Title:	Project: LIPE-SL
LAVA LIPE-SL Developer Manual	Revision: 2
Document Owner: PTV	Date: Nov.27, 2013

TABLE OF CONTENTS:

1 LIPE-SL MODULE INTRODUCTION

The LAVA Internet Protocol System (LIPS) adds networking capability to a new or existing embedded device. A LIPS implementation consists of a LAVA Internet Protocol Engine (LIPE) unit coupled to a user-selected microprocessor. The user-selected microprocessor is referred to as a “LAVA Internet Protocol User” (LIPU) in this document.

This document describes the programming interface for the LIPE-SL module. This is an engineering design document, used to track features added to the LIPE-SL.

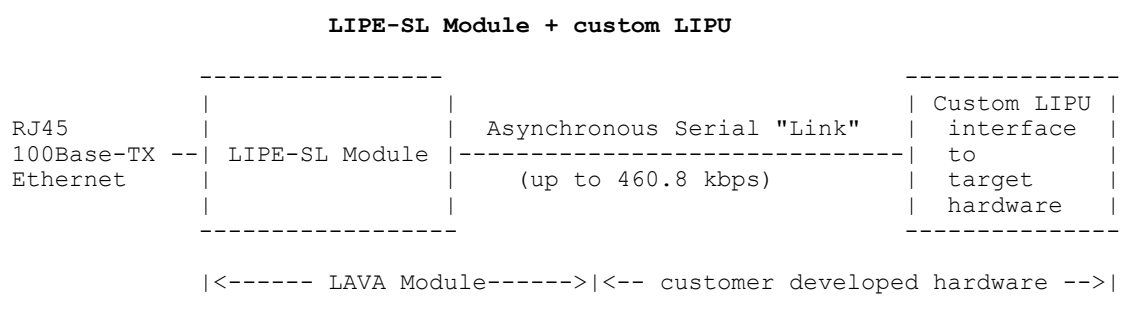
The LIPE-SL to LIPU interface is a simple asynchronous serial port with hardware handshaking. Two pins on the LIPE-SL module select the baud rate.

The LIPE-SL implements a TCP/IPv4 stack which supports the core protocols: TCP, UDP, ICMP, ARP, DNS, and DHCP. The LIPU and therefore the user application are isolated from the details of the TCP/IP stack. Only the payload of each transport protocol message is exchanged between the LIPE and LIPU. The transport protocols are TCP and UDP. Additional higher protocols such as SNTP, FTP, TFTP, Telnet, and SNMP can be easily implemented on the LIPU.

The LIPE-SL supports a mixture of up to 12 TCP and UDP connections over the common interface to the LIPU. Multiple services can be implemented on the LIPU, each with a dedicated TCP or UDP port. The ability to multiplex multiple connections is a key advantage to the LIPS system.

As all TCP/IP management is handled by the LIPE, the LIPU requires very few resources for many applications. A modest and therefore inexpensive user microprocessor is required. The code and data memory footprint associated with adding a TCP/IP stack to a system is isolated in the LIPE module.

The LIPE-SL network interface uses an IEEE 802.3 10BASE-T / 100BASE-TX compatible Fast Ethernet MAC and PHY.



A LIPE module runs a number system processes and user processes. Each user process handles a specific network access point and connection. Each user process is defined as a TCP or UDP network access point. The user processes are volatile, and are set up by the LIPU after a system reset.

A user process on a LIPE is "created", "configured", and then "opened" using a simple command set. Special "event messages" are sent to the LIPU to report LIPE state changes. Messages exchanged between LIPE and LIPU use a simple bracketing scheme based on the SLIP protocol to permit any range of data to be transferred, as well as dealing with the multiplexing of data between connections.

A TCP process is defined for either Client or Server operation. A TCP Client initiates a connection with a remote system. A TCP Server listens for a remote system to connect to it. Once a TCP connection is

established, data transfer is bidirectional. Each TCP process can be assigned a unique TCP Port. TCP processes can be assigned a common TCP Port for improved throughput to a particular service.

A UDP process listens for data from a remote system, or can send data at any time. Each UDP process is assigned a unique UDP Port.

The LIPE command format is a set of simple binary messages, which makes parsing and message generation easy for even the smallest microcontroller.

The LIPE commands due not attempt to follow any socket programming interface. A wrapper can easily be developed on the LIPU to create an interface, which further simplifies LIPU development.

The LIPU is selected by the user based on application requirements. The development environment used is determined by the choice of LIPU, company preferences, and personal preferences. The LIPE does not limit the development environment options.

A LIPS module is intended for devices that have the need for rapid customization, multiple network connections, any set of sensor requirements, modest communication requirements, and no limits on the nature of the data being transferred. Potential devices include light switches, data loggers, environment sensors (such as temperature), door monitors, or even the "intelligent" toaster.

Several sample LIPU frameworks are provided by LAVA to speed up the LIPU development. As the interface to the LIPE-SL is an asynchronous serial port with hardware handshaking, module evaluation and early development can generally be performed using a common Microsoft Windows or Linux based workstation.

2 GLOSSARY

Term	Description
LIPE	LAVA Internet Protocol Engine
LIPE-SL	LAVA Internet Protocol Engine with asynchronous serial port used as the link between the LIPE and LIPU.
LIPS	LAVA Internet Protocol System
LIPU	LAVA Internet Protocol User, the MPU/CPU that uses the LIPE.
MPU	Microprocessor Unit
mS	Millisecond
Octet	Eight bits of information
PID	Process Identifier, to identify the process the data is intended for.
Process	For the LIPE, a process is an instance of code running to perform a specific feature, typically managing a single TCP or UDP access point and connection.
Routing-PID	First octet after a break character indicates the Link Frame data destination or source.
SLIP	Serial Line Internet Protocol
SST	Simple Serial Terminal
MSB	Most Significant Bit
UART	Universal Asynchronous Receiver Transmitter, aka a "serial port".
End of Table	

3 LIPE-SL HARDWARE SUMMARY

The LIPE-SL Module is a component to be plugged, or soldered, into a board developed for a target application. The LIPE-SL module is available in a number of form factors to simplify integration into a customer design. This section provides a summary of the key pins common to all LIPE-SL modules.

The module requires only a +3.3 volt regulated supply rail for it to run.

The LIPE-SL module consists of a dedicated MPU running the LIPE-SL firmware, reset circuitry, interface pins for LIPU, and clocking. The module clock is not available to the LIPU.

The wired modules include a RJ45 Ethernet connector with 2 status LEDs.

RJ45 Network Connector	
Green LED	Description
Off	No network link
On	Network link established
Blink	Network data being sent or received
Yellow LED	Description
Off	10BASE-T connection, if Green LED is active.
On	100BASE-TX connection, if Green LED is active.
End of Table	

The interface signals to the LIPE-SL consist of a core set of 3.3V TTL level signals for serial communications, baud rate selection, and reset. Please see the module specific documentation for additional signals.

LIPE-SL Module Pin Definitions		
Signal Direction	**Signal Name**	**Description**
Output	Serial TX	Asynchronous Serial Transmit, data link from LIPE-SL to LIPU
Input	Serial RX	Asynchronous Serial Receive, data link from LIPU to LIPE-SL
Output	Serial RTS	Request To Send, logic 0 indicates that data exchange is permitted, and therefore logic 1 indicates that data exchanged is not permitted.
Input	Serial CTS	Clear To Send: logic 0 indicates that data exchange is permitted, and therefore logic 1 indicates that data exchanged is not permitted.
Input	Baud Select 1	See the Baud Select table.
Input	Baud Select 2	See the Baud Select table.
Output	Aux. Serial TX	Asynchronous Serial Transmit for the SST process.
Input	Aux. Serial RX	Asynchronous Serial Receive for the SST process.
Input	Reset	Reset the module. This pin is held low for at least 100 microseconds after the supply has stabilized.
End of Table		

The Baud Rate Select pins define the common baud rate used by Serial Transmit and Serial Receive. The Baud Rate Select pins are read on power up. The active Baud Rate selection is reported through a parameter defined as the Device Identifier, which is described later in the document.

Baud Select Pins		
Pin 1	**Pin 2**	**Baud Rate (bps)**
1	1	115200 (default)
0	1	19200
1	0	230400
0	0	460800
End of Table		

If the Baud Select pins are not connected, then weak pull-ups internal to the MPU result in a baud rate of 115200 being selected as the default.

The active baud rate selection can be detected by the LIPU in the Device Identifier, which is a data field described later in this document.

The serial communications always use 8 data bits, no parity and 1 stop bit.

The RTS and CTS flow control lines are always in use to regulate the transfer of data between the LIPE-SL and LIPU. There is no software flow control mechanism.

The LIPE-SL uses a UART with a 16 byte FIFO, therefore up to 16 characters may be issued after the CTS line is set inactive.

The LIPE Heart Beat LED flashes with a period of 600 to 700 milliseconds. If the Heart Beat LED stops flashing, the LIPE is no longer running. If the Heart Beat LED flashes at a very high rate, then the LIPE operation is suspended waiting for the LIPE to LIPU transmit buffer to clear. It is expected that the LIPU is collecting data from the LIPE in a timely manner. If the CTS line is not being driven by the LIPU or has been disconnected, the LIPE to LIPU transmit buffer cannot be cleared.

4 LIPE CONFIGURATION MANAGEMENT

The manufacturing default configuration of a LIPE module is:

```
Device Name             : "** ** ** LIPE"
IP Address              : 192.168.0.35
Subnet Mask             : 255.255.255.0
Default Gateway         : 0.0.0.0
DNS Server IP           : 0.0.0.0
HTTP Configuration Port: 8080
DHCP                    : Disabled
SST Port                : 0
```

The basic Network Configuration of a LIPE-SL can be managed using the HTTP Configuration server on TCP Port 8080. The HTTP Configuration server is a LIPE system process, which runs by default. All Network Configuration parameters can also be read and written via commands from the LIPU.

Changes to the LIPE Network configuration are non-volatile.

The "Device Name" is an ASCII string of up to 16 characters long. The default Device name is "** ** ** LIPE", where "** ** **" are the last 3 octets of the MAC address formatted in hexadecimal. The LIPE Device Name is used for the Host Name field in a DHCP transaction (DHCP option 12).

A static IP Address and Subnet Mask can be set through the HTTP Configuration server. The manufacturing default address should be set to a unique value as soon as possible after receiving a new LIPE module.

The Default Gateway and DNS Server IP Address are set as required for your network.

When DHCP is enabled, the LIPE uses DHCP to establish the following network parameters: IP Address, Subnet Mask, Gateway Address, and DNS Address. The LIPE automatically blocks all network operations from starting until DHCP has resolved the previous network parameters. The LIPU can create and configure LIPE user processes, however the "open" command does not take effect until DHCP has completed. The create, configure, and open operations for a user process are described later in this document.

The HTTP Configuration port sets the TCP port for the HTTP Configuration server, which is 8080 by default. The standard HTTP port 80 is therefore available for the LIPU to implement a custom web server. If the HTTP Configuration port is set to 0, the HTTP Configuration server is disabled on power up. The HTTP Configuration Server can also be disabled by LIPU command.

The Simple Serial Terminal (SST) port activates a dedicated TCP Server process. The SST process transfers data between the auxiliary serial port and the network once a remote system has connected to the SST port. The SST is a simple Ethernet to serial interface. The SST process can be enabled automatically on power up by setting the default SST Port to a non-zero value. The SST process can also be enabled, or disabled by LIPE commands. More information on the SST process is provided later in the LIPE Process Types section.

Application configuration storage is up to the LIPU. The one exception is the "Device Information string", which is a generic 80-octet field. This string is not managed through the HTTP Configuration server. This string is empty by default. The nature of the data in this string is application specific. For additional details on the "Device Information string" please see the LIPE command definitions. The value can be set and read by LIPE commands. This field can also be reported to the LAVA Manager application, via the LDP system process described later in this document.

All Network Configuration parameters can also be read and written via commands from the LIPU. The user can implement a custom web page on the LIPU to handle all network configurations, and can then suppress the default HTTP Configuration server.

The LIPE does not have a pin to restore the above default values. The LIPU is responsible for implementing a “restore” pin or equivalent mechanism. As all of the above configurations can be modified by the LIPU, any set of application specific defaults can be implemented.

5 LIPE-LIPU OPERATION SUMMARY

The LIPE operation summary is presented as a demonstration of how to set up and use the LIPE for a single TCP Server. Setting up TCP Client, UDP processes, or additional TCP Servers are simple variations on the example. Many of the messaging details are skipped over for this summary and are covered in the later sections of this document.

The example represents each octet (byte) transferred between the LIPE and LIPU as HH, where the HH is a hexadecimal value. The 0x prefix used in the remainder of the document is not used in this section. Decimal values are single digit or are noted as decimal.

The LIPE-LIPU communications is always performed in blocks of data referred to as “Link Frames”, or simply “frames” for short. All frames start with a C0 octet and end with a C0 octet. The C0 octet is referred to as a “break” character. The second octet after the opening break always identifies the nature of the frame. This second octet is referred to as a “Routing Process Identifier” (RPID).

For LIPU to LIPE frames, a RPID of 80 indicates the frame contains a command. All other LIPU to LIPE frames use RPID 01 to 0C and contain network data for the payload of a TCP or UDP message to be sent to the network. On power up any frame with RPID 01 to 0C received by the LIPE is dropped, as the related LIPE processes have not been set up yet.

For LIPE to LIPU frames, a RPID of 80 (128 decimal) indicates the frame contains an event message or a command acknowledgment. All other frames have an RPID of 01 to 0C and contain network data from the payload of a TCP or UDP message received from the network. On power up no frame with RPID 01 to 0C is produced, as the LIPU must first set up the related processes within the LIPE.

All octets between “break” characters of a frame are transmitted using special rules. Any C0 is sent as DB DC. Any DB is sent as DB DD. These special rules allow the break characters to be unique in the LIPE-to-LIPU exchanges. The receiver must check for these special sequences, and restore the original characters. The examples generally do not show the expanded sequences.

The LIPE always sends two frames of information to the LIPU on power up:

The first frame sent from LIPE to LIPU is 7 octets long, and indicates a LIPE reset has occurred:

```
C0 80 10 82 0A 80 C0
```

The “10 82 0A 80” portion of the first frame is an event message, which reports the LIPE has reset. Additional information on event messages and the specifics of the format are described later in the document.

