Application Note: JN-AN-1110a

JenNet-IP Border-Router Node

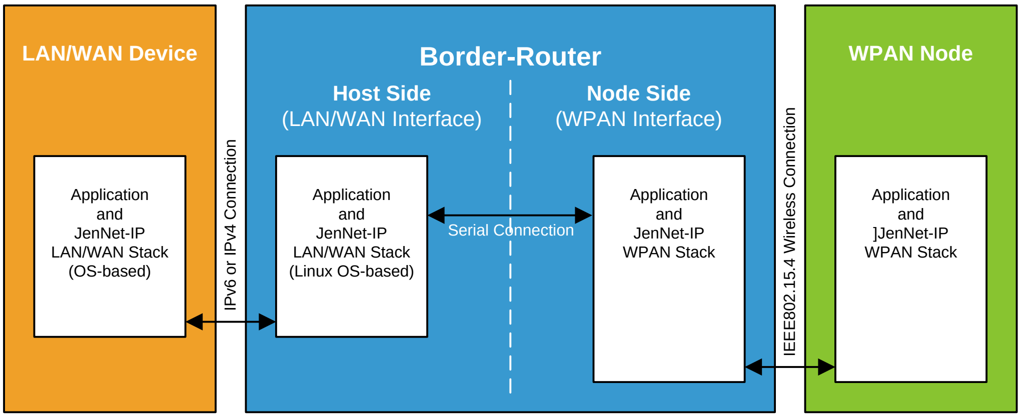
This Application Note describes the hardware and software components that are required to implement the Node side of a JenNet-IP Border-Router (which connects to the Host side, described in an accompanying document).

The Border-Router Node implements the following functionality:

- **IEEE802.15.4 WPAN Co-ordinator**
- **IEEE802.15.4 WPAN Router (Local Co-ordinator)**
- **IPv6 packet routing between WPAN and Border-Router Host**
- **IEEE802.15.4 security using JenNet-IP security model**
- **RADIUS client using PAP for joining node authentication**
- **Logging to Border-Router Host syslog**
- **Antenna diversity**
- **High/Low-power radio front-ends**

1 Application Overview

The JenNet-IP Border-Router allows the connection of one or more IEEE802.15.4 WPANs (Wireless Personal Area Networks) to the IPv6 Internet where they can exchange data with any other IPv6-connected device, enabling the “Internet of Things”. This document describes the software on the JN51xx device, which forms the “Border-Router Node”. It must be interfaced with a “Border-Router Host” in order to form a complete Border-Router.


LAN/WAN Device
Application and JenNet-IP LAN/WAN Stack (OS-based)
IPv6 or IPv4 Connection
Border-Router
Host Side (LAN/WAN Interface)
Node Side (WPAN Interface)
Application and JenNet-IP LAN/WAN Stack (Linux OS-based)
Serial Connection
Application and JenNet-IP WPAN Stack
IEEE802.15.4 Wireless Connection
WPAN Node
Application and]JenNet-IP WPAN Stack

2 Hardware

The required hardware to create the system described in this Application Note is:

- JN5168 wireless microcontroller to function as an IEEE802.15.4 WPAN interface (Border-Router Node)
- Any hardware platform (PC, Embedded device) capable of booting a Linux operating system (Border-Router Host)

3 Prerequisites

The following software must be installed to allow development of the Border Router Node.

- JN-SW-4141 Beyond Studio for NXP
- JN-SW-4165 JenNet-IP SDK (Beyond Studio)

4 Building the Border-Router Node Application

The Border-Router Node application source file should be copied into the **Eclipse workspace** directory within Beyond Studio for NXP. This Application Note includes Eclipse projects that can be opened in the Beyond Studio for NXP IDE supplied in the JN516x SDK Toolchain (JN-SW-4141). There is one build target defined, which compiles the application for the JN5168 device. Alternatively, there is a build script at the following location:

<workspace>/JN-AN-1110-JenNet-IP-Border-Router/Node/Build/Build.sh

This script can be run from the mingw command prompt, from within the **Build** directory. This script compiles the application using several baud rates defined in the script.

