imec
Kapeldreef 75
B-3001 Leuven

Neuropixels System User API

Author(s):	Jan Putzeys		
Date:	17-05-2019	*Doc. No:*	Neuropixels_System_User_API
Reviewed by:			
Keywords:	…		
Distribution list:	imec: Barun Dutta, Carolina Mora Lopez External: Tim Harris, Bill Karsh (HHMI), Josh Siegle (Allen Institute)		

Revision history

Version	Date	Description	Responsible
V1.0	July 05, 2018	Initial release	Jan Putzeys
V1.1	August 30, 2018	• BSC boot code updated to 140, BS boot code updated to 99, API Version updated to 0.130 • Chapter 3.5.4: updated module functionality • Chapter 4.2: updated status register • Chapter 5.2: updated information on trigger matrix and SYNC clock • Chapter 6.3: updated information on trigger and SYNC configuration • Chapter 6.4: updated information on data recording • Chapter 7: valid port numbers updated to 1 to 4 • Chapter 7.5: updated trigger configuration functions • Chapter 7.6.2: updated RAM FIFO functions • Chapter 7.8: updated NP_ErrorCode enum • Chapter 8: updated buglist	Jan Putzeys
V1.2	October 16, 2018	• API Version updated to 0.140 • Chapter 3.5.4: updated trigger/SYNC description • Chapter 5.2: updated trigger/SYNC description • Chapter 6.1: added PCIe buffer configuration • Chapter 6.3/6.4: updated trigger/SYNC configuration • Added chapter 7.1: global configuration functions • Chapter 7.6.2: modified enums Trigger matrix • Chapter 7.6: remove function setSyncClockFrequency • Chapter 8: updated bug list	Jan Putzeys
V1.3	December 3rd, 2018	• BS, BSC and API Version updated • Chapter 4.2 updated • Chapter 6.4: added example to SYNC configuration • Chapter 6.5: updated STATUS byte, updated ElectrodePacket • Chapter 7.1 updated setParameter min/max, updated NP_PARAM_SYNCSOURCE • Chapter 7.6.2 updated • Chapter 7.9 updated • Chapter 8 updated	Jan Putzeys
V1.4	December 18th, 2018	• Chapter 1: API Version updated • Chapter 2: delivered fileset updated • Chapter 5.2: updated • Chapter 6.3: added function setTriggerBinding	Jan Putzeys

		• Chapter 6.4 updated • Chapter 7.1: updated for NP_PARAM_BUFFERSIZE and NP_PARAM_SYNCMASTER • Chapter 7.6.2: added function setTriggerBinding • Chapter 7.8.2: modification to bistHB • Chapter 7.8.8: modification to bistSignal • Chapter 7.8.9: modification to bistNoise • Chapter 7.9: updated enum NP_ErrorCode • Chapter 8: updated bugs and issues	
V1.5	March 07, 2019	• Chapter 1: FPGA and API versions updated. • Inserted new chapter; (6.7): using the HST module functions • Inserted new chapter: (7.9): HST module functions • Chapter 8: updated bugs and issues	
V1.6	April 01, 2019	• Chapter 1: API version updated. • Chapter 7.5.3: enable connection of multiple electrodes to channels	
V1.7	May 17, 2019	• Chapter 1: API version updated • Chapter 7.2.9: Added function to set debug messages. • Chapter 7.2.10: Added function to read the last error message • Chapter 7.5.3: typo removed	

Related documents:

- [probe]**:** Neuropix Phase III: Final High-Density Neural probe, Specifications Report
- [Shank configuration]: Neuropix_PhaseIII_V2_Shank_Configuration.xlsx
- [Base configuration]: Neuropix_PhaseIII_V2_Base_Configuration.xlsx
- [Electrode to channel mapping]**:** Neuropix_PhaseIII_V2_Electrode_Mapping.xlsx
- [User manual]: 2019.03.12_Neuropixels_System_and_Probe_User_Manual_V1_6.pdf

Contact address data: Kapeldreef 75, 3001 Heverlee

Contents

List of abbreviations

ADC	Analog-to-Digital Converter
AP	Action potential
ASIC	Application Specific Integrated circuit
BIST	Built-In Self-Test
BRAM	Block RAM
BS	Basestation
BSC	Basestation Connect Board
CR	Carrier
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First-in, first-out
FMC	FPGA Mezzanine Card
FPC	Flexible Printed Circuit
FPGA	Field Programmable Gate Array
GPI	General Purpose Input
GPO	General Purpose Output
HB	Heart Beat
HS	Headstage
HST	Headstage Test
I2C	Inter-IC-bus
IC	Integrated Circuit
ID	Identification
I/O	Input/Output
LDO	Low-DropOut
LED	Light-emitting Diode
LFP	Local Field Potential
LSB	Least Significant Bit
MSB	Most Significant Bit
MUX	Multiplexer
MXI	Multisystem eXtension Interface
NDB	Neural Data Bus
PCB	Printed Circuit Board
PCB	Printed Circuit Board
PC	Personal Computer
PCIe	Peripheral Component Interconnect Express
POR	Power-On Reset
PSB	Parallel Serial Bus
PXI	PCI eXtensions for Instrumentation
P/N	Part Number

RAM	Random Access Memory
R/W	Read/Write
SMA	SubMiniature version A
SPI	Serial Peripheral Interface
S/N	Serial Number
TP	Twisted pair
UART	Universal Asynchronous Receiver-Transmitter
USB-C	Universal Serial Bus Type-C
ZIF	Zero Insertion Force

Definition of Technical Terms

- **ASIC**: integrated circuit which contains the recording electrodes, amplifiers and digitizers.
- **Probe Shank:** Implanted part of the probe ASIC.
- **Probe Base:** Non-implanted part of the probe ASIC.
- **Flex:** Flexible PCB on which the ASIC plus some additional components are mounted.
- **Headstage:** Miniature board located close to the ASIC, which configures and calibrates the ASIC and serializes ASIC data.
- **Headstage Test:** small test module which can be plugged in the ZIF connector of the headstage, which allows to test the HS functionality.
- **Basestation:** The PXIe plug-in module, the combination of Carrier, BSC, Enclustra FPGA module and front panel.
- **Carrier:** An interconnect/support board between the BSC and Enclustra FPGA module.
- **Basestation connect board:** Small board with deserializer chips, which plugs in the CR board, and which acquires and processes data transmitted by the probe.
- **Enclustra FPGA module:** Plug-in module which bridges between the BSC and the PCIe interface.
- **Slot:** a slot in the PXI chassis, in which a Basestation module is plugged in
- **Port:** The USB-C connector in which the cable to the HS is plugged in. Each BSC contains 4 ports.
- **Driver**: The software driver API, required by the probe end users to communicate to the system and to develop custom application software.
- **SerDes**: combination of serializer and deserializer chip.

1 Scope

This document specifies the Basestation functionality and API function descriptions. It serves as a reference for the user application developer.

The document is valid for the following VHDL code version and API version:

- BSC FPGA Boot code: Version 1, Revision 0, Build 151
- BS FPGA Boot code: Version 1, Revision 1, Build 123
- API: Version 1, Revision 20

2 General requirements

Each delivered VHDL code and API is version controlled. The version is split in two parts: version.revision (xx.yy). The major number (version) is incremented if the update is no longer compatible with previous versions, i.e. VHDL code with the same major number (version) are compatible with each other. The API checks compatibility with the hardware.

The delivered code is uploaded on a svn server. Bugs, issues and feature requests are tracked on a trac server, both set up and maintained by imec.

Delivered fileset:

- Enclustra FPGA
    - binary file for update via JTAG
    - binary file for remote update via API
- BSC FPGA
    - binary file for update via JTAG
    - binary file for remote update via API
- API
    - Dll for windows Visual Studio and Mingw
    - Lib file windows Visual Studio and Mingw
    - Drivers for Enclustra module
    - System requirements for user PC

The MCS, DLL and LIB files contain the version number in the file name.

3 System Architecture

3.1 Probe

This chapter lists the relevant probe specifications for the development of BS VHDL code and API development and usage of the API functions. For detailed information on the probe, refer to the probe specification document.

3.1.1 Probe architecture

The probe is a high-density neural probe which consists of two parts: shank and base. The shank area is a thin needle on which electrodes are located and which is inserted in the brain. The base contains most of the active signal circuitry. There is only one probe type, containing 960 recording electrodes, of which 384 can be recorded simultaneously.

The probe shank consists of:

- An array of 960 passive electrodes.
- A distributed switch matrix to select groups of 384 electrodes. Each signal line in the shank is connected to 2 or 3 different electrodes. Two additional switches will enable: applying a calibration signal at the input of the pixel and grounding the input of the pixel for noise measurements.
- A distributed shift register (SR) to program the electrode connectivity. This programming is not dynamic, i.e. it is not meant to scan the array at the sampling speed.
- An array of 4 internal reference electrodes distributed in the array: a large tip electrode and 3 of the recording electrodes (192, 576, 960). An internal reference electrode is available every 3.8 mm, and only one of them can be selected as common reference for each channel. When the internal electrode is not selected as reference, it can work normally as recording electrode (except for the tip electrode). An external reference electrode, connected through an input pin, can also be selected as common reference.

A complete readout architecture is implemented at the probe base. Figure 1 shows the high-level block diagram of the full probe. The full architecture consists of:

- Each of the 384 dual-band recording channels consists of:
    - An instrumentation amplifier with a fixed gain of 50.
    - Two analog filters to separate the AP signals (300 Hz high-pass and 10 kHz low pass) from LFP signals (1 kHz low pass).
    - Two programmable-gain amplifiers (PGAs) providing gains in the range between 1 and 60.
    - A buffer to drive the subsequent analog multiplexer and ADC.
- An analog multiplexer to combine the two outputs (AP and LFP channels) of 12 channels into a single line. Each MUX samples the AP and LFP output of 12 channels. The AP outputs are sampled at 12x the sampling frequency of the LFP outputs: for every 12 AP output samples, 1 LFP output is sampled.
- The sampling speed of the ADC is 390 kHz. Each AP output is sampled at 30 kHz, while each LFP output is sampled at 2.5 kHz.
- An array of 32 10-bit ADCs. Each ADC is connected to one MUX and, therefore, is digitizing the outputs of 12 channels. The sampling frequency is 390 kHz and

synchronized with the MUX clock. The ADC operates at the same clock frequency as the master clock (i.e. 93.6 MHz).

- A digital data interface to combine and send the outputs of the ADCs to a limited number of IOs, following a serial communication protocol (i.e. a parallel synchronous bus or PSB). Seven PSB data lines will be implemented to handle the data rate of the 32 ADCs.
- A digital control interface to configure the different programmable parameters and the electrode selectivity through an I2C interface.
- A general reset pin (NRST) which places the entire probe in reset mode, and a MUX reset pin (REC_NRESET) which resets the MUXes and ADCs.

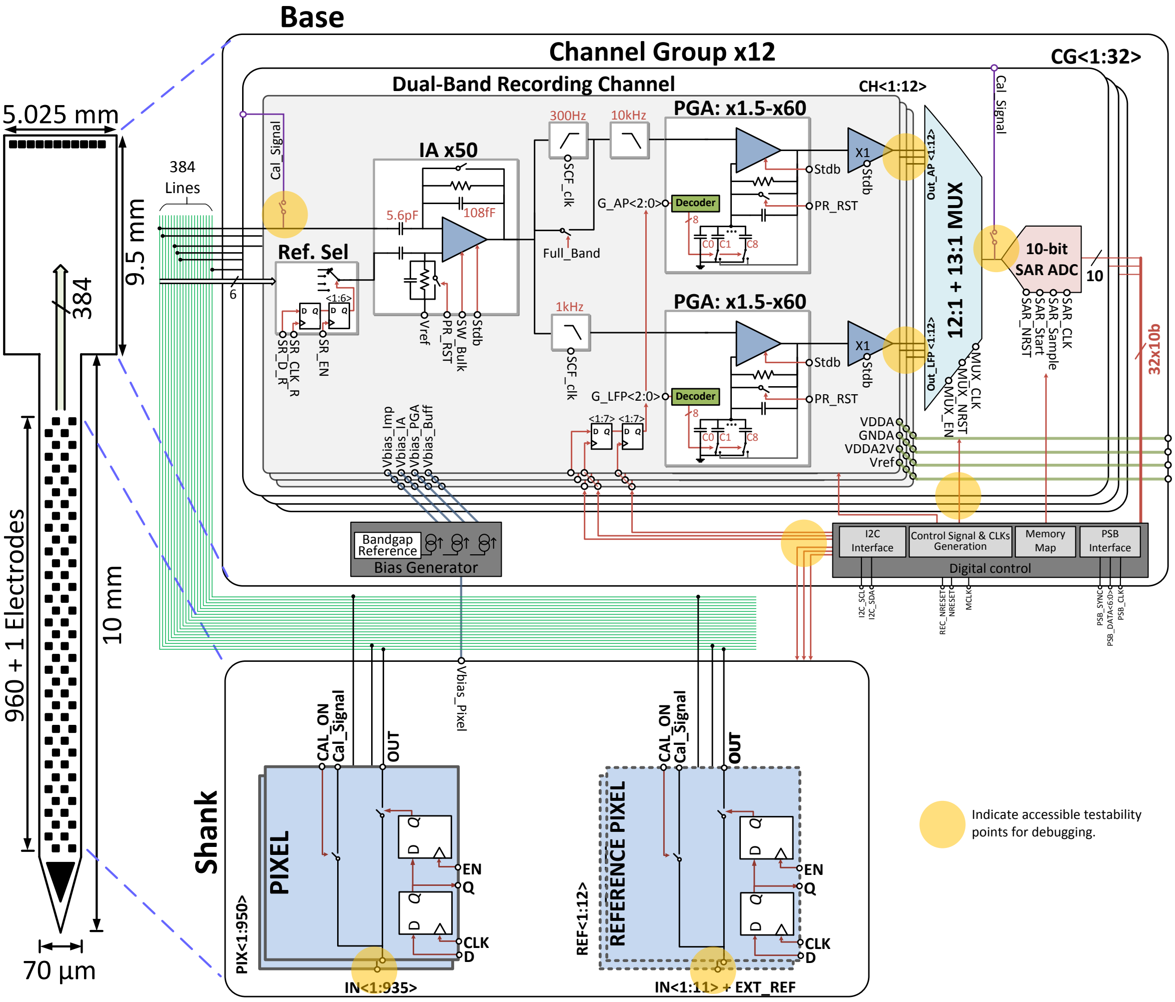

Figure 1: High level block diagram of the probe.

