



DxP Communications Protocol

Supports iPIO-8 Ver 1.1 and above

Overview

The Dataprobe Exchange Protocol (DxP) is the means by which Dataprobe monitor and control products (iPIO-8, iBoot-G2) communicate status and control messages to each other. It is published so that system developers can create systems using other devices to communicate with DxP supported products.

The DxP Communication protocol uses TCP messages transmitted on port 9100 (default). This port can be changed as required in the product setup.

The Protocol consists of a handshake, followed by a command and a response. The handshake uses an incremental sequence number to assist in validating messages and providing unique messages when AES encryption is used.

When AES encryption is enabled, the protocol must be AES encrypted using the same passphrase as the device being communicated with.

The protocol uses little endian ordering.

Protocol

Initial Handshake and Message Sequence

In this phase the client sends a hello message to the device. This hello message is a packet with the null terminated string "hello-000"

0x68 0x65 0x6c 0x6c 0x6f 0x2d 0x30 0x30 0x30 0x00

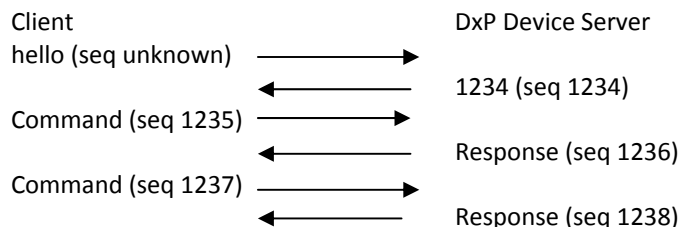
The DxP device server will respond with a packet containing the unsigned 16 bit sequence number.

This sequence number is incremented by both the client and server with each correct packet received.

Command / Response

In this phase the client sends commands as formatted in the sections below. The sequence number is incremented by both sides after receiving a packet.

Example



REF: protocols\dxp_proto_v100218e



Technical Support: 201-934-5111
Main: 201-934-9944

tech@dataprobe.com
Website: dataprobe.com

Packet

The packet consists of 2 parts. Header and Payload. Both Header and Payload are sent as a single packet.

Header

The header is defined by the following C structure.

```
typedef struct {
    uChar command;
    char[21] uName;
    char[21] password;
    uChar desc;
    uChar param;
    uint16 seq;
}THeader
```

Command

The command is an enumerated value that tells the server what class of descriptor is to follow. The current commands are:

```
0x03  command_Status
      This command is used to control the relays and get the status of
      the inputs and relays of the device.

0x04  command_Keepalive
      This command is used as part of the network connection supervision.
      It is sent to all remote devices every 2 seconds.
```

uName

This variable is reserved for future use. It will contain the user name of a user on the device that is being accessed.

Password

This variable is reserved for future use. It will contain the password of the user above.

desc

This is the command descriptor. It is an enumerated value that describes the payload that follows.

param

This is an optional parameter that can be passed to the device along with the payload.

seq

This is the packet's sequence number. It is used as part of the security scheme.

Payload

The payload is described by the descriptor. There is a different set of descriptors for each command type.

command_Status

The control command currently supports the following descriptors.

0x01 eCommand_ChangeRelay
This command is used to change the status of an individual relay.
The payload is as follows:

```
typedef struct{
    unsigned char relay;
    unsigned char state;
}TChangeRelay;
```

Where relay is the number of the relay to be affected 0x00 through 0x07 for relays 1 - 8 and state sets the state of the relay, 0x00 = open 0x01 = closed

0x04 eCommand_GetOutputs
This command is used to get the status of the outputs. This command does not require any payload.

0x06 eCommand_GetInputs
This command is used get the status of all the inputs. This command does not require any payload.

0x07 eCommand_PulseRelay
This command is used to pulse a relay. The payload for this command is as follows:

```
typedef struct {
    unsigned char relay;
    unsigned char state;
    uint16 pulseWidth;
}TPulseRelay;
```

Where

relay is the number of the relay to be affected 0x01 through 0x08 for relays 1 - 8;
state sets the state of the relay, 0x00 = open 0x01 = closed;
pulseWidth sets the pulse width in seconds 1 to 99.

command_KeepAlive

The keep alive command currently supports the following descriptors.

0x00 eKeepAlive_NULL

This is the only valid descriptor that the keep-alive command supports. It is define as null as it carries no payload.

Responses

Each command sent to the device will send a response. The response will differ depending on the command, and descriptor.

command_Status

0x01 eCommand_ChangeRelay
This command will return 0x00 on success 0x01 on error;

0x06 eCommand_GetRelays
This command returns the status of all 8 relays as an array of unsigned characters. It returns the following structure:

```
typedef struct{
    unsigned char status[8];    //Array of relay Status
}TRelayStatus;
```

The structure will be filled with the current status of all 8 relays where 0x00 = relay open and 0x01 = relay closed. LSB is Relay 1, MSB is Relay 8.

0x06 eCommand_GetInputs
This command returns the status of all 8 relays as an array of unsigned characters. It returns the following structure:

```
typedef struct{
    unsigned char status[8];    //Array of input Status
}TInputStatus;
```

The structure will be filled with the current status of all 8 inputs where 0x01 = input open and 0x00 = input closed. LSB is Input 1, MSB is Input 8.

0x07 eCommand_PulseRelay
This command will return 0x00 on success 0x01 on error

command_KeepAlive

0x00 eKeepAlive_NULL
This command returns a single byte containing the status of the command.
This command will return 0x00 on success 0x01 on error