The application makefile is at the following location:

<workspace>/JN-AN-1110-JenNet-IP-Border-Router/Node/Build/Makefile

It has many configuration options to customise the functionality of the Border-Router. The function of each of these options is described in the makefile.

5 Border-Router Node Functions

5.1 Serial Link

The Border-Router Host uses a UART-based serial connection to communicate with the Border-Router Node. This serial link may be configured to run at different baud rates by changing the HOST_BAUD_RATE variable in the Border-Router Node makefile. The Host baud rate should match this baud rate, by starting **6LoWPANd** with the `–baudrate=<baudrate>` option. The serial link protocol is described in Appendix 1.

5.2 JenNet-IP Network Configuration

The Border-Router Node is not pre-configured with any network settings. These are provided at run-time by the Host via the serial link. One basic configuration packet specifies IEEE802.15.4, JenNet and IP settings, including channel number, PAN ID and IPv6 prefix. A second, optional packet may be sent to configure security. This contains the network encryption key, authentication method and any information required for the selected authentication method. Once these configuration packets have been sent to the Node, the Host specifies a run method, which may be one of the following:

- **Run as PAN/JenNet-IP Co-ordinator**: This is the default mode of operation where the Border-Router Node acts as both the WPAN Co-ordinator and IPv6 gateway. If the network is running with security enabled, the device will attempt to authenticate joining nodes.
- **Run as a JenNet-IP Router**: The device will attempt to join a Co-ordinator using the configuration information passed from the Host. Note that in this mode, while IPv6 packets from the Host may be routed to devices in the WPAN, their replies will be routed to the Co-ordinator of the network, not back to this device or the Host.
- **Run in commissioning mode**: If the network has security enabled, the device will attempt to authenticate itself to the Co-ordinator. If this is successful, the Co-ordinator will share the network key with the device, which will then pass the security information on to the Host. This is used to allow a Border-Router to ‘learn’ network settings from a remote control unit or from another Border-Router.

5.3 IPv6 Packet Routing

The WPAN of a JenNet-IP network is within a /64 sized IPv6 address space. Within the WPAN, any packet destined to an address in this IPv6 prefix will be routed using JenNet to the destination node. If it is destined to an address outside of the /64 then it is routed to the Co-ordinator. When these packets arrive at the Co-ordinator, they may be received by the application using the function **i6LP_IpRecvFrom()**. The Border-Router Node application polls this function for any incoming packets. When these are received, they are forwarded to the Border-Router Host via the serial link. The Host can then route them to the destination IPv6 address.

Outside of the WPAN, the Border-Router Host has a route to the /64 IPv6 prefix of the WPAN, using **6LoWPANd**, via the serial link to the Border-Router Node. When IPv6 packets are received from the Host, the Border-Router node uses the **i6LP_IpSendTo()** function to pass the packet into the network stack, which then uses JenNet to route the packet to its destination.

5.4 JenNet-IP Security

If the Host enables JenNet-IP security, all traffic at the IEEE802.15.4 MAC layer will be encrypted using the 128-bit network encryption key specified in the security packet sent from the Host over the serial link. If new devices are to be commissioned into the network, an authentication method must also be specified in the security configuration packet. The following sub-sections describe the authentication methods available.

5.4.1 No Authentication

If no authentication method is configured then only devices with prior knowledge of the network key in use can join the network. Any nodes attempting to commission will be ignored. This mode is useful to create closed system of nodes, where all on-air traffic is encrypted using the network key. The network key must be provided to nodes that are permitted to join the network via an out-of-band method, such as NFC, or that use a pre-shared key in the application binary.

5.4.2 Authentication using RADIUS with PAP

If this method is configured then commissioning nodes will be authenticated by sending a RADIUS Access-Request message to the IPv6 address of a configured RADIUS server. If the server responds with an Access-Accept message then the Co-ordinator will broadcast (or unicast if supported) the commissioning key of the new node (received as the Access-Accept payload) through the network. The next time the node attempts to join, its packets can be decrypted using the commissioning key and the new parent will send the node the network key in use, encrypted using the commissioning key. In this way the network key is never transmitted in the clear.

