

Comparing the Digi® XBee® API with EmberZNet EM260 API

White Paper

Abstract

Digi's XBee ZB module and the EM260 co-processor are similar in many respects. Both are designed to provide an interface to a ZigBee® network while allowing external applications to be developed around their interface. The EM260 offers cheaper hardware than the XBee and provides access to the lowest ZigBee functions. Designing around the EM260 requires some RF design and additional regulatory testing (FCC, ETSI, etc.) than is required of an XBee-based design. Products that integrate the XBee module can avoid delving into RF design and the associated costs of RF testing. Aside from hardware and certifications differences, the other fundamental difference is in the software interface to the product (API). This document provides a comparison between the XBee ZB and the EM260 software interfaces.

Table of Contents

The Digi XBee API	2
API Construction	2
API Frames	2
EmberZNet EM260 API	4
API Construction	4
API Frames	4
XBee ZB and EM260 Characteristics	5
Digi API Characteristics	5
EM260 API Characteristics	5
Digi API and EM260 API Comparison by Topic	6
Forming/Joining a Network	6
Data Transmission and Reception	7
Appendix A – XBee ZB API Frames	8
Get/Set Command Registers	8
Data Transmission	9
Transmit Status	9
Data Reception	9
Modem Status	10
Appendix B – Abbreviated EM260 API	11

The Digi XBee API

API Construction

XBee API frame formats are constructed as follows.

Field	Description
Start delimiter	1-byte start delimiter (0x7E)
Length	2-byte frame length value
Frame Type	1-byte API frame type value that indicates what type of API command follows
Frame Data	Payload used by the API frame. Variable size depending on the API frame type
Checksum	1-byte checksum used to verify integrity of the API frame

API Frames

The Digi API can be summed up into the following core categories:

- Get/Set command register (local and remote)
- Transmit data
- Transmit status
- Receive data indicator
- Modem status

A short discussion of these frame types is included in Appendix A. See the XBee product manual for a full description of the API frames.

EmberZNet EM260 API

API Construction

The Ember EM260 API is formatted as follows:

Field	Description
Sequence	A sequence number set by the host microcontroller. The last received sequence number is included in response frames from the EM260.
Frame Control	Includes command options values in a request, and status values in a response frame.
Frame ID	The command identifier. This value determines how the rest of the command should be formatted.

API Frames

The EM260 API includes the following categories:

- Configuration frames
- Utilities frames
- Networking frames
- Messaging frames
- Security frames
- Trust center frames

Each API frame type includes various individual commands. The payload for the API depends on which command is being issued or received. An abbreviated listing of some EM260 API frames is found in Appendix B. See the Ember EZSP Reference Guide for a more complete listing.

XBee ZB and EM260 Characteristics

Digi API Characteristics

- Simple API frame to set/read command registers on any device in the network.
- API simplifies joining by hiding ZigBee details when not needed.
 - o Application sets the Scan Channels, PAN ID, and possibly security and stack profile registers, and the module takes care of joining.
 - o Networking and addressing parameters are readable using a simple API frame.
 - o An AT command register indicates the status of the last join attempt.
- Simple, but flexible API hides the ZigBee details in data transmissions.
 - o Applications that require endpoints, cluster IDs and profile IDs in addressing can use them.
 - o Applications that don't need these don't have to think about them.
- End device management.
 - o End device firmware supports either pin or cyclic sleep.
 - o Cyclic sleep wake and sleep times configurable with a few AT commands (SP, SN, SO).
 - o If an end device loses contact with its parent on the network, XBee firmware causes end device to reconnect to network automatically. The application does not have to manage end device connectivity.
 - o End device polls regularly when awake to ensure data reception. The application does not need to worry about scheduling polling, reacting to failures, deciding when to leave or join a network, etc.
- Advanced network discovery, commissioning, and diagnostics tools exist to alleviate the need to develop these tools. The user can instead focus on their application and tap into existing tools.
- The XBee modules are being tested and deployed in large networks.
- Module firmware supports advanced ZigBee functionality including mobile end devices, source routing, many-to-one routing, ZDO discovery and management support, etc.
- Modules can form/join a network out of the box.

EM260 API Characteristics

- Routing table, address table and neighbor table sizes are adjustable.
- More complicated API with various payload sizes, depending on the command.
- The external host microcontroller that talks to the EM260 must manage many low level ZigBee intricacies in the application. For example, the application must:
 - o Configure destination address, source and destination endpoint, cluster ID, profile ID, and data transmission options for each transmission.
 - o Manage the address table that maps 64-bit and 16-bit addresses on the EM260.
 - o Manage end devices – when should they poll? What should they do if the parent doesn't respond to a poll? What should they do if they are asked to leave? When and how should they attempt to join? When and how long can end devices sleep? How long should the parent buffer incoming RF data for the end device? How can the end device reconnect to the network if its parent goes away, if the PAN ID changes?

