HiQontrol Documentation

Release 0.0.2

Raphaël Doursenaud

Oct 05, 2017

Contents

Contents:

CHAPTER 1

Protocol reverse engineering

Miscillaneous notes

HiQnet Protocol Decoding

```
Soundcraft Si C16
  32    canaux de mixage
  8     Bus Mix Mono
  6(12) Mix Bus Stéréo
  4(8)  Matrix Bus Stéréo
  4(8)  FX Bus Stéréo
  1(2)  Main Stereo output
  1     Main Mono output
  1(2)  Monitor Stereo output

HiQnet (Harman iqnet)
Port: 3804

locate time 0x1000 produces a 3-4 s flash burst
any other value seem infinite

Device Manager Attributes
    0: [Si Compact 16] Class Name
    1: [Si Compact 16] Name
    2: Flags
    3: Serial Number
    4: [V2.0] Version

Virtual Devices Attributes
    0: Class Name
    1: Name

Parameter Attributes
    0:
```

```
    1:
    2:
    3:
    4:
    5:

Console VDs:
1.0.0.1 Mix busses master
  PID 1: ON Mix 1
  [...]
  PID 14: ON Mix 14
  PID 15: ON Mtx 1
  [...]
  PID 18: ON Mtx 4
  PID 19: Fader Mix 1
  [...]
  PID 32: Fader Mix 14
  PID 33: Fader Mtx 1
  [...]
  PID 36: Fader Mtx 4

1.0.0.2 ????
1.0.0.3 Main busses master
  PID 1: ON L&R
  PID 2: ON Mono
  PID 3: Fader L&R
  PID 4: Fader Mono
1.0.0.4 Mix 1 sends
  Cf Master sends
1.0.0.5 Mix 2 sends
1.0.0.6 Mix 3 sends
1.0.0.7 Mix 4 sends
1.0.0.8 Mix 5 sends
1.0.0.9 Mix 6 sends
1.0.0.10 Mix 7 sends
1.0.0.11 Mix 8 sends
1.0.0.12 Mix 9 sends
1.0.0.13 Mix 10 sends
1.0.0.14 Mix 11 sends
1.0.0.15 Mix 12 sends
1.0.0.16 Mix 13 sends
1.0.0.17 Mix 14 sends
1.0.0.19 Matrix 1 sends
  PID 1: ON Mix 1
  [...]
  PID 14: ON Mix 14
  PID 15: ON L
  PID 16: ON R
  PID 17: ON Mono
  PID 18: Fader Mix 1
  [...]
  PID 31: Fader Mix 14
  PID 32: Fader L
  PID 33: Fader R
  PID 34: Fader Mono
1.0.0.19 Matrix 2 sends
1.0.0.20 Matrix 3 sends
1.0.0.21 Matrix 4 sends
```

```
1.0.0.22 Master sends
  PID 1: ON CH1
    LONG 0 or 1
  [...]
  PID 36: ON ST4
  PID 37: Fader CH1
    LONG signed 0dB = 0x00000000 -inf (-138dB!?) = 0xffffdd80 +10dB = 0x00000280
  [...]
  PID 72: Fader ST4

1.0.0.23 Mix 1 GEQ
  PID 1: ON 31 Hz
  PID 28: ON 16kHz
  PID 29: Fader 31 Hz
    LONG signed -12dB = 0xfffffd00 0dB = 0x00000000 +12dB = 0x00000300
1.0.0.24 Mix 2 GEQ
1.0.0.25 Mix 3 GEQ
1.0.0.26 Mix 4 GEQ
1.0.0.27 Mix 5 GEQ
1.0.0.28 Mix 6 GEQ
1.0.0.29 Mix 7 GEQ
1.0.0.30 Mix 8 GEQ
1.0.0.31 Mix 9 GEQ
1.0.0.32 Mix 10 GEQ
1.0.0.33 Mix 11 GEQ
1.0.0.34 Mix 12 GEQ
1.0.0.35 Mix 13 GEQ