The second message is a minimum of 27 octets long and indicates the module’s network configuration has been set in the LIPE:

```
C0 80 10 94 10 80 XX 0F ii ii ii ii mm mm mm mm gg gg gg gg dd dd dd dd C0
```

From the “10 94” to the final “dd” is the Network Configuration event message. This event message reports the network configuration being used by the LIPE and how it was derived. The configuration can be obtained from a static configuration or DHCP.

The actual number of octets in the second frame depends on the values of the data between the break characters. Any CO or DB octets between the frame breaks must be expanded prior to transmission as noted earlier in this section.

When the network configuration is static, the second message is sent immediately after the first message. If using DHCP, there may be a delay while the DHCP process interacts with a DHCP server. The DHCP process can take up to 100 seconds to report success or failure. The XX field of the Network Configuration event message reports the status of the DHCP interaction.

The LIPE is now ready for the LIPU to set up “user processes”. The LIPE can be set up before the Network Configuration event message; however, the processes are blocked from interacting with the network until the Network Configuration is established, as is signified by the second frame.

The following frames are sent to the LIPE to define a TCP Server as user process **01** using TCP Port 4098 decimal:

```
C0 80 11 02 06 01 C0 C0 80 18 04 01 01 10 02 C0 C0 80 12 01 01 C0
 |               |   |                       |   |              |
 --- Frame #1  ---    ------- Frame #2 -------   --- Frame #3 ---
```

The LIPE responds to each frame and returns the following to acknowledge the commands were received correctly for user process **01**:

```
C0 80 11 83 00 01 06 C0 C0 80 18 83 00 01 01 C0 C0 80 12 82 00 01 C0
 |                  |   |                   |   |                |
 ----- Frame #1  -----   ----- Frame #2 ------   ---- Frame #3 ----
```

The first frame sent to LIPE creates a TCP Server as process **01** which uses Process Identifier (PID) **01**. The PID is shown in bold with an underline. The permitted values for the PID are 01 to 0C (1 to 12 decimal). Each PID corresponds to one of the 12 possible network connections supported by the LIPE-SL.

The second frame configures process **01** to use TCP Port 4098 decimal (or 1002 hexadecimal).

The third frame requests the LIPE open process **01** and allow it to listen to the network. The response to the open confirms the command was received; however the process is not immediately listening to the network.

The LIPE command acknowledgments can be ignored once the system has been debugged, which can simplify the LIPU further. The acknowledgment frames can be used by the LIPU to keep a state machine synchronized to the LIPE.

Within a moment the following frame is sent from LIPE to LIPU:

```
C0 80 10 82 01 01 C0
```

This frame is an event message, which reports that process **01** is now listening to the network on the specified TCP port. All user processes generate an event to report when the open was successful. A TCP Server generates the event immediately, while a TCP Client must first establish the TCP connection with the remote system. In the case of a TCP Client the remote system may not be found, resulting in a failed open attempt. A failed TCP Client open has a related event message reported to the LIPU.

When a connection is established by the remote system, the LIPE sends the following frame to the LIPU:

```
C0 80 10 8A 02 01 ii ii ii ii pp pp C0
```

The frame is an event reporting a remote system has opened a connection. The ii octets report the IP Address of the remote system. The pp fields report the TCP Port of the remote system. Values reported

which are larger than 1 octet are transferred big endian. One use for the auxiliary information in this event is to qualify which remote systems are allowed access.

The LIPU can now expect data from the remote system or send data to the remote system. A TCP connection is bidirectional.

Assume the remote system sends the text message “12345”. The LIPE handles all the required TCP handshaking, and sends the following to the LIPU:

```
C0 01 31 32 33 34 35 C0
```

The LIPU can send “6789” to the remote system by sending the following:

```
C0 01 36 37 38 39 C0
```

The nature of the data expected by the process **01** and the responses are application specific. If the TCP Port 50 (80 decimal) was used, the remote system has likely sent an HTTP request and is expecting a Web Server to be parsing the data. There is a special LIPE TCP process that simplifies Web Server development for the LIPU, which is described later in the document.

When the TCP connection remains open and no data is being transferred, the LIPE periodically sends a TCP Keep Alive to the remote system to ensure it is active. If the remote system does not respond, the LIPE closes the TCP connection and makes the user process available for a new TCP connection. An event message is sent to the LIPU to inform the LIPU that the connection is now closed.

The remote system and LIPU can exchange data for a long as is required. All data sent in each direction requires TCP handshaking and error checks between both ends of the connection. All the TCP handshaking and error checks are handled by the LIPU. Either the remote system or the LIPU can terminate the connection at any time, as with any TCP connection.

The LIPU can terminate the TCP connection by sending a LIPE Close command:

```
C0 80 13 02 00 01 C0
```

The LIPE Close command from the LIPU results in the following LIPE response:

```
C0 80 13 82 00 01 C0
C0 80 10 82 03 01 C0
```

The first frame acknowledges the “close” command was received for process **01**.

The second frame is an event message, which reports the TCP process and connection are closed. The LIPE Close operation can take several milliseconds to several seconds depending on the network.

Whether the TCP connection is closed by the remote system or by the LIPE, the same event message is generated to inform the LIPU. The remote system closes the connection using a TCP FIN, which is part of the network-to-LIPE exchange hidden from the LIPU.

When any LIPE process is in the closed state, the process remains ready to be opened again by the LIPU. The configuration of the process remains unchanged. When implementing a Web Server, the LIPU always closes the connection after sending the last byte of the requested file. Once the LIPE reports the connection has actually closed, the LIPU must then send an “open” request to the LIPE to prepare for a new web request. These rules follow the HTTP protocol requirements.

When the process is in the closed state, configuration changes for the process are permitted. Changing the configuration of a process that is in the open state is prevented by the LIPE.

The LIPU may decide to delete the TCP Server process. The LIPE Delete command uses the PID of the process as a reference. All resources that were used within the LIPE are released. The process should have been “closed” first to gracefully shut down the TCP connection. The delete command is immediate with no concern for the state of the TCP connection. The LIPE Delete command has an acknowledgment sent for the command, but no event. In many applications the user processes need never be deleted, they are created and simply left in the closed state ready to be activated by an open command.

There are a number of event messages that can be monitored by the LIPU, depending on the application requirements. These event messages are described later in the document.

The LIPU should always watch for the system reset event message. If there is a power brownout, the LIPE may reset and the LIPU may not. All user processes are lost when the LIPU is reset. The LIPU should watch for this event and reconfigure the LIPE.

The LIPU startup should always reset the LIPE to ensure a known starting point.

There are a number of system processes within the LIPE, which each has a unique PID. The Link Manager uses PID 80 (128 decimal). The Link Manager is the system process that receives commands from the LIPU, sends command responses to the LIPU, and is the source of event messages. The Link Manager is the only system process that transfers frames of information to and from the LIPU. The PID of the other system processes is only used as a reference in the LIPE Delete command.

To set up a second TCP Server, simply repeat the 3 frames (used for the create, configure, and open) with another PID from 02 to 0C. To set up a second TCP Server using PID **02**, send the following:

```
C0 80 11 02 06 02 C0 C0 80 18 04 02 01 10 02 C0 C0 80 12 01 02 C0
 |               |   |                       |   |             |
 --- Frame #1  ---    ------- Frame #2 -------   --- Frame #3 ---

C0 80 11 02 06 02 C0
C0 80 18 04 02 01 10 02 C0
C0 80 12 01 02 C0
```

The event messages described earlier apply to this new process. The event messages all use PID **02** as the reference.

The PID used need not be **01** or **02**. Any PID from **01** to **0C** that is not already in use may be selected.

Setting up a TCP Client is just as simple. The create command sends 07 rather than 06. The create command is the first frame of the three frames in the setup example.

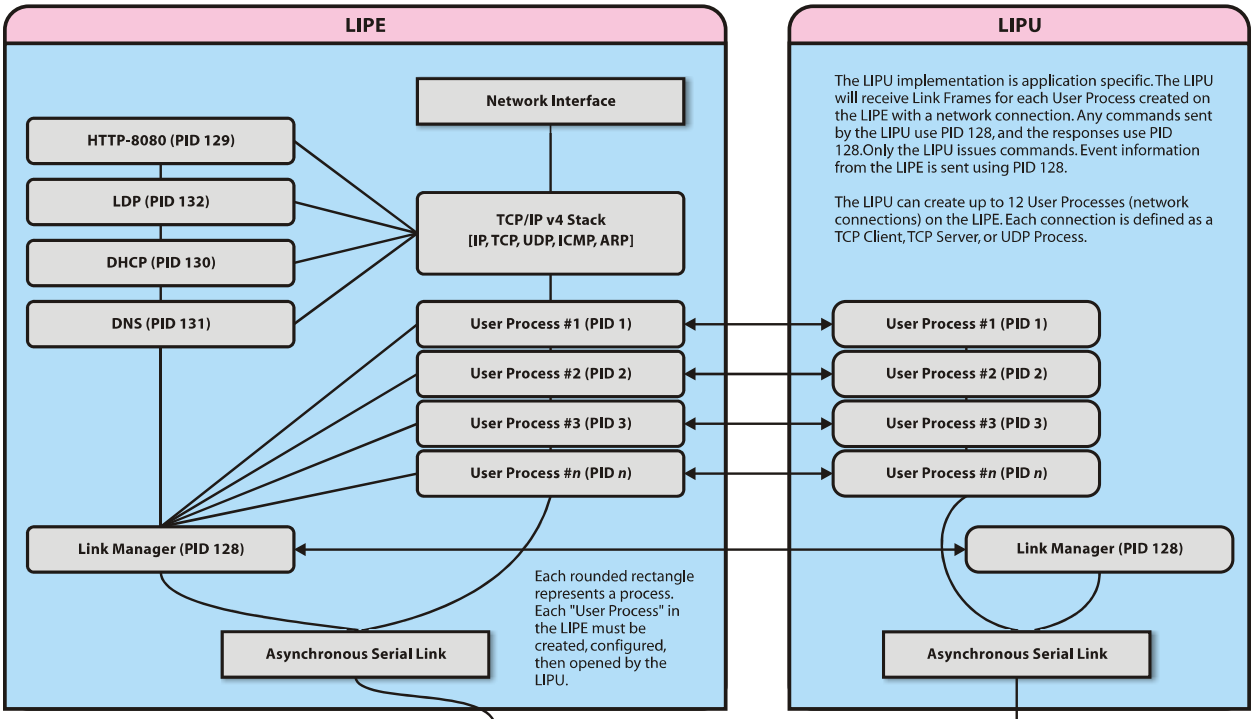
A TCP Server requires two additional messages to set the remote system address and TCP Port. The “listening” event is not generated; however the “connected” event is reported when the TCP connection to the remote system is established. If the remote address is specified as a name, the LIPE automatically uses DHCP to resolve the IP Address. The DHCP Server Address must be set up in the LIPE Network Configuration to use DHCP.

If the TCP Server open fails, the open fail event reports the specific failure.

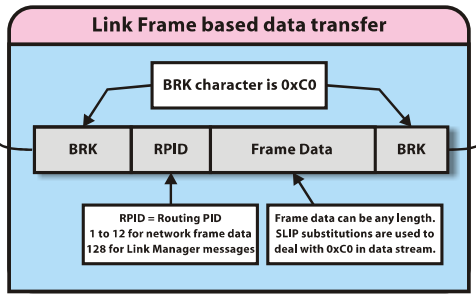
Setting up a UDP process is just as simple as setting up a TCP Server. The data frames transferred between the LIPE and LIPU have special rules that are defined later in the document.

The remainder of the document describes the format of the messages in detail along with the various message types. Additional LIPE user processes for UDP and special cases of TCP are also described.

LIPE-LIPU Architecture Summary



Each User Process is attached to a TCP or UDP port in the TCP/IP stack.
When the LIPE starts up there are no User Processes present.
The LIPE has a number of default system processes running, each with an associated PID. The Link Manager is the only system process that communicates with the LIPU. The system process PID can be used to shut down the process from the LIPU using a "Close" command.
The Link Manager cannot be shut down, as it is used to receive commands from the LIPU.
The DNS handler is active by default. DNS operations are initiated via commands sent to the Link Manager.
The DHCP handler is active by default if DHCP is enabled in the network configuration. This handler will set up the module's IP address, subnet mask, TCP port, gateway , and DNS server IP.
The LDP (LAVA Discovery Protocol) is active by default for use with the LAVA Manager.
The HTTP-8080 Web Server is active by default to provide an initial configuration mechanism for the LIPE device.



6 PROCESS IDENTIFIERS (PID)

Every process within the LIPE has a unique identifier referred to as a “Process Identifier” (or PID). All command and command acknowledgments use a PID to identify the target process.

The PID range 1..127 (0x01..0x7F) is reserved for user processes.

PID values 1..12 (0x01..0x0C) are used for LIPE-SL user processes, such as the Generic TCP Client and Generic TCP Server.

The PID range 13..127 (0x0D..0x7F) is reserved for future versions of the LIPE.

The PID range 128 to 255 (0x80..0xFF) is reserved for system processes, which include the Link Manager, HTTP-Configuration, DHCP, and DNS. These processes generally perform specific operations within the LIPE and need no interaction with the LIPU. Additional details on the system processes are provided later in the document. The PID values of 0x00, 0xC0 and 0xDB are never used.

A PID is always transferred as a single octet.

A Routing-PID (RPID) is the first octet after a <break> character in a Link Frame. This PID reports which process the frame data is being sent to or sent from. The RPID indirectly indicates the nature of the frame data.

A RPID of 128 (0x80) indicates the Link Frame contains a command message, command message response, or an event message. The Link Manager (PID 128) is the LIPE process that receives command messages, generates command message acknowledgments, and generates event messages.

An RPID of 1..12 (0x01..0x0C) indicates the payload of the Link Frame is data being sent to the network, or received from the network. This “payload” is the payload portion of a TCP or UDP frame. The PID, the TCP and UDP headers, IP headers, and Ethernet headers are hidden by the LIPE.

7 LIPE PROCESS TYPES

7.1 OVERVIEW

A LIPE module supports a number system processes and user processes.

Each system process provides a specific feature to the LIPU. Several standard TCP/IP services are implemented as LIPE system processes.

Each user process handles a specific network access point and connection for the LIPU. Each user process is defined as a TCP or UDP network access point. The user processes are volatile, and are set up by the LIPU after a system reset.