3.1.2 PSB bus

All the data coming from the 32 ADCs is multiplexed on a single PSB interface with 1 clock line, 1 sync line and 7 data lines.

The PSB_CLK signal is active as soon as the probe is switched on, NRST set high, and MCLK active. The other PSB signals are active when the REC bit in the OP_MODE register in the memory map (ref. chapter 3.1.3) is set to 1, and REC_NRESET is set to 1.

3.1.2.1 Clock signal

The **PSB_CLK** signal is internally generated on the probe, by dividing the MCLK clock. The default frequency of the PSB_CLK signal after power-up or after a general reset (*NRST* external signal) equals 23.4 MHz. Furthermore, the PSB_CLK frequency can be configured to be 11.7 MHz, while the clock frequency at which the PSB_DATA lines toggle remains 23.4 MHz. In this mode, the PSB bus can be demultiplexed on the HS from a 7-bit wide bus to a 14-bit wide bus, which makes a more efficient use of the serializers' parallel input bus (ref. chapter 3.3.2). The PSB_CLK frequency mode is set by the *PSB_F* data field of the *REC_MOD* register. The PSB_CLK runs constantly once the MCLK is provided and stops only during a general reset (*NRST* external signal).

The probe transmits a 11.7 MHz clock over the SerDes interface to the BSC FPGA. The 11.7 MHz clock originates from a 93.6 MHz oscillator on the HS. The oscillator has a specified accuracy of +-20 ppm.

3.1.2.2 Data

The data from 32 ADCs is output on the **PSB_DATA<6:0>** lines. PSB_DATA lines 0 to 5 contain the data of 5 ADCs each. PSB_DATA line 6 contains the data of 2 ADCs.

- PSB_DATA<0>: SAR_Data(1,2,3,4,5)
- PSB_DATA<1>: SAR_Data(6,7,8,9,10)
- PSB_DATA<2>: SAR_Data(11,12,13,14,15)
- PSB_DATA<3>: SAR_Data(16,17,18,19,20)
- PSB_DATA<4>: SAR_Data(21,22,23,24,25)
- PSB_DATA<5>: SAR_Data(26,27,28,29,30)
- PSB_DATA<6>: SAR_Data(31,32)

The 10-bit data from the ADCs is sequentially output on the PSB_DATA lines. The output data is divided into 10-bit-word frames, with one frame consisting of 6 words (5 ADC words + 1 fill word = "0000000000"). One frame contains 1 sample of each ADC. The fill word is required to match the frame rate to the ADC sampling frequency (390 kHz). 13 frames are required to cover the data from all the 384 AP channels plus the data of 32 LFP channels (one of the 13 frames includes the LFP data). A group of 13 frames is called a superframe. Therefore, 12 superframes are required to cover the data from all the 384 LFP channels. A group of 12 superframes is called an ultraframe.

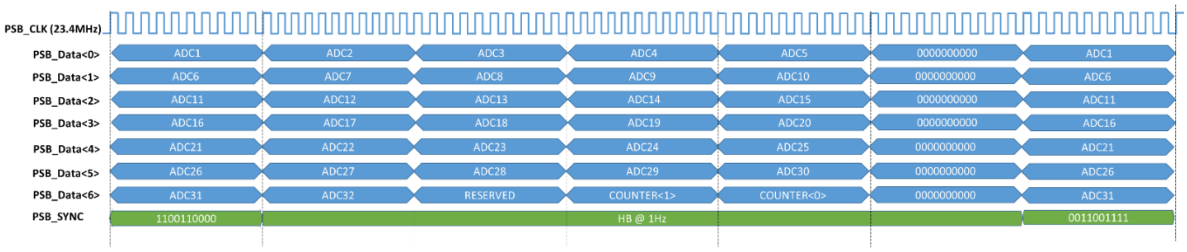

Figure 2: PSB timing diagram.

3.1.2.3 Counter

PSB_DATA line 6 also contains a 20-bit '**Counter**' value (2x 10 bit: COUNTER0 and COUNTER1). The counter clock is the same as the ADC sample clock (390 kHz) and the multiplexer clock which mux the channel outputs to the ADCs inputs. The multiplexers and the counter are both reset to zero by a general reset (*NRST* external signal or *SOFT_RESET* command), a digital reset (*DIG_NRESET* data field of the *REC_MOD* register) or a recording reset (*REC_NRESET* external signal).

The counter value can theoretically be used to check which channels are currently muxed to the ADCs, but this is not an efficient implementation. Instead, the start of recording must be triggered by the recording reset (*REC_NRESET* external signal), which will reset the multiplexers. The counter can be used to check whether frames are lost (the counter value increments with 1 for every frame).

3.1.2.4 FIXED

Furthermore, PSB_DATA line 6 also contains a 10-bit **'FIXED'** word ('Reserved' in the PSB diagram). The FIXED word contains the value of the data field *SYNC_DATA<7:0>* of the SYNC register. Since this register is only 1 byte long, bits <9:8> of the FIXED word are 00. This register is checked and updated in every frame. This serves as a synchronization method with an external device or to measure the delay of the I2C communication with the SerDes.

3.1.2.5 PSB_SYNC

The PSB_SYNC line contains a fixed 10-bit synchronization word, which differs between the start of a frame and the start of a superframe. The PSB_SYNC line is required for the system to detect when the first frame is presented on the PSB_DATA bus after a recording reset (*REC_NRESET* external signal).

The first synchronization word transmitted after a recording reset and at the start of each subsequent superframe = "1100110000".

The synchronization word transmitted at the beginning of a frame = "0011001111".

The remaining 50 bits on the PSB_SYNC line within the frame are filled with a HB signal: toggling between 0 and 1 at a frequency of 1 Hz.

3.1.2.6 Timing diagrams

When configured at 23.4 MHz, the data is clocked out MSB first and changing on the **rising** edge of *PSB_CLK* (the data is valid at the receiver on the **falling** edge of *PSB_CLK*). The PSB output lines are not 'highZ' when there is no activity. They simply remain stable.

When configured at 11.7 MHz, there is a 90 degrees phase shift between the data change and clock edge.

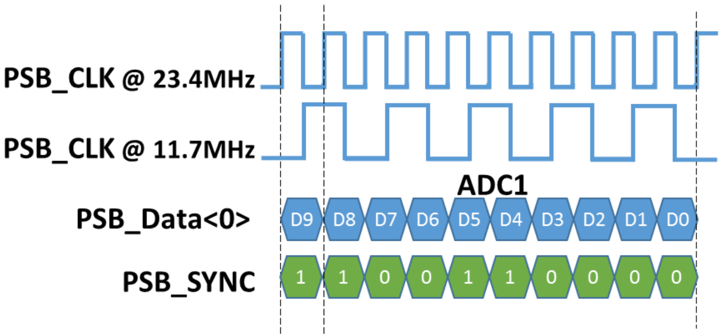

Figure 3: PSB_CLK and PSB_SYNC timing diagram.

3.1.3 Memory map

All probe configurations are set via memory map registers. The complete memory map register is shown in the table below.

Reg.	Reg.	Data index							
address	name	D7	D6	D5	D4	D3	D2	D1	D0
0x00	OP_MODE	P_DOWN	REC	CAL	DIG_TEST	TEST	X	X	X
0x01	REC_MOD	CH_NRESET	DIG_NRESET	SR_Default	PSB_F	X	X	X	X
0x02	CAL_MOD	PIX_CAL	CH_CAL	ADC_CAL	OSC_STDB	X	X	X	X
0x03	TEST_CONFIG1	TEST_CONTROL<39:32>							
0x04	TEST_CONFIG2	TEST_CONTROL<31:24>							
0x05	TEST_CONFIG3	TEST_CONTROL<23:16>							
0x06	TEST_CONFIG4	TEST_CONTROL<15:8>							
0x07	TEST_CONFIG5	TEST_CONTROL<7:0>							
0x08	STATUS	SR_OUT_OK	X	X	X	X	X	X	X
0x09	SYNC	SYNC_DATA<7:0>							
0x0a	SR_CHAIN5	Address reserved for the analog shift-register chain <4>: TA2							
0x0b	SR_CHAIN4	Address reserved for the analog shift-register chain <3>: TA1							
0x0c	SR_CHAIN3	Address reserved for the analog shift-register chain <2>: CA2							
0x0d	SR_CHAIN2	Address reserved for the analog shift-register chain <1>: CA1							
0x0e	SR_CHAIN1	Address reserved for the analog shift-register chain <0>: PA							
0x0f	SR_LENGTH2	SR_N_BYTES<15:8>							
0x10	SR_LENGTH1	SR_N_BYTES<7:0>							
0x11	SOFT_RESET	Address reserved for synchronous reset							

The memory map with 8-bit address registers is not large enough to contain all configuration settings of the probe. For this reason, base, shank and special test configurations are implemented as a shift register. The bits of the shift registers are shifted in by sequential writes to a dedicated memory map address.

The details of the memory map registers are given below:

OPMODE Register

Reg. address	Reg name	Data index							
		D7	D6	D5	D4	D3	D2	D1	D0
0x0000	OP_MODE	P_DOWN	REC	CAL	DIG_TEST	TEST	X	X	X

Index	NAME	Meaning
7	P_DOWN	0: Disable power down mode (default) 1: Enable power down mode
6	REC	0: Disable recording mode (default) 1: Enable recording mode
5	CAL	0: Disable calibration mode (default) 1: Enable calibration mode
4	DIG_TEST	0: Disable digital test mode (default) 1: Enable digital test mode
3	TEST	0: Disable test mode (default) 1: Enable test mode

REC_MOD Register

Reg. address	Reg name	Data index							
		D7	D6	D5	D4	D3	D2	D1	D0
0x0001	REC_MOD	CH_NRESET	DIG_NRESET	SR_Default	PSB_F	X	X	X	X

Index	NAME	Meaning
7	CH_NRESET	0: Enable reset of the channel pseudo-resistors 1: Disable reset of the channel pseudo-resistors (default)
6	DIG_NRESET	0: Enable reset of the MUX, ADC, and PSB counter 1: Disable reset of the MUX, ADC, and PSB counter (default)
5	SR_Default	0: Disable default values of analog SR chains (default) 1: Set analog SR chains to default values
4	PSB_F	0: Sets the PSB_CLK frequency to 23.4 MHz (default) 1: Sets the PSB_CLK frequency to 11.7 MHz

CAL_MOD Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x0002	CAL_MOD	PIX_CAL	CH_CAL	ADC_CAL	OSC_STDB	X	X	X	X

Index	NAME	Meaning
7	PIX_CAL	0: Disable pixel + channel gain calibration (default) 1: Enable pixel + channel gain calibration
6	CH_CAL	0: Disable channel gain calibration (default) 1: Enable channel gain calibration
5	ADC_CAL	0: Disable ADC calibration (default) 1: Enable ADC calibration
4	OSC_STDB	0: Enables the standby mode of the external oscillator (default) 1: Disables the standby mode of the external oscillator

TEST_CONFIG1-5 Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x0003	TEST_CONFIG1	TEST_CONTROL<39:32>							
0x0004	TEST_CONFIG2	TEST_CONTROL<31:24>							
0x0005	TEST_CONFIG3	TEST_CONTROL<23:16>							
0x0006	TEST_CONFIG4	TEST_CONTROL<15:8>							
0x0007	TEST_CONFIG5	TEST_CONTROL<7:0>							

Index	NAME	Meaning
[7:0] + [7:0] + [7:0] + [7:0] + [7:0]	TEST_CONTROL<39:0>	Reserved for control of the analog test functionality of the chip. It will control probe switches in the analog domain. Default: 0x00000.

STATUS Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x0008	STATUS	SR_OUT_OK	X	X	X	X	X	X	X

Index	NAME	Meaning
7	SR_OUT_OK	0: Indicates SR chain comparison is not OK (default) 1: Indicates SR chain comparison is OK

SYNC Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x0009	SYNC	SYNC_DATA<7:0>							

Index	NAME	Meaning
[7:0]	SYNC_DATA<7:0>	Reserved for a sync code that will be included in the PSB data frame

SR_CHAIN5-1 Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x000a	SR_CHAIN5	Address reserved for the analog shift-register chain <4>: TA2							
0x000b	SR_CHAIN4	Address reserved for the analog shift-register chain <3>: TA1							
0x000c	SR_CHAIN3	Address reserved for the analog shift-register chain <2>: CA 2							
0x000d	SR_CHAIN2	Address reserved for the analog shift-register chain <1>: CA 1							
0x000e	SR_CHAIN1	Address reserved for the analog shift-register chain <0>: PA							

Index	NAME	Meaning
--	Addresses reserved for the analog shift-register chains <4:0>	

SR_LENGTH2-1 Register

Reg. address	Reg name	Data index							
		D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**

0x0f	SR_LENGTH2	SR_N_BYTES<15:8>
0x10	SR_LENGTH1	SR_N_BYTES<7:0>

Index	NAME	Meaning
[7:0] + [7:0]	SR_N_BYTES <15:0>	Indicates the number of bytes to be read from the SR chain. Default: 0x0000.