1.0.0.36 Mix 14 GEQ
1.0.0.37 Matrix 1 GEQ
1.0.0.38 Matrix 2 GEQ
1.0.0.39 Matrix 3 GEQ
1.0.0.40 Matrix 4 GEQ
1.0.0.41 Master L+R GEQ
1.0.0.42 Master Mono GEQ
1.0.0.43 ????
1.0.0.44 Channels names
  PID 1: CH1
  [...]
  PID 36: ST4
1.0.0.45 Mix busses names
  PID 1: Mix1
  [...]
  PID 18: Mtx4
1.0.0.46 Main busses names
  PID 1: L
  PID 2: R
  PID 3: Mono
1.0.0.47 ????
1.0.0.48 CH1 Params
  PID 1: Gate On
  PID 2: Gate Threshold
  PID 3: Gate attack
  PID 4: Gate Release
  PID 5: Gate Depth
  PID 6: Gate HP Filter
  PID 7: Gate LP Filter
  PID 8: Comp On
  PID 9: Comp Threshold
```

```
  PID 10: Comp Attack
  PID 11: Comp Release
  PID 12: Comp Ratio
  PID 13: Comp Gain
  PID 14: EQ In
  PID 15: LF Freq
  PID 16: LF Gain

  PID 19: Lo Mid Freq
  PID 20: Lo Mid Gain
  PID 21: Lo Mid Q
  PID 22: Hi Mid Freq
  PID 23: Hi Mid Gain
  PID 24: Hi Mid Q
  PID 25: HF Freq
  PID 26: HF Gain

  PID 29: Delay
  PID 30: Phase
  PID 31: HPF On
  PID 32: HP Filter freq
  PID 33: Pan
    L = 0x00000000 C = 0x0000002d R = 0x00000059
  PID 34: L&R Assign
  PID 35: Mono Assign

  PID 38: ? (Sent when Assign changes [PID 34 or 35])
  PID 39: Gain
     LONG signed -5dB = 0xfffffec0 0dB = 0x00000000 58dB = 0x00000e80
  PID 40: +48V

1.0.0.49 CH2 Params
1.0.0.50 CH3 Params
1.0.0.51 CH4 Params
1.0.0.52 CH5 Params
1.0.0.53 CH6 Params
1.0.0.54 CH7 Params
1.0.0.55 CH8 Params
1.0.0.56 CH9 Params
1.0.0.57 CH10 Params
1.0.0.58 CH11 Params
1.0.0.59 CH12 Params
1.0.0.60 CH13 Params
1.0.0.61 CH14 Params
1.0.0.62 CH15 Params
1.0.0.63 CH16 Params
1.0.0.64 CH17 Params
1.0.0.65 CH18 Params
1.0.0.66 CH19 Params
1.0.0.67 CH20 Params
1.0.0.68 CH21 Params
1.0.0.69 CH22 Params
1.0.0.70 CH23 Params
1.0.0.71 CH24 Params
1.0.0.72 CH25 Params
1.0.0.73 CH26 Params
1.0.0.74 CH27 Params
1.0.0.75 CH28 Params
```

```
1.0.0.76 CH29 Params
1.0.0.77 CH30 Params
1.0.0.78 CH31 Params
1.0.0.79 CH32 Params
1.0.0.80 ST1 Params
  PID 1: Gate On
  PID 2: Gate Threshold
  PID 3: Gate attack
  PID 4: Gate Release
  PID 5: Gate Depth
  PID 6: Gate HP Filter
  PID 7: Gate LP Filter
  PID 8: Comp On
  PID 9: Comp Threshold
  PID 10: Comp Attack
  PID 11: Comp Release
  PID 12: Comp Ratio
  PID 13: Comp Gain
  PID 14: EQ In
  PID 15: LF Freq
  PID 16: LF Gain

  PID 19: Lo Mid Freq
  PID 20: Lo Mid Gain
  PID 21: Lo Mid Q
  PID 22: Hi Mid Freq
  PID 23: Hi Mid Gain
