IpuP-PRO Packet Definitions V2. (03/2022)

Time Stamp / Time Code structure

Time stamp = uint64 in nanoseconds

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;
```

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 (20 milliseconds). If packets are not returned within 20 milliseconds 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.

Max 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
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

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

[INPUT_HDR][TIME_CODE][STATUS][DATA_LEN][INP_DATA][CRC]

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

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 96 output channels comprising of 64 outputs driven from expressions or super-expressions and 32 direct drive outputs.

Max size = 2 + 4 + 2 + 2 + (max data= 1000 bytes) + 2 = 1012

[OUTPUT_HDR][TIME_CODE][STATUS][DATA_LEN][OUT_DATA][CRC]

OUTPUT_HDR = **0x5502**
TIME_CODE = uint32
STATUS_BITS = uint16
DATA_LEN = uint16
OUT_DATA = (uint16 * DATA_LEN) <= MAX_OUTPUTS (96) (little endian format)
CRC = uint16 (see below for details)

Conversion of uint16 to unary floating point for output data or input data:

```
float newValue[MAX_OUTPUTS];
for( unsigned t = 0; t < DATA_LEN; t++ )
{
	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;
}
```

Transport Control Messages

Transport messages are sent on their own UDP port, one higher than the UDP output port. E.G. the standard UDP output port is 10000, so transport messages are sent out on 10001.

Standard Transport messages are sent as per Xpress1 format as a four byte packet:

[XP2_MB_HEADER][COMMAND][0][MB_TRAILER]

XP2_MB_HEADER = 0xF0
COMMAND = uint8 (See below)
PAD_BYTE = 0
XP2_MB_TRAILER = 0xF7

```
#define     XP2_MB_HEADER                0xF0
#define     XP2_MB_TRAILER               0xF7

#define     XP2_MB_TRANSPORT_STOP        0x51
#define     XP2_MB_TRANSPORT_PLAY        0x52
#define     XP2_MB_TRANSPORT_RWD         0x53
#define     XP2_MB_TRANSPORT_FWD         0x54
#define     XP2_MB_TRANSPORT_STEP_FWD    0x55
#define     XP2_MB_TRANSPORT_STEP_BKWD   0x56
#define     XP2_MB_TRANSPORT_GOTO        0x57
#define     XP2_MB_TRANSPORT_PLAY_FULL   0x58
#define     XP2_MB_TRANSPORT_PLAY_HALF   0x59
#define     XP2_MB_TRANSPORT_PLAY_QUATER 0x5a
#define     XP2_MB_TRANSPORT_PLAY_DOUBLE 0x5b
#define     XP2_MB_TRANSPORT_PLAY_FOUR   0x5c
#define     XP2_MB_TRANSPORT_RWDPLAY     0x5d  // rewind and start playback
#define     XP2_MB_TRANSPORT_RECD        0x5e  // not implemented
```