SOFT_RESET Register

Reg. address	Reg name	Data index							
		D7	D6	D5	D4	D3	D2	D1	D0
0x0011	SOFT_RESET	Address reserved for synchronous reset							

Index	NAME	Meaning
--	Addresses reserved for synchronous reset	

3.1.4 I2C interface

The I2C interface is used to configure the probe, and to read back the configuration and status registers of the probe.

The memory map is accessed via an I2C interface. This interface is implemented to be compatible with the I2C interface of the serializer on the HS.

3.1.5 Shift registers

Apart from the memory map, 3 shift-register (SR) chains, with different number of registers and located in the analog domain, need to be programmed. For this, a I2C to SPI translation is implemented inside the probe. In this way, the data received through the I2C is sent serially to one of the shift-register chains. The chain is selected by a specific command address. The addresses of the analog shift-register chains are indicated in the memory map table (registers *SR_CHAIN5-1*). The number of bytes to be programmed into a SR chain (i.e. the length of the chain) is indicated by the *SR_N_BYTES<15:0>* data field of the *SR_LENGTH2-1* registers. If the number of bytes received in a write cycle exceeds the specified number, the additional bytes is ignored and not sent to the chain.

In order to test whether the SR chains are programmed correctly, the programming data is sent twice. During the second programming phase, the input data is compared with the output data of the SR chain (internal probe process) byte by byte. Once the comparison is finished, a flag will be stored in the data field *SR_OUT_OK* of the *STATUS* register. If the programming is correct, *SR_OUT_OK* will be set to ‘1’, if not it will be set to ‘0’.

The base configuration registers are accessed via the memory map address 0x000c and 0x000d. The shank configuration registers are accessed via the memory map address 0x000e. The shift registers on memory map addresses 0x000a and 0x000b are not used.

3.2 Flex

The probe chip is mounted on a flexible PCB, together with decoupling capacitors, a reference supply circuit and a memory device used for device identification storage.

The memory device is accessed (R/W) via the I2C interface. It’s an EEPROM type CAT24C04C4ATR by ON Semi.

The flex plugs in the HS via a ZIF connector.

3.3 Headstage

This chapter lists the relevant HS specifications.

3.3.1 MCLK signal

An oscillator on the HS generates a 93.6 MHz clock signal which is transmitted to the probe. The probe divides this clock signal and uses it for neural data sampling and data transmission. The probe transmits a 11.7 MHz clock which is connected to the PCLK input of the serializer on the HS. The 93.6 MHz oscillator has a specified accuracy of +-20 ppm.

3.3.2 Parallel data interface

The HS contains a serializer chip which transfers the neural data from the probes’ PSB interface to the BS. In order to make optimal use of the serializer bandwidth, a D-type flipflop is used to demultiplex the PSB_DATA<6:0> bus from 7 bits to 14 bits. In this mode the probes’ PSB bus can be configured to operate at 11.7 MHz.

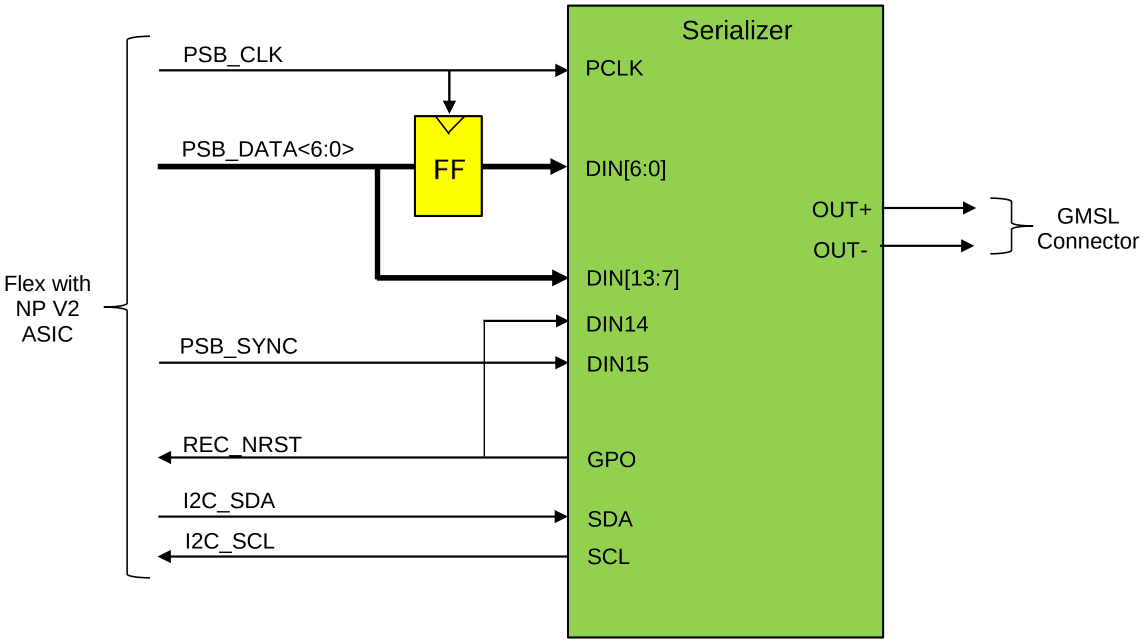

Figure 4: Serializer parallel/serial data lines.

3.3.3 Probe recording reset signal

When the recording reset signal of the probe (REC_NRESET) is low, the multiplexers and ADCs on the probe are reset. The probe starts to transmit data when the REC_NRESET is set high.

The SerDes chipset contains a GPI/GPO channel. The GPO signal located at the serializer side follows the GPI signal located at the deserializer side. This functionality is used to remotely control the REC_NRESET signal to the probe.

This REC_NRESET is transmitted from BS to HS over the GPI/GPO channel of the SerDes link.

3.3.4 Serializer configuration

There are specific serializer registers which require configuration at system start-up. These registers are accessed over the UART via the BS.

3.3.5 Test signal

The HS contains a square wave generator to create a test signal for verification of probe functionality. Amplitude and frequency of the test signal are fixed. The test signal can be placed in standby by a control signal from the probe. This control signal is controlled via the probes' memory map.

3.3.6 EEPROM

A small EEPROM which stores a version number and serial number of the HS is located on the HS. It is accessed via the SerDes's UART/I2C interface. The selected EEPROM is CAT24C04HU4I-GT3 by ON Semi.

3.3.7 LED

The HS contains a LED connected via a gated buffer to the PSB_SYNC line coming from the probe. The buffer is enabled via the GPIO1 signal from the serializer. The system is configured such that the LED is blinking with the probes' PSB_SYNC signal after the 'openProbe' API command. The LED stops blinking after the 'init' API command. The LED can be enabled again with a specific API function.

3.4 Cable

The HS connects to the PXI instrument via a 5 meter twisted pair cable. The cable is made of 1 pair of Cooner wire (CZ1322), transferring power and data over the same wire pair. The HS side of the cable contains a 4-pin Omnetics connector. The PXI side of the cable contains a USB-C connector. The cable can be plugged in either one of the two orientations in the USB-C connectors of the BSC card.

This cable is only to be used with the USB-C connectors on the BSC and should not be plugged in any other USB-C port.

3.5 Basestation

3.5.1 System components

The Basestation is a platform module to interface with Neuropixels probes. The BS is used in a PXIe chassis. It can connect to maximum 4 probes. The module interfaces between the HS and the PCIe interface of the PC. The module consists of 3 separate components:

- The Basestation Connect board: A probe-specific interface board which deserializes, processes and packetizes the neural data from the probe.
- The Enclustra FPGA module: A ZYNQ FPGA module which handles the PCIe communication and routing of trigger signals.
- The Carrier: An interconnect board which supports the BSC and Enclustra module and plugs in the PXIe slot.

The different components are described in more detail in the next chapters.

3.5.2 Basestation Connect board

The Basestation Connect Board is an interface board between the HS + 5 meter cable and the CR board. The BSC has 4 input ports (USB-C connector) to connect to maximum 4 probes. The data from the probe is transmitted to the Enclustra FPGA module.

3.5.2.1 Board schematic

A schematic drawing of the BSC is given in the figure below.

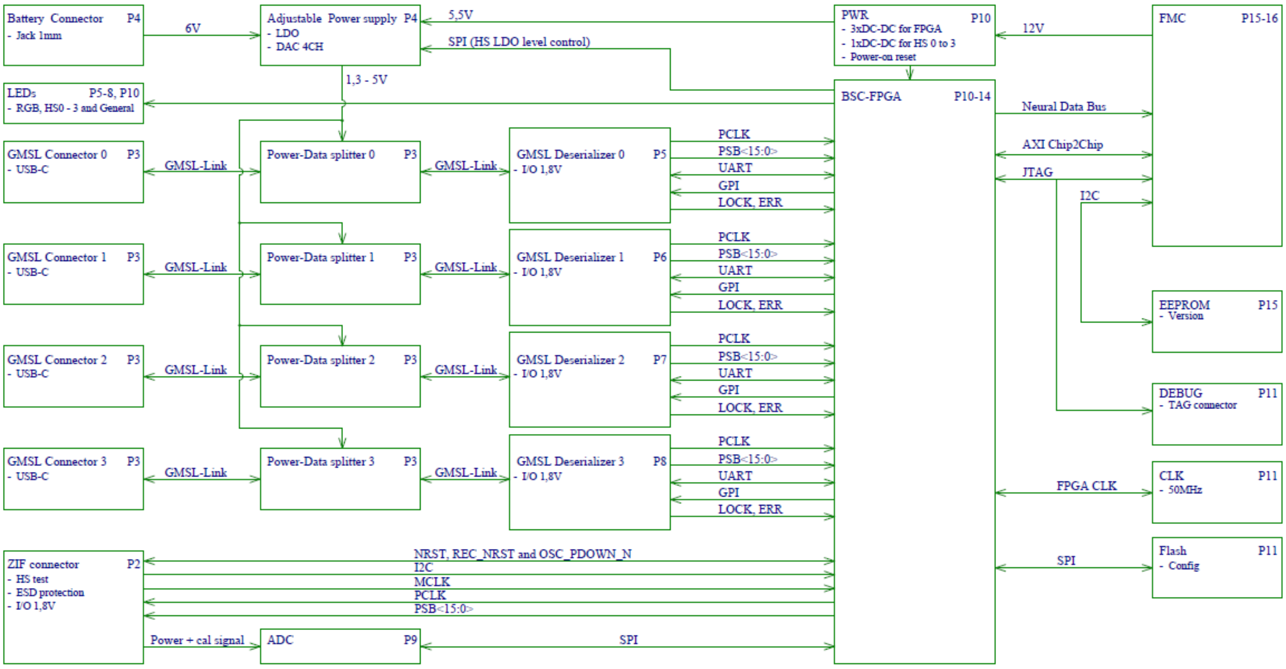

Figure 5: Schematic block diagram of the BSC.

3.5.2.1 USB-C connectors

Four USB-C connectors, 1 for each probe, are located on the front panel. These connectors are to be used with the single twisted pair cable only (chapter 3.4). The cable can be plugged in either one of the two orientations.

3.5.2.2 Deserializer

The BSC contains 4 deserializer chips, 1 for each port. The neural data is presented at the output of the deserializer as it was transmitted to the serializer chip on the HS: 14 parallel data lines clocked at 11.7 MHz, plus PSB_SYNC and REC_NRESET. Demultiplexing into 7 PSB datalines as it is generated by the probe ASIC is done by the BSC FPGA.

The deserializer registers are configured by the BSC FPGA via the UART bus. Configuration of the deserializer is part of the API ‘openProbe’ function.

The deserializer is configured in autoreset mode: a hiccup on the SerDes communication triggers the |ERR and/or LOCK signals. When the cause for the error has been resolved, the error flags are automatically cleared. These error flags are monitored by the BSC and transmitted as part of the data packets to the PC.

3.5.2.3 FPGA

The FPGA on the BSC is a Xilinx Artix-7 FPGA, XC7A50t. The following functionality is implemented on the FPGA:

- Configure probes and access EEPROMs on HS and flex via the SerDes control channel
- Interfacing with 4 deserializer chips
- Demultiplexing probe data into 384 AP and LFP channels
- Signal inversion, 0.6 V offset removal and gain correction
- Synchronization to PXI system clock
- Synchronization with start trigger and SYNC clock
- Packetizing
- Data path error signalling

The FPGA functionality is described in more detail in chapter 4.

3.5.2.4 LEDs

The BSC front panel contains 1 global status LED and 4 port status LEDs, 1 for each port.

The global status LED reflects the status of the BSC system:

- Off: The BSC FPGA is not powered or the boot code is not loaded successfully.
- Red: the BSC FPGA is powered and boot code is loaded, the AXI Chip2Chip interface link is not ready. This indicates an error on the Enclustra FPGA module or if the PXI bus clock is not available.
- Green: the BSC is powered and boot code is loaded, the AXI Chip2Chip interface link is ready, the BSC is armed. When armed, the timestamp is reset and hold at 0 until triggered.
- Blue: the BSC is triggered and is transmitting data to the Enclustra FPGA module.
- Purple: when the BSC is triggered and status led is blue, the red led is used to indicate the state of the neural data ready signal. This indicates of pc back pressure and thus data loss.

The port status LEDs indicate the status of the probe/HS:

- Off: no PSB clock detected. This occurs if the power to the port is not enabled, no probe is plugged in the port or the probe/HS/cable is faulty.

