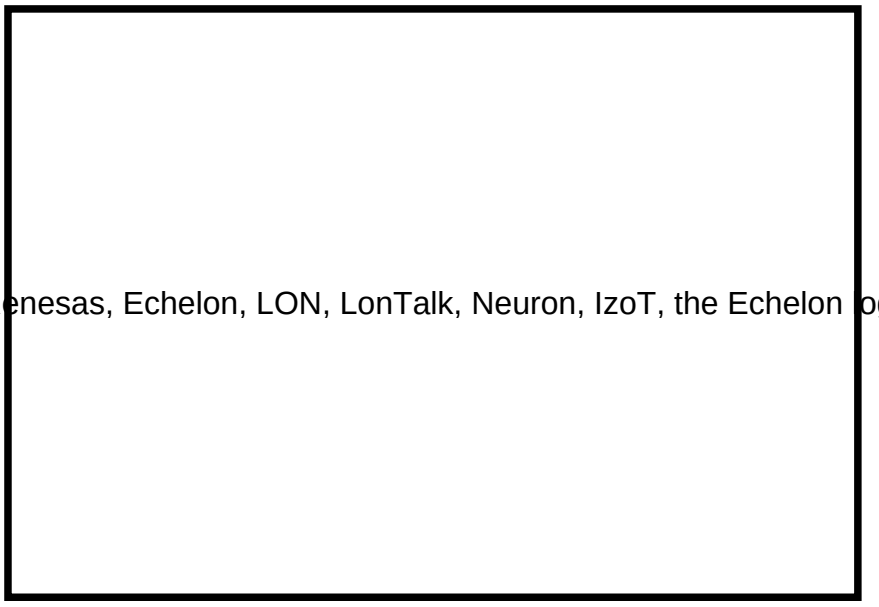
RENESAS
BIG IDEAS FOR EVERY SPACE

IzoT™ U60 FT Network Interface Module User's Guide

Describes the IzoT U60 FT Network Interface Module, a USB to TP/FT-10 network interface module in a compact board-level form factor. Explains the two protocol processing modes of operation, Layer 5 mode and Layer 2 mode.

005-0229-01B


Renesas, Echelon, LON, LonTalk, Neuron, IzoT, the Echelon log

Contents

The IzoT U60 Network Interface Module

The IzoT U60 FT Network Interface Module is a USB to TP/FT-10 network interface module in a compact board-level form factor.

Introduction

The IzoT U60 FT Network Interface Module is a board-level module that can be easily integrated into any controller or device. The U60 module has two protocol modes of operation—*Layer 2 mode* and *Layer 5 mode* that can be selected by the host application.

In Layer 2 mode, the U60 FT module implements Layers 1 and 2 of the ISO/IEC 14908-1 and 14908-2 protocol and is compatible with the LonTalk Device Stack EX included with the IzoT SDK Standard and Premium Editions. The LonTalk Device Stack EX provides source code for LonTalk /IP and classic LON compatible protocol stacks.

In Layer 5 mode, the U60 FT module implements Layers 1 through 5 of the ISO/IEC 14908-1 and 14908-2 protocol and is compatible with Echelon's previous generation MIP products.

On-board **Tx** and **Rx** LEDs provide an indication of FT packet transmission and reception. The **Tx** and **Rx** signals are provided on the U60 connector so that the host controller can provide transmit and receive indicators.