The process types are:

```
    1  (0x01) UDP-UB Process
*   2  (0x02) HTTP-Configuration-Server (fixed PID of 129)
*   3  (0x03) reserved for DHCP Process (fixed PID of 130)
*   4  (0x04) reserved for DNS Process  (fixed PID of 131)
*   5  (0x05) LDP Process               (fixed PID of 132)
    6  (0x06) TCP Server Process
    7  (0x07) TCP Client Process
    8  (0x08) TCP HTTP Server
    9  (0x09) UDP-U Process
    10 (0x0A) TCP SST Process
    99 (0x63) Link Loopback
```

The options marked by '*' are system processes created automatically by the LIPE-SL. LIPE-SL does not implement options 3 and 4 for the Create-Process command. The DNS process is always active. The Network Configuration controls the DHCP process.

All system processes run upon creation. All user processes must be issued an "open" command after being created. After creation and prior to a user process being opened, configuration changes can be made.

The System processes (HTTP-Configuration-Server, DHCP, DNS, and LDP) each have one instance in the LIPE. If a System process is created more than once, the success code is always returned. The process is not reset or altered in any way by a redundant "create". A system process is always created with the fixed associated PID, which is reported in the command acknowledgment. For example, attempting to create the HTTP-Configuration-Server as PID 200 will nonetheless result in PID 129 being used and PID 129 will be reported in the command acknowledgment.

The LIPE has a pool of 12 TCP/IP Transmit buffers, which are shared by the LIPE processes. The HTTP-Configuration-Server, TCP Server, TCP Client, TCP HTTP Server, and UDP Process each require a TCP/IP Transmit buffer. If the HTTP-Configuration-Server is not required, it can be "Deleted" to free up one of the 12 buffers. When a user process is "created" it is assigned 1 of these 12 buffers by the LIPE.

The TCP Client and UDP processes use a small LIPE ARP Cache to determine if an ARP Request is required. The cache stores the last 12 addresses. An ARP Cache entry is kept for a maximum of 10 minutes.

A single UDP process is expected per UDP Port. If two UDP processes listen to the same port, only the first UDP Process based on creation order receives an incoming frame. The LDP Process uses port 4101, DNS Client uses port 1060, and DHCP uses port 68.

7.2 SYSTEM PROCESS - HTTP CONFIGURATION SERVER

The HTTP-Configuration-Server is a web server used to set the basic networking configuration of the LIPE. The default TCP port is 8080. The TCP port used by this web page can be changed.

This process reduces the number of user processes by 1.

Only one instance of this process is permitted.

The PID is always 129. This process can be "deleted".

The process is started automatically on power up if the HTTP Configuration port is non-zero in the Networking Configuration.

The process can only be created if the HTTP Configuration port is non-zero.

All default web page parameters can be retrieved and modified by the LIPU.

The default Web page allows the following networking parameters to be managed:

1) Device Name

2) IP Address

3) Subnet Mask

4) Default Gateway

5) DNS Server IP

6) HTTP Configuration Port

7) DHCP (enabled or disabled)

8) SST Port

The networking parameters are described earlier in the document.

Changes to the LIPE Network configuration are non-volatile.

This process does not generate event messages.

7.3 SYSTEM PROCESS – DHCP CLIENT

The LIPE can use DHCP to establish the following network parameters: IP Address, Subnet Mask, Gateway Address, and DNS Address.

Use of DHCP is selected in the Network Configuration and activated on a reset. The DHCP process cannot be enabled by the Create-Process command on the LIPE-SL, and an error 11 (command not supported) results.

The DHCP process starts within 10 seconds of the LIPE powering up. The start-up time varies from module to module, to avoid all modules talking at once. The delay is tied to the last 2 octets of the MAC address.

When DHCP is enabled, the LIPE automatically blocks all network operations from starting until DHCP has resolved the network parameters. The LIPU can create and configure LIPE user processes; however, the “open” command does not take effect until DHCP has completed. The create, configure, and open operations for a user process are described later in this document.

The DHCP Client listens on UDP Port 68. A DHCP Server listens on UDP Port 67.

The maximum fixed lease time permitted is 21474000 seconds, which is approximately 248 days. The lease time is limited to the maximum value of 21474000 seconds. The minimum lease time permitted is 60 seconds. A lease time value lower than 60 is limited to 60 seconds. A lease time value of 0 is an unlimited lease. The LIPE never releases a configuration with a lease time value of 0.

The DHCP Client issues the Network Configuration event message to report status to the LIPU. When the configuration is established, the event message is issued with status 2.

On power up a user process can be opened until the Network Configuration event message reports a valid configuration. The LIPE accepts process open requests, however each process is held off automatically.

When the network configuration cannot be established by DHCP on power up, the Network Configuration event message is sent with status 4. The LIPE continues to issue DHCP requests after the event message is sent. The timeout for the initial event message is approximately 90 seconds. After the initial event message, the timeout is approximately 150 seconds. The LIPU continues to receive the event message with status 4, until a valid configuration can be acquired.

A DHCP acquired Network Configuration is silently updated by the LIPE each time the lease is coming due. If the lease renewal fails, the LIPE is issued the Network Configuration event message with status 6. The DHCP process halts after reporting status 6. The LIPE automatically closes any TCP or UDP User Process. Once the LIPU is ready to allow the DHCP operation to continue, the LIPU issues an Action Command with index 2 selected. This command allows the DHCP process to continue, and start to acquire a new network configuration as if the module had just started.

When the DHCP lease is renewed, there is a chance the DHCP Server may assign a new IP Address. A system, which is assigned a new IP Address, is obligated to shut down any process using the old address, and the switch to the new address. If the renewed Network Configuration changes, the Network Configuration event message is issued with status 3. The DHCP process halts after reporting status 6. The LIPE automatically closes any TCP or UDP User Process. Once the LIPU is ready to allow the DHCP operation to continue, the LIPU issues an Action Command with index 2 selected. This command allows the DHCP process to continue, and start to acquire a new network configuration as if the module had just started.

This process generates the following event messages:

#16 - Network Configuration

7.4 SYSTEM PROCESS – DNS CLIENT

The DNS process manages the DNS Network Configuration.

The DNS process is enabled automatically on the LIPE-SL and cannot be deleted. Attempting to create this process results in an error 11 (command not supported).

The LIPU can issue a command to the DNS process to lookup a DNS name and report back the IP Address via event message 9. When a TCP Client is configured to use a DNS Name rather than an explicit IP Address, it automatically uses the DNS lookup, however the DNS completion event is not sent to the LIPU. The TCP Client LIPU learns what IP address was located by the event message 2.

Only one DNS lookup can be performed at a time within the LIPE. When multiple TCP Clients attempt a DNS lookup, the user processes are processed in the order in which the lookup was attempted. When the LIPE DNS Lookup command is used, an error is reported if the system process is busy.

The DNS Client listens on UDP Port 1060 (53 + 1007). A DNS Server listens on UDP Port 53.

This process generates the following event messages:

```
#9 – DNS Lookup Complete
```

Note: The DNS Lookup Complete is only generated when a DNS lookup is started by a DNS Lookup command (see command opcode 27).

7.5 SYSTEM PROCESS - LDP SERVER

The LDP process supports the LAVA Discovery Protocol. A subset of LDP used by the LIPE. Only the basic discovery commands are implemented in the LIPE. The LAVA Discovery commands are used by the LAVA Manager application, and are described in a related document. This process is started by default on the LIPE and can be deleted by the LIPU.

The LDP process listens on UDP Port 4101.

This process does not generate event messages.

7.6 USER PROCESS – TCP SERVER

The TCP Server process is used to transfer TCP data to and from the network.

Once a TCP Server is created and opened, it listens on the network for a TCP connection attempt by a remote system. Once a TCP connection is established, an event message informs the LIPU. Once the TCP connection is established, data may be sent in either direction.

The TCP Server process uses a default Local TCP Port number of 4099 less the assigned PID. The Local TCP Port can be set by a LIPE Process-Cfg-Set command.

Either the remote system or the LIPU can end a TCP connection. The LIPU ends a connection by sending the user process a close command. The remote system ends a TCP connection by sending a TCP frame with the FIN flag bit set. The details of shutting down a TCP connection at the network level are handled by the LIPE. Once the connection has ended, and event message is sent to the LIPU. Once the connection has ended, the LIPU can send an open command to allow the process to listen for another remote system to establish a connection.

Data from the network is sent to the LIPU as Link Frame with the RPID set to the PID of the TCP Server user process. The maximum size of the data in the Link Frame is 970 octets. The 970 bytes does not include extra octets added to deal with 0xC0 and 0xDB octets, which is described later in the document.

The data from the LIPU is sent to the network as fast as possible, there is no attempt to keep the contents of a Link Frame intact by default.

The TCP Server can be set to “frame mode” using the Network Process Attribute parameter. The “frame mode” results in data forwarding only on a complete Link Frame, or process specific threshold. See the Network Process Attribute description for additional details.

This process generates the following event messages:

```
#1  – Open
#2  – Connected
#3  – Closed
#4  – TCP Reset
#5  – Invalid TCP Connection
#7  – TCP Keep Alive Received
#8  – TCP Keep Alive Sent
#15 – TCP Packet Dropped
```

7.7 USER PROCESS – TCP CLIENT

The TCP Client process is used to transfer TCP data to and from the network. A TCP Client user process initiates a TCP connection to a remote system. Once the TCP connection is established, data transfer is bidirectional using the same rules as the TCP Server.

The TCP Client process uses a default Local TCP Port number of 4099 less the assigned PID. The Local TCP Port can be set by a LIPE Process-Cfg-Set command. The TCP Client process requires a Remote TCP Port and Remote IP Address before it can be "opened". The Remote TCP Port and Remote IP Address can be set by a LIPE Process-Cfg-Set command.

The Remote IP Address is a string, which defines an IP Address or a system name for DNS lookup. See the LIPE Process-Cfg-Set command for additional details. The LIPE does not support an IP loopback address.

When using a DNS lookup for the remote system, there is a chance of failure. The LIPE performs an ARP Request to resolve the MAC address of the remote system. If the DNS lookup or ARP Request fails, then an Open Failure Event message is sent to the LIPU. See the event message descriptions for additional details.

When connecting to the remote system, there is a chance of failure. If the LIPE cannot contact the remote system, then an Open Failure Event message is sent to the LIPU. See the event message descriptions for additional details.

The "fixed local port" option locks the local port to the assigned value, else it is advanced between subsequent "close" and "open" operations. When not using the "fixed local port" operation, the Local TCP Port has 2048 added to it by the LIPE. The TCP Port behavior is derived from the LAVA ESL.

Either the remote system or the LIPU can end a TCP connection. The LIPU ends a connection by sending the user process a close command. The remote system ends a TCP connection by sending a TCP frame with the FIN flag bit set. The details of shutting down a TCP connection at the network level are handled by the LIPE. Once the connection has ended, and event message is sent to the LIPU. Once the connection has ended, the LIPU can send an open command to allow the process to listen for another remote system to establish a connection.

Data from the network is sent to the LIPU as Link Frame with the RPID set to the PID of the TCP Server user process. The maximum size of the data in the Link Frame is 970 octets. The 970 bytes does not include extra octets added to deal with 0xC0 and 0xDB octets, which is described later in the document.

The data from the LIPU is sent to the network as fast as possible, there is no attempt to keep the contents of a Link Frame intact by default.

The TCP Client can be set to “frame mode” using the Network Process Attribute parameter. The “frame mode” results in data forwarding only on a complete Link Frame, or process specific threshold. See the Network Process Attribute description for additional details.

This process generates the following event messages:

```
#1  - Open
#2  - Connected
#3  - Closed
#4  - TCP Reset
#5  - Invalid TCP Connection
#6  - Open Failure
#7  - TCP Keep Alive Received
#8  - TCP Keep Alive Sent
#15 - TCP Packet Dropped
```

7.8 USER PROCESS – TCP HTTP SERVER

The TCP HTTP Server process is a TCP Server with an HTTP filter. The filter eliminates most of the HTTP header information and presents a subset of the header data to the LIPU.

The general HTTP header format is:

```
<method> <resource> <http-version>CR+LF
<header line n + 1>  CR+LF
<header line n + 2>  CR+LF
 .....  .... . . .
<header line n + N>  CR+LF
CR_LF
<optional closing data> CR+LF
```

Each header line starts with a keyword. The only header line of interest to the TCP HTTP Server process starts with “Cookie:”. Each Cookie with a “lips” prefix is reformatted and presented in the same manner as a variable to the LIPU. The ‘;’ separators used in the Cookie field are replaced with ‘&’ separators. White spaces in a Cookie are not modified.

An HTTP GET has a file name and optional variables specified in the <resource> field. The filtered HTTP information sent to the LIPU for a "HTTP GET" is:

```
G<resource><cookies>CR+LF
```

An HTTP POST has a file name in the <resource> field and may have variables presented in the <optional closing data>. The filtered HTTP information sent to the LIPU for a "HTTP POST" is:

```
P<resource><optional closing data><cookies>CR+LF
```

Variable and Cookie names need to be kept as short as possible. The convention used by may of the LIPE examples use “Sn” and “Cn” names, where “n” is a decimal integer. These values are compact and easy to parse. The “S” is used for general String variables. The “C” is used for Configuration variables. The values used are arbitrary, as they are completely controlled by the LIPU developer.

The filtered HTTP information sent to the LIPU for any other HTTP method is:

```
?CR+LF
```

Note: The CR+LF refers to a 2-octet sequence with the values 0x0D and 0x0A.

If the LIPU requires all HTTP header information to be passed through, then a TCP Server is to be used. The LIPU is then in full control.

The TCP Port used to accept the TCP HTTP Server is configured by the LIPU just as for a TCP Server. The default TCP Port rules for a TCP Server process apply to the TCP HTTP Server.