Digi API and EM260 API Comparison by Topic

The following is a quick comparison of API functions/command registers that are used for two common scenarios:

- Forming/Joining a ZigBee network
- Data transmission and reception

Forming/Joining a Network

Digi XBee ZB API	EM260 Form/Join API
<p><i>XBee Command Registers</i></p> <p>SC – Scan channels bitmask ID – Extended PAN ID SD – Scan duration AI – Join status code (0 = joined) ZS – Set stack profile</p> <p>NR – Network reset (leave/rejoin) NJ – Permit Joining</p> <p>MY – Read node 16-bit address CH – Read channel EE – Security enable KY – Set security link key NK – Set network security key (coordinator)</p>	<p><i>EM260 Functions</i></p> <p>networkInit() networkState() stackStatusHandler() startScan() energyScanResultHandler() networkFoundHandler() scanCompleteHandler() stopScan() formNetwork() joinNetwork() scanAndFormNetwork() scanAndJoinNetwork() scanErrorHandler()</p> <p>leaveNetwork() findAndRejoinNetwork() permitJoining()</p> <p>childJoinHandler() getNodeId() getNetworkParameters() setInitialSecurityState() TrustCenterJoinHandler() broadcastNextNetworkKeyHandler() broadcastNetworkKeySwitch()</p>

Data Transmission and Reception

Digi XBee ZB	EM260 Data Transmission and Reception
XBee API Frames Data Transmission frame (0x10 or 0x11) Transmit Status frame (0x8B) Data Receive Indicator (0x90 or 0x91) Create Source Route (0x21) Source Route Indicator (0xA1)	EM260 Functions sendUnicast() sendBroadcast() sendReply() messageSentHandler() incomingSenderEui64Handler() incomingMessageHandler() setSourceRoute() incomingRouteRecordHandler()
XBee Command Registers AR – Set many-to-one route request time See Note 1 See Note 2	EM260 Functions sendManyToOneRouteRequest() incomingManyToOneRouteRequestHandler() incomingRouteErrorHandler() pollForData() pollCompleteHandler() pollHandler() addressTableEntryIsActive() setAddressTableRemoteEui64() setAddressTableRemoteNodeId() getAddressTableRemoteEui64() getAddressTableRemoteNodeId() replaceAddressTableEntry() lookupNodeByEui64() lookupNodeByNodeId()

Note 1 – XBee firmware manages incoming many-to-one route requests, route error messages and end device polling.

Note 2 – XBee firmware helps manage the address table. It provides indication of the current 16-bit address for a remote in the Transmit Status frame. The Data Transmission frames allow a 64-bit and optional 16-bit address to be specified in each transmission.

Appendix A – XBee ZB API Frames

Get/Set Command Registers

The XBee module supports various command registers for configuring ZigBee networking, addressing, sleep, and IO configuration settings. Some examples of the configuration registers are listed in table 1 below. See the XBee product manual for a detailed listing of commands.

AT Register	Description	Range
ID	Set/Read the 64-bit PAN ID. Set to 0 to join any PAN ID	0 – 0xFFFFFFFF FFFFFFFF
SC	Set/Read the scan channels mask used to form or join a network	0 – 0xFFFF
ZS	Set/Read the ZigBee stack profile	0 – 2
NJ	Permit joining time (seconds). Set to 0xFF to enable joining always	0 – 0xFF
AR	Set/Read the many to one broadcast interval rate (ten second units)	0 – 0xFF
SM	Set/Read the sleep mode setting on an end device	1 – pin sleep 4 – cyclic sleep
NI	Set/Read an ASCII identifier string for the device	String containing up to 20 ASCII characters
EE	Enable or disable security on the module	
EO	Set/Read the security options (bitmask) value	0 – 0xFF
D0	Set the AD0/DIO0 module IO pin to configure the pin with A/D or DIO functionality	2 – A/D 3 – Digital Input 4 – Digital Output, low 5 – Digital Output, high
BD, NB	Set/Read the module serial baud rate/parity settings	

These register settings can be read or set on the local XBee device or on any XBee device in the network.

The local command API frame used to set/read local configuration is formatted as follows:

Field	Description
Start Delimiter	0x7E
Length	2 bytes specifying the frame length
Frame Type (0x08)	API frame type for command register set/read operations
App Sequence Number	Sequence number set for each command frame
Cmd Register Name	2-byte value of the command register (i.e., 'NI' for the NI register)
Command Data	Optional data when setting a command register value
Checksum	1-byte checksum used to verify integrity of the API frame

The remote command API frame used to set/read configuration registers on remote XBee devices is similar, but adds the destination address field before the command register name. The API frame type for remotes commands is 0x17.

The XBee sends a command response frame in response to a command request, and a remote command response frame in response to a remote command request.

Data Transmission

The Data transmission API provides great flexibility for sending data to remotes. Applications that only care to get data from one radio to another can send a transmission by simply specifying the destination address. Advanced applications, such as those that operate on a public profile, can include endpoint, cluster ID, and profile ID information in the data transmissions if required.