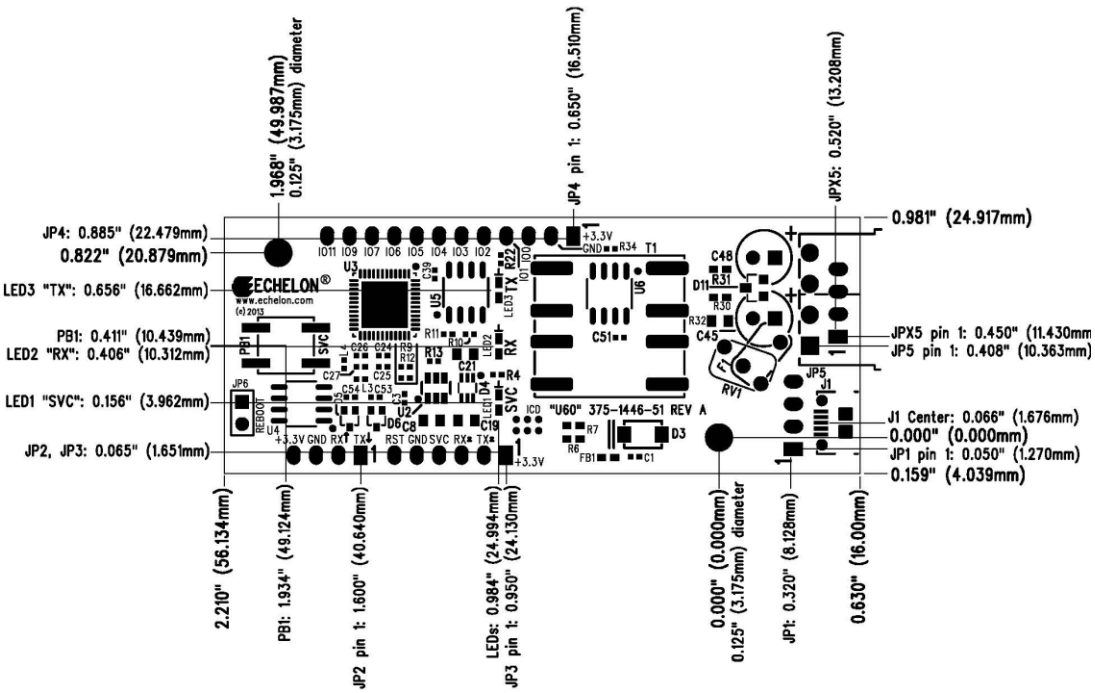
An on-board **Service** LED provides an indication of the network status of the U60 module. The **Service** signal is provided on the U60 connector so that the host controller can provide a network Service indicator.

Three pin header connectors are provided by the FT network. The USB, the power input, transmit/receive signals, and the service signal are described in this document.

Mechanical Dimensions

The dimensions for the U60 PCB are 72mm x 29mm (2.8" x 1.1"). Both the FT and USB connections are on the same edge of the PCB. Two mounting/tooling holes are provided. The form factor fits into a narrow DIN enclosure.

The following drawing indicates the PCB dimensions and mounting hole locations. Pin 1 for the USB header is indicated in the lower right corner. Pin 1 of the FT connector is in the upper right corner.



