EzLink

Easy-to-use PTP packet protocol

Version 0.1
April 1st, 2014

Author:
Jerry Johns

1.0 Overview

This document describes the packet specification for a new point-to-point packet protocol for data transfer between a host and a DUT running EFI diags. This was conceived to allow for sending of data without the overhead of converting it to ASCII (as would be the case when interacting over UART). It also allows for higher baud rates to be used, and enforces a transport protocol to ensure reliable data delivery.

1.1 Motivation

Communication with diags from a host is currently done over UART. More specifically, diags commands are executed from a diags shell to invoke functionality. Commands upon completion display status codes that convey the outcome of the execution.

While this is usually sufficient for simple interactions, it isn’t flexible or reliable for commands that either consume or produce large data sets. It is also extremely in-efficient to send non-ascii data as ascii as it introduces un-necessary overhead. Finally, a shell based medium doesn't provide verification mechanisms to handle errors that may be induced by the transport medium.

2.0 Packet specification

2.0.1 Transaction Overview

All transactions are sent through packets which contain a header and a payload. All requests from a transmitter are acknowledged by the receiver (and vice-versa). This ensures accurate state between both parties.

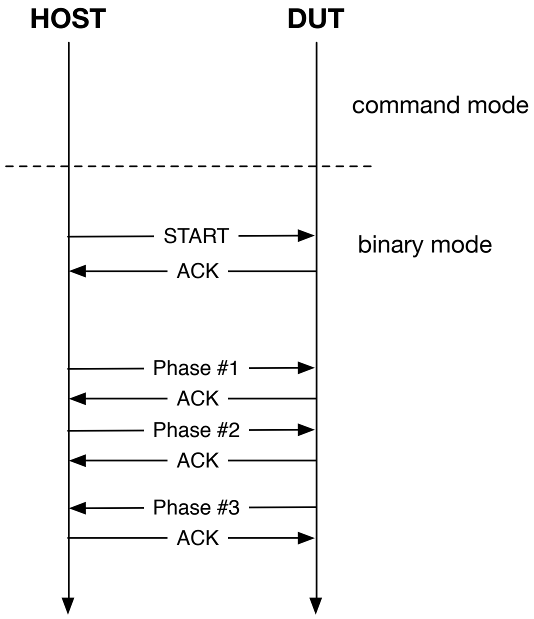
Entry into EzLink occurs after the host executes a diags-side command that utilizes this - once that command is executed, the host will switch over to APIs provided by the host-side library to continue its communications over the serial port (See API list)

Once the DUT switches over, it’ll wait for a START packet from the host indicating the beginning of transmission. Once it receives it, the DUT ACKs that packet and formal communication ensues.

Currently, only synchronous transactions are supported with this library (i.e only one outstanding request from either party is allowed on the bus). Hence, both parties know a-priori the sequence of transactions that are to follow and the transmitter/receiver roles each party is to play at the different points in the flow of transactions.

This model simplifies interaction and allows for the usage of APIs that don’t require callbacks.

Here’s a diagram outlining this flow:



2.0.2 Packet structure

All packets contain a fixed-size header and a variable-length payload. The packets contain a 32-bit field to signal the start of packet, a 16-bit modular sum checksum, type, sequence number and size fields.

The packet structure is listed below:

```
/* Serial packet types */
typedef enum {
    kEzLinkPktTypeStart,
    kEzLinkPktTypeData,
    kEzLinkPktTypeAck,
    kEzLinkPktTypeError
} EzLinkPktType;

#pragma pack(push, 1)

typedef struct {
    uint8_t Type;                       // Reflect by the EzLinkPktType above
    uint32_t Size;                      // Size of data payload that follows
    uint32_t SeqNum;                    // Monotonically increasing sequence number
    uint16_t Checksum;                  // 16-bit modular checksum
    uint8_t Data[];                     // Data payload
} EzLinkPkt;

#pragma pack(pop)
```

The various packet types are listed below:

Type	Description	Payload Type
`kHostBinPktTypeStart`	Sent by host to diags to signal the latter to start sample capture.	None
`kHostBinPktTypeAck`	The ACK packet used to signal acknowledgement from the receiver	None
`kHostBinPktTypeData`	Normal data packets that can be sent by either party	User-specific
`kHostBinPktTypeError`	Sent by either party to signal an error	ErrorPkt

2.0.3 Error packet

At any point during the data transaction, diags can send an error packet if it encounters an error condition. The following payload is sent in this situation:

```
typedef struct {
    int32_t Error;
    char ErrorStr[];
} ErrorPkt;
```

The Error is a 32-bit diags error code - the ErrorStr contains an ASCII error string that provide more descriptive error information.

3.0 Host Library API

A host library has been provided that will simplify the bring-up of applications that utilize this functionality on the host-side. This is provided in the “ezlink-host.c” file (Additional info on the method declarations and their parameters can be found in “ezlink-host.h”)