- Red: the red LED is actuated by the REC_NRESET signal to the probe. This status occurs after enabling the power to the ports (‘openBS’ function) and before enabling the probe ‘openProbe’ function). When the red LED is on, green and blue are inhibited to avoid color mixing.

- Green: the probe PSB clock is active and the REC_NRESET signal to the probe is not active. This status occurs after calling the ‘openProbe’ function and is the normal status during operation of the probe.

- Blinking green: the probe PSB clock is active but deviating from 11.7 MHz. This is an error status and could occur if the probe configuration during ‘openProbe’ was not successful.

- Blue: the internal data emulator on the BSC FPGA is active. The probe data is not used.

3.5.2.5 Supply, power/data splitter

The HS and probe are powered via the single twisted pair cable. A filter circuit combines the power and data signal on one twisted pair cable. The supply signal can be generated from the internal PXI chassis supply, or from an external battery supply (chapter 3.5.2.6). The power supply to the 4 ports is enabled or disabled via API functions (‘openBS’, ‘closeBS’), only for the 4 ports simultaneously, not individually.

3.5.2.6 Battery connector

A jack connector (CUI PJ-102AH) can be used optionally to connect an external power supply (4.0 - 5.0 V). The BSC detects if a power supply is applied to the connector and switches the supply of the 4 ports automatically between the external power supply and the internal supply from the PXI chassis.

3.5.2.7 EEPROM

A small EEPROM storing a version number, S/N and P/N of the BSC is located on the BSC. The selected EEPROM is CAT24C04HU4I-GT3 by ON Semi.

3.5.2.8 FMC

The BSC plugs in the HPC FMC connector of the CR board.

3.5.3 Carrier

The Carrier board is the interface board between the BSC, the Enclustra FPGA module and the PXI backplane. The CR board also contains the SMA trigger connector. The front panel and its’ extractor are mechanically fixed to the CR board. A schematic drawing of the CR is shown in the figure below.

The CR contains a start trigger I/O connector with active circuitry which enables bidirectional use of the start trigger line. The start trigger line is compatible with 5.0 V logic signals. When used as output, a high impedance load needs to be connected to the SMA connector.

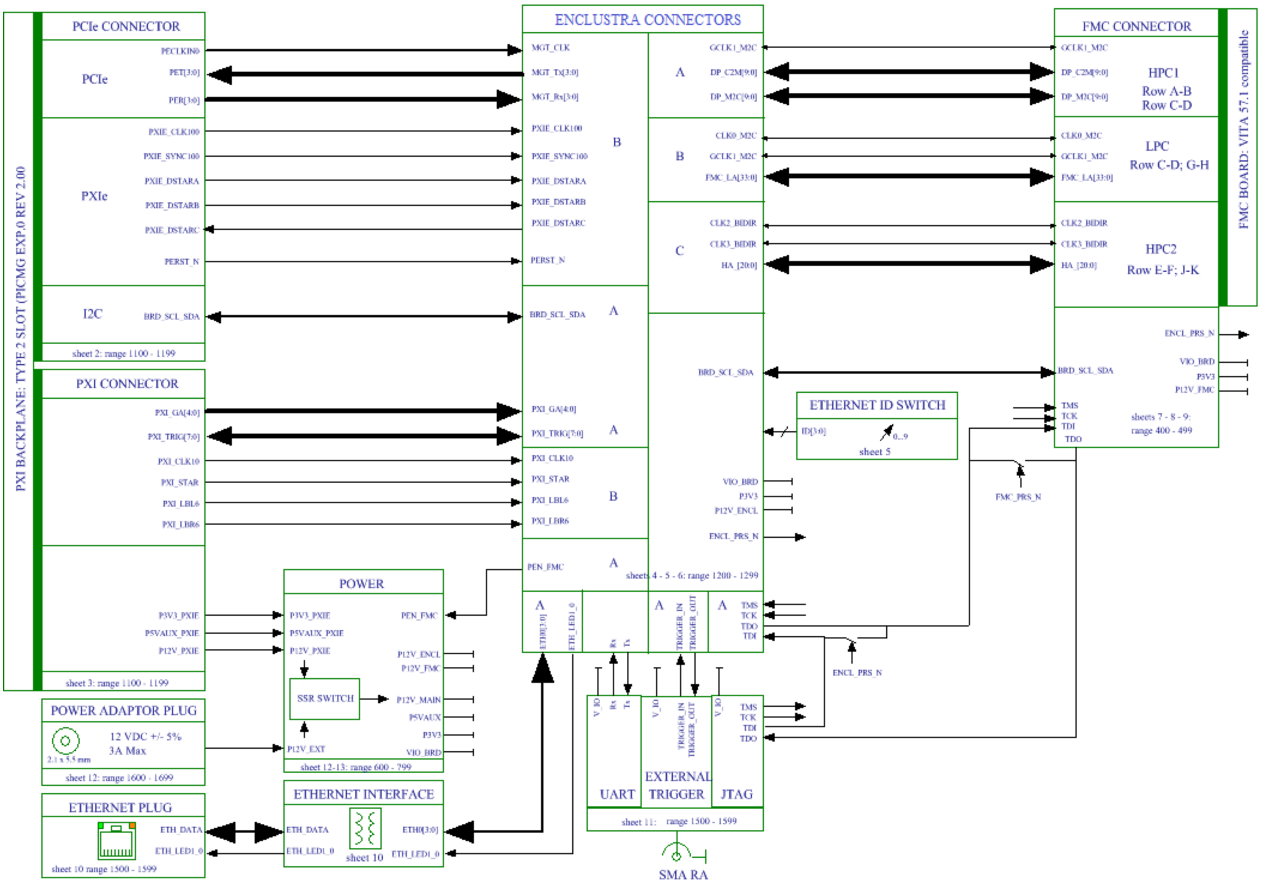

Figure 6: Schematic block diagram of the Carrier.

The CR board contains hardware provisions for an ethernet interface. The FPGA code for using the interface is currently not implemented on the Enclustra FPGA module. Therefore, the ethernet interface cannot be used.

3.5.4 Enclustra FPGA module

The Enclustra FPGA module (Mercury ZX5 ME-ZX5-15-2I-D10-R3) handles the transmission of neural data packets from the BSC over the PCIe interface to the PC.

Also, a switch matrix for start trigger connectivity is implemented on the FPGA. The trigger input signal is routed to the BSC, where it triggers the start of transmission of neural data packets or is included in a trigger field in the data packet in case neural data transmission is already ongoing. This trigger input signal can be configured to connect to the SMA connector on the CR, any of the first 7 trigger lines on the PXI bus (PXI_TRIG<0:6>), a user defined trigger on the neural data or a software trigger.

Any of the input triggers can also be routed as an output trigger to the SMA connector on the CR, and to any of the first 7 trigger lines on the PXI bus. This allows distribution of a common trigger throughout all modules in the PXI chassis or external instruments. For example, an input software trigger can be routed to the SMA connector on the CR as an output. Or an input trigger on the SMA connector on the CR can be transmitted to a line on the PXI_TRIG<0:6> bus, where it can be used by other BS modules or other PXI modules as an input trigger.

The Enclustra FPGA module can also generate a slow configurable clock (SYNC) which can be transmitted to the BSC where it is sampled synchronous with the neural data. It can also be transmitted to other BS modules in the PXI chassis.

4 BSC FPGA Specification

4.1 Top level block diagram

A functional block diagram of the BSC FPGA is shown in the figure below. The FPGA architecture is divided in two main parts: System and Application. The System contains all hardware functionality which is closely coupled to the functionality of the BS platform: BSC PCB, CR and interface to the Enclustra FPGA module. The Application part is designed to handle the bidirectional communication to the Neuropixels probe, compliant with the probes' interface specification.

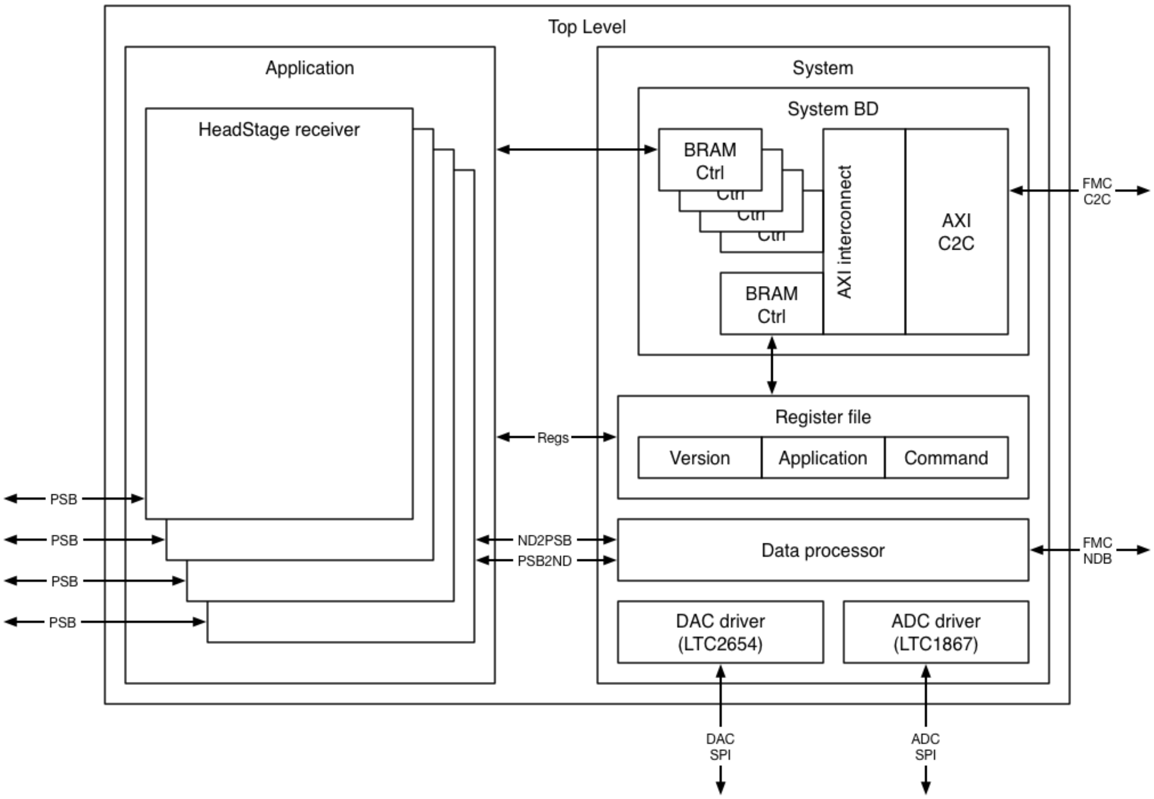

Figure 7: BSC FPGA block diagram.

4.2 Application module

The Application module contains 4 identical Headstage receiver modules, 1 for each port on the BSC. A functional block diagram of the Application module is shown in the figure below.

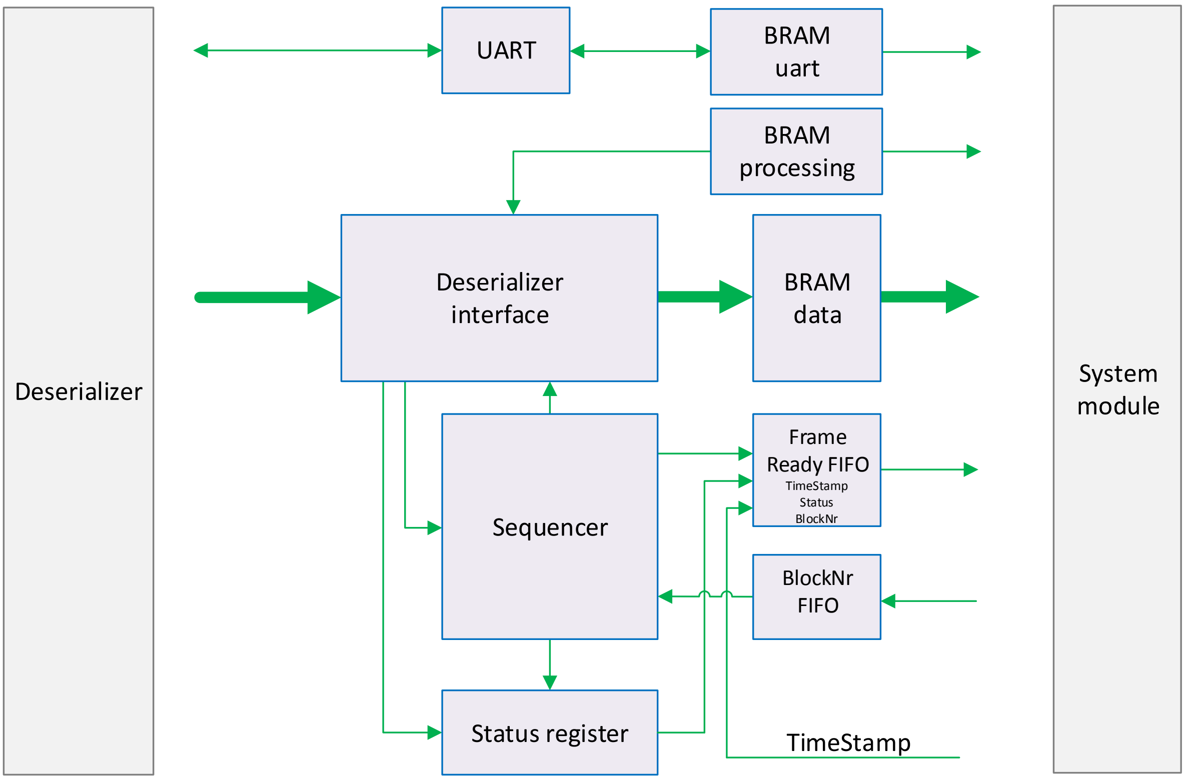

Figure 8: Block diagram Application module.