This process generates the following event messages:

```
#1  - Open
#2  - Connected
#3  - Closed
#4  - TCP Reset
#5  - Invalid TCP Connection
```

```
#7  - TCP Keep Alive Received
#8  - TCP Keep Alive Sent
#15 - TCP Packet Dropped
```

A simple example of a HTTP GET with variables and cookies is:

```
GET /io/file.bas?S1=some&data&S2=more&data HTTP/1.1
Accept: image/gif, image.jpeg, ……..
Referer: http://192.168.0.35/io/input.bas
Accept-Language: en-us
User-Agent: Mozilla/4.0 ……
Accept-Encoding: gzip, deflate
Host: 192.168.0.35
Connection: Keep-Alive
Cookie: lips-1:ck1; lips-2:ck2
```

The example has no optional data. The majority of the header line information has been removed from the example. An actual GET has many lines of information in the header to define the browser capabilities to the server. The example would be filtered and reported to the LIPU as:

```
G/io/file.bas?S1=some&data&S2=more&data&lips-1:ck1&lips-2:ck2
```

A simple example of a HTTP POST with variables and cookies is:

```
GET /io/file.bas HTTP/1.1
Accept: image/gif, image.jpeg, ……..
Referer: http://192.168.0.35/io/input.bas
Accept-Language: en-us
User-Agent: Mozilla/4.0 ……
Accept-Encoding: gzip, deflate
Host: 192.168.0.35
Connection: Keep-Alive
Cookie: lips-1:ck1; lips-2:ck2

S1=some&data&S2=more&data
```

The example show the variable in the optional data field. The majority of the header line information has been removed from the example. An actual POST has many lines of information in the header to define the browser capabilities to the server. The example would be filtered and reported to the LIPU as:

```
P/io/file.bas?S1=some&data&S2=more&data&lips-1:ck1&lips-2:ck2
```

The uniform presentation of the HTTP request simplifies the parsing effort on the LIPU.

7.9 USER PROCESS – UDP-UB PROCESS AND UDP-U PROCESS

The UDP processes are used to transfer UDP data to and from the network. The LIPU has special rules that must be followed to work with UDP.

The UDP-U process supports only unicast IP addresses. The process does not accept any broadcast messages sent to the local UDP port of the process.

The UDP-UB process supports unicast and broadcast addresses. The process does accept broadcast messages sent to the local UDP port of the process. The IP Address of a UDP outgoing message can be set to 255.255.255.255 to generate a broadcast message. The MAC destination address for a broadcast message is set to FF:FF:FF:FF:FF:FF. The highest IP address on the sub-set also is treated as a broadcast message.

A UDP process uses a default Local UDP Port number of 4099 less the assigned PID. The Local TCP Port can be set by a LIPE Process-Cfg-Set command.

Once "opened" the process will listen on the specified local port. Incoming UDP frames are forwarded to the LIPE, with 6 octets inserted at the start of the frame data. The 6 octets inserted are for the IP Address followed by the UDP Source Port of the remote system. The LIPU knows it's own UDP Local Port. The LIPU must deal with the inserted data and use it for any Acknowledgment.

The LIPU to LIPE UDP data is expected to have the remote systems IP and Port inserted into the start of the data block. The first 4 octets are the remote system IP Address. The next 2 octets are for the remote system UDP Port. The LIPE does not support an IP loopback address.

All UDP transfers are sent as blocks. The LIPE allows for a 494-byte payload for UDP messages, including the 6 bytes for remote system identification. The LIPU must not send more than 488 bytes of data. The IP Address and Source Port are sent big endian.

When a UDP message has been sent, the LIPU is sent event message 13. The third octet of the event report is 1 if the UDP message was sent. The third octet of the event report is 2 if the requested IP Address could not be resolved to a MAC Address.

This process generates the following event messages:

```
#1  – Open
#2  – Connected
#3  – Closed
#13 – UDP TX Done
```

7.10 USER PROCESS – SIMPLE SERIAL TERMINAL (SST) PROCESS

The Simple Serial Terminal (SST) port is a dedicated TCP Server process. The SST process transfers data between the auxiliary serial port and the network once a remote system has connected to the SST port. The SST is a simple Ethernet to serial interface.

The SST process can be enabled automatically on power up by setting the default SST Port to a non-zero value in the Network Configuration. The SST process can also be enabled or disabled by LIPE commands. When the SST process is enabled by LIPE command, it must also be opened before the process listens to the network.

Only one instance of this process is permitted. Each time the SST process is closed, it automatically re-opens. This automatic reopen occurs for a remote close or a LIPE command based close.

The SST process supports 9600 baud or 19200 baud. An odd SST TCP port results in 19200 baud, while an even SST TCP Port results in 9600 baud. The data format is always 8 bits with no parity. The serial port operates with no flow control. Once a remote system has connected to the SST, data received on the auxiliary serial port is sent to the network when 256 characters are received, or 100 milliseconds of no activity.

The auxiliary serial port has a 1024 byte receive buffer and a 1024 byte transmit buffer.

The SST process transfers data without using the special rules required for LIPE-LIPU communications. No <break> characters are used on the serial port. No RPID is used on the serial port. No LIPE command or event information is transferred through the auxiliary serial port.

The TCP Local port setting applies to the SST process. If a TCP Local Port is not assigned, the default rules of a TCP Server Process apply.

The SST process uses the resources of three LIPE user processes. When the process runs as a power up option, it is assigned PID 12.

When the SST is used with no LIPU present, the CTS signal on the LIPE to LIPU interface must be tied low. Event messages generated for the SST process are sent to the LIPU. If the CTS line is left to float active, no data can be sent in the path to the LIPU. If the transmit buffer to the LIPU fills up, the LIPE locks up waiting for the buffer to clear. When the transmit buffer cannot be cleared, the LIPE Heart Beat LED flashes at a very high rate. The normal rate is approximately every 600 to 700 milliseconds.

When the SST is used with no LIPU present, DHCP lease loss is handled automatically by the LIPE. In this case no DHCP Continue message is required for the DHCP process to continue operation after a lease loss. A DHCP lease loss results in the SST being automatically closed. The SST process is returned to the listen state once a new DHCP lease is acquired.

This process generates the following event messages:

```
#1  – Open
#2  – Connected
#3  – Closed
#4  – TCP Reset
#5  – Invalid TCP Connection
#7  – TCP Keep Alive Received
#8  – TCP Keep Alive Sent
#15 – TCP Packet Dropped
```

Note: SST Event messages are sent to the LIPU, not the auxiliary serial port.

7.11 USER PROCESS – LINK LOOPBACK PROCESS

The Link Loopback Process is a diagnostic process intended for factory use only. This process will echo any link data received by the LIPU back to the LIPU.

This process starts as soon as it is created, even though it is defined as a user process.

The command acknowledgment is generated with no Link Frame wrapper. After the command acknowledgment the process reports "LOOPBACK STARTED". There are a number of changes to the standard operation enabled by this command that are factory specific and not documented here.

The Link Loopback process is intended to be the only process enabled from the LIPU. Once this process is enabled, the LIPE stops parsing commands from the LIPU. The only way to remove the Link Loopback process is by a physical reset of the LIPE.

This process does not generate event messages.

8 LIPE-LIPU LINK PROTOCOL

8.1 OVERVIEW

The LIPE and LIPU communicate over an asynchronous serial link using RTS-CTS hardware handshaking. The electrical interface is simple TTL drivers and receivers. As the intended distance between the LIPE and LIPU is no more than a few centimeters, the connection is deemed error free.

The rules used to transport data between the LIPE and LIPU is referred to as the “Link-Protocol”. A block of information sent using the Link-Protocol is referred to as a "Link Frame".

The rules to transfer data from LIPE to LIPU and LIPU to LIPE are symmetrical. The document uses the term LIPE-LIPU communications to refer to data transfer in either direction.

8.2 THE LINK FRAME

All LIPE-LIPU communications use the following Link Frame format for transporting information between the two units:

```
<break> <rpid> <frame data> <break>
```

The '<' and '>' characters in sample messages are not part of the transferred data, and are used to delimit a field in the example.

Any '[' and ']' characters in sample messages are not part of the transferred data, and are used to delimit optional fields.

The fields in the examples may contain a text description, or hexadecimal data.

Hexadecimal data always uses the "0x" prefix.

The <break> is a single octet with a value of 0xC0.

The first octet after a <break>, is a Routing-PID (RPID). The RPID identifies how the frame data is used. For the LIPE-SL the RPID is 128 (0x80) or 1 to 12 (0x01 to 0x0C).

All octets after the RPID are frame data, until another <break>.

Any occurrence of 0xC0 in the frame data is substituted with 0xDB 0xDC.

Any occurrence of 0xDB in the frame data is substituted with 0xDB 0xDD.

The 0xC0 and 0xDB substitutions allow each octet of frame data transferred to be any value from 0 to 255 (0x00 to 0xFF).

The frame data portion of a Link Frame has no length field.

There is no time limit between octets, as a general rule. The exception is command information sent to PID 128 (0x80) on the LIPE, which must be received within 250 milliseconds, starting with the command opcode. A timeout results in data being dropped until the next <break>.

When TCP or UDP message is received and validated by a LIPE process, the payload of the message is forwarded to the LIPU using the PID of the process. The TCP/UDP message is received by the TCP/UDP process set up by the LIPU. There can be up to 12 user processes on the LIPE-SL, which use PIDs 1 to 12. Each received network frame generates one link frame. The maximum frame size of is described later for each process type.

When the LIPE receives a link frame from the LIPU with RPID 1 to 12, the data is sent to the network via the TCP/UDP process set up by the LIPU. All the handshaking and error checking with the network are handled by the LIPE. The only overhead to the block of data being sent is 2 break characters and a PID. The data can be sent in small frames or one large frame. It may be preferable to employ a sharing scheme on the LIPU when large frames are being generated, so that the data each active process is interleaved.

A TCP process with default configuration settings, will forwards data to the network as fast as possible potentially in many small frames. The process can be set to “frame mode” which waits for a <break> or at 536 characters before sending the data to the network. Some process types always have frame mode set.

A UDP process always waits for the LIPU to send the closing <break>, before sending data to the network. The maximum size of a UDP frame is described later in the document.

Any Link Frame with RPID 128 (0x80) contains control messages being transferred between the LIPE and LIPU. These control messages include commands, command acknowledgments, or event information. The process at each end of this connection is called the Link Manager in this document.

When sending multiple frames only a single <break> is required between the frames. The LIPE always sends a <break> at the end of a Link Frame, and another <break> when the next Link Frame is initiated. Sending the extra <break> has little impact on the overall performance, and is useful during testing.

Sending multiple breaks with no information in between is permitted, and is simply dropped by the LIPE.

Sending <break><rpid><break> has no frame data, and is dropped by the LIPE.

When the LIPE starts, all received characters are ignored until the first <break> is received.

When the LIPE receives a Link Frame with a RPID that is not valid or active, the content of the frame is dropped by the LIPE.

Bracketing the frame data with 0xC0 and the substitution rules described above is derived from the SLIP protocol. The Serial Line Internet Protocol (SLIP) is a transport mechanism for TCP/IP packets over a serial interface. The original definition can be found in RFC1055. The SLIP protocol is a simple mechanism to bracket blocks of binary information that works well for the LIPE Link Frame. Rather than being used to transfer TCP/Packets, LIPE uses SLIP to transfer LIPE commands, command acknowledgments, event messages, and the payload of a TCP or UDP network frame.

NOTE: Once a connection is opened, network data could be received at any time. In addition to network data, the LIPE may generate events to report a change or completion of a request. The LIPU parsing of data must deal with the chance that information being received can be interleaved between several PIDs. This interleaving becomes interesting when waiting for command response.

8.3 COMMANDS AND ACKNOWLEDGMENTS

Managing the LIPE from the LIPU is performed using a set of commands sent to LIPE PID 128. The commands are transferred as the data portion of a Link Frame.

The LIPE is managed from the LIPU using the following message types:

1) Command Messages sent from the LIPU to the LIPE

2) Command Message Acknowledgment sent from the LIPE to the LIPU

3) Event Messages sent from LIPE to the LIPU

Commands are sent to the LIPE to set up and configure the Processes to be run.

Each command sent to the LIPE generates a Command Acknowledgment.

Event Messages are formatted the same as a Command Acknowledgment to simplify parsing.

All messages consist of an opcode, a length, and a payload. The opcode value is from 1..255. The length value is from 0..127. The payload consists of 0 to 127 octets of data as required by the opcode.

Until a Command Acknowledgment is received, the next command is not to be issued.

All Commands are sent to the LIPE using Routing-PID (RPID) 128.

All Command Acknowledgments are reported to the LIPU using RPID 128.

All Event Messages are sent to PID 128, the same as a Command Acknowledgment.

All integer values are transferred big endian, unless otherwise specified.

A Command, Command Acknowledgment, or Event are transferred using the Link Transport

Framing described earlier in the document.

Command, Command Acknowledgment, or Event are transferred using the Link Framing described earlier in the document.

Basic Message Structure:

```
<opcode> <length> [payload]
```

Command Message + Link Frame:

```
<break> <rpid> <opcode> <length> [payload] <break>
```

Acknowledgment/Event Message + Link Frame:

```
<break> <rpid> <opcode> <0x80 + length> [payload] <break>
```

The <opcode> is a 1-octet field, and is a command identifier. The <length> is a 1-octet field, with a permitted range of 0 to 127. The MSB is always 0 for a message sent to the LIPE from the LIPU, hence all commands. The MSB is always 1 for a message sent from the LIPE to LIPU, hence all Command Acknowledgment and Event Messages.

The minimum payload length is checked for each opcode type. Any extra data sent in the payload is ignored.

When using Framing with SLIP, it is important to note that the <length> does not include any extra characters added by the SLIP substitutions.

The [payload] field is optional based on the generic structure. There is always a minimum of 1 payload octet in command response or event.

The first payload octet in a Command Acknowledgment or Event Messge is always a status field.

The second payload octet is typically a PID field. This PID identifies which process made the original request. The LIPE echoes back the PID from the Command to the Command Acknowledgment. As all Command Acknowledgments and Event Messages use a RPID of 128 for the framing, this is required to report information to a specific process.

Upon the LIPE receiving a Command opcode, the remainder of a Command must be received within 250 milliseconds, else a timeout error is reported as an Event Message.

Any error detected by the LIPE while parsing a Command results in no further parsing until a Break is received.

Command Acknowledgment Status	Status Description
0 (0x00)	no error
1 (0x01)	generic error
2 (0x02)	command parsing
3 (0x03)	process number invalid
4 (0x04)	process number out of range
5 (0x05)	wrong process type
6 (0x06)	wrong process state
7 (0x07)	parameter invalid
8 (0x08)	parameter too small
9 (0x09)	parameter too big
10 (0x0A)	configuration incomplete
11 (0x0B)	command not supported
12 (0x0C)	add process failed
13 (0x0D)	command opcode invalid
14 (0x0E)	command length too big
15 (0x0F)	command payload too long
16 (0x10)	command payload too short
17 (0x11)	command timeout
18 (0x12)	command index invalid
19 (0x13)	busy, request rejected
20 (0x14)	insufficient resources
End of Table	

8.4 EVENT MESSAGES

Unsolicited messages from the LIPE, report LIPE changes to the LIPU. The Event Messages include information from specific processes as well as generic system information

A LIPE Event Message is formatted the same as a Command Acknowledgment, with a minimum payload length of 2. The <status> field is redefined as the <event> field to report a specific event number.

