Fohhn-Net protocol

Fohhn-Net is based on EIA-485 (formerly RS-485), thus you need a device with a EIA-485 / RS-485 interface to connect to Fohhn-Net devices.

With the optional NA-3 ethernet adapter it is possible to send the data as a UDP package. The NA-3 is a bridge between ethernet and EIA-485 for the Fohhn-Net protocol.

Serial parameters:

19200 baud
1 startbit
8 databits
1 stopbit

Fohhn-Net uses a half-duplex 2-wire connection.

Most of the Fohhn-Net devices have RJ-45 or terminal blocks as interface socket.
Optional XLR cabeling is possible, with adapter cables.

Signal	RJ-45	XLR
Shield / GND	Shield	Pin 1
DATA +	Pin 3 + Pin 6	Pin 2
DATA -	Pin 4 + Pin 5	Pin 3

all other pins on the RJ-45 connector have not to be connected !

Possible cable types are:

- CAT5, CAT6 or CAT7 cable
- 2 x 2 x 0.6mm-0.8mm (J-Y(ST)Y 2x2x0.6)
- microphone cable

IP based communication (with NA-3 adapter, only):

Protocol: UDP
Port: 2101

You have to send the same binary data as a UDP package to the IP address of the NA-3 and UDP port 2021.

The NA-3 adapter transfers the data over its EIA-485 port to the Fohhn-Net devices.

The NA-3 adapter sends the reply from the Fohhn-Net device as a UDP package to the source ip address and source port of the last request.

Fohhn-Net protocol

Master sends

1. Byte	Startbyte	<SB>	
2. Byte	Device Address	<ADDR>	
3. Byte	Databyte Count	<COUNT>	
4. Byte	Command Byte	<CMD>	
5. Byte	Addr MSB	<ADDR_MSB>	
6. Byte	Addr LSB	<ADDR_LSB>	
7. Byte	Databyte 1	<DATA>	(min. one databyte)
N. Bytes	Optional	<DATA>	

Slave acknowledge (timeout 330 ms)

N	data bytes	<DATA>	(optional)
N+1	Byte Device Address	<ADDR>	
N+2	Byte Startbyte	<SB>	

Protocol encoding

The Fohhn-Net protocol has a startbyte <SB> and a controlbyte <CB>.

The startbyte <SB> or the controlbyte <CB> must never occur in the message.

If the data (every byte, except the startbyte) contains the value of startbyte or controlbyte it must be encoded with the controlbyte <CB>

0xF0 = startbyte <SB>

0xFF = controlbyte <CB>

Encoding:

If the values from <SB> and <CB> are needed in real data, you have to encode them with two bytes

<CB> + 0x00 = 0xF0

<CB> + 0x01 = 0xFF

The controlbyte is not counted as a extra byte in databyte count

Loading a preset

Master sends

```
1. Byte   Startbyte           0xF0
2. Byte   Device Address      0x01 .. 0xFE
3. Byte   Databyte Count      0x01
4. Byte   Command Byte        0x05 // command LoadPreset
5. Byte   ADR_MSB             0x01
6. Byte   ADR_LSB             0x01 .. 0x64 // preset nr.
7. Byte   Databyte 1          0x00
```

Acknowledge from Fohhn Audio DSP (slave)

```
1. Byte   Device Address      0x01 .. 0xFE
2. Byte   Startbyte           0xF0 // end of reply
```

Example: load preset 20 - 24 for device 1

```
Preset 20:   0xF0 0x01 0x01 0x05 0x01 0x14 0x00
Preset 21:   0xF0 0x01 0x01 0x05 0x01 0x15 0x00
Preset 22:   0xF0 0x01 0x01 0x05 0x01 0x16 0x00
Preset 23:   0xF0 0x01 0x01 0x05 0x01 0x17 0x00
Preset 24:   0xF0 0x01 0x01 0x05 0x01 0x18 0x00
```