- The Deserializer interface module demultiplexes the PSB data bus as coming from the deserializer chip on the BSC into 384 channels, AP and LFP. This module can also be configured to work in ADC mode, in which it will demultiplex the datastream into 32 ADCs. The data is written to the BRAM data, in packets (Blocks) of 384 channels with 1 sample for each channel (or 32 ADCs). To each sample, the Deserializer interface also appends the gain correction value and comparator correction value, which it obtained from the module BRAM processing. The gain correction value and comparator correction value are probe specific correction parameters which are uploaded to the BSC FPGA by API functions.

- The BRAM data module is a BRAM memory block used as a memory buffer between the Application and System module on the BSC FPGA. The BRAM has space for up to 8 Blocks of neural data (1 Block = 384 channels, 1 sample per channel). The Deserializer interface module writes to the BRAM data module, to a location specified by the Sequencer Module.

- The Sequencer module keeps track of which one of the 8 locations in the BRAM data can be written to by the Deserializer interface module. The Sequencer flags to the System module, via the Frame Ready FIFO, if 1 complete frame of data is available in the BRAM data module. The Sequencer gets flagged by the System module via the BlockNr FIFO module if the System module has finished reading a Block of data. The Sequencer tells the Deserializer interface which the next BRAM data location is it can write to.

- The Frame Ready FIFO: this register is used to flag to the System module which is the next BRAM data location (Blocknr) it needs to read from. A status byte from the Status register and a 30-bit timeStamp are added for each FIFO entry.

- The Status register contains 8 data bits:

  - Trigger: Status of the input trigger line.
  - LFP_nAP: indicates if the current frame is a AP (0) or LFP (1).
  - CountErr: the 20-bit counter on the PSB_DATA<6> line is monitored. An error flag is raised in case this counter is not incrementing in steps of 1.
  - SerDesErr: the deserializer |ERR signal is monitored.
  - LockError: the deserializer |LOCK signal is monitored.
  - PopError: No space left for storing incoming neural data in the BSC FIFO.
  - EXT_IN: SYNC clock coming from the Enclustra FPGA module, sampled synchronously with the neural data.
  - SyncError: the 10-bit synchronization word on the PSB_SYNC line is monitored. In case the received synchronization word is wrong, this bit is set.
- The BlockNr FIFO: When the System module has finished reading a Block of data from the BRAM data module, it writes the location of the Block to this FIFO. The Sequencer module can send this location to the Deserializer interface as free to write again.
- The BRAM processing module: Holds probe specific configuration information, related to gain correction and comparator correction which is appended to the neural data stream to the correct channel and applied in the System module. Data is written to this BRAM via the AXI Chip2chip interface.
- UART and BRAM UART module: The UART interface between BSC FPGA and deserializer is used for configuration of SerDes and probe configurations such as gain and electrode selection. Also, the EEPROM on flex and HS are accessed via this interface. Data is written to the BRAM UART via the AXI Chip2chip interface.

4.3 System module

The System module acquires the neural data from the 4 Headstage receiver modules in the Application module, optionally applies gain correction parameters, packetizes the data, and queues it in a FIFO buffer which is read out by the PCIe interface implemented on the Enclustra FPGA module.

A functional block diagram of the System module is shown in the figure below.

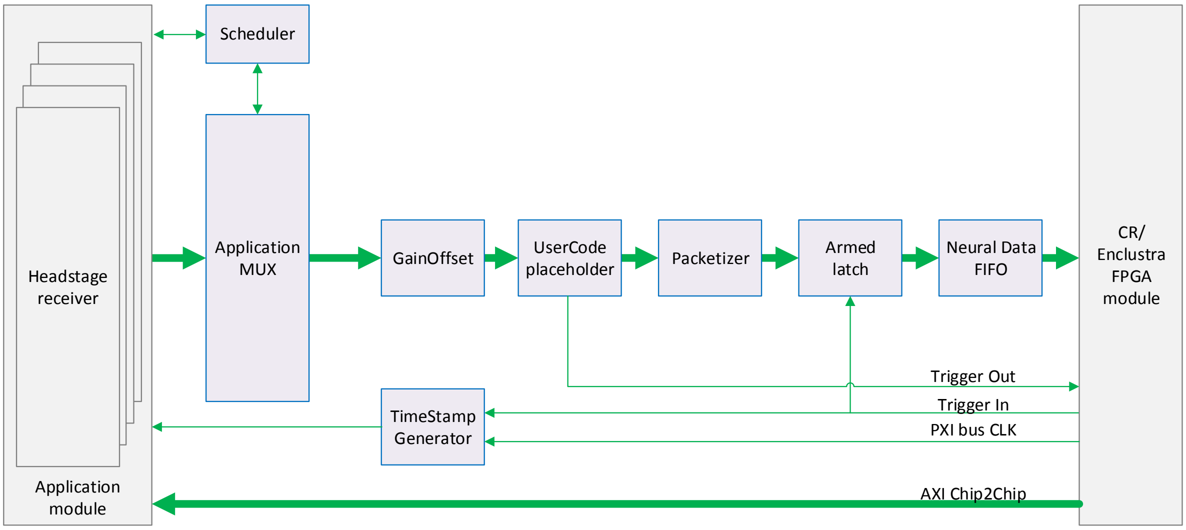

Figure 9: Block diagram System module.

- The Application MUX module: reads the neural data from the Application BRAM data modules as assigned by the Scheduler module.

- The Scheduler module: Monitors the Headstage receiver modules on the Application module to check where neural data is available. Interfaces with the Headstage receiver module the via the Frame Ready FIFO module and BlockNr FIFO module.

- The GainOffset module:
  - The neural data coming from the probe ASIC is centered around mid-supply (0.6V). This fixed offset is removed to obtain a 0-centered signal.
  - The analog channels on the probe ASIC have negative gain coefficients. The neural data is inverted again in this module to undo this effect (multiplied by gain -1).
  - Comparator correction parameters can be applied in this module to reduce the number of missing bits on the probe ADCs.
  - The analog channels on the probe ASIC have a slight tolerance on their gain factors. The deviation from the nominal gain on is measured during probe testing on a per electrode basis, and these gain correction parameters can be applied to the neural data.

- The Usercode module: a placeholder module which can optionally be used by the user to process or monitor neural data, such as spike sorting or threshold detection.

- The Packetizer module: wraps the neural data in packets, adding a header. The header contains timestamp, trigger, source and status information.

- The Armed latch module: Neural data packets from the Packetizer module are discarded while the BS is kept in armed mode. A start trigger arriving at the BSC starts transfer of neural data packets from the Packetizer module to the Neural data FIFO module. This enables synchronization of neural data over multiple BS modules in the PXI chassis, and for synchronization with other instruments. The data path from deserializer to Packetizer module is not suspended while the BS is kept in armed mode, such that the Usercode module can optionally monitor the neural data for generating a trigger.

- The Neural Data FIFO module: neural data packets are queued here, waiting to be read by the PCIe interface on the Enclustra FPGA module. If FIFO gets full, the System module stops reading data from the Application module.

- The TimeStamp generator module: a 30-bit timestamp code is generated and transmitted to the Application module where it is appended to the neural data packets. The TimeStamp generator module is kept in reset at 0 while the BS is armed and starts when a start trigger arrives to the BSC. Subsequent trigger signals do not reset the timestamp unless the BS is armed again. The timestamp is generated by a 100 kHz clock, which is derived from the PXI bus clock (PXI_CLK 100 MHz) and therefore synchronous with all BS modules in the PXI chassis.

  Each probe uses its' own 30 kHz sample clock oscillator, and due to slight tolerances on the oscillators' frequency, the number of samples from each probe would deviate between the probes over time. Using a 100 kHz timestamp clock allows to accurately align in time the samples from all probes.

5 Enclustra FPGA module

The Enclustra FPGA module transmits neural data and configuration data between PC and BSC. Selection and routing of trigger signals is implemented in this FPGA.

5.1 PCIe interface

Neural data is received from the BSC FPGA and transmitted over the PCIe interface on the PXI chassis backplane to the PC. Configuration data is received from the PC and used for configuration of trigger signals or forwarded to the BSC FPGA for configuration of probe or BSC.

5.2 Trigger and SYNC clock

The neural data stream generated by the probes connected to the BS can be synchronized with other BS modules in the PXI chassis, other modules in the chassis or external instruments.

The trigger signal starts the acquisition of neural data on the BSC. The SYNC clock is a clock signal which is recorded synchronously with the neural data.

Connectivity of Trigger and SYNC clock is implemented in a configurable switch matrix on the Enclustra FPGA module.

5.2.1 Trigger

The trigger signal is used to start the acquisition of neural data on the BSC. Subsequent triggers are also recorded and transmitted synchronously with the neural data. The trigger signal can be shared over all instruments in the PXI chassis via the PXI backplane.

The source for the trigger signal can be software, external from the SMA connector, another PXI module in the chassis (Neuropixels or third party) or a user defined trigger based on the neural data. Multiple trigger sources can be selected.

The PXI_TRIG<0:6> lines on the PXI backplane bus are used for sharing trigger signals between different modules in the PXI chassis.

The user can select for each BS module which trigger signals are used as input trigger. The following trigger lines can be selected as trigger input:

- Software trigger
- SMA connector on the CR
- PXI_TRIG<0:6> on the PXI bus.
- USER<0:9>: User defined trigger from the BSC FPGA (based on neural data)

The user can select for each BS module which trigger input signals are used as output trigger and to which trigger line. The following trigger inputs can be selected to drive the trigger output line:

- Software trigger
- SMA connector on the CR

- PXI_TRIG<0:6> on the PXI bus.
- USER<0:9>: User defined trigger from the BSC FPGA (based on neural data)

The following trigger lines can be selected to be used as trigger output line:

- SMA connector on the CR
- PXI_TRIG<0:6> on the PXI bus.
- None

Multiple PXI modules should not drive the same PXI_TRIG<0:6> line simultaneously. This can be prevented by configuring the output trigger for each PXI slot to another PXI_TRIG line. This is the responsibility of the user.

The SMA I/O connector can be used as input or output line for the trigger, When used as a trigger output, the user can configure the polarity of the trigger pulse on this signal, active low or active high. The output pulse has a fixed width of 1ms. When used as a trigger input, the user can configure the edge sensitivity. The trigger input requires a minimum pulse width of 16 ns.

- Example case 1:

  2 Neuropixels BS modules are plugged in the PXI chassis. The user wants to start data acquisition from software. BS module 0 is configured to use SW trigger as input trigger. BS module 0 is configured to use SW trigger signal as output trigger to the PXI_TRIG<0> line. BS module 1 is configured to use PXI_TRIG<0> as input trigger.

- Example case 2:

  2 Neuropixels BS modules are plugged in the PXI chassis. The user wants to start data acquisition triggered by an external instrument. BS module 0 is configured to use its' SMA connector as input trigger. BS module 0 is configured to use SMA I/O trigger signal as output trigger to the PXI_TRIG<0> line. BS module 1 is configured to use PXI_TRIG<0> as input trigger.

- Example case 3:

  2 Neuropixels BS modules and one DAQ module are plugged in the PXI chassis. The user wants to start data acquisition triggered by neural data on all BS ports and by the DAQ module. The DAQ module is configured to use PXI_TRIG<0> as output trigger. BS module 0 is configured to use USER<0>, PXI_TRIG<0> and PXI_TRIG<2> as input trigger, and to use USER<0> trigger signal as output to PXI_TRIG<1> line. BS module 1 is configured to use USER<0>, PXI_TRIG<0> and PXI_TRIG<1> as input trigger, and to use USER<0> trigger signal as output to PXI_TRIG<2> line.

5.2.2 SYNC

The SYNC clock is a slow clock signal which is recorded on the BSC FPGA synchronously with neural data.

The source of the SYNC clock can either be a local clock generator on the Enclustra FPGA, a clock signal coming from another BS module, or an external instrument. The source of the SYNC clock is selected with the switch matrix on the Enclustra FPGA.

The SYNC clock can be shared over all BS modules in the PXI chassis via the PXI backplane line PXI_TRIG<7>.

The SMA I/O connector can be used to connect a SYNC clock from an external source, or to output the internal SYNC clock generator.

6 API user manual

This chapter summarizes the procedure for calling basic API functions for using the probe. More detailed information on API functions can be found in chapter 7.

6.1 Switching on the system

1. After switching on the PXI system, the general status LED on the BS module is on (green), The power to the 4 ports on the BS module is disabled by default at switch-on. The user can safely connect probe, HS and cable to the BS ports.

2. The user configures the size and partitioning of the PCIe buffer on the PC, using the 'setParameter' function with parameter 'NP_PARAM_BUFFERSIZE' and 'NP_PARAM_BUFFERCOUNT'.

3. After connecting the probes to the BS ports, the 'openBS' function is called. This function opens a connection between the PC and the BS/BSC. The power supply to all ports on the BS module is enabled. The probe status LED of all ports which have a probe plugged in switches on to red.

4. The user calls the 'openProbe' function. This function enables the probe and initializes the HS and BSC:

    - The SerDes is configured.
    - The probes' PSB bus (by setting the REC bit in the OP_MODE register) is activated.
    - The probes PSB clock is set to 11.7 MHz (by setting the PSB_F bit in the REC_MODE register).

    If the configuration was successful, the HS heartbeat LED is blinking. The probe status LED on the BS front panel corresponding with the selected port switches to green.

    An error is returned if communication between probe and BS is malfunctioning.