The only parameter necessary for the RADIUS with PAP authentication is the IPv6 address of a RADIUS server. The Border-Router Node will send RADIUS Access-Request messages to this IPv6 address using the socket mechanism built into JenNet-IP. Any incoming packets on this socket are handled by the RADIUS client. Access-Accept messages cause the Border-Router Node to commission the joining device by calling **eApi_CommissionNode()**. This function is passed the MAC address of the node that is attempting to join as well as its commissioning key. The commissioning key is returned from the RADIUS server in the Access-Accept message payload. It is carried as a vendor-specific field. The vendor used is NXP Semiconductors (0x006DE9) and the `vendor-specific` attribute is 100 (0x64).

5.5 Logging to Border-Router Host Syslog

The Border-Router Node can be configured via the makefile to log either to a UART (by setting UART_DEBUG=1 in the application makefile) or to the Host’s syslog via the serial link. The default is to log to the Host. Messages from the Border-Router Node will appear in the syslog from **6LoWPANd**, prepended with the string “**module:** “.

5.6 Antenna Diversity

The Border-Router Node supports antenna diversity for improved receive sensitivity. It is enabled by sending an Antenna Diversity packet to the Border-Router Node via the serial link. Once enabled, it cannot be disabled again until the Border-Router Node is reset.

5.7 Radio Front-End Selection

The Border-Router Node supports multiple radio front-end types. These are selected by the Border-Router Host via the serial link. The supported radio front-ends are:

- Standard-power
- High-power
- ETSI mode (high-power but with limited transmit power for ETSI compliance)

Appendix: Serial Protocol

The Host and Node sides of the Border-Router communicate using a simple serial protocol at 1 Mbaud. This protocol allows IP messages and configuration information to be exchanged between the two devices.

The protocol reserves byte values less than 0x10 for use as special characters (for example, Start and End characters). Therefore, to allow data which contains these reserved values to be sent, a procedure known as 'byte stuffing' is used. This involves identifying a byte to be sent which falls into the reserved character range, then sending an Escape character (0x02) first, followed by the data byte XOR'd with 0x10.

For example, if a non-special character with the value of 0x05 is to be sent:

1. Send the Escape byte (0x02)
2. XOR the byte to be sent with 0x10 (0x05 ^ 0x10 = 0x15)
3. Send the modified byte

Message Fields

A message consists of the following fields:

- Start character (special character)
- Message type (byte stuffed)
- Message length (byte stuffed)
- Checksum (byte stuffed)
- Message data (byte stuffed)
- End character (special character)

Start Character

The Start character is a single-byte special character with the value 0x01 which is sent as the first byte of any message to allow the receiving end to synchronise. Since this is considered a special character, it will be sent without modification.

Message Type

The message type is a single-byte identifying the nature of the data contained in the message payload. The values implemented are as follows:

Message Type Value	Meaning
0x00	Version Request
0x01	Version
0x65	IPv6 Packet
0x66	Configuration Data
0x67	Run (Co-ordinator Mode)
0x68	Reset
0x69	IPv6 Address
0x70	Configuration Request
0x71	Security Configuration
0x72	Log
0x73	Ping
0x74	Profile
0x75	Run (Router)
0x76	Run (Commission to remote)
0x77	Enable Activity LED
0x78	Set Radio Frontend
0x79	Enable Antenna Diversity

Table 1: Message Types

Message Length

The message length is a 16-bit value equal to the number of bytes in the payload section of the message, sent most-significant-byte first.

Checksum

The checksum is an 8-bit value calculated by XORing the following (starting with a checksum of 0x00):

- Message type byte
- Message length, most significant byte
- Message length, least significant byte
- Data bytes

Message Data

The message data is a number of bytes equal in length to the value in the message length field. The number of bytes transmitted via the UART may be higher due to the presence of Escape bytes sent to identify values that fall into the reserved range. All multi-byte binary data is sent in network byte-order (big-endian).