  PID 24: Hi Mid Q
  PID 25: HF Freq
  PID 26: HF Gain

  PID 29: Delay
  PID 30: Phase
  PID 31: HPF On
  PID 32: HP Filter freq
  PID 33: Pan
    L = 0x00000000 C = 0x0000002d R = 0x00000059
  PID 34: L&R Assign
  PID 35: Mono Assign

  PID 38: ? (Sent when Assign changes [PID 34 or 35])
  PID 39: Trim
1.0.0.81 ST2 Params
1.0.0.82 ST3 Params
1.0.0.83 ST4 Params
1.0.0.84 Mix 1 Params
  PID 1: Comp On
  PID 2: Comp Threshold
  PID 3: Comp Attack
  PID 4: Comp Release
  PID 5: Comp Ratio
  PID 6: Comp Gain
  PID 7: EQ On
  PID 8: LF Freq
  PID 9: LF Gain

  PID 12: Lo Mid Freq
  PID 13: Lo Mid Gain
```

```
  PID 14: Lo Mid Q
  PID 15: Hi Mid Freq
  PID 16: Hi Mid Gain
  PID 17: Hi Mid Q
  PID 18: HF Freq
  PID 19: HF Gain

  PID 22: Delay
  PID 23: Phase
  PID 24: HPF On
  PID 25: HP Filter Freq
  PID 26: Pan
  PID 27: L&R Assign
  PID 28: Mono Assign

  PID 31: ? (Sent when Assign changes [PID 34 or 35])
[...]
1.0.0.97 Mix 14 Params
1.0.0.98 Matrix 1 Params
  PID 1: Comp On
  PID 2: Comp Threshold
  PID 3: Comp Attack
  PID 4: Comp Release
  PID 5: Comp Ratio
  PID 6: Comp Gain
  PID 7: EQ On
  PID 8: LF Freq
  PID 9: LF Gain

  PID 12: Lo Mid Freq
  PID 13: Lo Mid Gain
  PID 14: Lo Mid Q
  PID 15: Hi Mid Freq
  PID 16: Hi Mid Gain
  PID 17: Hi Mid Q
  PID 18: HF Freq
  PID 19: HF Gain

  PID 22: Delay
[...]
1.0.0.101 Matrix 4 Params
1.0.0.102 L&R Params
  PID 1: Comp On
  PID 2: Comp Threshold
  PID 3: Comp Attack
  PID 4: Comp Release
  PID 5: Comp Ratio
  PID 6: Comp Gain
  PID 7: EQ On
  PID 8: LF Freq
  PID 9: LF Gain

  PID 12: Lo Mid Freq
  PID 13: Lo Mid Gain
  PID 14: Lo Mid Q
  PID 15: Hi Mid Freq
  PID 16: Hi Mid Gain
  PID 17: Hi Mid Q
```

```
  PID 18: HF Freq
  PID 19: HF Gain

  PID 22: Delay

  PID 25: Balance
1.0.0.103 Mono Params
  Cf Matrix params
1.0.0.104 FX1 sends
  Cf Master sends
1.0.0.105 FX2 sends
1.0.0.106 FX3 sends
1.0.0.107 FX4 sends



Meters: UDP 3333 ?
  Payload 624o
  1 of 2 in VLAN 1
  AA BB CC DD
  AA BB -> VU value ?
  CC -> Comp gain reduction ?
  DD = 0x09 -> No gate
  DD = 0x01 -> Gate close
  DD = 0x04 -> Gate open
  DD = 0x0c -> Gate hold ????

  Groupes de 4 octects

  16 premiers = 16 CH
  16 suivants : voir meter_packet_decode.txt

  2 derniers -> Monitor

  VU ()
    32+4*2+8+6*2+4*2+2+1 = 71
  Gate (Bool ou tristate)
    32+4 = 36
  Comp
    32+4+8+6+4+2 = 56
```