Example: load preset 20 - 24 for device 2

```
Preset 20:   0xF0 0x02 0x01 0x05 0x01 0x14 0x00
Preset 21:   0xF0 0x02 0x01 0x05 0x01 0x15 0x00
Preset 22:   0xF0 0x02 0x01 0x05 0x01 0x16 0x00
Preset 23:   0xF0 0x02 0x01 0x05 0x01 0x17 0x00
Preset 24:   0xF0 0x02 0x01 0x05 0x01 0x18 0x00
```

Set absolute volume

Master sends

```
1. Byte   Startbyte           0xF0
2. Byte   Device Address      0x01 .. 0xFE
3. Byte   Databyte Count      0x03
4. Byte   Command Byte        0x87  // command SetVolume
5. Byte   Channels            0xXX  // channels
6. Byte   Index               0x01
7. Byte   Volume_H            0xXX  // volume
8. Byte   Volume_L            0xXX  // volume
9. Byte   Flags               0xXX  // flags
```

Acknowledge from Fohhn Audio DSP (slave)

```
1. Byte   Device Address      0x01 .. 0xFE
2. Byte   Startbyte           0xF0 // end of reply
```

Example: set volume for device 1, channel 1 + channel 2

```
 0.0dB On:    0xF0 0x01 0x03 0x87 0x03 0x01 0x00 0x00 0x01

 0.0dB Mute:  0xF0 0x01 0x03 0x87 0x03 0x01 0x00 0x00 0x00

-7.5dB On:    0xF0 0x01 0x03 0x87 0x03 0x01 0xFF 0x01* 0xB5 0x01

+6.0dB On:    0xF0 0x01 0x03 0x87 0x03 0x01 0x00 0x3C 0x01

-40.0dB On:   0xF0 0x01 0x03 0x87 0x03 0x01 0xFE 0x70 0x01
```

Notes

You can set multiple channels with one command.

Each channel has a value:

Channel 1: 0x01
Channel 2: 0x02
Channel 3: 0x04
Channel 4: 0x08
Channel 5: 0x10
Channel 6: 0x20

Examples:

For channel 1 and 2, this is 0x03
For all channels 1 to 6, the value is 0x3F

The volume is a 16-bit signed value. (value * 10)

For example:

+6.0dB has the value 60, in hex 0x003C
Volume_H: 0x00 and Volume_L: 0x3C

-40.0dB has the value -400, in hex 0xFE70
Volume_H: 0xFE and Volume_L: 0x70

Flags 0x01 is on, 0x00 mute.

*encoding of 0xFF

Set relative volume / mute

Master sends

```
1. Byte   Startbyte           0xF0
2. Byte   Device Address      0x01 .. 0xFE
3. Byte   Databyte Count      0x03
4. Byte   Command Byte        0x96  // command SetVolumeRelative
5. Byte   Channels            0xXX  // channels
6. Byte   Index               0x01
7. Byte   Volume_H            0xXX  // volume
8. Byte   Volume_L            0xXX  // volume
9. Byte   Flags               0xXX  // flags
```

Acknowledge from Fohhn Audio DSP (slave)

```
1. Byte   Device Address      0x01 .. 0xFE
2. Byte   Startbyte           0xF0 // end of reply
```

Example: relative volume change for device 1, channel 1 + channel 2

```
-1.0dB On:    0xF0 0x01 0x03 0x96 0x03 0x01 0xFF 0x01* 0xF6 0x01

+1.0dB On:    0xF0 0x01 0x03 0x96 0x03 0x01 0x00 0x0A 0x01
```

Example: output mute for device 1, channel 1 + channel 2

```
On (Unmute):  0xF0 0x01 0x03 0x96 0x03 0x01 0x00 0x00 0x05

Off (Mute):   0xF0 0x01 0x03 0x96 0x03 0x01 0x00 0x00 0x00
```

Notes

You can set multiple channels with one command.