The names used for data types throughout this document are as follows

Name	Type
Uint8_t	Unsigned 8-bit integer (one byte)
Uint16_t	Unsigned 16-bit integer (two bytes)
Uint32_t	Unsigned 32-bit integer (four bytes)
Uint128_t	Unsigned 128-bit integer (sixteen bytes)

Table 2: Data Types

End Character

The End character is a single-byte special character with the value 0x03 which is sent as the last byte of any message to allow the receiving end to synchronise. Since this is considered a special character, it will be sent without modification.

Message Descriptions

The message types listed above in Table 1 are described below.

Version Request [0x00]

This message has type 0x00. It has no payload. It is sent to allow the host processor to determine the version of (and hence features implemented by) the Border-Router Node.

Version [0x01]

This message has type 0x01. Its payload is the firmware version information of the source device. When sent from the Border-Router Node to the host processor, the message payload is as follows:

Data type	Name	Description
Uint8_t	Major Version Number	The major number of the firmware
Uint8_t	Minor Version Number	The minor number of the firmware
Uint8_t	Revision Number	The revision number of the firmware

Table 3: Payload of 'Version' Message

This version information allows the host processor to determine the supported features of the Border-Router Node.

IPv6 Packet [0x65]

This message has type 0x65. Its payload is a raw IPv6 packet. The message is used to pass IPv6 packets from the IEEE802.15.4 wireless network to the host processor, where they can be routed out to the Internet, if desired. It also allows IPv6 packets that originated either on the host processor or on the Internet to be routed into the IPv6 network of the IEEE802.15.4 network, if the wireless network is configured with a globally routable IPv6 prefix.

Configuration Data [0x66]

This message has type 0x66.

After the Border-Router Node has been reset, a Configuration Data packet must be sent to it in order to set the parameters of the wireless network that will be established.

The message payload is binary configuration data for the IEEE802.15.4 wireless network. When sent from the host processor to the Border-Router Node, the payload determines the network settings that the node will use to start or join a network.

This message is also sent from the Border-Router Node to the host processor to inform it of the configuration of the network that has actually been started. For example, if the Border-Router Node is initially sent a Configuration Data packet specifying channel 0, this will cause the node to perform an energy scan and choose the quietest channel on which to start a network. The Configuration Data packet sent back to the host processor will then contain the actual channel on which the network was formed.

The configuration parameters are as follows:

Data type	Name	Description
Uint8_t	Region	This allows the operating region of the Border-Router Node to be set, such that the radio complies with regional regulations. The following regions are defined: 0 = Europe 1 = USA 2 = Japan
Uint8_t	Channel	This allows the IEEE802.15.4 operating channel to be set. The channel can be set to any number in the range 11-26. A special value of 0 can be used to instruct the node to automatically select the quietest channel.
Uint16_t	PAN ID	This allows the PAN ID of the wireless network to be set. This should be unique for each wireless network to be established. A special value of 0xFFFF can be used to instruct the node to automatically choose a PAN ID that is not already in use within radio range.
Uint32_t	JenNet ID	This allows the JenNet 32-bit Network Application ID to be set. Nodes will only attempt to join a network with matching JenNet ID.
Uint32_t	IPv6 Prefix MSW	Most significant word of the IPv6 prefix.
Uint32_t	IPv6 Prefix LSW	Least significant word of the IPv6 prefix.

Table 4: Payload of 'Configuration Data' Message

Run (Co-ordinator Mode) [0x67]

This message has type 0x67. It has no payload. After the Border-Router Node has been configured, sending the Run (Co-ordinator Mode) message causes it to start an IEEE802.15.4 network with itself as the PAN Co-ordinator. After the network has been established, a Configuration Data message is sent from the Border-Router Node to the host processor, containing details of the network that has been established. IPV6 packets can then be exchanged.

Reset [0x68]

This message has type 0x68. It has no payload. It is used to cause a software reset of the Border-Router Node.

IPv6 Address [0x69]

This message has type 0x69. It has no payload when sent from the host processor to the Border-Router Node. It is used to query the IPv6 address of the Border-Router Node. It should only be sent after using the Configuration Data message to set the IPv6 prefix. In response, the Border-Router Node will send back a message of the same type with a payload containing the full IPv6 address of the node.