HiQnet Packet Decoding

```
[Header]
02      Version
1b      Header length (27)
00      Message length MSB
00
00
2f      Message length LSB (47)
f7      Source address MSB
f2
00
00
00
```

```
00      Source address LSB
06      Destination address MSB
53
00
00
00
00      Destination address LSB
01      Message ID MSB
29      Message ID LSB (locate)
01      Flags MSB
20      Flags LSB (0000 0001 0010 0000) [Session + Guaranteed]
04      Hop
00      Seq num MSB
14      Seq num LSB
6e      Sess num MSB
04      Sess num LSB
[Payload]
ff      locate MSB
ff      locate LSB (Always on)
00
10
53
69
43
6f
6d
70
61
63
74
00
00
00
00
00
00
00
```

Soundcraft Si Console Meter Packet Decoding

```
27201000 CH1
cfce0009
cfcf0009
cfcf0009
cece0009
cece0009
cece0009
cfce0009
cfcf0009
cfce0009
cfce0009
cece0009
cfce0009
cfcf0009
cfce0009
cfce0009 CH16
```

```
ffff0009 CH17
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009
ffff0009 CH32
9d930009 ST1 L
9c930009 ST1 R
ffff0009 ST2 L
ffff0009 ST2 R
d6c60009 ST3 L
c6c60009 ST3 R
ffff0009 ST4 L
ffff0009 ST4 R
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
```

```
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffffff01
ffff0000 Mix 1
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000 Mix 8
ffff0000 Mix 9 L
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000 Mix 14 R
8b890000
ffff0000
ffff0000
ffff0000
66600000
9d940000
ffff0000
ffff0000 Mtx 1 L
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000 Mtx 4 L
ffff0000 Mtx 4 R
7e79ff00 Main L
c1b8ff00 Main R
ffffff00 Mono
ffffff00
ffff0000 Mix 1
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
```

```
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000 Mix 14 R
8b890000
ffff0000
ffff0000
ffff0000
7e790000 Main L
c1b80000 Main R
ffff0000 Mono
ffff0000 Mtx 1 L
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000
ffff0000 Mtx 4 L
ffff0000 Mtx 4 R
7c77ff00 Mon L
bfb6ff00 Mon R
```

CHAPTER 2

Source code

HiQnet package

Submodules

hiqnet.protocol module

HiQnet protocol library.

class hiqnet.protocol.**Command**(*source=None*, *destination=None*, *command=None*)
Bases: object

HiQnet command.

address_used()
Build an Address Used command.

bytes_remaining = 0

commandlen

decode(*command*)
Decodes a binary command.

Parameters **command** – The binary command to decode

decode_discoinfo()
Decode discovery information command payload.

Payload: - HiQnet Device - Cost - Serial Number - Max Message size - Keep alive period - NetwordID - NetworkInfo

destination_address = None.0.0.0.0

disco_info(*device*, *disco_type='Q'*)
Build a Discovery Information command.

Parameters

- **device** (Device) – The HiQnet device sending the discovery command
- **disco_type** (*str*) – Discovery type message. I(nfo) or Q(uery)

error_code = 0

error_string = ‘’

flags = reqack:0 ack:0 info:0 error:0 res1:0 guaranteed:0 multipart:0 res2:0 session:0 res3:0 res4:0 res5:0 res6:0 res7:0

get_attributes()
Build a Get Attributes command.

get_vd_list(*workgroup=’‘*)
Build a Get VD List command.

Parameters **workgroup** (*str*) – The workgroup to get the VD list from.

header = ‘\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00’

headerlen

hello()
Build an hello command.

Starts a session.

Returns The session number

Return type int

hop_counter = 5

locate(*time*, *serial_number*)
Builds a Locate command.

The receiver makes itself visible. Usually this is done by flashing some LEDs on the front panel.

Parameters

- **time** (*bytearray*) – time the leds should flash in ms 0x0000 turns off locate led(s) 0xffff turns on locate led(s)
- **serial_number** (*str*) – The target device’s serial number

See also:

locate_on(), *locate_off()*

locate_off(*serial_number*)
Builds a locate command asking for the visual clue to be inactive.

Parameters **serial_number** (*str*) – The target device’s serial number

locate_on(*serial_number*)
Builds a locate command asking for the visual clue to be active.

Parameters **serial_number** (*str*) – The target device’s serial number

message = DISCOINFO

new_sequence_number = count(0)

optional_headers = ‘’

payload = ‘’

recall()
Build a Recall command.

Recalls a preset.

request_address(*req_addr*)
Build a Request Address command.

Parameters **req_addr** (*int*) –

sequence_number = 0

session_number = 0

source_address = None.0.0.0.0

start_seq_no = 0

store()
Build a Store command.

Stores current state to a preset.

version

class hiqnet.protocol.**FullyQualifiedAddress**(*device_address=None*, *vd_address='x00'*, *object_address='x00x00x00'*, *devicevdobject=None*)
Bases: object

Fully Qualified HiQnet Address.

classmethod broadcast_address()
Get the Fully Qualified HiQnet Broadcast Address.