Event Messages always use RPID 128.

All Event Messages sent from the LIPE to LIPU use the following format:

```
<opcode> <0x80 + length> <event> <pid> [event specific data]
```

The <opcode> is 0x10 for a LIPE Event Message.

The <length> is typically 0x82, as there is a minimum of two standard octets in the payload. The first octet is <event> the second <pid>.

The <event> field reports the specific event code.

The <pid> field reports which process is to be notified of the event. The message is always routed to process 128, however final recipient is generally another process. The <pid> is generally that of the network process managing the TCP or UDP connection.

Events are used to report the result of operations such as a DNS lookup, which will have been started by a LIPU process other than PID 128. The RPID of the event must always be 128 to avoid being confused with network traffic. Example: If a TCP Server connection is created and assigned a PID of 4 by the LIPE, any TCP Stack event issued for that connection is reported with <pid> set to 4 in a related Event message.

<table>
<tr><th>Event Field</th><th colspan="2">Event Description</th></tr>
<tr><td>0 (0x00)</td><td colspan="2">Name: “Null Event”
Null event, everything is ok. Reported if the LIPE receives a command opcode of 0x10. The command payload is ignored by the LIPE. This behavior can be used to check if the LIPE module is present. A PID of 128 is always reported in the event message payload.
Length field is always 2.</td></tr>
<tr><td>1 (0x01)</td><td colspan="2">Name: “Open”
A user process is now opened. The TCP/IP Stack is in listen state. This event occurs after an Open command has taken effect.
Length field is always 2.</td></tr>
<tr><td rowspan="6">2 (0x02)</td><td colspan="2">Name: “Connected”
A remote system has established a connection to a TCP Server or TCP Client process in the LIPE.
Length field is always 8.</td></tr>
<tr><td>Payload Offset</td><td>Data</td></tr>
<tr><td>0</td><td>0x02 – the Event/Status field</td></tr>
<tr><td>1</td><td>PID of the user process (1..12)</td></tr>
<tr><td>2..5</td><td>IP Address of the remote system</td></tr>
<tr><td>6..7</td><td>TCP Port of the remote system</td></tr>
<tr><td>3 (0x03)</td><td colspan="2">Name: “Closed”
A user process is now close. A TCP/IP connection is now closed.
Length field is always 2.</td></tr>
<tr><td>4 (0x04)</td><td colspan="2">Name: “TCP Reset”
A TCP connection has received a Reset (RST Flag), the connection is now being shut down. Once the TCP connection has been shut down by the LIPE, a close event is</td></tr>
</table>

<table>
<tr><th>Event Field</th><th>Event Description</th></tr>
<tr><td></td><td>generated.
Length field is always 2.</td></tr>
<tr><td>5 (0x05)</td><td>Name: “Invalid TCP Connection”
An Invalid TCP connection attempt has been made. A TCP connection attempt is initiated with a packet, which has the SYN Flag bit set in the Flag field of the TCP header. This event reports a TCP SYN was received with the stack in the wrong state.

When a single LIPE process is handling a TCP Port, this event occurs if a host which had established a connection, issues another SYN request without having closed the connection. The host must be using the original TCP source port. The default LIPE action is to issue a Keep Alive. If the Keep Alive fails, the connection is closed. The "Process-Cfg-Set" command can disable the default action, leaving corrective action to the LIPE.

When a multiple LIPE processes are handling a TCP Port, this event occurs when all processes appear to be busy. The first busy process receives the packet and issues the event. To be deemed busy the process must be in the TCP Connected state and the process has the Keep Alive Timeout disabled. The default LIPE action is to issue a Keep Alive. If the Keep Alive fails, the connection is closed. The "Process-Cfg-Set" command can disable the default action, leaving corrective action to the LIPE.

If there is no matching process to handle the packet with the SYN, it is dropped with no notification as a default action. The "Module-Cfg-Set" Option 14 can be used to enable reporting of dropped TCP packets.

Length field is always 8.
This event only occurs only for a TCP based process.
<table>
<tr><th>Payload Offset</th><th>Data</th></tr>
<tr><td>0</td><td>0x09 - the Event/Status field</td></tr>
<tr><td>1</td><td>PID of the network process (1..12).</td></tr>
<tr><td>2..5</td><td>IP Address of the remote system</td></tr>
<tr><td>6..7</td><td>TCP Port of the remote system</td></tr>
<tr><td>8</td><td>Action taken by the LIPE.

1= No action taken by the LIPE, as the LIPU disabled the automated handling. The LIPU must decide what action should be performed. A connection uses the automated rules by default.

2 = The IP Address and source TCP port of the host system matches that which initially established the connection. The LIPE assumes the host has been reset, and is starting over. The connection is automatically shut down, so the normal establish connection rules can be applied.

3 = Automatically determine if the original host is still present by sending a TCP Keep alive. If the Keep Alive is not responded to the connection is automatically closed, a close event is issued.

4= Data was about to be sent by the LIPE, so no TCP Keep Alive needed to be sent. If the data about to be sent is not acknowledged, the connection is automatically closed as a standard action.</td></tr>
</table>
</td></tr>
<tr><td>6 (0x06)</td><td>Name: “Open Failure”
Open failure by TCP Client or TCP Server.
Length field is always 3.
<table>
<tr><th>Payload Offset</th><th>Data</th></tr>
<tr><td>0</td><td>0x09 - the Event/Status field</td></tr>
<tr><td>1</td><td>PID of the network process (1..12).</td></tr>
<tr><td>2</td><td>Reason for the Open failure.

1 = No TCP connect response from the remote system within a one second timeout from final successful ARP Request. After a TCP connection failure, another ARP Request is issued. If the ARP Request succeeds the TCP connection attempt is made up to a limit of 5 attempts.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Event Field</th><th colspan="2">Event Description</th></tr>
<tr><td></td><td></td><td>2 = ARP retry limit reached (5 attempts made, 1 every 10 seconds). The remote system is not responding to ARP Requests.

3 = No TCP/UDP transmit buffer available.

4 = DNS lookup failed, no IP address could be found. The DNS request is issued up to 5 times, with a retry interval of 5 seconds.

5 = No DNS support present in LIPE. This should never occur, as the DNS process is activated as a default.

6 = Remote IP Address is invalid (0.0.0.0).

7 = No Gateway defined, the ARP Request could not be sent.

8 = No DNS Server Address defined.

9 = No internal resources for the connection.</td></tr>
<tr><td>7 (0x07)</td><td colspan="2">Name: “TCP Keep Alive Received”
Received a TCP Keep Alive. This event only occurs for a TCP based process. The network process PID is reported.
Length field is always 2.</td></tr>
<tr><td>8 (0x08)</td><td colspan="2">Name: “TCP Keep Alive Sent”
The requested TCP Keep Alive has been sent.
This event only occurs for a TCP based process.
The network process PID is reported.
Length field is always 2.</td></tr>
<tr><td>9 (0x09)</td><td colspan="2">Name: “DNS Lookup Complete”
An LIPU requested DNS Lookup was completed. If the IP Address is 0, the DNS lookup failed. The DNS request is issued 5 times maximum, with a 5 second retry interval.
Length field is always 7.</td></tr>
<tr><td></td><td>Payload Offset</td><td>Data</td></tr>
<tr><td></td><td>0</td><td>0x09 - the Event/Status field</td></tr>
<tr><td></td><td>1</td><td>PID of the requesting process.</td></tr>
<tr><td></td><td>2</td><td>DNS lookup status
0=ok
3=task number invalid
8=parameter too small
9=parameter too big
11=command not supported (DNS process is not active)
17=command timeout
19=busy, request rejected

Note: The status 11 will not occur on the LIPE-SL, as the DNS process always present in this module.

Note: The status 11 occurs if the DNS server could not be contacted, or if the DNS Address in the network configuration is 0.0.0.0. When the DNS Address is 0.0.0.0 no network is attempted.</td></tr>
<tr><td></td><td>3..6</td><td>IP Address of the remote system</td></tr>
<tr><td>10 (0x0A)</td><td colspan="2">Name: “System Reset”
A LIPE reset has occurred; at this moment only the default LIPE processes are active. The LIPU can now configure the LIPE. An unexpected LIPE will reset result in this event. A PID of 128 is always reported.
Length field is always 2.</td></tr>
<tr><td>11 (0x0B)</td><td colspan="2">Name: “System Reset Pending”
System reset pending. A reset request has been received from the LIPU and has not been acted on just yet. A PID of 128 is always reported.
Length field is always 2, no extra payload data.</td></tr>
<tr><td>12 (0x0C)</td><td colspan="2">Name: “Link Data Flushed”
Data received from LIPU has been flushed.
Length field is always 3.
Generation of this event must be enabled using Module-Cfg-Set.</td></tr>
</table>

Event Field	Event Description	
	Payload Offset	Data
	0	0x0C - the Event/Status field
	1	PID of 128 always reported
	2	Reason for data being flushed. 1 = Process not defined in LIPE. 2 = Process not accepting data. 3 = Process not permitted, RPID of 0 detected. 4 = TCP Process not connected
13 (0x0D)	Name: “UDP TX Done” The last UDP transmit has been completed.	
	Payload Offset	Data
	0	0x0D – the Event/Status field
	1	PID of the requesting process.
	2	Status of the transmit request: 1 = Success 2 = Failed, ARP could not resolve the IP address. 3 = Failed, Gateway Address not set in Network Configuration
14 (0x0E)	Name: “Network Connection” Change to Network Connection State (optional). This event is not permitted as a default, it must be enabled using "Module-Cfg-Set" Option 16. Length field is always 3.	
	Payload Offset	Data
	0	0x0E - the Event/Status field
	1	PID of 128 always reported.
	2	Network connection speed, where 0 = No connection 1 = 10BASE-T, half duplex 2 = 10BASE-T, full duplex 3 = 100BASE-TX, half duplex 4 = 100BASE-TX, full duplex
15 (0x0F)	Name: “TCP Packet Dropped” A TCP Packet was dropped as could not be supported by any available process (optional). This event is not permitted as a default; it must be enabled using "Module-Cfg-Set" Option 14. Length field is always 17.	
	Payload Offset	Data
	0	0x0F - the Event/Status field
	1	PID of 128 always reported.
	2..5	IP Address of the remote system
	6..7	Source TCP Port
	8..9	Destination TCP Port
	10	TCP Flags
16 (0x10)	Name: “Network Configuration” Report the Network Configuration parameters now in use by the LIPE. Length field is always 20.	
	Payload Offset	Data
	0	0x10 - the Event/Status field
	1	PID of 128 always reported.
	2	Network Configuration status: 1 = Using a static configuration. 2 = The configuration has been set by DHCP. 3 = The DHCP allocated configuration has changed by the DHCP Server. The LIPE DHCP is now halted, waiting for a Continue command from the LIPU. The LIPU must shut down Any process using the network prior to sending the Continue command. Once the Continue is sent, the LIPE starts to use the new configuration. 4 = The DHCP configuration lookup has failed, the LIPE

Event Field	Event Description	
		keeps trying automatically. The event is to let the LIPU know there is a problem, and that LIPE is still trying. This event repeats every few minutes until a configuration is acquired. 5 = The IP Address in the static configuration is 0.0.0.0, which is not allowed. User processes which interact with the network are not allowed to be opened by the LIPE. 6 = The lease on the DHCP assigned configuration has expired, and a new configuration could not be acquired. The LIPE DHCP is now halted, waiting for a Continue command from the LIPU. The LIPU must shut down. Any process using the network prior to sending the Continue command. Once the Continue is sent, the LIPE starts to use the new configuration. Additional information can found in the section describing the DHCP System Process.
	3	Reserved for future use. Always reported as 0.
	4..7	IP Address
	8..11	IP Network Mask
	12..15	Gateway Address
	16..19	DNS Server Address
		NOTE 1: It is permitted to set up and open any number of network connections within the LIPE prior to the Network Configuration being reported as set. With no valid IP address, no network connection is ever allowed to proceed by the LIPE. Each LIPE process that has been "opened" waits for the systems IP address to be set. When a LIPE process is held off waiting for the IP address to be set, no warnings or errors are issued to the LIPU.
End of Table		

9 LIPE COMMANDS

9.1 SUMMARY

This section defines the commands supported by the LIPE. A LIPE command consists of an Opcode, a Length, and a Payload. The following table summarizes the supported commands.

Summary of LIPE Command Opcodes		
Opcode	**Name**	**Description**
16 (0x10)	LIPE-Event	Report LIPE Event

17 0x11)	Create-Process	Create a LIPE process, where the following options are available in the LIPE-SL:
		
		`1 = UDP-UB Process`
		`2 = HTTP-Configuration Server (PID of 129) *`
		`3 = reserved for DHCP Process (PID of 130) *`
		`4 = reserved for DNS Process (PID of 131) *`
		`5 = LDP Process (PID of 132) *`
		`6 = TCP Server Process`
		`7 = TCP Client Process`
		`8 = TCP HTTP Server`
		`9 = UDP-U Process`
		`10 = TCP SST Process`
		`99 = Link Loopback (factory use only)`
		
		The options marked by '*' are system processes created automatically by the LIPE. Options 3 and 4 are not implemented for the Create-Process command in the LIPE-SL. The DHCP process is always active. The DHCP process is control by the Network Configuration.

18 (0x12)	Open-Process	Tell a "created" process to start network operations.
19 (0x13)	Close-Process	Tell a running process to shut down network operations. The configuration of a process is not altered. Internal LIPE resources are not released. The process can be opened to restart network operations.
20 (0x14)	Delete-Process	Remove a process and related resources.
21 (0x15)	Reset-System	Reset the LIPE.
22 (0x16)	Module-Cfg-Set	Set Network Configuration and general LIPE parameters
23 (0x17)	Module-Cfg-Get	Get Network Configuration and general LIPE parameters.
24 (0x18)	Process-Cfg-Set	Set the configuration of a user process.
25 (0x19)	Not Used	Reserved for Get Process Configuration.
26 (0x1A)	Action-Request	Send a special command to a Process
27 (0x1B)	DNS-Lookup	Initiate a DNS lookup.
28 (0x1C)	Base64-Encode	Encode a string using Base64
29 (0x1D	Base64-Decode	Decode a Base64 string.
End of Table		

The Opcode values skip the 0x00 to 0x0F values to make reading the raw data traffic a little easier. There are no undocumented commands in this range.