5. The user can read the probes' ID code using the function 'readId'.

6. The user writes the ADC calibration data to the probe, using the function 'setADCCalibration', with the probes' ADC calibration file.

7. The user writes the probes' gain correction data to the BSC, using the function 'setGainCalibration', with the probes' gain calibration file.

8. The user calls the 'init' function. This function initializes the probe in a default state. The default settings for the probe are listed in [Shank configuration] and in [Base configuration]:

    a. Channels are connected to the electrodes of Bank 0.

    b. All channels are set to use the external reference electrode.

    c. Internal reference electrode and tip electrode are disconnected.

d. All channels are programmed for AP gain setting 4 (x1000) and LFP gain setting 0 (x50).

e. All channels are active (not standby) and in default AP bandwidth setting (300 Hz high pass cut-off).

f. The probe is set to recording mode via the REC bit in the OP_MODE register.

g. The ADC calibration settings (‘setADCCalibration’) and gain correction settings (‘setGainCalibration’) are not overwritten by this function.

The heartbeat signal is no longer visualized on the HS.

The system is now ready for further configuration steps and recording.

6.2 Configuring the probe

0. Optionally, the user can select the CAL_SIGNAL as an input to the probe, instead of the recording electrodes. The CAL_SIGNAL can be used to connect a test signal from a function generator to the probe. Call the functions ‘setOPMODE’ and ‘setCALMODE’ to select the input configuration. By default, the probe is configured to use the input from the recording electrodes after calling the ‘init’ function. Therefore, calling the ‘setOPMODE’ and ‘setCALMODE’ is only required if the user wants to use the CAL_SIGNAL.

1. The user selects the recording electrode for each channel (excluding channel 191), using the function ‘selectElectrode’.

2. The user selects the reference input (External, Internal or Tip) for each channel (excluding channel 191), using the function ‘setReference’.

3. The user configures the AP and LFP gain of each channel (excluding channel 191), using the function ‘setGain’. It is not required to call the setGainCalibration again.

4. If required the user can set the bandwidth limit for each channel, using function ‘setAPCornerFrequency’.

5. If required the user can set selected channels in standby, using the function ‘setStdb’.

6. The above-mentioned probe configuration functions only modify the API parameters, but do not transmit the configuration to the probe. In order to write the updated configuration to the probe, the user calls the function ‘writeProbeConfiguration’.

6.3 Configuring trigger

After calling the ‘openProbe’ command, the probe is continuously acquiring and transmitting data. This data is transmitted with negligible delay to the BSC FPGA, where it is discarded until a start trigger is received.

The triggers are routed between the PXIe modules via the configurable trigger matrix which is implemented on the Enclustra FPGA module.

Two function sets are provided to configure the trigger matrix. The user can choose either which meets the applications' requirements:

1. 'setTriggerInput' and 'setTriggerOutput' which allows the user to select one trigger source or trigger sink.

2. 'setTriggerBinding' and 'getTriggerBinding' which allows to set multiple trigger sources and trigger sinks using a mask parameter. This function provides a more intuitive way to configure the trigger matrix.

To prepare the system for triggering, the user follows this procedure:

1. The user configures the trigger mode of the BS modules in the chassis, using functions 'setTriggerInput' and 'setTriggerOutput'. Alternatively, the 'setTriggerBinding' function can be used.

2. If required, the user sets the polarity of the input or output trigger using the function 'setTriggerEdge'.

3. The user sets the system in arm mode, using the function 'arm'. This function sets the timestamp generator in the BSC FPGA to zero, and the system waits for an incoming trigger. The general status LED on the front panel is green.

The software trigger is by default enabled after calling the 'openBS' funtion.

6.4 Configuring SYNC

Use the following procedure to configure the SYNC clock:

1. Select which BS is master over the SYNC clock. All other BS modules in the PXI chassis use the same SYNC clock as the master. Use function 'setParameter' with parameter 'NP_PARAM_SYNCMASTER' to select the master BS.

2. Configure the source of the SYNC clock on the master module: either SMA connector or internal SYNC clock generator. Use function 'setParameter' with parameter 'NP_PARAM_SYNCSOURCE'.

3. If the internal SYNC clock source is selected, set the frequency/period of the clock source. Use function 'setParameter' with parameter 'NP_PARAM_SYNCFREQUENCY_HZ' or with parameter 'NP_PARAM_SYNCPERIOD_MS'.

4. In case the selected SYNC clock signal needs to be output on the SMA connector, use the function 'setTriggerOutput()' with parameters (slotID, TRIGOUT_SMA, TRIGIN_SHAREDSYNC). This can be done for any BS module in the chassis, it does not necessarily have to be the master BS. Alternatively, the function 'setTriggerBinding' with parameters (slotID, SIGNALLINE_SMA, SIGNALLINE_SHAREDSYNC) can be used also.

Example:

1. Set slot #2 to be master of the SYNC signal:

setParameter(NP_PARAM_SYNCMASTER, 2)

2. Set the SYNC master to use the internal SYNC clock generator:

   setParameter(NP_PARAM_SYNCSOURCE, TRIGIN_SYNCCLOCK)

   Alternatively, set the SYNC master to use the SMA input connector:

   setParameter(NP_PARAM_SYNCSOURCE, TRIGIN_SMA)

3. Set the frequency of the SYNC clock generator to 10 Hz:

   setParameter(NP_PARAM_SYNCFREQUENCY_HZ, 10)

4. Output the SYNC signal on the SMA connector of slot #3:

   setTriggerOutput(3, TRIGOUT_SMA, TRIGIN_SHAREDSYNC)

   or:

   setTriggerBinding(3, SIGNALLINE_SMA, SIGNALLINE_SHAREDSYNC)

6.5 Recording data

The system buffers the incoming neural data in the RAM memory of the PC. The API can stream this data to binary files on the PC without data loss. Functions are provided to read data from these binary files. The API also provides functions to read neural data from the RAM memory directly, in different formats.

The API can stream data from RAM memory to disc without data loss. It is a prerequisite to have a SSD drive with sufficient write speed (each probe generates data at about 160 Mbps). If the user reads data from the RAM FIFO directly without using the APIs' built-in data to disc streaming functionality, it is the responsibility of the users' application to read data from the FIFO on time. Data which is not read from the FIFO in time is overwritten with new data. A function is provided to monitor the filling level of the FIFO.

1. If applicable, prior to receiving the trigger signal, enable streaming of neural data to a defined binary file. 1 File per BS module. Use the functions 'setFileStream' and 'enableFileStream'.

2. If applicable, generate a software trigger using function 'setSWTrigger'. When a trigger arrives on the BS module, the general status LED turns from green to blue.

4. The neural data can be read by the user application from the FIFO memory using the functions 'readAPFifo', 'readLFPFifo' or 'readElectrodeData'.

5. The data acquisition by the BSC FPGA is stopped and the timestamp is reset to 0 by calling the 'arm' function again.

6. If applicable, call the function 'enableFileStream' to disable data streaming to disc.

The API built-in data to disc streaming functionality saves the probe data to the binary files using a dedicated data packet format. The packet consists of a header and a payload. Data packets are 32 bits aligned. The 10-bit neural data samples are packed together in 32-bit words. For 384 channels, the payload length is fixed to 120 words of 32 bit. A data packet has either AP or LFP data. This is indicated by a bit in the header.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MagicID[28]																												Type[4]			
1	Format[8]								SeqNr[8]								SampleCount[16]															
2	Timestamp[32]																															
3	CRC[16]																SourceID[8]								Status[8]							

Table 1: Data packet header.

<table>
<tr><td>Field ID</td><td>#bits</td><td>Description</td></tr>
<tr><td>MagicID</td><td>28</td><td>Constant value that indicates the start of a packet.
The value is chosen such that the probability of occurrence of the value inside the data stream is minimal. Value = 0xF00BABE
This allows software to scan through an arbitrary binary data dump to quickly find the start of a data packet. While scanning, the validity of a matching MagicID value can be assured using the header’s CRC.
Note: For Neuropixels, data packets are always of the same size and are guaranteed to be complete (including header). In this case the MagicID merely acts as a sanity check field.</td></tr>
<tr><td>Type</td><td>4</td><td>Type of header. This field can be regarded as part of the MagicID field since the type of header is unique for each variant of this protocol.<table><tr><td>0</td><td>Not used</td></tr><tr><td>1</td><td>Neuropixels format</td></tr><tr><td>2 - 15</td><td>Reserved</td></tr></table></td></tr>
<tr><td>Format</td><td>8</td><td>Data sample format<table><tr><td>Bitspersample[4]</td><td>Reserved[3] (“000”)</td><td>Packed</td></tr></table>**Bitspersample**: number of bits per sample-1. Allowed range is 1 to 16 bits per data element. Example: 0xF = 16-bit data. For Neuropixels, this field is set to 10.
Packed: if set, the data is stored in packed format, if not set, the data is aligned to the next octet that is able to store a full sample. For Neuropixels, this bit is always set.</td></tr>
<tr><td>SeqNr</td><td>8</td><td>Packet sequence number. This value can be used as a logical sequence in a packet stream. The value automatically wraps from 255 to 0.</td></tr>
<tr><td>SampleCount</td><td>16</td><td>Number of samples in the packet payload.
Total payload size in bytes = ceiling (SampleCount * bitspersample / 8)</td></tr>
<tr><td>TimeStamp</td><td>32</td><td>100 kHz timestamp clock value. 30-bit value. 2 MSBs are fixed to 0.</td></tr>
<tr><td>CRC</td><td>16</td><td>Header CRC. Polynomial = CRC16 X25. Seed = 0xFFFF.
The CRC is calculated 16-bits at a time. Since CRC is located in the most significant 16-bit word of the last 32 header bits, the CRC calculation starts with 16 LSBs of word 0, followed by 16 MSBs of word 0, …</td></tr>
<tr><td>SourceID</td><td>8</td><td>Identification of the source of data in the current packet<table><tr><td>0..2</td><td>Port number on the BS module</td></tr><tr><td>3..7</td><td>Geographical address of the BS module in the chassis</td></tr></table></td></tr>
</table>

Status	8	Status data	
		0	Trigger: This flag is set if a trigger occurred during receiving the payloads’ data at the BSC FPGA.
		1	LFP_nAP; The packet contains LFP (=1) or AP (=0) data.
		2	CountErr; Every PSB frame has an incrementing count index. If the received frame count value is not as expected, data loss has occurred, and this flag is set.
		3	SerDesErr; If a deserializer error (hardware pin) occurred during reception of this frame this flag is set.
		4	LockErr; If a deserializer loss of lock (hardware pin) occurred during reception of this frame this flag is set.
		5	PopError; Set whenever the ‘next blocknumber’ round-robin FIFO is flagged empty during request of the next value. This indicates a full FIFO buffer on the QBSC FPGA, and occurs typically when the PCIe data transfer gets interupted.
		6	EXT_IN: The SYNC clock generated on the Enclustra FPGA is sampled on the BSC FPGA and added to the data packet.
		7	SyncError; Front-end receivers are out of sync. One or more of the PSB frames is invalid.

Table 2: Description of data packet header fields.

The probe data in the recorded binary files can be read using the API functions described in chapter 7.7.

The ‘readElectrodeData’ function acquires data directly from RAM FIFO and returns the data to the user in the structure type ‘electrodePacket’. The electrodePacket contains one sample of each of the 384 LFP channels, and 12 samples of each of the 384 AP channels. The structure contains the following members:

- uint32_t timestamp[12]: 12 samples of the 100 kHz timestamp clock as sampled synchronously with the incoming neural data on the BSC FPGA.
- int16_t apData[12][384]: 12 samples for each of 384 AP channels.
- int16_t lfpData[384]: 1 sample for each of 384 LFP channels.
- uint16_t Status[12]: 12 samples of the STATUS byte as described in Table 2: contains trigger status, SYNC clock and error flags.

6.6 Switching off

The user can switch off the power supply to the HS cable by calling the ‘closeBS’ function. This way the user can connect another probe or HS to the system, without switching off the PXI chassis.

6.7 Using the HST module functions

The functionality of the HS can be verified with the Headstage Test module. Several tests are implemented on a microcontroller on the HST module and these are activated by calling API functions.

1. Connect the HS to the BS with the cable as you would do for an experiment with a probe.

2. Connect the HS to the HST module by plugging the flex of the HST module in the ZIF connector of the HS.

3. Switch on the power to the HS and HST module by calling the 'openBS' function. The 'openBS' function switches on the power supply to all ports on the selected slot. The red OK led on the HST module should switch on. If this is not the case, either the cable between BS and HS or the supply on the HS is malfunctioning.

4. Open a connection to the HST module by calling the 'openHST' function. If this function fails with error 24 (NO_LOCK), the PSB_CLK signal connection between the HST module and the serializer on the HS or the serializer itself is failing.

5. Test the functionality of the I2C bus by calling the 'HSTestI2C' function. If the function fails with error 8 (TIMEOUT), the I2C bust between the HS and HST module is failing.

6. Check the PSB_CLK signal by calling the 'HSTestPCLK' function. This test is redundant: if the 'openHST' function has completed successfully, the PSB_CLK clock signal is present.

7. Check the VDDA1V8 supply. The function 'HSTestVDDA1V8' checks if both pins on the ZIF connector contain an analog 1.8 V supply signal within the required tolerances. If the test fails, the functions returns 'BISTERROR', and prints a message with more details on the failure.

8. Check the VDDD1V8 supply. The function 'HSTestVDDD1V8' checks if both pins on the ZIF connector contain a digital 1.8 V supply signal within the required tolerances. If the test fails, the functions returns 'BISTERROR', and prints a message with more details on the failure.