Configuration Request [0x6A]

This message has type 0x6A. It has no payload. It is sent by the Border-Router node at reset to request configuration data from the host processor. The host processor should respond with a Configuration Data message, and optionally a Security Configuration message, and then a Run message to start the network.

Security Configuration [0x6B]

This message has type 0x6B. If a secure network is to be established, this message must be sent from the host processor to the Border-Router Node after the Configuration Data message and before the Run message. If it is not sent, the IEEE802.15.4 network will be unsecured. The configuration parameters are as follows:

Data type	Name	Description
Uint128_t	Security Key	This is the 128-bit AES encryption key that will be used to secure the network.
Uint8_t	Authorisation Scheme	This setting configures the authorisation scheme that will be used to authenticate nodes that do not have the network key and allow them access to the network. Valid values are: 0 = None 1 = RADIUS PAP Authentication
	Authorisation Scheme Data	This is a placeholder in which authorisation scheme-specific data is sent.

Table 5: Payload of ‘Security Configuration’ Message

Each supported authorisation scheme is described below.

None

Nodes that do not have the network key cannot be authorised and therefore cannot join the network. In this case, there is no additional authorisation scheme data passed in the Security Configuration message.

RADIUS PAP Authentication

The Border-Router Node includes a RADIUS client implementation that supports PAP (Password Authentication Protocol). When the Border-Router Node is configured to use RADIUS with PAP authentication, it is also given the IPv6 address of the RADIUS server that it should contact for node authorisation. This may be (but does not have to be) resident on the host processor.

In this case, the authorisation scheme configuration data passed in the Security Configuration message is as follows:

Data type	Name	Description
Uint128_t	RADIUS Server IPv6 Address	This is the IPv6 address of the RADIUS server for authentication.

Table 6: RADIUS PAP Authentication Scheme Data

The configured RADIUS server is responsible for determining whether a node is allowed access to the network.

Log [0x6C]

This message has type 0x6C. It is used to allow the Border-Router Node to send log messages to the host processor. As the Border-Router Node operates, it logs events, including any processor exceptions. These are passed to the host processor for logging. The packet data is as follows.

Data type	Name	Description
Uint8_t	Log Level	This is the severity level of the log message. Values are equivalent to the Unix syslog values, i.e. 0 Emergency: system is unusable 1 Alert: action must be taken immediately 2 Critical: critical conditions 3 Error: error conditions 4 Warning: warning conditions 5 Notice: normal but significant condition 6 Informational: informational messages 7 Debug: debug-level messages
String	Message	The ASCII string of the log message. The string should be NULL terminated, but this should not be relied upon. The length of the message can be determined from the length field in the header.

Table 7: Payload of ‘Log’ Message

Ping [0x6D]

This message has type 0x6D. It has no payload. This message can be sent regularly by the host processor to determine whether the Border-Router Node is still attached to the system. The Border-Router Node will respond to a ping message with another ping message.

Profile [0x6E]

This message has type 0x6E. It is used to configure the JenNet profile that should be used by the Border-Router Node. The network profile can be used to configure properties such as the ping rate and the number of missed pings allowed.

Data type	Name	Description
Uint8_t	Profile	This is the identifier of the JenNet network profile to use. For details of the profiles, refer to the *JenNet-IP WPAN Stack User Guide (JN-UG-3080)*.

Table 8: Payload of ‘Profile’ Message

Run (Router Mode) [0x6F]

This message has type 0x6F. It has no payload. Once the Border-Router Node has been configured, sending the Run (Router Mode) message to it causes it to attempt to join an existing IEEE802.15.4 network using the configuration supplied in the Configuration Data message. After the network has been established, a Configuration Data message is sent from the Border-Router Node to the host processor containing details of the network that has been joined.

Run (Commission to Remote Mode) [0x70]

This message has type 0x70. It has no payload. Once the Border-Router Node has been configured, sending the Run (Commission to Remote Mode) message to it causes it to attempt to join an existing IEEE802.15.4 network that has a remote control unit as its Co-ordinator using the configuration supplied in the Configuration Data message. The Border-Router Node will attempt to securely join the remote control unit using its commissioning key. After the network has been established, a Configuration Data message is sent from the Border-Router Node to the host processor containing details of the network that has been joined. A Security Configuration message is also sent giving the security configuration of the network that has been joined.