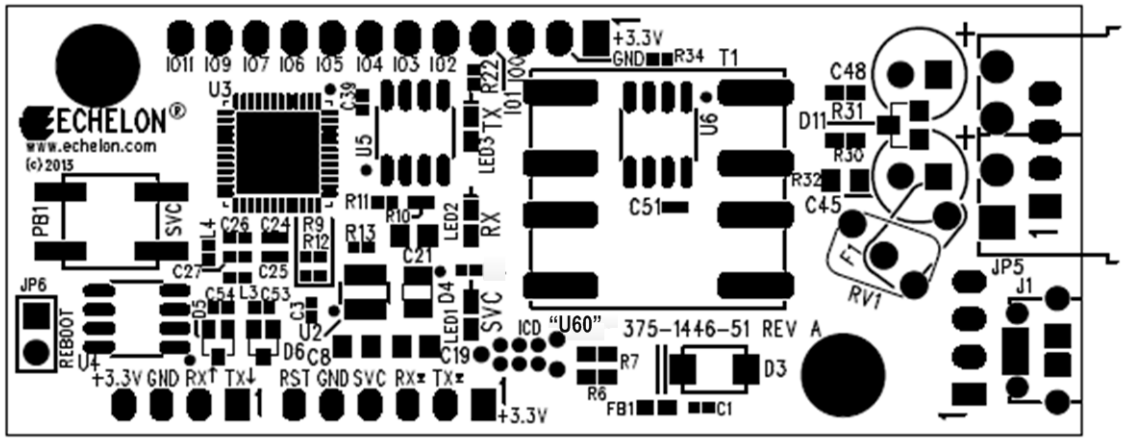
DRILL DRAWING

ALL HOLE SIZES ARE IN MILS
ARE FINISH HOLE SIZES

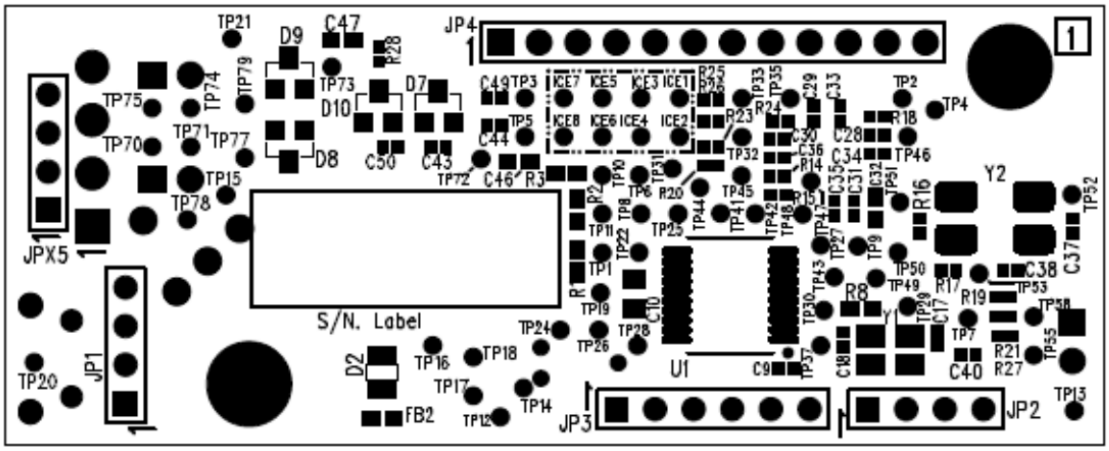
SIZE	QTY	SYM	PLATED	TOL
19.69 x 45.28	2	⊕	YES	+3/-3
21.65 x 33.46	2	⊗	YES	+3/-3
35	6	⊡	YES	+3/-3
40	38	⊠	YES	+3/-3
54	4	⋈	YES	+3/-3
125	2	⊠	NO	+3/-0

Tall components on the U60 PCB are the FT-X3 transformer (T1), the electrolytic capacitors (C45 and C48), Metal Oxide Varistor (RV1) and the FT connector (JP5). The tallest of these components is the FT-X3 transformer, with a height of 12.5mm (0.492”). Consider these components as keep out/away areas for ESD.

Areas to keep noise sensitive circuits away from are the two crystals (Y1 and Y2) for the two processors on the PCB. These locations are both around the push-button for the Service message (the push-button can be seen on the left side in the following picture). Y1 and Y2 are on the secondary side of the PCB. Y2 is located under the area of the push-button. Y1 is located near the header pins labeled **RX** and **RST**.



Primary Side



Secondary Side

Visual Indicators

Three LEDs are located on the Primary side of the PCB, centered between the two mounting holes. The **SVC** LED blinks yellow for Service indication. The **RX** LED flashes green when data is received on the FT network. The **TX** LED flashes green when data is transmitted on the FT network.

Connectors

The U60 module has three connectors—an FT Network Connector, a USB Connector, and an I/O Connector. These connectors are described in the following sections.

FT Network Connector

The FT Network Connector is a 0.1" pitch single row square pin header at **JPX5**. This header is populated on the Secondary side of the PCB.