9. Check the VDDA1V2 supply. The function 'HSTestVDDA1V2' checks if both pins on the ZIF connector contain an analog 1.8 V supply signal within the required tolerances. If the test fails, the functions returns 'BISTERROR', and prints a message with more details on the failure.

10. Check the VDDD1V2 supply. The function 'HSTestVDDD1V2' checks if both pins on the ZIF connector contain a digital 1.8 V supply signal within the required tolerances. If the test fails, the functions returns 'BISTERROR', and prints a message with more details on the failure.

11. Check the MCLK signal. The function 'HSTestMCLK' can detect if the MCLK line is floating, grounded or pulled high to one of the supplies. The function returns 'BISTERROR' if the MCLK signal is not correctly detected and prints a message with more details on the failure.

12. Check the functionality of the PSB bus by calling the function ‘HSTestPSB’. This function checks the PSB_SYNC line and the PSB_DATA<0:6> lines. The function first checks PSB_SYNC and subsequently PSB_DATA<0> to <6> and returns ‘BISTERROR’ at the first failing signal without checking the remaining signals. The function prints a message indicating which signal has failed.

    Note: PSB_SYNC and PSB_DATA are routed to the input port of the serializer via a flip-flop on the HS. An electrical short-circuit on the input port of the serializer cannot be detected with the current version of the HST module.

13. Check the functionality of the NRST signal by calling the ‘HSTestNRST’ function. This function checks if the NRST is kept high by the POR circuit on the HS. The function returns ‘BISTERROR’ if the NRST signal is floating or kept low.

14. Check the functionality of the REC_NRESET signal by calling the ‘HSTestREC_NRESET’ function. This function checks if the REC_NRESET can be set correctly by the HS. The function returns ‘BISTERROR’ if the REC_NRESET signal is floating or fixed to GND or VDD.

15. Check the functionality of the test signal generator which is located on the HS. The test signal generator itself is a test circuit used for checking the signal path integrity on the probe with the ‘bistSignal’ API function, by applying a test signal to the CAL_SIGNAL input of the probe. The ‘HSTestOscillator’ function checks if the test signal oscillator can be correctly switched on and off with the OSC_PDOWN signal, and if the test signal oscillator output signal falls within the specified tolerances. The function returns ‘BISTERROR’ if the test signal generator is not functioning correctly.

16. Switch off the power to the ports by calling the ‘closeBS’ function. The HST module can now be unplugged from the HS.

7 API Specification

7.1 Global parameter settings

Global parameters configure the system settings which affect all Neuropixels PXIe modules in the chassis. The function can configure the PCIe buffer which is used by the Enclustra driver and the SYNC clock.

Function
setParameter(np_parameter_t paramid, int value)
Arguments
paramid: which parameter to configure
value: value to assign to the selected variable
Returns

A version which takes a value of type double is available.

Function
setParameter_double(np_parameter_t paramid, double value)
Arguments
paramid: which parameter to configure
value: value to assign to the selected variable
Returns

Use the enum np_parameter_t to select which parameter is configured with the function:

- NP_PARAM_BUFFERSIZE: set the size of a partition of the PCIe buffer.
    - Unit: Bytes
    - Max: TBD
    - Min: 32768
    - Default: 128kB

  Each Neuropixels module in the chassis gets its own PCIe buffer. All modules get an equally sized PCIe buffer partition, regardless of the number of probes connected to the module.

  The partition is subdivided into 4 subpartitions. Each probe connected to the module fills its’ own subpartition. If less than 4 probes are connected to the slot, certain subpartitions remain empty. When a subpartition is full, it is scheduled to be transmitted to the user FIFO by the API. The size of the partition therefore defines the latency of data transfer. Latency for a module with 4 probes or 1 probe remains the same because the filling of the subpartition triggers the data transfer. The number of neural data samples which fits in a certain partition size can be calculated from the size of the neural data packets as described in chapter 6.5. The size of a neural data packet which contains 1 sample for each of the 384 channels on 1 probe, either AP or LFP, equals 496 bytes.

NP_PARAM_BUFFERSIZE must be set before calling the openBS function.

- NP_PARAM_BUFFERCOUNT: set the number of partitions of the PCIe buffer.
  - Unit: partitions
  - Max: 1024
  - Min: TBD
  - Default: 64 partitions

  The number of partitions in combination with the partition size defines the total PCIe buffer size.

  NP_PARAM_BUFFERCOUNT must be set before calling the openBS function.

- NP_PARAM_SYNCMASTER: sets the slot which acts as a master on the SYNC clock. The master slot transmits the SYNC clock signal over the PXI bus (PXI_TRIG_7) to the other slots in the chassis. The other slots use the signal on the PXI_TRIG_7 line as SYNC input.

  After calling openBS, each slot uses its own internal SYNC clock signal, no slot is master on PXI_TRIG_7. Therefor the SYNC master must be set in order to have the same SYNC clock on all Neuropixels modules in the chassis.

  Note: setting the SYNC clock master is also possible using the setTriggerBinding functions. It is however safer to use the setParameter function as this makes sure that only one slot is set as master.

- NP_PARAM_SYNCFREQUENCY_HZ: sets the frequency of the SYNC clock. This parameter overwrites the setting made by NP_PARAM_SYNCPERIOD_MS.
  - Unit: Hz
  - Max: Hz
  - Min: TBD
  - Default: TBD Hz

- NP_PARAM_SYNCPERIOD_MS: set the period of the SYNC clock. This parameter overwrites the setting made by NP_PARAM_SYNCFREQUENCY_HZ.
  - Unit: ms
  - Max: TBD
  - Min: TBD
  - Default: TBD ms

- NP_PARAM_SYNCSOURCE: sets the source for the SYNC clock. This can be either the SMA connector or the internal SYNC clock generator on the BS module. Use the enum triggerInputline_t (ref. chapter 7.6.2) to select the SYNC clock source:

  - TRIGIN_SMA: an external SYNC clock generator can be connected to the SMA connector of the BS module which is configured as master (NP_PARAM_SYNCMASTER).
  - TRIGIN_SYNCCLOCK: the internal SYNC clock generator of the BS module which is configured as master.

Default: the internal SYNC clock generator is selected by default after calling openBS.

7.2 Opening and initialization functions

7.2.1 Open BS

The ‘openBS’ function initializes the CR FPGA and BSC FPGA:

- setup configuration link over PCIe interface.
- CR FPGA and BSC FPGA reset.
- Check hardware and software versions: BSC FPGA, BS FPGA, API.
- Power is applied to all 4 ports on the BSC.

The ‘openBS’ function does not initialize the probe or HS.

Function
openBS (unsigned char slotID)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

The user needs to wait 1 second after calling the ‘openBS’ command before calling the ‘openProbe’ command, to allow the switch-on sequence of the probe to finish.

7.2.2 Scan BS

The ‘scanBS’ function verifies whether a Neuropixels BS module plugged in a chassis slot. The function returns ‘SUCCESS’ if a BS module is plugged in, ‘NO_SLOT’ if not.

Function
scanBS (unsigned char slotID)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

7.2.3 Open probe

The ‘openProbe’ function initializes the SerDes link and HS, but not the probe itself. The argument selects which of the 4 HS ports on the BSC to enable.

Calling the ‘openProbe’ function with a certain port number does not close any previously opened ports.

The ‘openProbe’ function initializes the SerDes and HS:

- Configure the deserializer registers.
- Configure the serializer registers.
- Toggle the DIG_NRESET bit in the REC_MOD register of the probe memory map.
- Set the REC bit in the OP_MODE register of the probe memory map to 1 (required to enable the PSB_SYNC signal).
- Set the REC_NRST signal to the probe to enable the PSB bus.
- Enable the heartbeat signal on the HS.
- Set the BS FPGA for this port in electrode mode.

The ‘openProbe’ function can be used to quickly query if a port has hardware connected.

Function
openProbe (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: which HS/probe (valid range 1 to 4)
Returns

7.2.4 Initialize

The function resets the connected probe to the default settings: load the default settings for configuration registers and memory map; and subsequently initialize the probe in recording mode:

- Set the memory map and test configuration registers to the default values (ref. chapter 3.1.3).
- Set the shank configuration register map to the default values: recording electrodes of bank 0 connected, external reference electrodes connected.
- Set the base configuration register map to the default values, excluding the ADC comparator values (ADCxxSlope, ADCxxFine, ADCxxCoarse, ADCxxCfix, ADCxxCompP, ADCxxCompN).

    - All channels are set to use the external reference electrode.
    - All channels are programmed for AP gain setting 4 (x1000) and LFP gain setting 0 (x50).
    - All channels are active (not standby) and in default AP bandwidth setting (300 Hz high pass cut-off).
- Set the REC bit in the OP_MODE register in the memory map to high (ref. chapter 3.1.3). All other bits in the OP_MODE register are set low.
- Disable the heartbeat signal on the HS.

Function
init (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: which HS/probe (valid range 1 to 4)
Returns

7.2.5 Load calibration data

The probe calibration is programmed via the probes' base configuration register. The probe calibration data is stored in a .csv file.

The following function reads the .csv file containing the calibration data, updates the base configuration register variable in the API. The function does not transmit the base configuration shift register to the memory map of the probe. **The user needs to call the 'wrtieProbeConfiguration' function to effectively write the calibration data to the probe.**

The function also checks if the selected calibration file is for the connected probe, by checking the S/N of the connected probe with the S/N in the calibration csv file.

This function sets the ADC calibration values (compP and compN, Slope, Coarse, Fine and Cfix):

Function
setADCCalibration (unsigned char slotID, signed char port, const char* filename)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
filename: the filename to read from (should be .csv)
Returns

The name of the file is: [probe serial number] ADCCalibration.csv

7.2.6 Load gain correction data

The probe gain correction data is stored in a .csv file. The gain correction parameters are loaded to the BSC FPGA, where they are applied in the system module (ref. chapter 4.3).

The following function reads the gain correction parameters from the specified .csv file and writes these to the BSC FPGA. The function also checks if the selected gain correction file is for the connected probe, by checking the S/N of the connected probe with the S/N in the gain correction csv file.

Function
setGainCalibration (unsigned char slotID, signed char port, const char* filename)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
filename: the filename to read from (should be .csv)
Returns

The name of the file is: [probe serial number]_GainCalValues.csv.

7.2.7 Close BS

The ‘closeBS’ function shuts down the power supply to the ports on the selected BS module. The communication between PC and BS module is terminated.

Function
closeBS (unsigned char slotID)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

7.2.8 Log file

The system can log commands and status information to a text file (api_log_slot_port.txt). This functionality is default disabled, after calling the ‘openProbe’ function.

The log is saved in separate text files per SerDes port and per PXI slot. The log file name contains the SerDes port number and PXI slot number.

The following function enables/disables the logging:

Function
setLog (unsigned char slotID, signed char port, bool enable)

Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
enable: enable (true) or disable (false) the logging
Returns

7.2.9 Set the debug messages level.

The API functions generate status information for the user via ‘fprintf’ functions. The user can set which level of information is included and thus the number of messages which is generated. Macros are defined in NeuropixAPI.h to set the debug messages level:

‘level’ value	Macro	Type of information
0	DBG_PARANOID	Same as 1, plus PCIE register transactions
1	DBG_VERBOSE	Same as 2, plus procedural info on all functions
2	DBG_MESSAGE	Same as 3, plus procedural info on BIST and HST functions
3	DBG_WARNING	Same as 4, plus warnings
4	DBG_ERROR	Errors only

The default value of ‘level’ is set to 2.

Function
void dbg_setlevel (int level)
Arguments
level: which type of information is printed for the user via the API.
Returns
No return value

7.2.10 Get the last error message

The user can call the last generated error message. The function only returns error messages, the type of message is not controlled by the ‘dbg_setlevel’ function.

Function
size_t getLastErrorMessage (char* bufStart, size_t bufsize)
Arguments
bufStart: destination buffer to write the error message to
bufsize: the maximum amount of characters to write to bufStart
Returns
The actual number of characters written to bufStart

7.3 Serial/part numbers, hardware and software versions

7.3.1 Probe

The probe ID, part number, version and revision number are stored on the EEPROM memory on the flex. The probe ID is a 16-digit number. The probe part number is an ASCII string of maximum 40 characters. The flex part number is an ASCII string of maximum 20 characters. Version and revision are each 1-byte integers.

These EEPROM locations are accessed over the UART/I2C interface using the following functions:

Function
readId (unsigned char slotID, signed char port, uint64_t* id)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
id: the probe ID code
Returns

Function
readProbePN (unsigned char slotID, signed char port, char* pn, size_t len)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
pn: the probe part number
len: the number of characters to read
Returns

Function
readFlexPN (unsigned char slotID, signed char port, char* pn, size_t len)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
pn: the flex part number
len: the number of characters to read
Returns

Function
getFlexVersion (unsigned char slotID, signed char port, unsigned char* version_major, unsigned char* version_minor)

Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
version_major: the flex board version number
version_minor: the flex board revision number
Returns

7.3.2 Headstage

The HS serial number, part number, version and revision number are stored on the EEPROM memory on the headstage. The HS S/N is a 16-digit number. The P/N is a 20-character ASCII string; version and revision are each 1-byte integers.

These EEPROM locations are accessed over the UART/I2C interface using the following functions:

Function
readHSSN (unsigned char slotID, signed char port, uint64_t* sn)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which HS port (valid range 1 to 4)
sn: the HS serial number
Returns

Function
readHSPN (unsigned char slotID, signed char port, char* pn, size_t maxlen)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
pn: the HS part number
maxlen: the number of characters to read
Returns

Function
getHSVersion (unsigned char slotID, signed char port, unsigned char* version_major, unsigned char* version_minor)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
version_major: the HS board version number
version_minor: the HS board revision number
Returns

7.3.3 BSC

The BSC serial number, part number, version and revision number are stored on the EEPROM memory on the BSC. The BSC S/N is a 16-digit number. The P/N is a 20-character ASCII string; version and revision are each 1-byte integers.