This process allows a host processor to learn the details of a standalone network that has been started using a remote control unit, then start as a Co-ordinator for that network.

Enable Activity LED [0x71]

This message has type 0x71. It is used to configure a DIO pin of the JN51xx device as an 'activity LED' on the Border-Router Node. The activity LED will subsequently flash a 'heartbeat' during network activity. The message has a single-byte payload containing the DIO number of the pin that should be set up as the activity LED. The Border-Router Node will determine whether the requested DIO is already in use for another function and, if so, will not allow its selection. Otherwise, the DIO will be configured and activated.

Data type	Name	Description
Uint8_t	DIO number	This is the number of the DIO that should be set up as an 'activity LED'.

Table 9: Payload of 'Enable Activity LED' Message

Set Radio Frontend [0x72]

This message has type 0x72. It is used to specify the type of radio front-end that is fitted on the Border-Router Node and the mode in which it should operate. The message payload is as follows:

Data type	Name	Description
Uint8_t	Frontend type/mode	Radio front-end to set up. Values are as follows: 0 Standard-power module 1 High-power module 2 High-power module (ETSI mode)

Enable Antenna Diversity [0x73]

This message has type 0x73. It has no payload. Upon receiving this message, the Border-Router Node will enable antenna diversity mode in the IEEE802.15.4 MAC layer of the stack.

IPv6 Multicast

The Border-Router Node sends regular MLDv2 packets to the host processor (IPv6 address ff02::2, all routers link local address) indicating the multicast groups that exist within the wireless network. These packets can be used to set up IPv6 multicast routing on the host processor, allowing IPv6 multicast packets to be routed into the wireless network.

Starting a Network

To start a wireless network using the Border-Router Node, the following sequence of messages should be exchanged:

Host Processor		**Border-Router Node**
Get Version	→	
	←	Firmware version
Configuration Data (network settings)	→	
	←	Configuration Data
Security Settings (optional)	→	
Run	→	
	←	Configuration Data Security Settings (optional)
	←	Log (Event-dependent)
Ping	↔	Ping
IPv6	↔	IPv6

Following a reset of the Border-Router Node, it sends a Configuration Request message. This allows the host processor to be aware of the need to re-configure the node, as before. The following sequence of messages is exchanged:

Host Processor		**Border-Router Node**
	←	Configuration Request
Configuration Data (network settings)	→	
		Configuration Data
Security Settings (optional)	→	
Run	→	
	←	Configuration Data Security Settings (optional)
	←	Log (Events Dependant)
Ping	↔	Ping
IPv6	↔	IPv6

Note: At start-up, if the host processor misses the initial Configuration Request from the Border-Router Node, it can assume that the node is awaiting its configuration data and it should send the data immediately.

Example Packets

The following sections show example packets to be exchanged with the Border-Router Node.

Configuration Data

The following is an example serial byte stream sent to the Border-Router Node to configure it.

Serial Data stream:

```
0x01 0x66 0x02 0x10 0x10 0x7b 0x02 0x10 0x02 0x1b 0x3a 0x77 0x11 0x10
0x11 0x10 0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x12
0x03
```

Unescaped serial data stream:

```
0x01 0x66 0x00 0x10 0x7b 0x00 0x0b 0x3a 0x77 0x11 0x10 0x11 0x10 0xfd
0x04 0x0b 0xd3 0x80 0xe8 0x00 0x02 0x03
```

Serial Data	Payload	Description
0x01 0x66		Start character, message type = 0x66
0x02 0x10 0x10	0x00 0x10	Length=16
0x7b	0x7b	Message checksum = 0x7b
0x02 0x10	0x00	Region = 0
0x02 0x1b	0x0b	Channel = 11
0x3a 0x77	0x3a 0x77	PAN ID = 0x3a77
0x11 0x10 0x11 0x10	0x11 0x10 0x11 0x10	JenNet ID = 0x11101110
0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x12	0xfd 0x04 0x0b 0xd3 0x80 0xe8 0x00 0x02	IPv6 prefix = 0xfd040bd380e80002
0x03		End character

IPv6

The following is an example serial byte stream from a ping (ICMP Echo Request) packet sent to a Border-Router Node with IPv6 address fd04:bd3:80e8:2:200:0:400:dec from a Linux PC with IPv6 address fd04:bd3:80e8:1::2.