Return type *FullyQualifiedAddress*

device_address = None

object_address = None

vd_address = None

class hiqnet.protocol.**Message**(*identifier=None*, *name=None*)
Bases: object

HiQnet messages handling.

MESSAGES = {'\x00\x02': 'GETNETINFO', '\x00\x03': 'RESERVED1', '\x00\x00': 'DISCOINFO', '\x00\x01': 'RESERV

identifier = None

name = None

hiqnet.device module

HiQnet device architecture.

Node (Device)

- At least one virtual device (The first is the device manager)
- Parameters and/or objects
- Objects contains parameters and/or other objects

Attributes everywhere

Either STATIC, Instance or Instance+Dynamic

Virtual devices, objects and parameters

Have a Class Name and a Class ID

class hiqnet.device.**Attribute**(*atr_type*)
Bases: object

Member variables of the HiQnet architecture.

Static are basically constants. Instance are variables that are set at device bootup. Instance+Dynamic are regular variables that can change during the life of the device.

type = None

class hiqnet.device.**Device**(*name*, *hiqnet_address=54399*, *network_info=<hiqnet.networkinfo.IPNetworkInfo object>*)
Bases: object

Describes a device (aka node).

address
Get the device manager address

Returns The fully qualified address of the device manager

Return type *FullyQualifiedAddress*

hiqnet_address
Get the device HiQnet address

Returns The device HiQnet address

Return type int

manager = None

name
Get the device name

Returns The device name

Return type str

network_info = None

virtual_devices = None

class hiqnet.device.**DeviceManager**(*name_string*, *class_name=None*, *flags=0*, *serial_number=None*, *software_version=None*)
Bases: *hiqnet.device.VirtualDevice*

Describes a HiQnet device manager.

Each device has one and this is always the first virtual device.

flags = reqack:0 ack:0 info:0 error:0 res1:0 guaranteed:0 multipart:0 res2:0 session:0 res3:0 res4:0 res5:0 res6:0 res7:0

serial_number = None

software_version = None

class hiqnet.device.**Object**
Bases: object

HiQnet objects.

May contain other objects or parameters.

class hiqnet.device.**Parameter**
Bases: *hiqnet.device.Object*

HiQnet parameters.

Represents the manipulable elements and their attributes.

control_law = None

data_type = None

flags = res1:0 sensor:0 res2:0 res3:0

maximum_value = None

minimum_value = None

name_string = ‘’

class hiqnet.device.**VirtualDevice**
Bases: object

Describes a HiQnet virtual device.

This is the basic container object type.

attributes = None

class_name = <hiqnet.device.Attribute object>

name_string = <hiqnet.device.Attribute object>

objects = None

parameters = None

hiqnet.device.**negotiate_address**()
Generates a random HiQnet address.

The address is automatically checked on the network.

hiqnet.networkinfo module

HiQnet device network informations.

class hiqnet.networkinfo.**IPNetworkInfo**(*mac_address*, *dhcp*, *ip_address*, *subnet_mask*, *gateway_address='0.0.0.0'*)
Bases: *hiqnet.networkinfo.NetworkInfo*

IPv4 network informations.

classmethod autodetect()
Get infos from the interface.

We assume that interface to the default gateway is the one we want and fallback to the second interface since the first is usually “lo”.

Return type *NetworkInfo*

dhcp = True

gateway = None

ip_address = None

mac_address = None

subnet_mask = None

class hiqnet.networkinfo.**NetworkInfo**(*network_id=1*)
Bases: object

Network informations.

NET_ID_RS232 = 4

NET_ID_TCP_IP = 1

network_id = None

- 1: TCP/IP
- 2: reserved
- 3: reserved
- 4: RS232

class hiqnet.networkinfo.**RS232NetworkInfo**
Bases: *hiqnet.networkinfo.NetworkInfo*

RS232 network informations.

Note: Not implemented

- com_id 1 byte
- baud_rate 4 bytes
- parity 1 byte
  - 0 - None
  - 1 - Odd
  - 2 - Even
  - 3 - Mark
  - 4 - Space
- stop_bits 1 byte
  - 0 - 1 bit
  - 1 - 1.5 bits
  - 2 - 2 bits
- data_bits 1 byte
  - 4-9
- flow_control 1_byte
  - 0 - None
  - 1 - Hardware

–2 - XON/OFF

hiqnet.flags module

HiQnet Flags.