JPX5 Pin Assignment	
Pin Number	**Signal Name**
1	FT NETWORK B
2	FT NETWORK A
3	RS-485 GND (Not Connected)
4	RS-485 SHIELD (Not Connected)

USB Connector

The USB Connector is a 0.1" pitch single row square pin header at **JP1**. This header is populated on the Secondary side of the PCB.

JP1 Pin Assignment	
Pin Number	**Signal Name**
1	+5V
2	USB -
3	USB +
4	GND

I/O Connector

The I/O Connector is a 0.1" pitch single row square pin header at **JP3**. This header is populated on the Secondary side of the PCB.

JP3 Pin Assignment	
Pin Number	**Signal Name**
1	+3.3V
2	TX LED
3	RX LED
4	Service
5	GND
6	Neuron Reset

Host Communications

The U60 communicates with a host over a USB connection using the U60 link-layer protocol (LLP).

A driver is required on the host that provides an open, close, read, and write interface to the U60. The driver manages the host end of the U60 LLP, and initializes the U60 when the driver is opened by an application. The OpenLDV driver for Windows includes a U60 driver, starting with OpenLDV 5. Source code for a Linux driver for the U60 is included with the IzoT SDK Premium Edition.

Protocol Processing Selection

The U60 has two protocol processing modes of operation, *Layer 5 mode* and *Layer 2 mode*.

- In *Layer 5 mode*, Layers 1 to 5 of the ISO/IEC 14908-1 and 14908-2 protocol stack are handled by the U60, and Layer 5 messages are exchanged between the U60 and the host using the U60 LLP. Layer 5 mode requires less code on the host than Layer 2 mode since Layer 5 mode leverages the ISO/IEC 14908-1 Layers 3 through 5 protocol stack implemented by the Neuron firmware. The U60 Layer 5 interface is compatible with the MIP/P20 and MIP/P50 interface, with the exception of the changes to the link-layer protocol (LLP) described in this section.
- In *Layer 2 mode*, Layers 1 and 2 of the ISO/IEC 14908-1 and 14908-2 protocol stack are handled by the U60, and Layer 2 messages are exchanged between the U60 and the host using the LLP. The host must interface Layers 3 through 7 when the U60 is running in Layer 2 mode. You can use Layer 2 mode with the Echelon LonTalk Device Stack EX running on the host to create high performance controllers that support more address table entries and simultaneous transactions than can be supported by the U60 with a Layer 5 interface. The LonTalk Device Stack EX includes a complete implementation of ISO/IEC 14908-1 Layers 3 through 5 that runs on a host and interfaces with a native LON interface such as the U60 module, or interfaces directly with an Ethernet or Wi-Fi interface on the host.

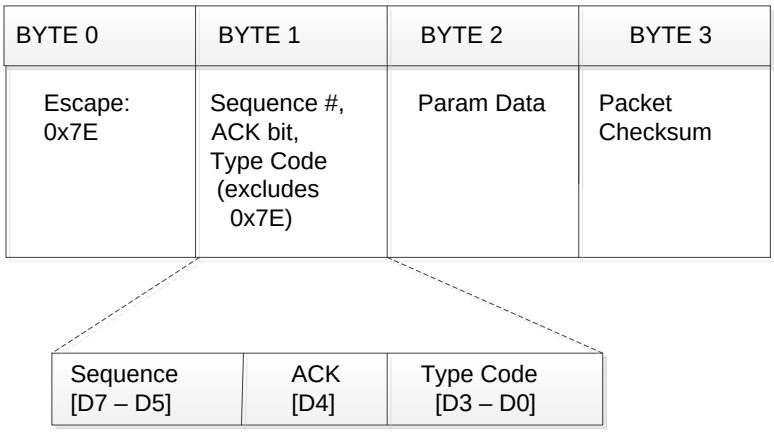
If you are using the OpenLDV driver for Windows, version 5 or newer, your application can select the protocol processing mode of operation by opening the driver with the **ldv_open_cap()** function and specifying the mode with the **nDeviceCaps** parameter. Set **nDeviceCaps** to **LDV_DEVCAP_L2** for Layer 2 mode or **LDV_DEVCAP_L5** for Layer 5 mode.