Each channel has a value:

Channel 1: 0x01
Channel 2: 0x02
Channel 3: 0x04
Channel 4: 0x08
Channel 5: 0x10
Channel 6: 0x20

Examples:

For channel 1 and 2, this is 0x03
For all channels 1 to 6, the value is 0x3F

The volume is a 16-bit signed value. (value * 10)

Flags 0x01 is on, 0x00 mute
0x05 is switch on (unmute)

If you send 0x01, the channels stay on - but they will not unmute if they are muted.

For unmute send 0x05

*encoding of 0xFF

Set Routing

Master sends

```
1. Byte   Startbyte           0xF0
2. Byte   Device Address      0x01 .. 0xFE
3. Byte   Databyte Count      0x03
4. Byte   Command Byte        0x81  // command SetRouting
5. Byte   Channels            0xXX  // output channels
6. Byte   Index               0x0X  // input channel
7. Byte   Volume_H            0xXX  // gain
8. Byte   Volume_L            0xXX  // gain
9. Byte   Flags               0xXX  // flags
```

Acknowledge from Fohhn Audio DSP (slave)

```
1. Byte   Device Address      0x01 .. 0xFE
2. Byte   Startbyte           0xF0 // end of reply
```

Example: set routing for device 1, output channel 1

```
Input 1 0.0dB On:  0xF0 0x01 0x03 0x81 0x01 0x01 0x00 0x00 0x01

Input 3 0.0dB On:  0xF0 0x01 0x03 0x81 0x01 0x03 0x00 0x00 0x01

Input 1 0.0dB Off: 0xF0 0x01 0x03 0x81 0x01 0x01 0x00 0x00 0x00

Input 3 0.0dB Off: 0xF0 0x01 0x03 0x81 0x01 0x03 0x00 0x00 0x00
```

Notes

You can set multiple output channels with one command.

Each output channel has a value:

Output channel 1: 0x01
Output channel 2: 0x02
Output channel 3: 0x04
Output channel 4: 0x08
Output channel 5: 0x10
Output channel 6: 0x20

Examples:

For channel 1 and 2, this is 0x03
For all channels 1 to 6, the value is 0x3F

You can set one input channel with one command, only.

Input channel 1: 0x01
Input channel 2: 0x02
Input channel 3: 0x03
Input channel 4: 0x04

Input channels can not be combined.

The gain is a 16-bit signed value. (value * 10)

Flags 0x01 is on, 0x00 mute.

Set Standby

Master sends

```
1. Byte   Startbyte           0xF0
2. Byte   Device Address      0x01 .. 0xFE
3. Byte   Databyte Count      0x01
4. Byte   Command Byte        0x0C // command SetStandby
5. Byte   ADR_MSB             0x01
6. Byte   ADR_LSB             0x00
7. Byte   Databyte 1          0x0X // Standby
```

Acknowledge from Fohhn Audio DSP (slave)

```
1. Byte   Device Address      0x01 .. 0xFE
2. Byte   Startbyte           0xF0 // end of reply
```

Example: standby for device 1

```
Standby:    0xF0 0x01 0x01 0x0C 0x01 0x00 0x01

On:         0xF0 0x01 0x01 0x0C 0x01 0x00 0x00
```

ReadFunctionData

Data readback example for volume (command 0x87)

Device 1, volume for channel 1

Master send:

```
1. Byte Startbyte          0xF0
2. Byte Device Adress      0x01
3. Byte Databyte Count     0x01
4. Byte Command Byte       0x0A
5. Byte Channel            0x01
6. Byte Index              0x01
7. Byte Command            0x87
```

Slave returns (-0.2dB) in this example:

```
1. Byte <CB>               0xFF // volume
2. Byte Volume_H           0x01 // volume
3. Byte Volume_L           0xFE // volume
4. Byte Flags              0x01 // flags
5. Byte Device Address     0x01
6. Byte Endbyte            0xF0
```

Notes

With the command ReadFunctionData you can readback DSP functions. You have to send the ReadFunctionData command and add the command number from the „Set-Command“ for the specific function as data byte 1.

This way you can readback all dsp functions.