9.2 OPCODE:0X10 "LIPE EVENT"

```
Event Format: <OP=0x10><L=0x82..0x88><event><pid>[event specific]
```

The LIPE Event opcode is a special case. This opcode is normally only used in a message sent from the LIPE to the LIPU to report a change in the LIPE. These "Event Messages" are sent to LIPU with no prompting by the LIPU.

An "Event Message" is formatted the same as a Command Acknowledgment, which means the LIPU does not need to deal with special parsing rules.

An "Event Message" is sent using a Link Frame with RPID 128.

An "Event Message" payload has 2 octets minimum.

An "Event Message" always has the most significant bit in the length field set.

The first payload octet reports the specific event. The LIPE Events are listed in the "LIPE Event Messages" section of the document.

The second payload octet reports the PID of the process to which initiated the event. If the event is general to the LIPE module, then PID 0x80 (128) is used.

Additional payload octets may be present to provide additional information specific to an event. See the "LIPE Event Messages" section of the document for additional details.

If the LIPE receives a message with the 0x10 opcode, an acknowledgment is generated with a 2 octet payload. The first octet is 0 and the second octet is 0x80 (128). This prompted "event message" can be used to detect if the LIPE is present.

Example of Event - System Reset

Octets	Description
0xC0	<break>
0x80	RPID = 128
0x10	Opcode = 16, LIPE Event
0x82	Length = 3, with MSB set
0x0A	Event = 10, reports a system startup has occurred.
0x80	PID = 128
0xC0	<break>

9.3 OPCODE:0X11 "CREATE-PROCESS"

```
Command Format #1:     <OP=0x11><L=0x01><type>
Command Format #2:     <OP=0x11><L=0x02><type><pid>
Acknowledgment Format: <OP=0x11><L=0x83><status><pid><type>
```

The "Create-Process opcode is used to activate a LIPE system or user process.

The command <length> is either 1 or 2.

The first payload octet defines the process type to be created. The create-process types are:

```
  1  (0x01) UDP-UB Process
* 2  (0x02) HTTP-Configuration-Server (fixed PID of 129)
* 3  (0x03) reserved for DHCP Process (fixed PID of 130)
* 4  (0x04) reserved for DNS Process  (fixed PID of 131)
* 5  (0x05) LDP Process               (fixed PID of 132)
  6  (0x06) TCP Server Process
  7  (0x07) TCP Client Process
  8  (0x08) TCP HTTP Server
  9  (0x09) UDP-U Process
  10 (0x0A) TCP SST Process
  99 (0x63) Link Loopback
```

The process options marked by '*' are system processes created automatically by the LIPE. Options 3 and 4 are not implemented for the Create-Process command in the LIPE-SL. The DHCP process is always active. The DHCP process is control by the Network Configuration.

All system processes run upon creation. All user processes must be issued an "open" command after being created. Prior to a user process being opened, configuration changes can be made.

The second octet of the command payload sets the PID to be used. The permitted range for a user process PID is 1 to 12 (0x01 to 0x0C). If the PID is already allocated, then error 12 is reported in the status field of the acknowledgment.

If the second payload octet is 0, or if the second octet is absent, the LIPE allocates the next available PID in the 1 to 12 range for a user process.

When the LIPE assigns the PID, the value is reported in the acknowledgment. When the LIPU selects the PID, the value from the command is echoed in the acknowledgment.

The second octet of the command payload is ignored for a system process. A system process is always assigned the fixed PID noted in the above table.

The LIPU uses the reported PID for all other commands that apply to this process.

Example of Create-Process for a TCP Server

Octets Description - Command

0xC0 <break>
0x80 RPID = 128
0x11 Opcode = 17, Create-Process
0x01 Length = 1
0x06 Payload[0] = 6, to select "TCP Server"
0x01 Payload[1] = 1, the PID to be used
0xC0 <break>

Octets Description - Acknowledgment

0xC0 <break>
0x80 RPID = 128
0x11 Opcode = 17, Create-Process
0x83 Length = 3, with MSB set
0x00 Payload[0] = 0, Status = ok
0x01 Payload[1] = 1, the PID from the command
0xC0 <break>

9.4 OPCODE:0X12 "OPEN-PROCESS"

```
Command Format:         <OP=0x12><L=0x01><pid>
Acknowledgment Format: <OP=0x12><L=0x82><status><pid>
```

This opcode is used to "open" a process already created and configured.

This opcode puts the process into the normal running state.

TCP and UDP processes create the TCP/IP stack connection for network operation upon the "Open-Process".

Any process configuration changes must take place before the open.

The command <length> is always 1.

The payload contains the PID of the process to be acted on.

Once a process is opened, it can be closed, configuration changes applied, and then opened again.

The command and acknowledgment are sent using a Link Frame with RPID 128.

Example of Open-Process

Octets Description - Command

0xC0 <break>
0x80 RPID = 128
0x12 Opcode = 18, Open-Process
0x01 Length = 1
0x06 Payload[0] = 1, user process #1
0xC0 <break>

Octets Description - Acknowledgment

0xC0 <break>
0x80 RPID = 128
0x12 Opcode = 18, Open-Process
0x82 Length = 2, with MSB set
0x00 Payload[0] = 0, Status = ok
0x01 Payload[1] = 1, user process #1
0xC0 <break>

9.5 OPCODE:0X13 "CLOSE-PROCESS"

```
Command Format:         <OP=0x13><L=0x01><pid>
Acknowledgment Format: <OP=0x13><L=0x82><status><pid>>
```

This opcode is used to "close" a user process already "opened".

For a TCP process, the stack is gracefully shut down and any required internal cleanup is performed.

For a UDP process any required internal cleanup is performed.

The command acknowledgment indicates the close has only been accepted by the LIPE. Event message 3 "Stack is now closed" is sent to the LIPU when the close is completed for TCP and UDP processes.

The process is not deleted, simply made dormant. In this state the process can have configuration changes made, it can be opened again, or it could be deleted.

The “close” is not used for system processes, and results in an error 4 (process number out of range.

The command <length> is always 1.

The payload contains the PID of the process to be acted on.

The permitted PID range is 1 to 12.

The command and acknowledgment are sent using a Link Frame with RPID 128.

Example of Close-Process

Octets Description - Command

0xC0 <break>
0x80 RPID = 128
0x13 Opcode = 19, Close-Process
0x01 Length = 1
0x01 Payload[0] = 1, user process #1

0xC0 <break>

Octets Description - Acknowledgment

0xC0 <break>
0x80 RPID = 128
0x13 Opcode = 19, Close-Process
0x82 Length = 2, with MSB set
0x00 Payload[0] = 0, Status = ok
0x01 Payload[1] = 1, user process #1
0xC0 <break>

9.6 OPCODE:0X14 "DELETE-PROCESS"

```
Command Format:         <OP=0x14><L=0x01><pid>
Acknowledgment Format: <OP=0x14><L=0x82><status><pid>
```

This opcode results in the immediate termination of the specified process.

The process gets no warning or time to shutdown.

If a user process requires time to shut down, then it must be "closed" first. The preferred sequence for any user process (PID less than 128), is to first close it, wait for the "Stack is now closed" Event, and then "delete".

The command <length> is always 1.

The payload contains the PID of the process to be acted on.

Deleting a process with a PID in the range 1..12 releases all resources used by the process immediately.

There is no graceful shutdown of an open TCP user process, when using the Delete-Process command. A TCP user process should first be closed.

The delete does apply to the system processes HTTP-Configuration-Server and the LDP process. Deleting the HTTP-Configuration-Server releases internal resources that can then be used by an additional user process. The LDP process does not use resources that limit the number of user processes. The LDP process can be deleted if the LIPU simply does not want it running, or is implementing an equivalent service of the same UDP port.

The Link Manager, DNS and DHCP process cannot be deleted using this command on the LIPE-SL. Attempting to delete these system processes results in error 1 in command acknowledgment.

Once a process is deleted, it can re-activate using a "create-process" request. Remember that once a process is deleted, any configuration setting that had been set, are of course lost.

The command and acknowledgment are sent using a Link Frame with RPID 128.

Example of Delete-Process

Octets Description - Command

0xC0 <break>
0x80 RPID = 128
0x14 Opcode = 20, Delete-Process
0x01 Length = 1
0x01 Payload[0] =1, user process #1
0xC0 <break>

Octets Description - Acknowledgment

0xC0 <break>
0x80 RPID = 128
0x14 Opcode = 20, Delete-Process
0x82 Length = 2, with MSB set
0x00 Payload[0] = 0, Status = ok
0x01 Payload[1] = 1, user process #1
0xC0 <break>

9.7 OPCODE:0X15 "RESET-SYSTEM"

```
Command Format: <OP=0x15><L=0x00>
Acknowledgment Format: none
```

This opcode resets the LIPE. There is no command acknowledgment. Just prior to the reset taking effect, the LIPE issues Event #11 "System reset pending".

When the LIPE comes out of reset, Event #10 "Reset has occurred" is sent to the LIPU.

The command <length> is always 0.

The command is sent using a Link Frame with RPID 128.

Example of RESET-SYSTEM

Octets Description - Command

0xC0 <break>
0x80 RPID = 128
0x15 Opcode = 21, Reset-System
0x00 Length = 0
0xC0 <break>

9.8 OPCODE:0X16 "MODULE-CFG-SET"

```
Command Format:  <OP=0x16><L=variable><pid><index><data, variable length>
Acknowledgment Format: <OP=0x16><L=0x83><status><pid><index>
```

This opcode is used to set the LIPE Module configuration. The configuration can be set in a block or as individual elements.

The command <length> varies depending on the configuration item being set.

The first payload octet in the command is the PID for the process making the request. Although a PID is not critical for the request, it is reported in the acknowledgment. The PID in the request is echoed in the acknowledgment. The receiving process on the LIPU may need to know which process initiated the command.

The second payload octet in the command is the requested configuration item, as specified by the <index> field.

The remainder of the payload contains the data for the item being set. All items, which do not fit in one octet, are big endian.

The command and acknowledgment are sent using a Link Frame with RPID 128.

Module configuration items:

```
 1:  Module Configuration        Sets items 2 through 9 in single report
 2:  IP Address                  Requires a 32-bit integer
 3:  IP Subnet Mask              Requires a 32-bit integer
 4:  Gateway Address             Requires a 32-bit integer
 5:  DNS Server IP               Requires a 32-bit integer
 6:  HTTP Configuration Port     Requires a 16-bit integer
 7:  DHCP Mode                   Requires a 8-bit integer, 0=off, else on
 8:  Device Name                 Requires ASCII string
 9:  Administrator Password      Reports ASCII string
10:  RFU                         Reserved for future use
11:  RFU                         Reserved for future use
12:  Device Information String   Accepts up a 80 character string.
13:  RFU                         Reserved for future use
14:  Default Network Setup       See text.
15:  RFU                         Reserved for future use
16:  Network Status Events       Requires an 8-bit integer (*)
17:  Dropped TCP Packet Events   Requires an 8-bit integer (*)
18:  Permit Rx Flush Event       Requires an 8-bit integer (*)
19:  SST Port                    Requires a 16-bit integer
20:  TCP Timeout To Close        Requires a 16-bit integer (1000..65535 mS)
21:  TCP Timeout To Connect      Requires a 16-bit integer (5000..65535 mS)
22:  TCP Time Wait Timeout       Requires a 16-bit integer (250..65535 mS)
23:  TCP Close Request Delay     Requires a 16-bit integer (0..65535 mS)
```

The three items marked with (*) are volatile, hence they always return to a default value after a reset. All other items are stored in non-volatile memory just before the command acknowledgment is issued.

The acknowledgment does not echo back the data. The acknowledgment is always a fixed length.

Changes to the IP Address, IP Subnet Mask, HTTP Configuration Port, DHCP Mode, and SST port take effect only on a module reset. Reading back these parameters reports the value written, not that which is active. After changing these values it is recommended that the LIPE be reset. All other Module Configuration parameters take effect immediately.

Using index 1 (Module Configuration) allows items 2 through 9 to be set in a single message. The Device Name field is a fixed length field of 17 characters. The Device Name can be up to 16 characters, and the remainder of the field is padded with 0x00. The 17'Th character for the Device Name field must be a 0x00. The Administrator Password field is a fixed length field of 9 characters. The Administrator Password can be up to 8 characters, and remainder of the field padded with 0x00. The 9'Th character for the Device Name field must be a 0x00.

The Device Name is an ASCII string of up to 16 characters long. The default Device name is "** ** ** LIPE", where "** ** **" are the last 3 octets of the MAC Address formatted in hexadecimal. The LIPE Device Name is used for the Host Name field in a DHCP transaction (DHCP option 12).

Setting the Device Name using index 8, requires an ASCII string of up to 16 characters. A null character terminates the string, which is not required in the set message. The set message does not require the string be padded to 16 characters.

The Administrator Password is an ASCII string of up to 8 characters long. A null character terminates the string, which is not required in the set message. The set message does not require the string be padded to 8 characters.

The Device Information String is a 80 character string of binary data. The format of the data is application specific. The LAVA Manager can retrieve this string using the LAVA Discovery Protocol. The string is intended as an enhanced Device Identifier, for future customization. The length of data transferred is inherent in the payload length. When the payload length field is less than 3, no Device Information is present.

The Default Network Setup command restores the LAVA manufacturing defaults for items 2 to 9. The LAVA manufacturing defaults are:

```
Device Name             : "** ** ** LIPE"
IP Address              : 192.168.0.35
Subnet Mask             : 255.255.255.0
Default Gateway         : 0.0.0.0
DNS Server IP           : 0.0.0.0
HTTP Configuration Port: 8080
DHCP                    : Disabled
SST port                : 0
```

The Network Status Events option determines if Network Status Event messages are permitted. A value of 0 disables the generation of the event messages. A non-zero value enables the generation of the event messages. The default is 0. This parameter is volatile.

The Dropped TCP Packet Events option determines if an event is generated to inform the LIPE of unsupported TCP packets received by the LIPE. The event contains the IP Address, Source TCP Port, Destination TCP Port, and the TCP Flags. A value of 0 disables the generation of the events. A non-zero value enables the generation of the events. The default is 0. This parameter is volatile.

The "Permit Rx Flush Event" is not active as a default. A value of 1 enables the event generation. A value of 0 disables the event generation. This configuration item is volatile. The event can be useful during early development of a system. This parameter is volatile.

