Fohhn-Net Technical Manual

1 Introduction

This manual is intended for users who want to develop their own media control drivers for direct **Fohhn-Net** access.

1.1 Fohhn-Net basics

Fohhn-Net is based on RS-485. We recommend you to equip your media control with a RS-485 (EIA-485) module in order to connect to and control **Fohhn-Net** devices. In doing so, you can also control several **Fohhn-Net** devices with just one RS-485 module, since signals can be linked through to the next device (daisy chain).

1.2 Interface parameters

Fohhn-Net uses a half-duplex 2-wire RS-485 connection. Most of the **Fohhn-Net** devices have RJ-45 connectors or terminal blocks as interface socket.

1.3 Electrical parameters

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

1.4 Fohhn-Net serial RS-485 parameters

- 19200 baud
- 1 start bit
- 8 data bits
- 1 stop bit
- no parity
- no flow control

Some systems only allow baud-rate, start bit and data bit configurations. Current systems only use one stop bit as default.

1.5 Cable types

Most modern installations will require twisted pair CAT-7 cables. However, the RS-485 standard will also support other types of cables:

- twisted pair CAT-5 or CAT-7
- J-Y(ST)Y 2 x 2 x 0.6 (2 x 2 x 0.6 mm – 0.8 mm) – often used for ISDN infrastructures
- microphone cables

Please make sure that no analogue audio signal is transferred via the same cable, for you might hear some interference caused by the data signal.

1.6 Large installations

If you have more than 32 devices, you must use several separate RS-485 lines. However, even if you have less than 32 devices, it may be useful to split the system into multiple RS-485 lines for easier handling and faster response.

1.7 NA-3 Fohhn-Net Ethernet adapter

If mere IP based communications are possible, you can use **Fohhn® NA-3 Ethernet Adapters** as a bridge between Ethernet and **Fohhn-Net**. The **NA-3** and the **Fohhn-Net** devices have no authentication possibilities, thus you have to make sure that the **NA-3** can only be reached via your control devices and your control PC (with a dedicated Ethernet or a port based VLAN configuration, for example). Each **NA-3** requires a fixed IP address and your control device to be in the same IP range (the **NA-3** cannot reply via router). To avoid communication drop outs, you have to make sure that the **NA-3** is not flooded with broadcast packages from other applications (the **NA-3** has a 10 Mbit Ethernet connection). Identical binary data must be sent as a UDP package to both the IP address of the **NA-3** and UDP port 2101. In turn, the **NA-3** adapter transfers the data via its RS-485 port to the respective **Fohhn-Net** devices. The adapter also sends a reply from the **Fohhn-Net** device as a UDP package to both the source IP address and the source port of the last request. The **NA-3** uses the **Fohhn-Net** ID 1 as default. For your other **Fohhn®** devices, such as controllers or amplifiers, you will have to change its ID or use a **Fohhn-Net** ID other than 1.

1.8 Devices with integrated Dante input interfaces (Brooklyn II module only)

Set up a static IP address to the Dante input interface.

Identical binary data must be sent as a UDP package to both the IP address of the Dante input interface and UDP port 2101. In turn, the Dante input interface transfers the data to the device. The adapter also sends a reply from the **Fohhn®** device as a UDP package to both the source IP address and the source port of the last request.

However, we recommend the use of a text-based TCP protocol. For more information, please see the **Fohhn-Net TCP Text Protocol Instructions** document.

2 The Fohhn-Net Protocol

2.1 Basics

Fohhn-Net is a binary protocol. In this document, each byte is represented as a hex value, marked with 0x (the hex value 0xF0 is equivalent to the decimal value 240, for example). The byte order for 16-bit values is always big-endian. As Fohhn-Net is not based on strings, you don't have to send any carriage return or line feed characters at the end.

2.2 Fohhn-Net structure

A command consists of seven bytes at the minimum, a six bytes header and at least one data byte. The header begins with the start byte <SB> followed by the device address, data byte count, the command and two address bytes.

```
1. Byte Startbyte       <SB>
2. Byte Device ID       <ID>
3. Byte Databyte Count <COUNT>
4. Byte Command Byte    <CMD>
5. Byte Address MSB     <ADR_MSB>
6. Byte Address LSB     <ADR_LSB>
7. Byte Databyte 1      <DATA>      // min. one databyte
N. Bytes Optional       <DATA>
```

