iPuP-PRO Packet Definitions V2. (09/2024)

Packet Definitions

V1: To keep within a single UDP frame, packets should be no bigger than 512 bytes size.

V2: Output packets are now over 512 bytes. Input packets remain the same.

Input Sync Packet – Input sync packets are sent from the iPuP-PRO at the start of a processed frame to request input data from external devices. Input data packets should be returned before the start of the next frame. If packets are not returned within one frame period they will be dropped or stacked, however the last / most recent input frame received will be used to process the new frame.

Additional data can be sent within the sync packet such as system status, currently no status data is being sent, so DATA_LEN=0.

Absolute max packet size = 2 + 4 + 2 + (DATA_LEN, max = 502) + 2 = 512

[SYNC_HDR],[TIME_CODE],[DATA_LEN],[STATUS_DATA],[CRC]

SYNC_HDR = **0xAA03**
TIME_CODE = uint32 (see below for time code structure)
DATA_LEN = uint16
STATUS_DATA = (uint8 * DATA_LEN)
CRC = uint16 (see below for details)

Input Data Packet – All inputs are uint16 precision during transmission, and converted to unary float precision upon reception. Therefore DATA_LEN is in uint16 word size not byte size! If DATA_LEN = 64, there are 64 * uint16 WORDS = 128 uint8 BYTES. Uint16 input data is in little endian format. The iPuP-Pro has two UDP input sources, each having up to 64 input control channels. The two UDP channels are configured within the iPuP-Pro port configuration. Default input sync send addresses are: 10.0.0.1:20000 and 10.0.0.1:20001

Absolute max packet size = 2 + 4 + 2 + 2 + (max data = 500 bytes) + 2 = 512

[INPUT_HDR],[TIME_CODE],[STATUS],[CHAN_COUNT],[INP_DATA],[CRC]

INPUT_HDR = **0xAA02**
TIME_CODE = uint32 (see below for time code structure)
STATUS_BITS = uint16
CHAN_COUNT = uint16
INP_DATA = (uint16 * CHAN_COUNT) <= MAX_INPUTS (64) (little endian format)
CRC = uint16 (see below for details)

In the case of a iPuP-Pro UDP Input Packet with 64 channels we would have:

[2 byte header],[4 byte time code],[2 byte status],[2 byte channel count],[2 byte x channel count].[2 byte CRC] = **140 bytes**

Conversion of uint16 to unary floating point for input data uses the following method:

```
float newValue[MAX_OUTPUTS];
for( unsigned t = 0; t < CHAN_COUNT; t++ )
{
	newValue[t] = OUT_DATA[t]/65535.0;
}
```

UDP Output Packet - All outputs are uint16 precision during transmission, and converted at the receiving end to the receivers desired precision. Uint16 output data is in little endian format. Currently the iPuP-PRO is limited to 128 output channels comprising of 80 outputs driven from expressions or super-expressions and 48 direct drive outputs. The UDP output packet is sent to the IP address and port number defined by the iPup-Pro configuration. Default send address is: 10.0.0.1:10000

This protocol can have larger channel counts than 128, but is limited to a maximum channel count of 500, however the iPuP-Pro is internally limited to 128 channels:

Absolute max packet size = [2] + [4] + [2] + [2] + [(max data= 1000 bytes)] + [2] = 1012

Packet Definition

[OUTPUT_HDR],[TIME_CODE],[STATUS],[CHAN_COUNT],[OUT_DATA],[CRC]

OUTPUT_HDR = **0x5502**
TIME_CODE = uint32 (see below for time code structure)
STATUS_BITS = uint16
CHAN_COUNT = uint16
OUT_DATA = (uint16 * CHAN_COUNT) <= MAX_OUTPUTS (128) (little endian format)
CRC = uint16 (see below for details)

In the case of a iPuP-Pro UDP Output Packet we would have:

[2 byte header],[4 byte time code],[2 byte status],[2 byte channel count],[2 byte x channel count].[2 byte CRC] = **268 bytes**

Conversion of uint16 to unary floating point for output data uses the following method:

```
float newValue[MAX_OUTPUTS];
for( unsigned t = 0; t < CHAN_COUNT; t++ )
{
	newValue[t] = OUT_DATA[t]/65535.0;
}
```

UDP Feedback Packet – The feedback packet sent from the iPuP-Pro is the actual position of the servos rather than the desired position sent by UDP output packet [0x5502]. The packet is sent to the same address and port as the UDP Output Packet above but with a differing packet header. This array of position data is the last know position of any connected servos. The packet format is identical to the UDP Output Packet as described above with the following differences:

- The header is **0x5503**
- The uint16 value range for conversion is from 1 to 65535, 0 is reserved to indicate a disconnected device.

Conversion of uint16 to unary floating point for feedback data uses the following method:

```
float newValue[MAX_OUTPUTS];
for( unsigned t = 0; t < CHAN_COUNT; t++ )
{
	// check for disconnected device condition
	if( OUT_DATA[t] == 0 )
	{
		// adjust data
		OUT_DATA[t] = 1;
		// set appropriate servo disconnected flag.
	}
	// convert data
	newValue[t] = OUT_DATA[t]/65535.0;
}
```

Packet CRC Generation

The following code is used to generate the 16bit CRC for the data packets above. The CRC is calculated from the packet header through to the end of the data section:

```
const   uint16_t cCrc16Table[16] = {
                                0x0000, 0xcc01, 0xd801, 0x1400,
                                0xf001, 0x3c00, 0x2800, 0xe401,
                                0xa001, 0x6c00, 0x7800, 0xb401,
                                0x5000, 0x9c01, 0x8801, 0x4400
                                };
/*
* crcGenerate()
* dPtr = char* pointer to data buffer
* pLength = length of data buffer in bytes
*
* To generate the CRC for an output packet the following would be passed to this function
*
* OUTPUT_HDR = 2 bytes, TIME_CODE = 4 bytes, STATUS_BITS = 2 bytes, DATA_LEN = 2 bytes
* crc16 = crcGenerate( buffer, 2 + 4 + 2 + 2 + (2 * DATA_LEN) );
*/

uint16_t        crcGenerate(char *dPtr, unsigned pLength)
{
uint16_t rl,crc16=0;

for( unsigned t = 0; t < pLength; t++, dPtr++ )
        {
        rl = cCrc16Table[crc16 & 0x0f];
        crc16 = (crc16>>4) & 0x0fff;
        crc16 = crc16 ^ rl ^ cCrc16Table[*dPtr & 0x0f];

        rl = cCrc16Table[crc16 & 0x0f];
        crc16 = (crc16>>4) & 0x0fff;
        crc16 = crc16 ^ rl ^ cCrc16Table[(*dPtr>>4) & 0x0f];
        }

return crc16;
}
```

Time Code structure used within UDP output and sync packets

Time code structure = uint32 H:M:S:F

```
typedef union
        {
        uint32 timecode;
        unsigned frames : 8; // 0 > 255 FPS
        unsigned seconds : 6; // 0 > 59
        unsigned minutes : 6; // 0 > 59
        unsigned hours : 10; // 0 > 1023
        }s_TimeCode;
```