The SST Task TCP Port sets the default value for the SST TCP Process TCP Port. This parameter is non-volatile. If the value is non-zero, the TCP SST Process is enabled by default on power up. Please see the create-process command for additional details on this process. The value is used on power up to determine the default state of the process.

The TCP Timeout To Close (index 20) requires a 16-bit integer, in the range of 1000 to 65535. The units are milliseconds. The value is truncated to an increment of 10. A value lower than 1000, is treated as

1000. The default value is 5000 milliseconds. This is the maximum allowed time for a TCP connection is complete the handshaking when a connection is being closed.

The CP Timeout To Connect (index 21) requires a 16-bit integer, in the range of 5000 to 65535. The units are milliseconds. The value is truncated to an increment of 10. A value lower than 5000, is treated as 5000. The default value is 10000 milliseconds. This timeout is the maximum allowed time for a TCP Client to attempt to connect to a remote system.

The TCP Time Wait Timeout (index 22) requires a 16-bit integer, in the range of 250 to 65535. The units are milliseconds. The value use truncated to an increment of 10. A value lower than 250, is treated as 250. The default value is 250 milliseconds. This is the time spent in the TIME-WAIT state waiting for a final acknowledgment. Until the TCP/IP Stack exists this state, the resources used by a connection are not released. For an embedded system with a limited number of permitted connections, the value of the Times Wait timeout is a tradeoff based on your network characteristics.

The TCP Close Request Delay (index 23) requires a 16-bit integer, in the range of 0 to 65535. The units are milliseconds. The value is truncated to an increment of 10. The default value is 100 milliseconds. This parameter delays the activation of a LIPE Close command. This parameter was added for in-house testing and does not need to be adjusted. The value is locked in per process, when the process is created.

9.9 OPCODE:0X17 "MODULE-CFG-GET"

```
Command Format:        <OP=0x17><L=0x02><pid><index>
Acknowledgment Format: <OP=0x17><L=0x80+'N'><status><pid><index><'N' octets>
```

This command reads the LIPE module configuration. The configuration can be read as a block or as individual elements.

The command <length> is always 2. The acknowledgment length varies depending on the item being read.

The first payload octet in the command is the PID for the process making the request. Although a PID is not critical for the request, it is used in the acknowledgment. The PID in the request is echoed in the acknowledgment. The receiving process on the LIPU may need to know which process initiated the command.

The command's second payload octet is the requested configuration item, as specified by the <index> field.

The acknowledgment <length> varies depending on the configuration item being read.

The command and acknowledgment are sent using a Link Frame with RPID 128.

Module configuration items:

```
 1:  Module Configuration        Reports items 2 through 9 in single report
 2:  IP Address                  Reports a 32-bit integer
 3:  IP Subnet Mask              Reports a 32-bit integer
 4:  Gateway Address             Reports a 32-bit integer
 5:  DNS Server IP               Reports a 32-bit integer
 6:  HTTP Configuration Port     Reports a 16-bit integer
 7:  DHCP Mode                   Reports a 8-bit integer,  0=off, 1=on
 8:  Device Name                 Reports ASCII string with Null terminator
 9:  Administrator Password      Reports ASCII string with Null terminator.
10:  Firmware Revision String    Reports ASCII string with Null terminator
11:  Module Definition           See text.
12:  Device Information String   Reports an 80-character string.
13:  RFU                         Reserved for future use.
14:  RFU                         Reserved for future use.
15:  RFU                         Reserved for future use.
16:  Network Status              Reports an 8-bit integer.
17   RFU                         Reserved for future use.
18   RFU                         Reserved for future use.
19:  SST Task TCP Port           Report a 16-bit integer.
20:  TCP Timeout To Close        Reports a 16-bit integer (1000..65530 mS)
21:  TCP Timeout To Connect      Reports a 16-bit integer (5000..65530 mS)
22:  TCP Time Wait Timeout       Reports a 16-bit integer (250..65530 mS)
23:  TCP Close Request Delay     Reports a 16-bit integer (0..65530 mS)
```

```
Example - Request the IP Address:

  Command  = 0x17 0x02 0x05 0x02
  acknowledgment = 0x17 0x87 0x00 0x0F 0x02 0xC0 0xA8 0x00 0x23

  The command length field is 0x02, the PID is 0x05, and the index is 0x02.
  The acknowledgment information after the length field is <status>, <pid>, <index>,
  and then <data>, where the length of the data depends on the item requested.
  The reported value in the example is 0xC0A80023 (192.168.0.35).
  The example shows the data after SLIP decoding has taken place, therefore
  the 0xC0 for the IP address is not shown as 0xDB 0xDC.
```

The Module Configuration reports the network configuration parameters of the LIPE, which can be modified. The Module Configuration can be retrieved through a bulk read using item 1, or as individual items.

The acknowledgment data field produced for item 1 is a single octet string with no dividers between the fields. The configuration item 1 reports the values for items 2 through 9. The values in the string are always big-endian.

The number of octets used for each value matches the size used for the individual reports.

The Module Configuration reports the following information in the payload:

```
[0]      Status (standard for an LIPE acknowledgment)
[1]      PID of the request (standard for an LIPE acknowledgment)
[2]      Index of the request, which is 1
[3..6]   IP Address, a 32-bit integer
[7..10]  IP Subnet Mask, a 32-bit integer
[11..14] Gateway Address, a 32-bit integer
[15..18] DNS Server IP, a 32-bit integer
[19..20] HTTP Port, a 16-bit integer
[21]     DHCP Mode, an 8-bit integer, 0=off, 1=on
[22..38] Device Name, an ASCII string with Null terminator.
[39..47] Administrator Password, an ASCII string with Null terminator.

Note: The Device Name and Admin Password are fixed length fields.
```

Using index 1 (Module Configuration) allows items 2 through 9 to be read in a single message. The Device Name field is a fixed length field of 17 characters. The Device Name can be up to 16 characters, and the remainder of the field is padded with 0x00. The 17'th character for the Device Name field is always 0x00. The Administrator Password field is a fixed length field of 9 characters. The Administrator Password can be up to 8 characters, and remainder of the field padded with 0x00. The 9'th character for the Device Name field is always 0x00.

The Device Name is an ASCII string of up to 16 characters long. The default Device name is "** ** ** LIPE", where "** ** **" is the last 3 octets of the MAC Address formatted in hexadecimal. The LIPE Device Name is used for the Host Name field in a DHCP transaction (DHCP option 12).

The Firmware Revision String is typically as 22-character null terminated string, therefore a total of 23 characters. The revision string formatting matches the presentation used by the HTTP Configuration Server.

The Module Definition reports the following read-only attributes of the LIPE.
The Module Definition reports the following information in the payload:

```
[0]      Status (standard for an LIPE acknowledgment)
[1]      PID of the request (standard for an LIPE acknowledgment)
[2]      Index of the request, which is 11
[3..8]   MAC Address of the LIPE
[9]      Firmware Version - Major
[10]     Firmware Version - Minor
[11..12] Firmware Version - Test
[13..14] Firmware Release Year
[15]     Release Month (1..12)
[16]     Release Day (1..31)
[17..18] Device Identifier of the LIPE
[19..20] Vendor Identifier of the LIPE
[21..22] System Identifier of the LIPE
[23..26] LIPE Production Date Code
[27..30] LIPE Production PCB Name
```

The Module Definition payload fields 18 and 19 report the LIPE Device Identifier. The LAVA Manager Application uses this value. The Device Identifier for the LIPE will be one of the following values:

Device Identifier	Description
0x33B0	LAVA LIPE-SL with baud rate set to 115200
0x23B0	LAVA LIPE-SL with baud rate set to 57600
0x13B0	LAVA LIPE-SL with baud rate set to 19200
0x03B0	LAVA LIPE-SL with baud rate set to 230400

The Module Definition payload fields 20 and 21 report the LIPE Vendor Identifier, which is 0x1407 for LAVA. This value is used by LAVA for product management.

The Module Definition payload fields 20 and 21 report the LIPE System Identifier. This value is used by LAVA for product management.

The Module Definition payload fields 23 to 27 report the date of manufacture for the module. The first 2 octets are the year, followed by month, then the day.

The Module Definition payload fields 28 to 31 report the product PCB Name, which consists of 4 ASCII characters.

The Device Information String is an 80-character string of binary data. The format of the data is application specific. The LAVA Manager can retrieve this string using the LAVA Discovery Protocol. The string is intended as an enhanced Device Identifier, for future customization. The length of data transferred is inherent in the payload length. No Device Information is present when the payload length field is less than 4.

The Network Status reports an 8-bit integer to indicate the network connection status. The values reported are the same as the Event message "Network Connection State". The possible values reported are:

Network Status	Description
0	No connection
1	10BASE-T, half duplex
2	10BASE-T, hull duplex
3	100BASE-TX, half duplex
4	100BASE-TX, full duplex

The SST Task TCP Port value reports the default value for the SST TCP Process TCP Port. This parameter is non-volatile. If the value is non-zero, the TCP SST Process is enabled by default on power up. Please see the create-process command for additional details on this process. The value is used on power up to determine the default state of the process.

The TCP Timeout To Close (index 20) reports a 16-bit integer, in the range of 1000 to 65530. The units are milliseconds. The default value is 5000 milliseconds. This is the maximum allowed time for a TCP connection is complete the handshaking when a connection is being closed.

The TCP Timeout To Connect (index 21) reports a 16-bit integer, in the range of 5000 to 65530. The units are milliseconds. The value use is truncated to an increment of 10. The default value is 10000 milliseconds. This timeout is the maximum allowed time for a TCP Client to attempt to connect to a remote system.

The TCP Time Wait Timeout (index 22) reports a 16-bit integer, in the range of 250 to 65535. The units are milliseconds. The default value is 250 milliseconds. This is the time spent in the TIME-WAIT state waiting for a final acknowledgment. Until the TCP/IP Stack exists this state, the resources used by a connection are not released. For an embedded system with a limited number of permitted connections, the value of the Times Wait timeout is a tradeoff based on your network characteristics.

The TCP Close Request Delay (index 23) reports a 16-bit integer, in the range of 0 to 65535. The units are milliseconds. The default value is 100 milliseconds. This parameter delays the activation of a LIPE Close command. This parameter was added for in-house testing and does not need to be adjusted. The value is locked in per process, when the process is created.

9.10 OPCODE:0X18 "PROCESS-CFG-SET"

```
Command Format:         <OP=0x18><L=0x02+N><pid><index><'N' octets>
Acknowledgment Format: <OP=0x18><L=0x83><status><pid><index>
```

This command sets the LIPE Process Configuration.

There is no get operation for the Process Configuration.

The command <length> is a minimum of 2. The command length varies depending on the configuration item being set.

The first payload octet in the command is the PID for the process, which is being configured. The second command payload octet is an "index" used to identify which configuration item is being set. The remainder of the command payload contains the configuration data.

The "process-set-cfg" items are:

Index	Description
1	Local TCP/UDP port - used for listen
2	Remote TCP Port - for TCP Client operations
3	Remote IP Address - for TCP Client operations
4	Network Process Attributes (bit field)
5	TCP Keep Alive Timeout
6	TCP Retry Timeout

The acknowledgment length is always 3, with the first payload field being the status, followed by the PID from the command, and then the index from the command.

The command and acknowledgment are sent using a Link Frame with RPID 128.

The Local TCP/UDP Port is (index 1) a 16-bit integer. The valid range is 1 to 65535. This parameter defines on which port a process receives network traffic. The TCP and UDP ports do not overlap.

The Remote TCP Port (index 2) is a 16-bit integer. The valid range is 1 to 65535. This parameter defines on which port a TCP process sends network traffic. This option is only applicable to a TCP Client process.

The Remote System Name (index 3) is an ASCII string of up to 40 characters. The minimum length of the string is 7-characters. The string sent to the LIPE does not require a NULL (0x00) termination character. The first NULL (0x00) is the string sent to the LIPE will be treated as the end of the string. The Remote System Name defines the destination system for a TCP Client process. If the string starts with a numeric character ('0' to '9'), the string is treated as an IP Address using decimal dotted notation. Any other string is treated as a name to be used for an automatic DNS lookup by a TCP Client process. This option is only applicable to a TCP Client process. The LIPE does not support an IP loopback address.

The Network Process Attribute (index 4) is an 8-bit integer. The Network Process Attribute is a general-purpose parameter used to define special features for a network-based process. The 8-bit integer is a bit field.

Network Process Attribute	
Bit	Description
0	Reserved for future use. Always set to 0.
1	Force a TCP Client to used the configured Local TCP Port, with no modifications. Normally the configured value is treated as a base value, which is changed every time a connection re-opened. The TCP server and UDP Server ignore this bit.
2	Report invalid TCP connection attempts to the LIPU, with no automated recovery by the LIPE.
3	Forward data to network only on closing <break>. If the LIPU data exceeds the network payload limit, then the data is also forwarded. The TCP Payload limit is 970 bytes. The UDP payload limit is 494 bytes. This attribute puts the connection into a "frame mode" rather than the default “stream mode”. Normally LIPU data is sent to the network as soon as possible. The TCP HTTP Process and the UDP Process has this attribute always set internally.
4	Must be 0, else an invalid parameter error is reported.
5	Must be 0, else an invalid parameter error is reported.
6	Must be 0, else an invalid parameter error is reported.
7	Must be 0, else an invalid parameter error is reported.
End of Table	

The TCP Keep Alive Timeout (index 5) is an 8-bit integer. A TCP process automatically sends a Keep Alive after a period of inactivity defined by this parameter. If the Keep Alive fails to get a acknowledgment, the connection is closed. This feature allows the LIPE to automatically free up processes, which are no longer part of an active session. This command sets the timeout in increments of 10 seconds, hence a 3 yields 30 seconds. The default timeout is 30 seconds. The maximum value of 255 yields 42.5 minutes. A value of 0 disables the Keep Alive timeout. With no Keep Alive timeouts, the LIPU application must keep track of network connections that appear "dead". As the name implies, this option only applies to a TCP process. The standard TCP Retry rules apply to the Keep Alive message.

The TCP Retry Timeout (index 6) is an 8-bit integer. A TCP process retries a data operation every second by default. The time out can be set to 1, 2, 3, 4, or 5 seconds with this option where a value of 1 is active by default. Any other value results in an error. The TCP Stack reties an operation up to 15 times before declaring a failed attempt. The Keep Alive always repeat at half the TCP Retry Timeout.