If you are using the U60 driver for Linux, you have to specify the protocol processing mode in the driver source code, and rebuild the driver for your Linux platform. The protocol processing mode is selected in the call to **U50LinkStart()** in the **u50_dev_open()** function. Specify the mode in the last parameter in the call to **U50LinkStart()**. Specify **U50_OPEN_LAYER2** for Layer 2 mode or **U50_OPEN_LAYER5** for Layer 5 mode.

Code Packet Layout

The basic component of the U60 LLP is the code packet, which starts with an 0x7E escape code. If an escape code appears in a normal data stream it is followed by

another escape code and interpreted as a single data byte value rather than the start of the code packet. The following figure illustrates the code packet layout.



The second byte contains a 3-bit sequence number in the sequence bits (D7-D5), a single ACK bit (D4), and a 4-bit Type Code bits (D3-D0). You cannot form the escape code by combining a Type Code value with the sequence number (0x0E is not allowed). As a result, you can create 15 different codes. The Type Code values are described in the next section.

The sequence number is cycled through between values '1' and '7', with '0' being an initialization value. When a code packet is received that has the same sequence number as the previous code packet received, the new packet (and any following data) is rejected (this transfer will be acknowledged if necessary). The exception is if the sequence number is zero. The zero sequence number may be used for any idempotent code packet or message.

The third and fourth bytes may contain the escape code value, but they will not be interpreted as escape codes. This may cause minor re-synchronization issues if this packet is broken; however, it ensures a constant code packet size.

The packet checksum is an 8-bit value that, when added to the first three bytes, results in an 8-bit zero result.

The code packet has the following features:

- Asynchronous method of presenting itself at any time (by using an escape sequence)
- Checksum verification
- Data transfer initiation
- Duplication detection using a sequence number—duplicates may be sent when an expected response is lost or broken and does not occur within a time-out period

Data outside of the code packet is restricted to either a *message* or a *multi-byte local network interface command or response* and is always enclosed with a length byte in the front and a checksum at the end. The length byte does not account for the checksum at the end—the inclusion of the checksum is implied. The checksum covers the data that preceded it; therefore, it does include the length byte.

All data outside a code packet must include an escape prefix before any escape data values. For example, if a 0x7E value appears in the sequence it must be followed by

another 0x7E (the first 0x7E value is the escape prefix). Checksum bytes and length bytes must be preceded with escape prefixes.

The following table summarizes the layout of the data message used in Layer 5 mode and for the **niNETMGMT** local network interface command (Layer 2 mode is required for the IzoT Device Stack EX). When sending local network interface commands (NI Commands other than **niCOMM** or **niNETMGMT**) that have additional data, the commands are contained in a data message and they are preceded by a **CpMsg** code packet.

BYTE 0	BYTE 1	BYTE 2..N	BYTE N+1
Length (bytes to follow except checksum)	NI Command	SICB starting w/ message header	Message Checksum of bytes 0-N

The following table summarizes the layout of the data message used in Layer 2 mode.

BYTE 0	BYTE 1	BYTE 2..N	BYTE N+1
Length (bytes to follow except checksum)	NI Command	Priority/AltPath/DeltaBacklog byte, NPDU, CRC (2 Bytes)	Message Checksum of bytes 0-N

The U60 supports extended messages that may be up to 1280 bytes in length. For these messages the BYTE 0 length byte will be 0xFF, followed by the MS byte of the 16-bit length, followed by the LS byte of the 16-bit length, followed by the NI command:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4..N	BYTE N+1
0xFF	MS byte of length	LS byte of length (bytes to follow except checksum)	NI Command	SICB starting w/ message header	Message Checksum of bytes 0-N

For Layer 2 mode:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4..N	BYTE N+1
0xFF	MS byte of length	LS byte of length (bytes to follow except checksum)	NI Command	Priority/AltPath/DeltaBacklog byte, NPDU, CRC (2	Message Checksum of bytes 0-N

Type Code Values

The following table lists the values for the Type Code byte. Uplink means that data is transferred from the U60 to the host processor. Downlink means that data is transferred from the host processor to the U60.