The simple data transmission API frame is formatted as follows:

Field	Description
Start Delimiter	0x7E
Length	2 bytes specifying the frame length
Frame Type (0x10)	API frame type for simple data transmission
App Sequence Number	Sequence number set for each command frame
Destination Address	Destination address of the remote device, or the broadcast address for broadcast transmissions
Broadcast Radius	1-byte value specifying the broadcast radius for this transmission (applies to broadcasts only).
Transmit Options	1-byte bitmask to specify any transmission options
Data	Data payload to include in the transmission
Checksum	1-byte checksum used to verify integrity of the API frame

Applications that need to support multiple endpoints, cluster IDs, and/or profile IDs can use the explicit data transmission API frame. This frame is formatted similarly to the simple data transmission frame, but it includes source and destination endpoint fields, a 2-byte cluster ID field, and a profile ID field after the destination address. The explicit frame can even be used to send ZDO commands to any radio in the network. The API frame type for the explicit data transmission frame is 0x11.

Transmit Status

After a data transmission is sent, an API transmit status frame is sent, a transmit status message is received. This status message indicates the success or failure status of the transmission.

Data Reception

The receive data indicator API frames include received RF data and addressing information about the device that sent the data. Similar to the transmission API frames, the receive API frames provide great flexibility in how much ZigBee addressing information is revealed. Advanced applications, such as those that operate on a public profile, can expose ZigBee endpoint, cluster ID and profile ID information. Simple applications that do not need this level of detail can omit these fields.

The simple receive data indicator frame is constructed as follows:

Field	Description
Start Delimiter	0x7E
Length	2 bytes specifying the frame length
Frame Type (0x90)	API frame type for simple receive data indicator frame
Source Address	The address of the remote device that sent the data transmission. This includes both the 64-bit and 16-bit addresses if included in the transmission.
Rx Options	1-byte bitmask of receive data options
Data	The RF data received by the radio
Checksum	1-byte checksum used to verify integrity of the API frame

The explicit receive data indicator frame is similar to this frame, but includes the source and destination endpoint, cluster ID, and profile ID values of the received data after the source address. The API frame type for the explicit receive data indicator frame is 0x91.

Modem Status

The modem status frame provides an indication of radio state changes. The status values include the following indications:

- Hardware reset
- Watchdog reset
- Joined network
- Started network (coordinator)

Appendix B – Abbreviated EM260 API

EM260 configuration values are set using the configuration frames “Get Configuration Value” and “Set Configuration Value,” similar to the Digi command registers model. These frames can be used to set/read stack parameters such as:

- Security level
- End device poll timeout
- Tx power mode
- APS Ack timeout (unicast timeout)

The networking API frames include the following commands:

Command Request	Request Parameters	Response Parameters
Network Init	-	1
Start Scan	3	1
Join Network	2	1
Set Radio Power	1	1
Get Network Parameters	-	3

Received Status Message	Response Parameters
Stack Status Handler	1
Network Found	3

The messaging API frames include:

Command Request	Request Parameters	Response Parameters
Send Unicast	6	2
Send Broadcast	6	2
Poll for Data	3	1
Address Table Entry Is Active	1	1
Set Address Table Remote Address	2	1
Get Address Table Remote Address	1	1

Received Status Message	Response Parameters
Message Sent Handler	7
Incoming Message Handler	9
Poll Handler	1
ID Conflict Handler	1

WHEN
RELIABILITY
MATTERS™

Digi International

11001 Bren Road E.
Minnetonka, MN 55343
U.S.A.
PH: 877-912-3444
952-912-3444
FX: 952-912-4952
email: info@digi.com

Digi International France

31 rue des Poissonniers
92200 Neuilly sur Seine
PH: +33-1-55-61-98-98
FX: +33-1-55-61-98-99
www.digi.fr

Digi International KK

NES Building South 8F
22-14 Sakuragaoka-cho,
Shibuya-ku
Tokyo 150-0031, Japan
PH: +81-3-5428-0261
FX: +81-3-5428-0262
www.digi-intl.co.jp

Digi International (HK) Limited

Unit 3206-08A, 32/F,
AIA Tower
183 Electric Road,
North Point, Hong Kong
PH: +852-2833-1008
FX: +852-2572-9989
www.digi.cn

Digi International, the leader in device networking for business, develops reliable products and technologies to connect and securely manage local or remote electronic devices over the network or via the web. With over 20 million ports shipped worldwide since 1985, Digi offers the highest levels of performance, flexibility and quality.

www.digiembedded.com
email: info@digiembedded.com

© 2008 Digi International Inc.

Digi International Inc. Digi, Digi International, the Digi logo, the When Reliability Matters logo and XBee are trademarks or registered trademarks of Digi International Inc. in the United States and other countries worldwide. All other trademarks are the property of their respective owners.

91001488
A1/1008