Serial data stream:

```
0x01 0x65 0x02 0x10 0x68 0x58 0x60 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10
0x40 0x3a 0x40 0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x11
0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02
0x12 0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x12 0x02 0x12
0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x14 0x02 0x10 0x02 0x1d 0xec 0x80 0x02
0x10 0x37 0x80 0x28 0xdd 0x02 0x10 0x02 0x11 0x84 0x64 0x67 0x4e 0x1a 0xff
0x02 0x16 0x02 0x10 0x02 0x18 0x02 0x19 0x02 0x1a 0x02 0x1b 0x02 0x1c 0x02
0x1d 0x02 0x1e 0x02 0x1f 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19
0x1a 0x1b 0x1c 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28
0x29 0x2a 0x2b 0x2c 0x2d 0x2e 0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37
0x03
```

Unescaped serial data stream:

```
0x01 0x65 0x00 0x68 0x58 0x60 0x00 0x00 0x00 0x00 0x40 0x3a 0x40 0xfd 0x04
0x0b 0xd3 0x80 0xe8 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0xfd
0x04 0x0b 0xd3 0x80 0xe8 0x00 0x02 0x02 0x00 0x00 0x00 0x04 0x00 0x0d 0xec
0x80 0x00 0x37 0x80 0x28 0xdd 0x00 0x01 0x84 0x64 0x67 0x4e 0x1a 0xff 0x06
0x00 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14 0x15
0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x23 0x24
0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e 0x2f 0x30 0x31 0x32 0x33
0x34 0x35 0x36 0x37 0x03
```

Serial Data	IPv6 Packet	Payload
0x01 0x65		Start character, message type = 0x65
0x02 0x10 0x68		Length=104
0x58		Message checksum = 0x58
0x60	0x60	IP version = 6
0x02 0x10 0x02 0x10 0x02 0x10	0x00 0x00 0x00	Traffic class = 0
0x02 0x10 0x40	0x00 0x40	IP payload length = 64
0x3a	0x3a	Next header = ICMPv6
0x40	0x40	Hop limit = 64
0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x11 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x12	0xfd 0x04 0x0b 0xd3 0x80 0xe8 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02	Source IPv6 address = fd04:bd3:80e8:1::2
0xfd 0x02 0x14 0x02 0x1b 0xd3 0x80 0xe8 0x02 0x10 0x02 0x12 0x02 0x12 0x02 0x10 0x02 0x10 0x02 0x10 0x02 0x14 0x02 0x10 0x02 0x1d 0xec	0xfd 0x04 0x0b 0xd3 0x80 0xe8 0x00 0x02 0x02 0x00 0x00 0x00 0x04 0x00 0x0d 0xec	Destination IPv6 address = fd04:bd3:80e8:2:200:0:400:dec
0x80	0x80	ICMPv6 Type = 128, Echo request
0x02 0x10	0x00	ICMPv6 Code = 0
0x37 0x80	0x37 0x80	Checksum = 0x3780
0x28 0xdd	0x28 0xdd	ICMPv6 ID: 0x28dd
0x02 0x10 0x02 0x11	0x00 0x01	Sequence: 1
0x84 0x64 0x67 0x4e 0x1a 0xff 0x02 0x16 0x02 0x10 0x02 0x18 0x02 0x19 0x02 0x1a 0x02 0x1b 0x02 0x1c 0x02 0x1d 0x02 0x1e 0x02 0x1f 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e 0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37	0x84 0x64 0x67 0x4e 0x1a 0xff 0x06 0x00 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e 0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37	ICMPv6 Echo request payload data

Revision History

Version	Notes
1.0	First release
1.1	Formatting and minor changes made.
1.2	Updated for Beyond Studio for NXP

Important Notice

NXP Semiconductors

For the contact details of your local NXP office or distributor, refer to:

www.nxp.com