Value	Type	Description	Uplink / Downlink
0	**CpNull**	No data is being sent. Use this value for acknowledge-only packets, or for pinging.	U/D
1	**CpFail**	The previous transfer was broken or in error. This is typically due to checksum errors or fragmented transfers. The correct response to this command is to resend the previous transfer.	U/D
2	**CpMsg**	One message follows. A message is any multi-byte network interface transfer and requires an application output buffer or packet buffer in the downlink case. The number of messages is stored in the **Param Data** field. This value is limited to one in both the downlink and uplink case. The message follows the code packet checksum byte, and consists of a length byte, the NI command, the message itself, and a checksum.	U/D
3	**CpMsgReq**	Optional. Sent by the host for requesting both the attention of the U60 and requesting an uplink **CpMsgAck**. The **Param Data** field contains either a **0** or a **1** informing the U60 that the following message is either a non-priority message (0) or a priority message (1). The U60 driver does not use priority messaging; therefore, this value is ignored. This allows the U60 to respond to the **CpMsgReq** based on actual buffer availability. Alternatively, you can send the entire **CpMsg** plus message to the U60 and a **CpMsgReject** may be sent uplink if there are no available buffers.	D
4	**CpMsgAck**	The U60 is entering the ready-receive state. This is the U60's response to the **CpMsgReq**.	U

5	**CpMsgReject**	An attempt to transfer a message downlink is rejected because of a lack of buffer space. This code packet will be a response to a downlink **CpMsgReq** code packet or a response to a **CpMsg** without the **CpMsgReq** being sent. This code indicates that the offered downlink traffic is more than the U60 can handle (it has no more APP or NET output buffers). Upon receiving this code, the device driver should repeat the message send process until it succeeds.	U
6	**CpNiCmdShort**	Sends a single byte local network interface command. The command is stored in the **Param Data** field.	U/D
7	**CpResync**	This command is sent by the host to start a new session with the U60. When received it will reset the transmit and receive sequence numbers to zero so that any subsequent sequence numbers will be accepted rather than rejected. This command is also a good way to establish that communications are functioning with the U60. The U60 always responds to this command with a **CpNull** packet, so the host can determine that the serial link is not simply echoing data.	D

Notes:

- Use symmetrical timeouts on the host processor. There is an inter-byte timeout on the client side of 40ms when receiving either a code packet or a message body. There is a timeout of 250ms when waiting for the start of a downlink message **CpMsg** following a **CpMsgReq** / **CpMsgAck** sequence.
- All length fields do not count for escape prefixes. Instead, they reflect the length of the real data stream above the link-layer protocol. All length fields do not account for the checksum.
- Broken code packets (code packets with missing or corrupt bytes) are not responded to at all, and they rely on time-out mechanisms for re-transmission.
- Broken message streams are responded to as soon as the error condition is observed. The response is the **CpFail** code packet.

Acknowledgement Rules

The **Ack** bit is used to send a success response to a previous transfer, specifically transfers that send data that is above the LLP (non-LLP data). These are transfers of messages and transfers of local network interface commands.

To reduce traffic, not all data transfers require acknowledgement. The following are the acknowledgement rules:

- If a transfer requires an acknowledgment and there is no other data that needs to be sent, a **CpNull** packet is sent with the **Pass/Ack** bit set. Otherwise, if there is outgoing traffic to be sent the **Ack** bit will be included in the following transfer. This is true for both uplink and downlink data messages.