The device replies with at least two bytes, its device address and the start byte as finish. Depending on the command, the device replies with some data bytes before the last two bytes. If the reply is not received within 350 ms, you can presume a timeout. Your media control system must always wait for the reply or a timeout before sending a command to the next device. Please note that **Fohhn-Net** is a serial protocol, whereby a reply from the device is not guaranteed. Your control system should be fault-tolerant and retry each command three times if it does not receive a reply. If the control system does not receive a reply for three consecutive times, one can assume that the device is not reachable. Make always sure that the system waits for at least 350 ms before trying again.

```
N   Data bytes          <DATA>   // optional
N+1 Byte Device ID      <ID>
N+2 Byte Startbyte      <SB>
```

2.3 Encoding of reserved bytes

As the **Fohhn-Net** protocol has a start byte <SB> and a control byte <CB>, both must never occur in one message. If the data (each byte, except the start byte) contains the value of a start byte or a control byte, it must be encoded with the control byte <CB>.

- The value of the start byte <SB> is 0xF0.
- The value of the control byte <CB> is 0xFF.

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

- <CB> + 0x00 = 0xF0
- <CB> + 0x01 = 0xFF

The control byte does not count as an extra byte in data byte count. In the same way, the device encodes its reply data, with the exception of the last byte <SB>.

2.4 Data types

Data type	**Size**	**Min**	**Max**
unsigned byte	8-bit	0	255
signed byte	8-bit	-128	127
unsigned word	16-bit	0	65.535
signed word	16-bit	-32.768	32.767

3 Presets

3.1 Load a user preset

A user preset contains all DSP settings of a device. A preset always loads ALL DSP settings: One cannot load just certain parts of a preset. A user preset does not include a speaker preset or an amplifier preset; these system settings are not exchangeable with user presets.

Command structure for preset loading

```
Startbyte         0xF0
Device ID         0x01 .. 0xFE
Databyte Count    0x01
Command Byte      0x05
ADR_MSB           0x01
ADR_LSB           0x01 .. 0x64          // preset nr.
Databyte 1        0x00
```

Reply from device

```
Device ID 0x01 .. 0xFE
Startbyte 0xF0
```

Examples: load preset 20 – 24 for device 1

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

Examples: load preset 20 – 24 for device 2

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

4 Device settings

4.1 Change device standby

Most devices switch off the amplifier section in standby mode.

Command structure for setting device standby

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x01
Command     0x0C // command Standby
ADR_MSB     0x00
ADR_LSB     0x00
Flags       0xXX // Standby
```

Reply from device

```
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

Example for changing standby for device 1

```
0xF0 0x01 0x01 0x0C 0x00 0x00 0x01 // Standby
0xF0 0x01 0x01 0x0C 0x00 0x00 0x00 // On
```

4.2 Get device standby

(Not available on all devices!)

Command structure for getting device standby

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x01
Command     0x0A // command Readback
ADR_MSB     0x00
ADR_LSB     0x00
DATA        0x0C // get Standby
```

Reply from device

```
Flags       0xXX // Standby Flag
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

5 Check status information

5.1 Get device information

Device request command structure

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x01
Command     0x20 // command GetDeviceInfo
ADR_MSB     0x00
ADR_LSB     0x00
DATA        0x01
```

Reply from device

```
CLASS_H     0xXX
CLASS_L     0xXX
VERSION0    0xXX
VERSION1    0xXX
VERSION2    0xXX
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

5.2 Read signals

Command structure to request signal levels

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x01
Command     0x8D
ADR_MSB     0x00
ADR_LSB     0x00
DATA        0x00
```

Reply from device

```
OUTPUT1     0xXX
.
.
OUTPUT6     0xXX
INPUT1      0xXX
.
.
INPUT4      0xXX
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

Signal reply depends on the number of output and input channels. A device with one input channel and one output channel only returns two bytes, a device with six output channels and four input channels returns ten bytes. Each byte (signed value) represents the gain in dB.

5.3 Read protect and temperature

Command structure to request protect and temperature

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x01
Command     0x07
ADR_MSB     0x00
ADR_LSB     0x00
DATA        0x00
```

Reply from device

```
PROTECT     0xXX
TEMPH       0xXX
TEMPL       0xXX
OPT         0xXX
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

PROTOTECT byte structure

```
BIT0 // Protect Status for Channel 1
BIT1 // Protect Status for Channel 2
BIT2 // Protect Status for Channel 3
BIT3 // Protect Status for Channel 4
```

The PROTECT byte contains bit flags. A high bit means a failure. **AIREA®** devices use the second bit for AES/EBU valid; **Linea Focus** and **Focus Modular** devices (such as **LFI-120/220/350/450** and **FMI-100/110/400**) use the second bit for pilot tone status. TEMP is a 16-bit signed value for temperature; you have to divide the value by 10 to get the exact temperature in °C. OPT is used for **AIREA-Net** voltage.

6 Set DSP parameters

Each DSP channel has a specific value. These channel values are valid for **FC-8**, **FC-9**, **D-Series** amplifiers, current **Linea Focus and Focus Modular** speaker systems, **AIREA** system extensions and active **Fohhn®** subwoofers.

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

6.1 Set absolute volume

Command structure for setting absolute volume

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x03
Command     0x87        // command SetVolume
Channels    0xXX        // channels
Index       0x01
VolumeH     0xXX        // volume
VolumeL     0xXX        // volume
Flags       0xXX        // flags
```

Reply from device

```
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

Examples for setting volume for device 1 on channel 1

```
0xF0 0x01 0x03 0x87 0x01 0x01 0x00 0x00 0x01          // 0.0 dB On
0xF0 0x01 0x03 0x87 0x01 0x01 0x00 0x00 0x00          // 0.0 dB Mute
0xF0 0x01 0x03 0x87 0x01 0x01 0xFF 0x01 0xB5 0x01     // -7.5 dB On
0xF0 0x01 0x03 0x87 0x01 0x01 0x00 0x3C 0x01          // +6.0 dB On
0xF0 0x01 0x03 0x87 0x01 0x01 0xFE 0x70 0x01          // -40.0 dB On
```

The volume is a 16-bit signed value (value * 10). +6.0 dB has a value of 60, in hex 0x003C, VolumeH 0x00 and VolumeL 0x3C. -40.0 dB has a value of -400, in hex 0xFE70, VolumeH 0xFE and VolumeL 0x70. Flag 0x01 is on, 0x00 is mute. The flag 0x02 will invert the output, thus 0x02 is off and inverted, and 0x03 is on and inverted.

6.2 Set relative volume

A relative volume change is useful for volume up and down buttons. It's not possible to set a range.

Command structure for relative volume settings

```
Startbyte   0xF0
Device ID   0x01 .. 0xFE
Count       0x03
Command     0x96 // command SetVolumeRelative
Channels    0xXX // channels
Index       0x01
VolumeH     0xXX // volume
VolumeL     0xXX // volume
Flags       0xXX // flags
```

Reply from device

```
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

Examples for relative volume change for device 1 on channel 1

```
0xF0 0x01 0x03 0x96 0x01 0x01 0xFF 0x01 0xF6 0x01    // -1.0 dB On
0xF0 0x01 0x03 0x96 0x01 0x01 0x00 0x0A 0x01         // +1.0 dB On
```

Set output mute

The relative volume command can be used for an output mute without changeing the volume. Just use a relative volume change of zero.

Examples for output mute for device 1 on channel 1

```
0xF0 0x01 0x03 0x96 0x01 0x01 0x00 0x00 0x05   // On ( Unmute )
0xF0 0x01 0x03 0x96 0x01 0x01 0x00 0x00 0x00   // Off ( Mute )
```

Flag 0x01 is on, 0x00 is mute and 0x05 is switching on (unmute). If you send 0x01, the channels stay on, but will not unmute if they are already muted. In order to unmute them, send 0x05.

6.3 Set routing and input gain

Command structure for set routing

```
Startbyte         0xF0
Device ID         0x01 .. 0xFE
Count             0x03
Command           0x81 // command SetRouting
Channels          0xXX // channels
Index             0x0X // input channel
GainH             0xXX // gain
GainL             0xXX // gain
Flags             0xXX // flags
```

Reply from device

```
Device ID   0x01 .. 0xFE
Startbyte   0xF0
```

Channel values for input channels:

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

Examples for routing settings for device 1 on output channel 1

```
0xF0 0x01 0x03 0x81 0x01 0x01 0x00 0x00 0x01 // Input 1 0.0 dB On
0xF0 0x01 0x03 0x81 0x01 0x02 0x00 0x00 0x01 // Input 2 0.0 dB On
0xF0 0x01 0x03 0x81 0x01 0x01 0x00 0x00 0x00 // Input 1 0.0 dB Off
0xF0 0x01 0x03 0x81 0x01 0x02 0x00 0x00 0x00 // Input 2 0.0 dB Off
```