DOREMI LABS, INC.

NUGGET MACHINE CONTROL
PROTOCOL
Version 1.0 Ethernet

4.2 Controlling the V1 using the Ethernet port:

The V1 can handle IP based network communication using connectionless (UDP) or connection-oriented messages (TCP). The V1 uses a very simple protocol on top UDP and TCP.
Important note: The byte ordering in the structure is <u>big endian:</u> When filling or reading a data structure from a computer using little endian byte ordering (eg. Intel ix86), the programmer must manually swap the structure member.

4.2.1 Using UDP for Control

BYTE ORDERING IS BIG ENDIAN

The V1 receives UDP messages sent on port 0x8080, and replies on the same port. Every message should be accompanied with a 10 byte header which has the following format:

```
typedef  struct  {
  unsigned  short
  unsigned  short
  unsigned  short
  unsigned  short
  unsigned short
} ComHdr;
```

the type field can have on of the values of the following enums :

```
enum {
  Sony9P_Protocol = 1,
  ComputerLink = 2,
  Mgmt_Protocol= 3
};
```

Example : To send a Sony 9 pin Play command, you need to construct the following :

0x0001, 0x0003, 0x0000, 0x0000, 0x0000, 0x20, 0x01, 0x21

The first word indicate that the message is a Sony9P_Protocol message, the second word indicate the size of the message excluding the 10 byte header, the following 3 words are " don't care" . Then the play message 20.01.21

you will receive an ack

0x0001, 0x0003, 0x0000, 0x0000, 0x0000, 0x10, 0x01, 0x11

4.2.2 Using TCP/IP for Control

4.2.2.1 Description

At boot time, the V1 opens two listening socket on port 5000 (0x1388).

Operations are initiated using a simple protocol that uses messages of 12 bytes structured as follows:

```
typedef struct {
  int32 type; // 32 bits int32
  param1; // 32 bits


  int32 param2; // 32 bits }
cnxn_msg;
```

This structure is the header used for all TCP/IP communication, both requests and replies. However, the header might be followed by one or more bytes depending on the *type* of the message described below.

4.2.2.2 Simple messages

- **Nop** (No operation): This message does not do anything. It is basically used to prevent communications time out.
  - ***Request message***
    type is set to the value 0x616c6976 (‘ aliv‘) *param1* is not used and should be set to zero *param2* is not used and should be set to zero *data* no data should follow the header
  - ***Reply message***
    There is no reply to this message
  - ***Example***
    Outgoing: 61 6c 69 76 00 00 00 00 00 00 00 00
    Incoming: (none)

- **Version:** This message retrieves the protocol version used on the V1. The present document describes protocol 1.0
  - ***Request message***
    type is set to the value 0x76657220 (‘ ver ‘) *param1* is not used and should be set to zero *param2* is not used and should be set to zero *data* no data should follow the header
  - ***Reply message***
    type is set to the value 0x76657220 (‘ ver ‘) *param1* contains the protocol version number *param2* contains the protocol revision number *data* none
  - ***Example***
    Outgoing: 76 65 72 20 00 00 00 00 00 00 00 00
    Incoming: 76 65 72 20 00 00 00 01 00 00 00 00

- **Sony 9 pin protocol embedded message**
  - ***Request message***
    type is set to the value 0x73397020 (‘ s9p ‘)
    param1 is initialized with the sony 9 pin message length (checksum included). *param2* is not used and should be set to zero *data* the sony 9 pin command should be immediately after the header.
  - ***Reply message***
    type is set to the value 0x73397020 (‘ s9p ‘)
    param1 is initialized with the sony 9 pin message length (checksum included). *param2* is not used
    data the sony 9 pin reply follow immediately the header.
  - ***Example***
    Outgoing: 73 39 70 20 00 00 00 03 00 00 00 00 20 00 21
    Incoming: 73 39 70 20 00 00 00 03 00 00 00 00 10 00 11