- Code packets that do not require an acknowledgment via the **Ack** bit are packets that result in an immediate response anyway. The response implies acknowledgment. The following table lists these types of code packets:

Code Packet	Description
CpMsgReq	The U60 will respond with either the **CpMsgAck**, or a criss-cross could occur and an unrelated code packet will be sent by the U60.
CpMsgAck	The host does not need to acknowledge this message. Instead it provides an implied acknowledgement by sending the **CpMsg** code packet followed by a message packet

- Data that is sent and requires acknowledgement will persist in the source until the acknowledgment is received. While the persistent data is waiting for acknowledgement, the **Acknowledge Wait** timeout, which is 300ms for the U60 and the driver, causes the persistent data to be re-sent.

Sequence Number Cycling and Duplicate Detection

The sequence number is used to reject duplicate non-LLP data. Duplicate LLP data does not affect the LLP; therefore, the sequence number is only cycled for each transfer of non-LLP data. For example, two consecutive **CpMsgReq** packets have no effect—the second **CpMsgReq** packet reinstates the ready-receive state and the **CpMsgAck** is re-sent.

U60 Command Set

Uplink local network interface commands that must also convey additional data (for example, the **niL2MODE** response command) will always result in a data message that is at least 3 bytes in length. This is because the host driver expects at least 4 bytes of data (3 bytes plus checksum) to appear following a code packet.

Uplink means that data is transferred from the U60 to the host processor. Downlink mans that data is transferred from the host processor to the U60. The following table lists the U60 commands you can use in your U60 driver.

Value	Name	Description	Uplink / Downlink
0x1x	**niCOMM**	Passes completion events to the host processor.	
0x2x	**niNETMGMT**	Performs local network management or network diagnostic commands on the U60.	
0x31	**niERROR|niCRC**	The Layer 2 mode U60 will send this command uplink whenever it senses a CRC error (0x31).	U
0x50	**niRESET**	When sent downlink this will reset the U60. Following any reset the U60 will send this command uplink.	U/D
0x60	**niFLUSH_CANCEL**	Exits the Flush state.	D
0x70	**niONLINE**	Sets the U60 state to Soft-online.	D
0x80	**niOFFLINE**	Sets the U60 state to Soft-offline. No messaging can be handled in this state.	D

Value	Name	Description	Uplink / Downlink
0x90	**niFLUSH**	Sets the U60 to the Flush state.	D
0xA0	**niFLUSH_IGNORE**	Sets the U60 to the Flush Ignore Comm state.	D
0xCx	**niIO_SET**	Directly controls the U60's four I/O pins, IO0 – IO3, for general purpose I/O control from the U60. The L.S. 4 bits are used to control these pins. These pins are not accessible on a standard U60 module or U60 DIN.	D

0xD0	**niMODE_L5**	Sets the U60 to Layer 5 mode. If already in this mode the U60 will reply with this command. Otherwise the U60 will reset and resume in the new mode. This change is persistent across resets. This command is issued by the U60 driver.	U/D
0xD1	**niMODE_L2**	Sets the U60 to Layer 2 mode. If already in this mode the U60 will reply with this command. Otherwise the U60 will reset and resume in the new mode. This change is persistent across resets. This command is issued by the U60 driver.	U/D
0xE0	**niSSTATUS**	Provides status information. When sent downlink, the U60 responds with the **niSSTATUS** command followed by the following 4 bytes of data: [*TXID Value*], [*MIP Version*], [**1** *(Layer 2 Mode)* or **0** *(Layer 5 Mode)*], [Serial checksum error count]	U/D
0xE6	**niSERVICE**	In Layer 5 mode, sends a Service Pin message. Ignored in Layer 2 mode.	D

Product Query Network Management Command

The U60 supports the **Product Query** network management command from the host only. The code for this command is the Network Management Escape code [0x7D] followed by the sub-command and command values of [0x01], [0x01]. The response includes **PRODUCT** and **MODEL** values based on whether the U60 is currently in Layer 2 or Layer 5 mode. The App Version is 3.1 (31); the TXID is **4** for a TP/FT-10 channel.