9.11 OPCODE:0X1A "ACTION-REQUEST"

```
Command Format:         <OP=0x1A><L=0x02><pid><index>
Acknowledgment Format: <OP=0x1A><L=0x83><status><pid><index>
```

This opcode is used to send an Action Request to a LIPE process.

The first payload octet in the command is the PID for the process that requested the action. The second command payload octet is an "index" used to identify which action is being issued. The PID is reported in the command acknowledgment and then event used to report action completion.

The command acknowledgment length is always 3, with the first payload field being the status, followed by the PID from the command, and then the index from the command.

The command and acknowledgment are sent using a Link Frame with RPID 128.

An index of 1 requests a "Sent TCP Keep Alive" action. This action only applies to a TCP Client process. This action requests the process defined sent a TCP Keep Alive. When the requested TCP Keep Alive is issued, Event Message #8 "TCP Keep Alive has been sent" is reported to the LIPU. The <pid> in the command must be for a valid user process.

An index of 2 requests the DHCP Process “continues” after a configuration change or a lease timeout. When the Network Configuration event message is received with a Network Configuration Status of 3 or 6, the LIPU has to deal with a loss of permission to use the previous network configuration. Either the lease on the configuration has expired or the renewed configuration resulted in a change. The DHCP Process halts until the LIPU issues the “continue”. The LIPE automatically closes all TCP and UDP User Processes. Additional details can be found in the DHCP System Process description. The <pid> in the command must be 130 (0x82).

9.12 OPCODE:0X1B "DNS LOOKUP"

```
Command Format:         <OP=0x1B><L=0x01+'N'><pid><'N' octets>
Acknowledgment Format: <OP=0x1B><L=0x82><status><pid>
```

This opcode is used to request a DNS lookup for a specified System Name.

The first payload octet in the command is the PID for the process that made the request. The remainder of the command payload contains a null terminated ASCII string. The ASCII string is the Domain Name being searched for. The PID from the command is reported in the command acknowledgment and the related event. The PID in the request is echoed in the response.

The maximum length of the DNS Name (the ASCII string) is 40 octets.

The minimum length of the string is 7 octets.

A command acknowledgment confirms the command was accepted, not that the action is completed.

When the DNS Lookup is complete, Event #9 "DNS Lookup Completed" is reported to the LIPU. This event also reports the IP Address returned from the DNS lookup. When more than 1 address is reported in the DNS network response, only the result from the first answer field is reported.

The “DNS Lookup Complete” is also generated when the command reports an initial error.

The DNS lookup status will be one of the following:

```
 0  (0x00) no error
 3  (0x03) process number invalid, hence the PID is invalid
 8  (0x08) parameter too small
 9  (0x09) parameter too big
11  (0x0B) command not supported, no DNS Process
17  (0x11) command timeout (only used by event message)
19  (0x13) busy, request rejected
```

The LIPE has a single DNS process that must be shared. The first process to initiate a DNS lookup owns the DNS Process until it completes. If the DNS process is busy, it is up to the LIPU to implement a retry policy.

The command and acknowledgment are always sent using RPID 128.

Note: When a TCP Client process is using DNS to resolve the remote system name, the process automatically waits for the shared resource. In this case the TCP Client Open event is held off until the DNS lookup is completed.

Note: The PID in the command is echoed in the response. The PID had no effect on the DNS lookup operation. The PID can be any active LIPE process. The PID could be 128 (0x80) for the Link Manager or even 131 (0x83) for the DNS process.

9.13 OPCODE:0X1C "BASE64-ENCODE"

```
Command Format:        <OP=0x1C><L=0x01+'N'><pid><'N' octets>
Acknowledgment Format: <OP=0x1C><L=0x82+’X’><status><pid><’X’ octets>
```

This opcode is used to perform encode a string in BASE64. The maximum input string length is 90. The maximum result string is 120. The input and output strings are null terminated ASCII. The null characters are included in payload.

9.14 OPCODE:0X1D "BASE64-DECODE"

```
Command Format:         <OP=0x1D><L=0x01+'N'><pid><'N' octets>
Acknowledgment Format: <OP=0x1D><L=0x82+’X’><status><pid><’X’ octets>
```

This opcode is used to decode a string that is in BASE64. The maximum input string length is 120. The maximum result string is 90. The input and output strings are null terminated ASCII. The null characters are included in payload.

10 TCP SERVER EXAMPLE

The following example demonstrates how to use the LIPE command to create a single TCP Server process and have it accept data from the network, which is then sent to the LIPU. The examples are shown as a Link Frame with SLIP break characters included.

1) LIPE Command: Create a TCP Server

The LIPU issues a command with opcode 0x11 and a payload containing 0x06 to the LIPE to create the TCP Server. For this example the “create” command is going to have the LIPE assign the PID for the User Process.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x11   Opcode = 17, which is “Create-Process”
 0x01   Length = 1
 0x06   Payload[0] = 6, to select "TCP Server"
 0xC0   <break>
```

2) LIPE Response: Report success and the assigned PID

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x11   Opcode = 17, which is “Create-Process
 0x82   Length = 2, with the MSB set
 0x00   Payload[0] = 0, Status = ok
 0x01   Payload[1] = 1, PID #1 assigned
 0xC0   <break>
```

he LIPE has activated a TCP Server with a default TCP Port of 4098. The default TCP Port assigned is 4099 - PID. This can be set to an application specific value using "Process-Cfg-Set". For this example the default port is used.

The LIPE has dynamically assigned PID #1 to this process.

3) LIPE Command: Send command to Open the process:

As the default process configuration is to be used, the TCP Server is now “opened”. Once opened, the process will be in the TCP Listen state waiting for a network connection on TCP Port 4098.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x12   Opcode = 18, which is "Open-Process"
 0x01   Length = 1
 0x01   Payload[0] = 1, to specify PID #1
 0xC0   <break>
```

4) LIPE Response: Report success of the Open.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x12   Opcode = 18, which is "Open-Process"
 0x82   Length = 2, with the MSB set
 0x00   Payload[0] = 0, Status = ok
 0x01   Payload[1] = 1, PID #1
 0xC0   <break>
```

The acknowledgment indicates the TCP Server process has started and is now attached to the network. Configuration changes such as setting the TCP Port cannot be made after an Open.

6) LIPE Event:

The LIPE reports Event #1 once the stack enters the Listen state for the TCP Server process.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x10   Opcode = 16, which is an "LIPE Event"
 0x82   Length = 2, with the MSB set
 0x01   Payload[0] = 1, Event Field = Stack now in listen state.
 0x01   Payload[1] = 1, User Process #1 is the source of the event
 0xC0   <break>
```

5) Using any convenient test program, open a TCP connection to port 4098. Either Hyperterminal or the Open Source TeraTerm programs are good options.

6) LIPE Response:

When the remote host connected to the TCP Server, the "Stack in now connected" event was sent from PID #1. The extra data on the payload reports the IP Address of the connecting system and the TCP Port it is using.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x10   Opcode = 16, which is an "LIPE Event"
 0x88   Length = 8, with the MSB set
 0x02   Payload[0] = 2, Event Field = Stack is now connected.
 0x01   Payload[1] = 1, PID - Process #1 is the source of the event
 0xDB   Payload[2] = 219, IP Address octet 3, see following text
 0xDC   Payload[3] = 220, IP Address octet 3, see following text
 0xA8   Payload[4] = 168, IP Address octet 2
 0x00   Payload[5] = 0,   IP Address octet 1
 0x87   Payload[6] = 135, IP Address octet 0
 0x06   Payload[7] = 6,   TCP Port, upper 8 bits
 0x9D   Payload[8] = 157, TCP Port, lower 8 bits
 0xC0   <break>
```

The IP Address in this case was 192.168.0.135, which is 0xC0 0xA8 0x00 0x87 in hexadecimal. The 0xC0 was expanded to 0xDB and 0xDC as per the rules of SLIP.

The length field refers to the number of octets in the payload when there are no slip substitutions present.

The TCP Port of the remote system is 0x069D (1693).

The receiving process must check for the 0xDB SLIP escape character for every octet in the Link Frame. The LIPU architecture should implement a layering scheme to deal with the transport issues, and present a "de-slipped" frame to the application. How the LIPU deals with "de-slipping", is not within the scope of this document.

7) Send "Hello" to TCP Port 4098 from the Terminal Application.

8) LIPE to LIPU Link Frame:

Assuming a single TCP frame is sent to the LIPE with "Hello" in it, the following Link Frame is sent from the LIPE to LIPU. If the TCP frame contained 1000 characters, the Link Frame sent to the LIPU would be 0xC0 0x01 <the 1000 characters> 0xC0.

```
Octets Description
------ -----------
 0xC0   <break>
 0x01   RPID = 1, the frame data is from process #1
 0x48   'H'
 0x65   'e'
 0x6C   'l'
 0x6C   'l'
 0x6F   'o'
 0xC0   <break>
```

Any Link Frame information received by the LIPE from the LIPU in the same format, is simply forwarded to the network on the appropriate TCP port.

Until the connection is closed, the LIPE will simply forward Link Frame data from the LIPU to the network, and any received network data to the LIPU.

The LIPU can monitor the state of the TCP Stack by checking for Event Messages. If the TCP connection from 192.168.0.135 is closed, the Event #3 "Stack is now closed" is sent to the LIPU.

Simple state machine in the LIPU can track the state of the connection to know when data can be expected or sent to the network.

11 UDP PROCESS EXAMPLE

The following example demonstrates how to use the LIPE command to create a single UDP Process and have it accept data from the network, which is then sent to the LIPU. The examples are shown as a Link Frame with SLIP break characters included.

1) LIPE Command: Create a UDP Process

The LIPU issues a command with opcode 0x13 and a payload containing 0x09 to the LIPE to create the UDP Process. For this example the “create” command is going to request the use of PID #2, rather than having the LIPE assign a value. This example assumes that User Process #2 does not already exist.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128, send Link Frame to the LIPE Link Manager
 0x11   Opcode = 17, which is “Create-Process”
 0x02   Length = 2
 0x09   Payload[0] = 9 to select "UDP Process"
 0x02   Payload[1] = 2, request use of PID #2
 0xC0   <break>
```

2) LIPE Response: Report success and the assigned PID

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x11   Opcode = 17, which is “Create-Process
 0x82   Length = 2 with the MSB set
 0x00   Payload[0] = 0, Status = ok
 0x02   Payload[1] = 2, PID #2 assigned
 0xC0   <break>
```

The LIPE has activated a UDP Process with a default UDP Port of 4097. The default UDP Port assigned is 4099 - PID. This can be set to an application specific value using "Process-Cfg-Set". For this example the default port is used.

The LIPE now uses PID #2 to identify this process.

3) LIPE Command: Send command to Open the process:

As the default process configuration is to be used, the UDP Process is now “opened”. Once opened, the process will be attached to the TCP/IP Stack and listen on UDP Port #4097.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x12   Opcode = 18, which is "Open-Process"
 0x01   Length = 1
 0x02   Payload[0] = 2, to specify PID #2
 0xC0   <break>
```

4) LIPE Response: Report success of the Open.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x12   Opcode = 18, which is "Open-Process"
 0x82   Length = 2, with the MSB set
 0x00   Payload[0] = 0, Status = ok
 0x02   Payload[1] = 2, PID #2
 0xC0   <break>
```

The acknowledgment indicates the UDP Process has started and is now attached to the network. Configuration changes such as setting the UDP Port cannot be made after an Open.

5) LIPE Event:

The LIPE reports Event #1 once the stack enters the Listen state for the UDP Process.

```
Octets Description
------ -----------
 0xC0   <break>
 0x80   RPID = 128
 0x10   Opcode = 16, which is and "LIPE Event"
 0x82   Length = 2, with the MSB set
 0x01   Payload[0] = 1, Event Field = Stack now in listen state.
 0x02   Payload[1] = 2, User Process #2 is the source of the event
 0xC0   <break>
```

6) LIPE to LIPU Link Frame:

Assuming a single UDP frame is sent to the LIPE with "Hello" in it, the following Link Frame is sent from the LIPE to LIPU.

```
Octets Description
------ -----------
 0xC0   <break>
 0x02   RPID = 2, the frame data is from User Process #2
 0xDB   [0] = 219, IP Address octet 3, see following text
 0xDC   [1] = 220, IP Address octet 3, see following text
 0xA8   [2] = 168, IP Address octet 2
 0x00   [3] = 0,   IP Address octet 1
 0x87   [4] = 135, IP Address octet 0
 0x0F   [5] = 6,   TCP Port, upper 8 bits
 0xA0   [6] = 157, TCP Port, lower 8 bits
 0x48   [7] = 'H'
 0x65   [8] = 'e'
 0x6C   [9] = 'l'
 0x6C   [10]= 'l'
 0x6F   [11]= 'o'
 0xC0   <break>
```

The IP Address in this case was 192.168.0.135, which is 0xC0 0xA8 0x00 0x87 In hexadecimal. The 0xC0 was expanded to 0xDB and 0xDC as per the rules of SLIP.

The UDP Port of the remote system in this example is 0x0FA0 (4000).

The receiving process must check for the 0xDB SLIP escape character for every octet in the Link Frame. The LIPU architecture should implement a layering scheme to deal with the transport issues, and present

a "de-slipped" frame to the application. How the LIPU deals with "de-slipping" is not within the scope of this document.

As UDP is connectionless, the data forwarded to the LIPU needs to report the IP Address and Port of the remote system. Any UDP message sent by the LIPU to the Network (hence LIPE) must inject the IP Address and destination UDP Port into the message.

When generating the UDP frame for the Network, the LIPE knows the UDP Source Port for this connection. The UDP Source Port was established before the Process, hence connection, was “opened”.

12 REVISION HISTORY

Revision	Date	Change Description
1	Nov.12, 2013	Initial revision.
2	Nov.15, 2013	Typos and text cleanup only
	Nov.25, 2013	Corrected Create-Process Acknowledgment format.
	Nov.25, 2013	Corrected Action-Request Index 2. The LIPE automatically closes TCP and UDP user processes when the DHCP lease is lost. The same behavior is also described in the DHCP Client process description.
	Nov.25, 2013	Description of the Simple Serial Terminal process now includes information for operation with no LIPU.
	Nov.25, 2013	The Hardware summary now has a description of the Heart Beat LED.
	Nov.27, 2013	LIPE-LIPU Architecture Summary diagram updated.
	Nov.27, 2013	Replace text in section 6.8 TCP HTTP Server with new.