Simplified flags management with direct bitfields access.

See also:

https://wiki.python.org/moin/BitManipulation

class `hiqnet.flags.`**DeviceFlags**
Bases: `_ctypes.Union`

Device flags.

ack
Structure/Union member

asByte
Structure/Union member

b
Structure/Union member

error
Structure/Union member

guaranteed
Structure/Union member

info
Structure/Union member

multipart
Structure/Union member

reqack
Structure/Union member

res1
Structure/Union member

res2
Structure/Union member

res3
Structure/Union member

res4
Structure/Union member

res5
Structure/Union member

res6
Structure/Union member

res7
Structure/Union member

res8
Structure/Union member

res9
Structure/Union member

session
Structure/Union member

class hiqnet.flags.**DeviceFlagsBits**
Bases: _ctypes.Structure

Bitfields for the device flags.

ack
Structure/Union member

error
Structure/Union member

guaranteed
Structure/Union member

info
Structure/Union member

multipart
Structure/Union member

reqack
Structure/Union member

res1
Structure/Union member

res2
Structure/Union member

res3
Structure/Union member

res4
Structure/Union member

res5
Structure/Union member

res6
Structure/Union member

res7
Structure/Union member

res8
Structure/Union member

res9
Structure/Union member

session
Structure/Union member

class hiqnet.flags.**ParameterFlags**
Bases: _ctypes.Union

Parameter flags.

asByte
Structure/Union member

b
Structure/Union member

res1
Structure/Union member

res2
Structure/Union member

res3
Structure/Union member

sensor
Structure/Union member

class hiqnet.flags.**ParameterFlagsBits**
Bases: _ctypes.Structure

Bitfields for the parameters flags.

Bits 0, 2, and 3 are reserved. Bit 1 is the Sensor Attribute. 0 = Non-Sensor 1 = Sensor

res1
Structure/Union member

res2
Structure/Union member

res3
Structure/Union member

sensor
Structure/Union member

hiqnet.service.ip module

HiQnet IP communication.

class hiqnet.service.ip.**Connection**(*udp_transport*, *tcp_transport*)
Bases: object

Handles HiQnet IP connection.

> **Warning:** Other connection types such as RS232, RS485 or USB are not handled yet.

sendto(*command*, *destination='<broadcast>'*)
Send command to the destination.

Parameters

- **command** (Command) – Message to send
- **destination** (*str*) – Destination IPv4 address or '<broadcast>'

tcp_transport = None

udp_transport = None

class hiqnet.service.ip.**Factory**(*app*)
Bases: twisted.internet.protocol.Factory

HiQnet Twisted Factory.

protocol
alias of *TCPProtocol*

class hiqnet.service.ip.**TCPProtocol**
Bases: twisted.internet.protocol.Protocol

HiQnet Twisted TCP protocol.

dataReceived(*data*)
Called when data is received.

Parameters **data** (*bytearray*) – Received binary data

name = 'HiQnetTCP'

startProtocol()
Called after protocol started listening.

class hiqnet.service.ip.**UDPProtocol**(*app*)
Bases: twisted.internet.protocol.DatagramProtocol

HiQnet Twisted UDP protocol.

datagramReceived(*data*, *addr*)
Called when data is received.

Parameters

- **data** (*bytearray*) – Received binary data
- **addr** (*tuple*) – IPv4 address and port of the sender

name = 'HiQnetUDP'

startProtocol()
Called after protocol started listening.

Soundcraft package

Submodules

soundcraft.soundcraft module

Soundcraft meters library.

Main script

HiQontrol: an attempt at building a free, open source, multi–platform ViSi Remote alternative.

class main.**Control**(*source_device*, *udp_transport*, *tcp_transport*)
Bases: object

init(*hiqnet_dest*)

locate = False

locate_toggle(*hiqnet_dest*, *ip_dest*, *serial_dest*)

source_device = None

tcp_transport = None

udp_transport = None

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

m

s

Index

T

U

V