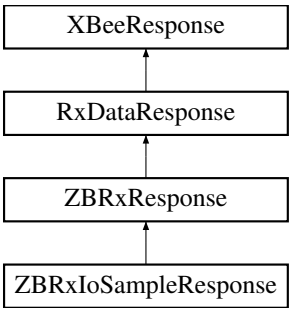
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.24 ZBRxResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for ZBRxResponse:



Public Member Functions

- XBeeAddress64 & **getRemoteAddress64** ()
- uint16_t **getRemoteAddress16** ()
- uint8_t **getOption** ()

- uint8_t getDataLength ()
- uint8_t getDataOffset ()

3.24.1 Detailed Description

Represents a Series 2 RX packet

3.24.2 Member Function Documentation

3.24.2.1 uint8_t ZBRxResponse::getDataLength ( ) [virtual]

Returns the length of the payload

Implements RxDataResponse.

3.24.2.2 uint8_t ZBRxResponse::getDataOffset ( ) [virtual]

Returns the position in the frame data where the data begins

Implements RxDataResponse.

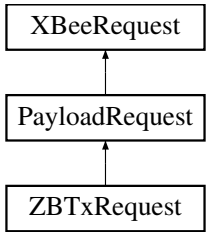
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.25 ZBTxRequest Class Reference

`#include <XBee.h>`

Inheritance diagram for ZBTxRequest:



Public Member Functions

- ZBTxRequest (XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)

- **ZBTxRequest** (XBeeAddress64 &addr64, uint16_t addr16, uint8_t broadcastRadius, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameId)
- ZBTxRequest ()
- XBeeAddress64 & **getAddress64** ()
- uint16_t **getAddress16** ()
- uint8_t **getBroadcastRadius** ()
- uint8_t **getOption** ()
- void **setAddress64** (XBeeAddress64 &addr64)
- void **setAddress16** (uint16_t addr16)
- void **setBroadcastRadius** (uint8_t broadcastRadius)
- void **setOption** (uint8_t option)

Protected Member Functions

- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()

3.25.1 Detailed Description

Represents a Series 2 TX packet that corresponds to Api Id: ZB_TX_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 72 bytes is the maximum for ZNet firmware and ZB Pro firmware provides the AT-NP command to get the max supported payload size. This command is useful since the maximum payload size varies according to certain settings, such as encryption. ZB Pro firmware provides a PAYLOAD_TOO_LARGE that is returned if payload size exceeds the maximum.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 ZBTxRequest::ZBTxRequest (XBeeAddress64 & *addr64*, uint8_t * *payload*, uint8_t *payloadLength*)

Creates a unicast ZBTxRequest with the ACK option and DEFAULT_FRAME_ID

3.25.2.2 ZBTxRequest::ZBTxRequest ( )

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

3.25.3 Member Function Documentation

3.25.3.1 uint8_t ZBTxRequest::getFrameData (uint8_t *pos*) [protected, virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements XBeeRequest.

3.25.3.2 uint8_t ZBTxRequest::getFrameDataLength ( ) [protected, virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements XBeeRequest.

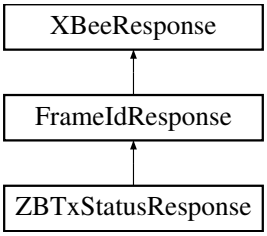
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

3.26 ZBTxStatusResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for ZBTxStatusResponse:



Public Member Functions

- uint16_t **getRemoteAddress** ()
- uint8_t **getTxRetryCount** ()
- uint8_t **getDeliveryStatus** ()
- uint8_t **getDiscoveryStatus** ()
- bool **isSuccess** ()

3.26.1 Detailed Description

Represents a Series 2 TX status packet

The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp