4.12.3 Analog Output

There is a 24-pin Centronics connector providing 16 channels analog output on the Main Unit. In the NeurOne PC Software, the user can define the channels or the difference between two channels to be directed to the analog output. See Appendix 3 for the pinout of the analog output connector.

The output range of the NeurOne analog output is 0 – 2.5 V (zero level 1.25 V). For making your own analog output cable, please refer to Appendix 3 or contact Bittium Biosignals for more information.

External devices connected to Amplifier shall be always connected via EN/IEC 60601-1 approved 4 kV isolator!

Failure to do this may cause electrical shock and serious injuries!

4.12.4 Digital OUT

NeurOne system can be configured to emit digital measurement data via an optional add-on Ethernet port called “Digital OUT”. This is ideally suited for real-time solutions that require data with minimal latency.

4.12.4.1 Specification

The following parameters can be configured by the user:

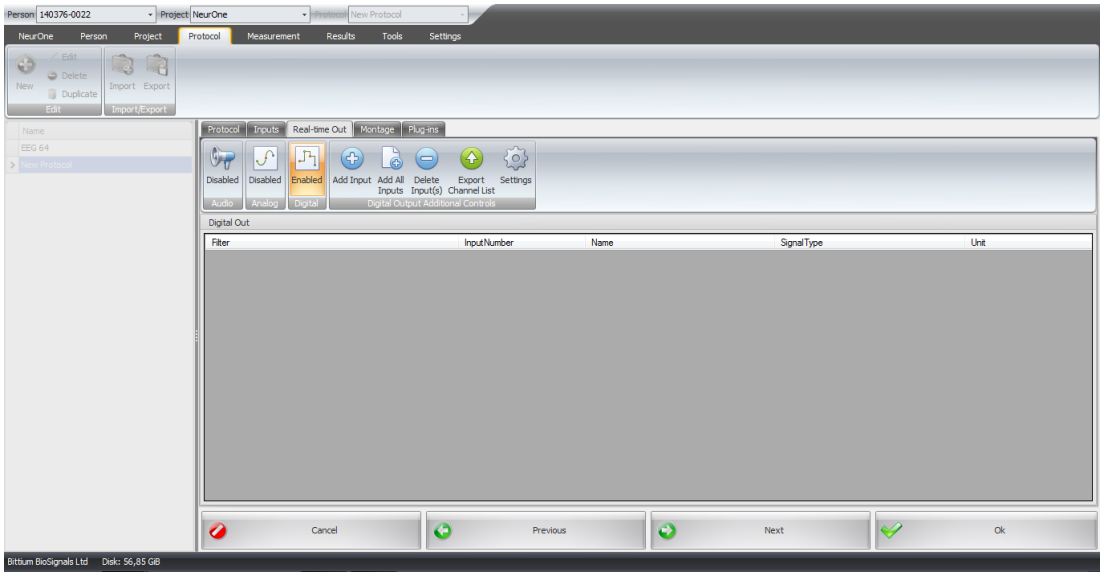
- Inputs that are directed to digital out port
- Destination IP address and UDP port for emitted digital out packets
- Digital out packet delivery frequency

The configuration itself is dependent on the user application receiving real-time data from NeurOne Main Unit is thus out of scope of this manual.

4.12.4.2 Features

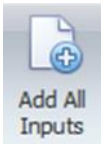
The emitted data is delivered at the sampling rate the measurement is running (i.e. there’s no additional downsampling performed), at regular intervals. The chosen delivery rate (i.e. how often UDP datagrams are emitted) depends on measurement configuration and cannot be higher than the measurement sampling rate. User can request delivery rates of 100 Hz, 250 Hz, 500 Hz, 1 kHz, 2 kHz, 3 kHz, 4 kHz and 5 kHz.

Digital Out:



If enabled Digital Output provides additional control buttons on the tool bar:

Add the input from defined input list of the protocol to the Digital OUT list. The inputs on that list will be transferred through Digital OUT. Inputs are added in order as they are defined in the input list



Add all inputs from defined input list of the protocol to be the Digital OUT list.

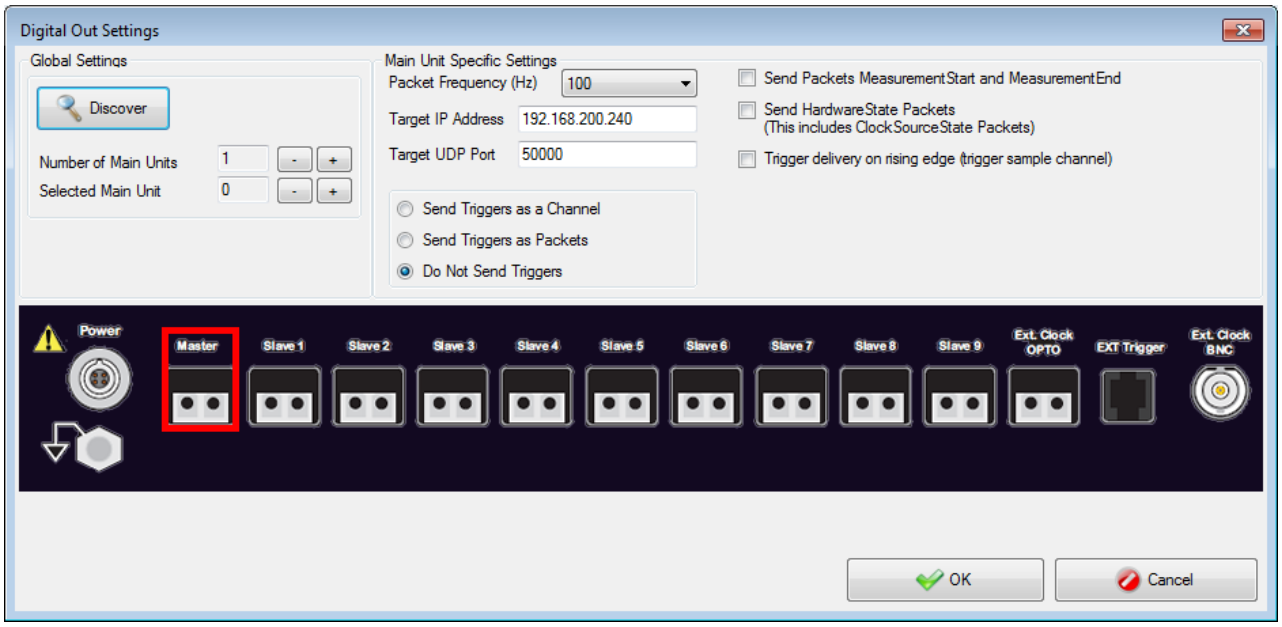
Delete selected input from the Digital OUT list.

Exports the list of Digital OUT channels as *.txt file to the desired target folder.

Settings of Digital OUT.

5.3.4.1 Digital OUT settings

The UI of Digital OUT settings contains the graphic presentation of SyncBox rear panel for Main Unit selection.



Discover button scans the NeurOne system in order to find out if SyncBox is available and the number of Main Units in the system and updates that parameter on the UI.

Select the desired Main Unit for Digital OUT definition (the parameter: Selected Main Unit). Red rectangle indicates the selection. Main Unit number of Master = 0 and unit number 1…9 stands for Slave 1… Slave 9.

In case there is only one Main Unit (standalone) without SyncBox in the system, the Master is the only available choice on the UI and Number of Main Units = 1.

Main Unit specific Settings

Packet frequency (Hz) The emitted data is delivered at the sampling rate the measurement is running (i.e. there's no additional downsampling performed), at regular intervals. The chosen delivery rate (i.e. how often UDP datagrams are emitted) depends on measurement configuration. User can request delivery rates of 100 Hz, 250 Hz, 500 Hz, 1 kHz, 2 kHz, 3 kHz, 4 kHz and 5 kHz. The higher the delivery rate, the less latency there'll be.

Target IP Address IP Address of the receiving real-time system where the Digital OUT is transmitting the data packets to.

Make sure that IP address definition in is defined correspondingly in TCP/IP settings of TARGET PC!

Make sure that Digital OUT cable is connected to the target PC prior to system start-up.

Target UDP Port UDP Port of receiving real-time system.

CheckBox: **Send packets MeasurementStart and MeasurementEnd**

If checked, MeasurementStart and MeasurementEnd packets are sent.

MeasurementStart packet is sent by NeurOne when it starts measuring, before any measurement data (samples / triggers) is emitted and also upon reception of Join* packet.

MeasurementEnd packet is sent by NeurOne to indicate the end of the measurement. It's the last packet emitted when measurement hardware is stopped.

* The Join packet is sent by the system receiving NeurOne digital out data. Main purpose of this packet is to enable MeasurementStart packet re-emission by NeurOne in cases where the receiver system comes online after NeurOne measurement has already been started.

CheckBox: **Send HardwareState Packets**
(This includes ClockSourceState Packets)

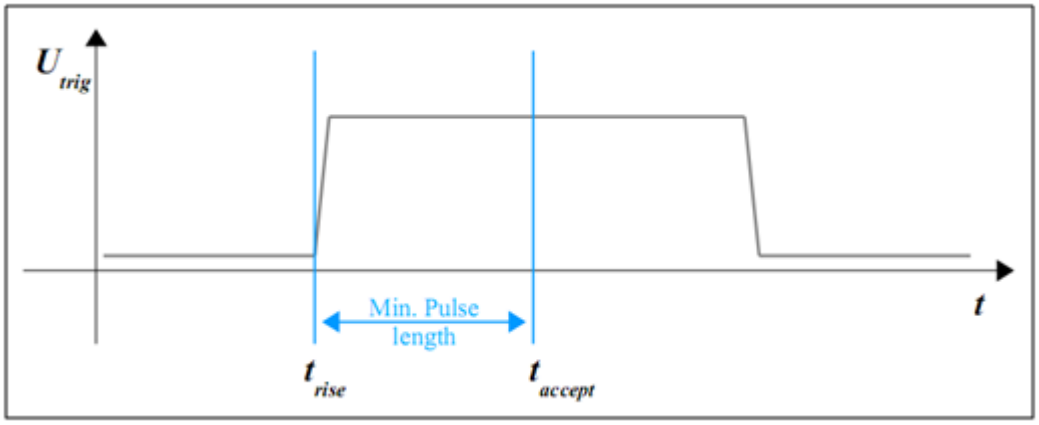
If checked, HardwareState packet is sent.

HardwareState packet relays information about measurement hardware state, which can be considered metadata in regards to the measurement. It's sent by applicable NeurOne main units at the start of the measurement after MeasurementStart packet before any measurement data (samples / triggers) is emitted and also upon reception of a Join packet. Note that this packet can contain different payloads which depend on the situation.

CheckBox: **Trigger delivery on rising edge (trigger sample channel)**

If checked and triggers are sent as a channel, trigger delivery takes place at the site of rising edge of incoming trigger pulse.

Sending triggering information among the samples poses an additional problem; after a rising edge on a trigger port the hardware must wait for a specified time to ensure the configured trigger pulse length is satisfied (see figure below). If trigger must occur on the sample of the rising edge the system must stall digital out sample delivery in order to deliver the trigger on the correct sample (t_{rise}). Alternatively, the trigger can be delivered on the sample it is accepted on, i.e. the sample when trigger pulse length is satisfied on (t_{accept}). The former causes jitter in sample packet delivery timing. The latter causes trigger to be sent late in relation to the rising edge.



Settings of sending triggers

There are three alternatives for trigger sending:

- Send Triggers as Packets
- Do Not Send Triggers
- Send Triggers as a Channel

Update firmware… initiates the firmware update procedure to NeurOne Main Unit.

NeurOne Manager table view shows all devices or only Main Units or Amplifiers.

Refresh devices update the table view.

Report copies the information on the table to clipboard.

5.8.2 Remote Control

NeurOne TCP Remote Control is a new feature introduced in NeurOne 1.4.1.39. It enables remote controlling (start recording, stop recording, stop session) of a running measurement and supports multiple clients. All connected clients are notified of measurement state changes (Idle, Monitoring, Recording).

5.8.2.1 Remote control settings

The Remote control has the following settings:

Enabled – Enables or disables the Remote Control component. If the server is disabled, it's not listening for connections. It's advisable to disable the server for security reasons if it's not being used. Defaults to false.

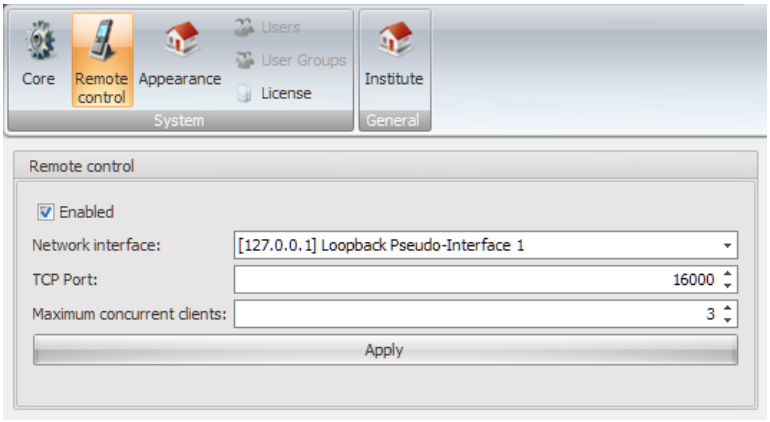
Network Interface - The network interface the remote control listens to. If you control NeurOne from the same computer, choose 127.0.0.1. Else choose the interface that's connected to your LAN. This setting has no default value.

TCP Port – Length of output pulse from trig out.

Maximum concurrent clients – Maximum number of clients that can be connected to the server at any given time. Allowed range is 1...10. If maximum number of clients is connected the server will immediately disconnect any new clients trying to connect.

Apply – Immediately applies the settings. Remote control is restarted with the new settings. Any previously connected clients will be disconnected.

The network interface selection input shows only interfaces that are operational and support IPv4 (i.e. network cable connected and have an IPv4 address).



For further information about Remote Control refer to Appendix 9.

Appendix 9 REMOTE CONTROL

Supported commands

Command protocol is line-based. Supported line terminators are CR (`\r`), LF (`\n`) or CRLF (`\r\n`). The used text encoding is ASCII. One line constitutes one command. The server responds to a single command with a single line, terminated by a CRLF.

Single command length is limited to 1000 characters. If a longer command is received the server terminates the client connection immediately.

For most commands the server immediately responds by sending

`OK:`*`commandText`*`\r\n`

where *commandText* is the command initially sent by the client (see command examples below). E.g.

```
OK:RECSTART\r\n
```

If the server encounters an error while processing the given command (e.g. unknown command or wrong session state) it responds with an error:

`ERROR:`*`errorIdentifier`*`:`*`description`*`\r\n`

where *errorIdentifier* is a textual identifier for error reason and *description* is a human readable more specific reason for the error; e.g.

```
ERROR: StateNotMonitoring:To start recording the system needs to
be in monitoring state.\r\n
```

List of all commands is shown below:

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>SESSTART</td><td>Starts a session. Corresponds to clicking the “Start” button in session toolbar. Using simulation is not supported.

List of all parameters is shown below:

<table>
<tr><th>Parameter name</th><th>Parameter value</th><th>Description</th></tr>
<tr><td>Person</td><td><Person Id></td><td>Selects the person by given person id. Case insensitive.</td></tr>
<tr><td>Project</td><td><Project name></td><td>Selects the project by given project name. Case insensitive.</td></tr>
<tr><td>Protocol</td><td><Protocol name></td><td>Selects the protocol by given protocol name. Case insensitive.</td></tr>
</table>
</td></tr>
<tr><td>RECSTART</td><td>Starts recording. Corresponds to clicking the “Start” button in record toolbar.</td></tr>
<tr><td>RECSTOP</td><td>Stops recording. Corresponds to clicking the “Stop” button in record toolbar.</td></tr>
<tr><td>SESSTOP</td><td>Stops the monitoring session. Corresponds to clicking the “Stop” button in session toolbar.</td></tr>
<tr><td>IMPSTART</td><td>Starts impedance test. Corresponds to clicking the “Start” button in impedance test toolbar. Using simulation is not supported.

If impedance test is started without starting the session first, then startup parameters are required (see SESSTART command above).</td></tr>
<tr><td>IMPSTOP</td><td>Stops impedance test. Corresponds to clicking the “Stop” button in impedance test toolbar.</td></tr>
<tr><td>STATUS</td><td>Responds with the current session state. See session state descriptions below.

Response format is: STATE:<StateName>[*].

The client can use this command to query the current session state e.g. right after it has connected. If the StateName is succeeded by an asterisk it means that the system is currently changing state.</td></tr>
</table>

MINIMIZE	Minimizes the application window.
MAXIMIZE	Maximizes the application window.
HIDE	Hides the application window.
SHOW	Shows the application window.
QUIT	Quits the application. Corresponds to clicking the "x" button in the application window.

Session state	Description
Idle	A measurement isn't running.
Monitoring	The measurement hardware is running and supplying data. Nothing is being recorded yet.
Recording	The measurement hardware is running and PC software is recording data.
Monitoring+Impedance	Monitoring, impedance test active
Recording+Impedance	Recording, impedance test active

Error identifier	Description
UnknownError	Reason for the error is unknown.
CommandUnknown	An unknown command was issued.
StateInTransition	A state transition is already in progress.
StateNotIdle	The system needs to be in idle state for the given command.
StateNotMonitoring	The system needs to be in monitoring state for the given command.
StateNotRecording	The system needs to be in recording state for the given command.
StateNotTestingImpedance	The system needs to be in impedance test state for the given command.
SessionStartError	Session couldn't be started.
ImpedanceStartError	Impedance test couldn't be started.
InvalidCommandParam	Invalid command parameter was given.
ParamMissing	Mandatory parameter is missing.
PersonNotFound	Given person wasn't found.

`ProjectNotFound`	Given project wasn't found.
`ProtocolNotFound`	Given protocol wasn't found.

Command examples

Here blue text denotes the commands sent by the client and red text responses sent by the server. Green text is comments.

Please note that the characters `\r\n` in the following examples are not to be input literally, but instead represent a line terminator (e.g. pressing of ENTER key if you're accessing the functionality with a terminal program).

Command example 1: Start a session

Client issues session start with parameters:
Client: `SESSTART person="New Person", project="New Project", protocol="New Protocol"\r\n`
When the session has been started the server notifies of state change:
Client: `STATUS:Monitoring\r\n`
Server responds that the command is processed:
Client: `OK:SESSTART\r\n`

Command example 2: Start recording

Client issues recording start:
Client: `RECSTART\r\n`
Server immediately responds that the command is being processed:
Client: `OK:RECSTART\r\n`
Later when the recording has been started the server notifies of state change:
Client: `STATUS:Recording\r\n`

Command example 3: Stop recording (with 2 clients)

Client 1: `RECSTOP\r\n`
Client 1: `OK:RECSTOP\r\n`
Client 1: `STATUS:Monitoring\r\n`
The server also notifies client 2 of session state change:
Client 2: `STATUS:Monitoring\r\n`

Command example 4: Stop the session

Client: `SESSTOP\r\n`
Client: `OK:SESSTOP\r\n`
Client: `STATUS:Idle\r\n`

Command example 5: Start impedance test during the session

Client issues impedance test start:

Client: `IMPSTART\r\n`
When impedance test has been started the server notifies of state change:
Client: `STATUS:Monitoring+Impedance\r\n`
Server responds that the command is processed:
Client: `OK:IMPSTART\r\n`

Command example 6: Start impedance test when the session is not running

Client issues impedance test start with parameters:
Client: `IMPSTART person="New Person", project="New Project", protocol="New Protocol"\r\n`
When impedance test has been started the server notifies of state change:
Client: `STATUS:Monitoring+Impedance\r\n`
Server responds that the command is processed:
Client: `OK:IMPSTART\r\n`

Command example 7: Stop impedance test

Client: `IMPSTOP\r\n`
Client: `OK:IMPSTOP\r\n`
Client: `STATUS:Monitoring\r\n`

Command example 8: Status query

Client requests current state information:
Client: `STATUS\r\n`
Server responds that it's currently idle:
Client: `STATUS:Idle\r\n`

Command example 9: Minimizing the application window

Client: `MINIMIZE\r\n`
Client: `OK:MINIMIZE\r\n`

Command example 10: Maximizing the application window

Client: `MAXIMIZE\r\n`
Client: `OK:MAXIMIZE\r\n`

Command example 11: Hiding the application window

Client: `HIDE\r\n`
Client: `OK:HIDE\r\n`

Command example 12: Showing the application window

Client: `SHOW\r\n`
Client: `OK:SHOW\r\n`

Command example 13: Quitting the application

Client: `QUIT\r\n`
Client: `OK:QUIT\r\n`

Command example 14: Session start fails

Client issues session start with parameters:
Client: `SESSTART person="New Person", project="New Project", protocol="New Protocol"\r\n`
Server immediately responds that the command cannot be executed because no main units seem to be present:
Client: `ERROR:SessionStartError: No main units seem to be present. Please check the Ethernet cabling, your TCP/IP settings, and ensure that amplifiers are powered on.\r\n`

Command example 15: Recording start fails

Client issues recording start:
Client: `RECSTART\r\n`
Server immediately responds that the command cannot be executed because the measurement system isn't running:
Client: `ERROR:StateNotMonitoring: To start recording the system needs to be in monitoring state.\r\n`

Appendix 10 DIGITAL OUT

Features

The data is delivered at the sampling rate the measurement is running at regular intervals. The chosen delivery rate (i.e. how often UDP datagrams are emitted) depends on measurement configuration. User can request delivery rates of 100 Hz, 250 Hz, 500 Hz, 1 kHz, 2 kHz, 3 kHz, 4 kHz and 5 kHz. Data packets cannot be delivered at 1-ms intervals if measurement sampling rate is below 1 kHz.

NeurOne digital out Ethernet port will be assigned an IP address relative to the control Ethernet port IP address. The control Ethernet port address is in turn determined by connection to SyncBox:

Main unit	Control IP address	Digital Out IP address
Stand-alone	192.168.200.200	192.168.200.220
SyncBox master	192.168.200.101	192.168.200.121
SyncBox slave1	192.168.200.102	192.168.200.122
SyncBox slave2	192.168.200.103	192.168.200.123
...		
SyncBox slave 9	192.168.200.110	192.168.200.130

Triggering information can be transmitted in packet or sample mode (or both). If packet mode transmission is enabled digital out will emit Triggers packets. If sample mode transmission is enabled digital out will emit an extra sample channel as the last channel in Samples packet.

There is no receival acknowledged mechanism for digital output. NeurOne handles only a single type of packet sent to its digital output interface (the Join packet). All other packets received by NeurOne are discarded.

UDP datagram length is limited by Ethernet MTU (IP layer fragmentation isn't supported). Each packet has at most 1472 bytes.

Packet types

There are many kinds of packets that NeurOne can emit to digital out. The emission of most of these can be enabled / disabled, depending on the actual needs of the user and the capabilities of the system receiving digital out data.

Packet type specifics are described in the following subchapters. Every field is encoded in big-endian byte order (i.e. most significant byte first). A packets type is identified by its first

byte. This way the receiving system may trigger conditional parsing based on packet type – or even decide to discard certain packets.

The data types referenced in packet type descriptions are:

Type	Description
`int8`	8-bit signed integer
`uint8`	8-bit unsigned integer
`int16`	16-bit signed integer
`uint16`	16-bit unsigned integer
`int32`	32-bit signed integer
`uint32`	32-bit unsigned integer
`int64`	64-bit signed integer
`uint64`	64-bit unsigned integer
`t[N]`	Array of values, each of type *t*. If *N* is specified it denotes array length. If N isn't given (i.e. "`[]`") the array is variable-length.

MeasurementStartPacket

This packet is sent by NeurOne when it starts measuring, before any measurement data (samples / triggers) is emitted and also upon receival of Join packet. The emission of this packet is optional and may be enabled / disabled per-measurement. By default this packet isn't emitted.

The MeasurementStart packet structure is shown below:

Type	Name	Description
`uint8`	`FrameType`	1 = MeasurementStart
`uint8`	`MainUnitNum`	Indicates which main unit the packet is coming from (see MainUnitNum field in Samples packet).
`uint8[2]`	`Reserved`	Reserved for future use
`uint32`	`SamplingRateHz`	Sampling frequency (Hz)
`uint32`	`SampleFormat`	Currently this field has a fixed value (`0x80000018`),
`uint32`	`TriggerDefs`	Indicates which trigger type is assigned to each source port. The type of each port is expressed as a value encoded in 3 bits: Bits / Trigger port 0-2 Isolated A 3-5 Isolated B 6-8 Parallel trigger 9-11 SyncBox button 12-14 SyncBox external Value meaning: Value / Trigger type 0 Disabled 1 Stimulus 2 Video 3 Mute 4 Parallel

<table>
<tr><td></td><td></td><td><table><tr><td>5-7</td><td>Reserved</td></tr></table></td></tr>
<tr><td>uint16</td><td>NumChannels</td><td>Number of active digital out channels.</td></tr>
<tr><td>uint16[N]</td><td>SourceChannels</td><td>Variable-length array of input numbers, indicating which analog input is mapped to each active digital out channel.
(N = NumChannels).
E.g.
SourceChannels[0] = 2 => digitalOut1 = input2
SourceChannels[1] = 5 => digitalOut2 = input5
SourceChannels[2] = 4 => digitalOut3 = input4
SourceChannels[3] = 65535 => digitalOut4 = triggers

Main units may transmit input numbers as follows<table><tr><th>Main unit</th><th>Input range</th><th>Trigger Ch#</th></tr><tr><td>Stand-alone</td><td>1..160</td><td>65535</td></tr><tr><td>SyncBox master</td><td>1..120</td><td>65534</td></tr><tr><td>SyncBox slave 1</td><td>121..240</td><td>65533</td></tr><tr><td>SyncBox slave 2</td><td>241..360</td><td>65532</td></tr><tr><td colspan="3">…</td></tr><tr><td>SyncBox slave 9</td><td>1080..1200</td><td>65524</td></tr></table></td></tr>
<tr><td>uint8[N]</td><td>ChannelTypes</td><td>Variable-length array of input channel metadata (N = NumChannels). Each 8-bit entry is as follows:<table><tr><th>Bits</th><th>Meaning</th></tr><tr><td>0-2</td><td>Channel type (AC / DC)
000: AC
001: DC
Rest of the values are reserved</td></tr><tr><td>3-4</td><td>Amplifier type (EXG / Tesla)
00: EXG
01: Tesla
Rest of the values are reserved</td></tr><tr><td>5-7</td><td>Reserved</td></tr></table>Trigger channel is an exception to the channel type presented above; its channel type is 0x80 (1000 0000$_2$).</td></tr>
</table>

Samples Packet

This packet contains measured digital samples. The samples are delivered in bundles. A bundle contains a sample for every channel for the same time instant, and cannot be broken into multiple sample packets. In other words, the samples are delivered as channel-interleaved and the samples of every channel from a single moment in time are delivered within the same sample packet.

This packet also describes the sample index and time stamp for the first sample contained. The Samples packet structure is shown below:

Type	Name	Description

uint8	1	FrameType	2 = Samples
uint8	2	MainUnitNum	Indicates which main unit the samples packet is coming from: 0: Stand-alone 1: SyncBox master 2: SyncBox slave 1 … 10: SyncBox slave 9
uint8[2]	3, 4	Reserved	Reserved for future use
uint32	5-8	PacketSeqNo	Packet sequence number (incremented by 1 for each sample packet sent)
uint16	9, 10	NumChannels	Number of channels (constant throughout the measurement)
uint16	11, 12	NumSampleBundles	Number of sample bundles in this packet
uint64	13-20	FirstSampleIndex	Sample index of first bundle in this packet
uint64	21-28	FirstSampleTime	Timestamp of first bundle in this packet
int24[N][C]		Samples	Sample bundles; each bundle contains 1 sample from each channel. (N = NumSampleBundles, C = NumChannels).

It should be noted that the sample values may need to be scaled by the receiver. The sample scaling factor depends on the amplifier hardware (amplifier type, input type (AC / DC)). This information is supplied in the MeasurementStart packet. The scaling factors (i.e. dividers) are shown below:

Amplifier type	Channel type	Scaling factor
EXG	AC	1
EXG	DC	100
Tesla	AC	20
Tesla	DC	100

Send Triggers as a Channel Mode

When Send Triggers as a Channel mode is enabled the samples packet will contain an additional sample channel (always the last channel in the packet). The triggering information contains only the source port the trigger occurred on. Trigger types are defined in MeasurementStart packet. When a trigger occurs its corresponding bit will be high for the duration of 1 sample in the sample channel.

As triggers are relayed as samples, the temporal accuracy of triggering information is dictated by the measurement sampling rate. If the same trigger occurs multiple times during a single sample triggering information will be lost. If more accurate timing information is needed, it's advisable to use the Triggers packet which enables microsecond accuracy.

If trigger must occur on the sample of the rising edge the system must stall digital out sample delivery in order to deliver the trigger on the correct sample (t_{rise}). Alternatively, the trigger can be delivered on the sample it is accepted on, i.e. the sample when trigger pulse length is satisfied on (t_{accept}) (see Figure 1).

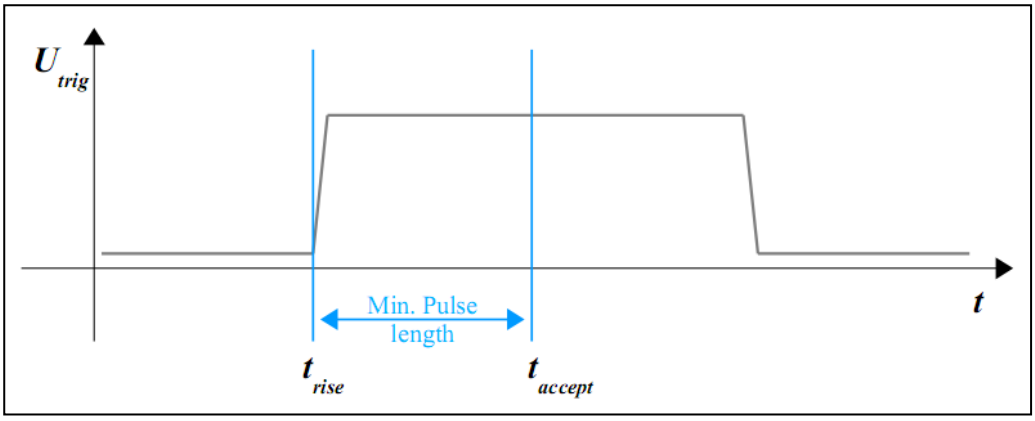


Figure 1. Trigger signal rising edge and accept timing.

Each trigger sample has the following information:

Bits	Name	Description
0	Res	Reserved for future use
1-2	Ai/Ao	Isolated A in (bit 1) / out (bit 2)
3-4	Bi/Bo	Isolated B in (bit 3) / out (bit 4)
5	SBB	SyncBox button
6	SBEi	SyncBox external trigger in
7	Res2	Reserved for SyncBox external trigger out
8-15	Par8	8-bit trigger code
16-23	Res3	Reserved for future use

Trigger Packets

This packet contains information about one or more trigger events registered by the measurement hardware. The emission of this packet is optional and may be enabled / disabled. By default this packet isn't emitted.

The Trigger packet structure is shown below:

Type	Name	Description
uint8	FrameType	3 = Triggers
uint8	MainUnitNum	Indicates which main unit the triggers packet is coming from (see MainUnitNum field in Samples packet).
uint16	NumTriggers	Number of triggers in this packet
uint32	Reserved	Reserved for future use
Trigger[N]	Triggers	Array of structures, each describing a single trigger event: (N = NumTriggers)

Type	Name	Description
uint64	MicroTime	Trigger position relative to start time of measurement (microseconds)
uint64	SampleIndex	Index of sample tied to trigger
uint8	Type	4/4 bits; trigger source channel id and mode.
uint8	Code	Parallel trigger code.
uint8[2]	Reserved	Reserved for future use

The value of trigger type field is divided into upper and lower 4 bits; the upper 4 bits denote the physical channel the trigger came from. The lower 4 bits indicate the mode the trigger hardware was configured for:

Source id (upper 4)	Description
1	Isolated port A
2	Isolated port B
3	Parallel trigger
4	SyncBox button
5	SyncBox external trigger

Mode (lower 4)	Description
1	Stimulation trigger
2	Video trigger
3	Mute trigger
4	Parallel trigger
5	Trigger output

MeasurementEnd Packet

This packet is sent by NeurOne to indicate the end of the measurement. It's the last packet emitted when measurement hardware is stopped. The emission of this packet is optional and may be enabled / disabled per-measurement. By default this packet isn't emitted.

The MeasurementEnd packet structure is shown below:

Type	Name	Description
uint8	FrameType	4 = MeasurementEnd
uint8	MainUnitNum	Indicates which main unit the packet is coming from (see MainUnitNum field in Samples packet).
uint8[2]	Reserved	Reserved for future use
uint64	FinalSampleCount	Total number of sample bundles sent during this measurement

HardwareState Packet

This packet relays information about measurement hardware state, which can be considered metadata in regards to the measurement. It's sent by applicable NeurOne main units at the start of the measurement after MeasurementStart packet before any measurement data (samples / triggers) is emitted and also upon receival of a Join packet. Note that this packet can contain different payloads which depend on the situation.

The emission of this packet is optional and may be enabled / disabled per-measurement. By default this packet isn't emitted.

The HardwareState packet structure is shown below:

Type	Name	Description
`uint8`	`FrameType`	5 = HardwareState
`uint8`	`MainUnitNum`	Indicates which main unit the packet is coming from (see MainUnitNum field in Samples packet).
`uint8`	`StateType`	Type of the payload: 1 = ClockSourceState
`uint8`	`Reserved`	Reserved for future use
`uint8[]`	`StatePayload`	Payload depending on StateType

ClockSourceState Packet

This state payload contains information about the current clock source of SyncBox system. It's not emitted on standalone systems. Every main unit sends this payload at the start of the measurement after MeasurementStart packet before any measurement data (samples / triggers) is emitted and also upon receival of a Join packet.

The payload structure is as follows:

Type	Name	Description
`uint64`	`MicroTime`	Clock source change time relative to start time of measurement (microseconds), according to sampling clock.
`uint32`	`ClockFreq`	Actual input clock frequency in Hz.
`uint32`	`TargetClockFreq`	Target (ideal) clock frequency in Hz.
`uint16`	`ClockSrc`	The clock source used by SyncBox: 1 = SyncBox internal 2 = BNC clock port of SyncBox 3= Fiber clock port of SyncBox

Join Packet

The Join packet is sent by the system receiving NeurOne digital out data. Main purpose of this packet is to enable MeasurementStart packet re-emission by NeurOne in cases where the receiver system becomes online after NeurOne measurement has already been started.

The Join packet structure is shown below:

Type	Name	Description
`uint8`	`FrameType`	128 = Join
`uint8[3]`	`Reserved`	Must be filled with zeros, contents are discarded.

Upon receival of Join packet NeurOne sends a MeasurementStart in the following circumstances:

a) if digital out is sending to a unicast address: NeurOne responds to Join only if it originates from that same address.

b) If digital out is sending to a broadcast address: NeurOne responds to Join only if it originates from the same subnet (255.255.255.0). The MeasurementStart packet is sent as unicast to the IP address the Join was received from. This is for situations where a single client needs to acquire MeasurementStart.

It should be noted that the action of submitting a Join packet to NeurOne and NeurOne responding with a MeasurementStart packet may cause momentary jitter to digital out frame delivery timing.

Note: the receiving system does not need to send this packet unless it explicitly wants a new MeasurementStart packet.

Note: NeurOne will respond to Join packet only if emission of MeasurementStart packet is enabled in NeurOne configuration.

Note: The Join packet must be sent to UDP port 5050.

Examples

This chapter contains a number of packet flow examples for different digital out configurations. The time column indicates wall-clock time elapsed from the start of the measurement.

Case 1: MeasurementStart / MeasurementEnd / Trigger packet emission isn't enabled. Sampling rate 5 kHz, delivery rate 1 kHz.

Time (ms)	Packet	Description
Measurement is started		

1.0	`Samples`	Samples 0..4 for each output channel
		`PacketSeqNo = 0`
		`NumSampleBundles = 5`
		`FirstSampleIndex = 0`
		`MicroTime = 0`

2.0	`Samples`	Samples 5..9 for each output channel
		`PacketSeqNo = 1`
		`NumSampleBundles = 5`
		`FirstSampleIndex = 5`
		`MicroTime = 1 000`

3.0	`Samples`	Samples 10..14 for each output channel
		`PacketSeqNo = 2`
		`NumSampleBundles = 5`
		`FirstSampleIndex = 10`
		`MicroTime = 2 000`

…		
Measurement is stopped		

Case 2: MeasurementStart / MeasurementEnd packet emission isn't enabled. Sampling rate 5 kHz, delivery rate 500 Hz.

Time (ms)	Packet	Description
Measurement is started		

2.0	`Samples`	Samples 0..9 for each output channel
		`PacketSeqNo = 0`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 0`
		`MicroTime = 0`

4.0	`Samples`	Samples 10..19 for each output channel
		`PacketSeqNo = 1`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 10`
		`MicroTime = 2 000`

4.2	`Triggers`	One or more triggers relating to samples sent in previous Samples packet.

6.0	`Samples`	Samples 20..29 for each output channel
		`PacketSeqNo = 2`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 20`
		`MicroTime = 4 000`

…
Measurement is stopped

Case 3: Emission of all packet types is enabled. Sampling rate 10 kHz, delivery rate 1 kHz.

Time (ms)	Packet	Description
Measurement is started		
0.0	`MeasurementStart`	Indicates start of measurement
0.0	`HardwareState (ClockSourceState)`	Relays hardware state metadata

1.0	`Samples`	Samples 0..9 for each output channel
		`PacketSeqNo = 0`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 0`
		`MicroTime = 0`

2.0	`Samples`	Samples 10..19 for each output channel
		`PacketSeqNo = 1`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 10`
		`MicroTime = 1 000`

2.2	`Triggers`	One or more triggers relating to samples sent in previous Samples packet

3.0	`Samples`	Samples 20..29 for each output channel
		`PacketSeqNo = 2`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 20`
		`MicroTime = 2 000`

…		

1501.0	`Samples`	Samples 15000..15009 for each output channel
		`PacketSeqNo = 1500`
		`NumSampleBundles = 10`
		`FirstSampleIndex = 15000`
		`MicroTime = 150 000`

Measurement is stopped		
1501.7	`MeasurementEnd`	Indicates end of measurement

Next part contains examples of how to parse sample packets. Examples can be repeated using the GNU Octave codes. GNU Octave is freely distributed software that is intended for numerical computations. The GNU Octave scripts need the GNU Octave sockets package. By default, the received values in the sample packets are in 8-bit unsigned format.

Example 1. Only one channel is directed to the digital out and both the sampling rate and packet delivery frequency are set to be 500 Hz.

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value uint8	2	0	0	0	0	0	0	24	0	1	0	1	0	0	0	0
Byte	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Value uint8	0	0	0	24	0	0	0	0	0	0	187	128	255	114	58	

Type	Description	Value uint8	Value
uint8	Sample packet identifier	2	2
uint8	Main unit indication	0	0
uint8	Not in use	0	0
uint8	Not in use	0	0
uint32	Packet sequency number	0 0 0 24	24
uint16	Number of channels	0 1	1
uint16	Number of sample bundles	0 1	1
uint64	Sample index of the first bundle	0 0 0 0 0 0 0 24	24
uint64	Time stamp of the first bundle	0 0 0 0 0 0 187 128	48000
int24	Data samples	255 114 58	-36294

Example 2. Two channels are directed to the digital out and both the sampling rate and packet delivery frequency are set to be 500 Hz.

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Value uint8	2	0	0	0	0	0	0	30	0	2	0	1	0	0	0	0	0
Byte	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Value uint8	0	0	0	30	0	0	0	0	0	234	96	248	231	55	248	232	51

Type	Description	Value uint8	Value
uint8	Sample packet identifier	2	2
uint8	Main unit indication	0	0
uint8	Not in use	0	0
uint8	Not in use	0	0
uint32	Packet sequency number	0 0 0 30	30
uint16	Number of channels	0 2	2
uint16	Number of sample bundles	0 1	1
uint64	Sample index of the first bundle	0 0 0 0 0 0 0 30	24
uint64	Time stamp of the first bundle	0 0 0 0 0 0 234 96	60000
int24	Data sample channel1	248 231 55	-465097
int24	Data sample channel2	248 232 51	-464845

Example 3. Only one channel is directed to the digital out and the sampling rate and packet delivery frequency are set to be 500 Hz and 100 Hz, respectively.

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value uint8	2	0	0	0	0	0	0	51	0	1	0	5	0	0	0
Byte	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Value uint8	0	0	0	0	255	0	0	0	0	0	7	200	48	249	247
Byte	31	32	33	34	35	36	37	38	39	40	41	42	43		
Value uint8	34	249	233	27	249	218	135	249	210	6	249	205	203		

Type	Description	Value uint8	Value
uint8	Sample packet identifier	2	2
uint8	Main unit indication	0	0
uint8	Not in use	0	0
uint8	Not in use	0	0
uint32	Packet sequency number	0 0 0 51	51
uint16	Number of channels	0 1	1
uint16	Number of sample bundles	0 5	5
uint64	Sample index of the first bundle	0 0 0 0 0 0 0 0 255	255
uint64	Time stamp of the first bundle	0 0 0 0 0 7 200 48	60000
int24	Data sample sample index255	249 247 34	-395486
int24	Data sample sample index256	249 233 27	-399077
int24	Data sample sample index257	249 218 135	-402809
int24	Data sample sample index258	249 210 6	-404986
int24	Data sample sample index259	249 205 203	-406069

GNU Octave script for receiving UDP sample packets:

```
pkg load sockets %loads the needed sockets packet
buff_size=50; %buffer size must be greater than the packet size
rcv_port=50000; %target port number
rcv_sck=socket(AF_INET, SOCK_DGRAM, 0);
bind(rcv_sck,rcv_port);
[str,len_s]=recv(rcv_sck,buff_size); %the first packet is empty
packet

for n=1:10
  [pckt(n,:),len_s]=recv(rcv_sck,60); % buffer must be greater that
the packet size
end
```

Note: This script is intended only for test purposes. It is not suitable for real-time measurements.