These EEPROM locations are accessed over the UART/I2C interface using the following functions:

Function
readBSCSN (unsigned char slotID, uint64_t* sn)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
sn: the BSC serial number
Returns

Function
readBSCPN (unsigned char slotID, char* pn, size_t len)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
pn: the BSC part number
len: the number of characters to read
Returns

Function
getBSCVersion (unsigned char slotID, unsigned char* version_major, unsigned char* version_minor)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
version_major: the BSC version number
version_minor: the BSC revision number
Returns

7.3.4 API

The API software version is returned using the following function:

Function
getAPIVersion (unsigned char* version_major, unsigned char* version_minor)
Arguments
version_major: the API version number

version_minor: the API revision number
Returns

7.3.5 BSC FPGA

The BSC FPGA boot code version is hardcoded in the FPGA VHDL code, and is returned using the following function:

Function
getBSCBootVersion (unsigned char slotID, unsigned char* version_major, unsigned char* version_minor, uint16_t* version_build)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
version_major: the BSC FPGA boot code version number
version_minor: the BSC FPGA boot code revision number
version_build: the BSC FPGA boot code build number
Returns

7.3.6 BS FPGA

The BS FPGA boot code version is hardcoded in the FPGA VHDL code, and is returned using the following function:

Function
getBSBootVersion (unsigned char slotID, unsigned char* version_major, unsigned char* version_minor, uint16_t* version_build)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
version_major: the BS FPGA boot code version number
version_minor: the BS FPGA boot code revision number
version_build: the BS FPGA boot code build number
Returns

7.4 System configuration

7.4.1 HS LED

The HS LED heartbeat blinking is enabled/disabled by the GPIO1 signal from the serializer which is controlled through serializer registers.

The default blinking status of the HS LED after calling the ‘openProbe’ function is enabled. The default status after calling the ‘init’ function is disabled.

Function
setHSLed (unsigned char slotID, signed char port, bool enable) const
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
enable: enable (true) or disable (false) HS heartbeat LED
Returns

7.4.2 Temperature

These functions return the BS FPGA and BSC FPGA temperature.

Function
getBSTemperature (unsigned char slotID, float * temperature)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
temperature: the BS FPGA temperature, in degrees Celsius.
Returns

Function
getBSCTemperature (unsigned char slotID, float * temperature)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
temperature: the BSC FPGA temperature, in degrees Celsius.
Returns

7.5 Probe configuration

7.5.1 Operating mode

The following function sets the operating mode of the probe, by configuring the OPMODE register of the probes' memory map.

Function
setOPMODE (unsigned char slotID, signed char port, probe_opmode_t mode)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
mode: the selected probe operation mode.
Returns

The function supports the following modes, use the enum probe_opmode_t:

Parameter mode	Operation mode
RECORDING	Recording
CALIBRATION	Calibration
DIGITAL_TEST	Digital test

- Recording: the default operation mode. The REC bit in the OP_MODE register is set high. All the other bits in the OP_MODE register are set low. The pixels are connected to the channels. The recorded signal is digitized and output on the PSB bus.

- Calibration: The CAL and REC bit in the OP_MODE register are set high. All the other bits in the OP_MODE register are set low. The test signal input (CAL_SIGNAL) is connected to either the pixel, channel or ADC input (selected via the CAL_MOD register). The recorded signal is digitized and output on the PSB bus.

- Digital test: The DIG_TEST and REC bit in the OPMODE register are set high; all other bits in the OP_MODE register are set low. The data transmitted over the PSB bus is a fixed data pattern.

7.5.2 Calibration mode

The CAL_SIGNAL input of the probe, which is connected to the HS test signal via the HS and flex, can be connected to either the probes' pixel inputs, channel inputs, or ADC inputs. The following function configures the CAL_MOD register of the probes' memory map.

Function
setCALMODE (unsigned char slotID, signed char port, testinputmode_t mode)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
mode: the selected test signal mode
Returns

The function supports the following modes, use the enum testinputmode_t:

Parameter mode	Calibration mode
PIXEL_MODE	Pixel
CHANNEL_MODE	Channel
NO_TEST_MODE	Cal_SIGNAL disconnected
ADC_MODE	ADC

- Pixel: The PIX_CAL bit in the CAL_MOD register is set high. All other bits in the CAL_MOD register are set low. The CAL_SIGNAL signal is connected to the pixel inputs.

- Channel: The CH_CAL bit in the CAL_MOD register is set high. All other bits in the CAL_MOD register are set low. The CAL_SIGNAL signal is connected to the channel inputs.

- None: The CAL_SIGNAL signal remains unconnected.

- ADC: The ADC_CAL bit in the CAL_MOD register is set high. All other bits in the CAL_MOD register are set low. The CAL_SIGNAL signal is connected to the ADC inputs.

7.5.3 Electrode selection

The probe base contains 384 channels. On the shank however, 960 electrodes are available. This way, each channel can record from 2 or 3 locations along the shank.

A complete table of channels mapping to electrodes is provided in document [Electrode to channel mapping]. The following table shows an extract of the complete mapping:

Table 3: Electrode to channel mapping.

	Bank number 0	**Bank number 1**	**Bank number 2**
Channel	**Electrode**	**Electrode**	**Electrode**
0	0	384	768
1	1	385	769
…	…	…	…
383	383	767	

The following figures show the grouping of electrodes into banks:

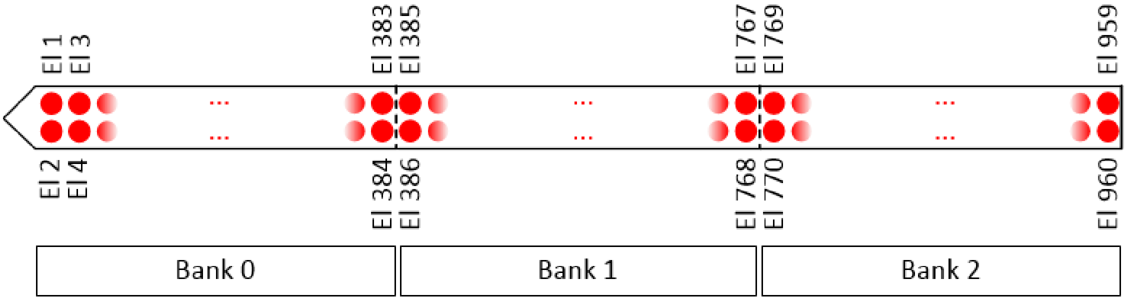

Figure 10: Electrode bank distribution over the shank.

The function below sets which electrode is connected to a channel.

There are 3 electrodes on the shank which can be used as a reference electrode only (191, 575 and 959). This function does not select these electrodes. Refer to chapter 7.5.4.

The function checks whether the input parameters ‘slotID’ and ‘port’ and ‘channel’ are valid and updates the contents of the API variable shank configuration register (ref. [Shank configuration]). It does not transmit the contents of this register to the memory map of the probe.

Function
selectElectrode (unsigned char slotID, signed char port, uint32_t channel, uint8_t electrode_bank)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
channel: the channel number (valid range: 0 to 383, excluding 191)
electrode_bank: the electrode bank number to connect to (valid range: 0 to 2 or 0xFF)
Returns

The valid range for the parameter ‘electrode_bank is limited to 0 to 1 or 0xFF for channel numbers 192 to 383.

From the channel number and electrode bank number can be calculated which electrode is connected to the channel:

Electrode number = (channel number) + 384 * electrode bank number

For example: Channel number = 10 and electrode bank number connection = 1. Channel 10 is connected to electrode 394.

If the parameter ‘electrode_bank’ is set to 0xFF, the channel is disconnected from any electrode. In order to connect only one electrode at a time to a channel, call ‘selectElectrode’ with parameter ‘electrode_bank’ set to 0xFF, prior to calling ‘selectElectrode’ with ‘electrode_bank’ set to the required bank number.

7.5.4 Reference selection

For each channel, the user can select between 3 inputs as a reference input.

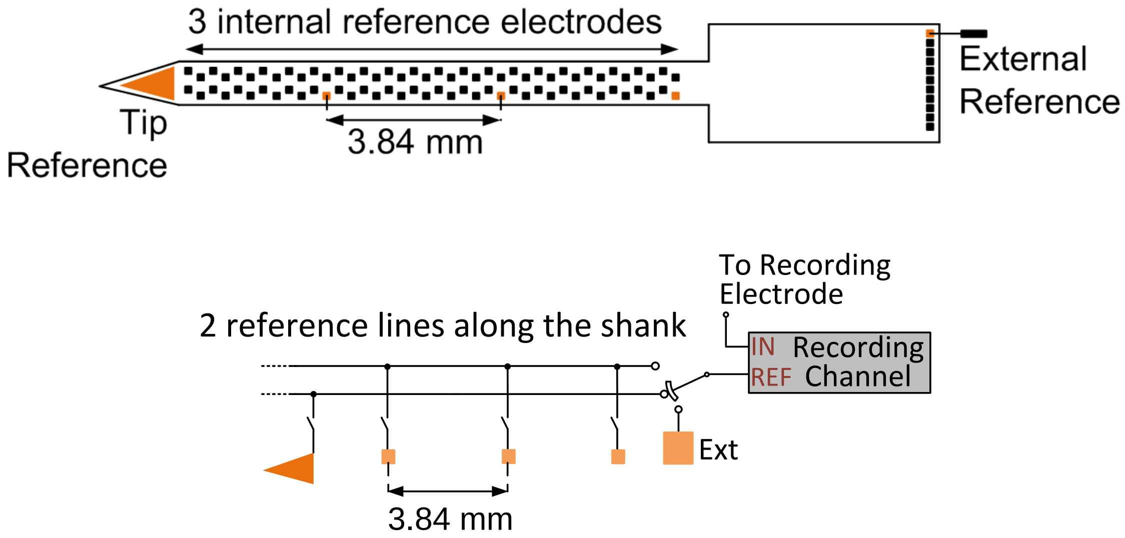

Figure 11: Neural probe functionality: Reference electrode selectivity concept.

These 3 inputs are:

- External reference electrode: an input pin located on the probe flex.
- Tip electrode: a large electrode located at the tip of the shank.
- Internal reference line (which can connect to recording electrodes 191, 575 or 959: 1 in each of the three electrode banks on shank)

The reference selection is made for each channel individually. The reference selection per channel is stored in the channel configuration shift register variable (ref. [Base configuration]). The bit CHxxxExtRef, CHxxxTipRef or CHxxxIntRef is set according to the following table:

‘reference’ value	CHxxxExtRef	CHxxxTipRef	CHxxxIntRef
0	‘1’	‘0’	‘0’
1	‘0’	‘1’	‘0’
2	‘0’	‘0’	‘1’

Leaving all three bits set to ‘0’ would leave the reference input of the channel floating. This status is not allowed.

The reference input selection also affects settings in the shank configuration register. The API takes care that a valid reference configuration is made.

- In case any of the channels uses the external reference electrode, the Ext. Electrode 1 and Ext. Electrode 2 bits in the shank configuration register are set to 1.
- In case none of the channels uses the Ext. Electrode 1 and Ext. Electrode 2 bits in the shank configuration register are set to 0.
- In case any of the channels uses the tip electrode, the Tip Electrode 1 and Tip Electrode 2 bits in the shank configuration register are set to 1.
- In case none of the channels uses the Tip Electrode 1 and Tip Electrode 2 bits in the shank configuration register are set to 0.
- In case any of the channels uses the internal reference line, one of the three reference electrodes in the shank configuration register are set to 1. Which of the three electrodes is defined by the ‘intRefElectrodeBank’ parameter. Only one out of 3 internal reference electrodes can be activated.

  The function input parameter ‘intRefElectrodeBank’ sets the selected reference. The following table shows the settings of specfic bits in the shank configuration register:

‘intRefElectrodeBank’ value	Selected electrode	Electrode 191	Electrode 575	Electrode 959
0	192	‘1’	‘0’	‘0’
1	576	‘0’	‘1’	‘0’
2	960	‘0’	‘0’	‘1’

- In case none of the channels uses the internal reference line, all of the three reference electrode bits in the shank configuration register are set to 0: Electrode 191, Electrode 575, Electrode 959. The value of the input parameter 'intRefElectrodeBank' is ignored in this case.

The function checks whether the input parameters 'slotID' and 'port' are valid and updates the contents of the API base configuration register variable (ref. [Base configuration]), and if necessary of the API shank configuration register variable (ref. [Shank configuration]). It does not transmit the shank and base configuration shift registers to the memory map of the probe.

Function
setReference (unsigned char slotID, signed char port, unsigned int channel, channelreference_t reference, uint8_t intRefElectrodeBank)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
channel: the channel number (valid range: 0 to 383)
reference: the selected reference input (valid range: 0 to 2)
intRefElectrodeBank: the selected internal reference electrode (valid range: 0 to 2)
Returns

Use the enum channelreference_t to select the reference input:

channelreference_t
EXT_REF
TIP_REF
INT_REF

7.5.5 Gain

The user can set the gain of each channel individually. The gain of the AP and LFP band can be set separately. AP and LFP gain can be selected from 8 predefined values.

The gain setting of the probe is programmed via the probes' base configuration register (ref. [Base configuration]). The function sets the CHxxx_APy and CHxxx_LFPy bits in the base configuration register, according to the following table:

parameter 'ap_gain'	**CHxxxG_AP0**	**CHxxxG_AP1**	**CHxxxG_AP2**	**Gain**
0	0	0	0	50
1	1	0	0	125
2	0	1	0	250
3	1	1	0	500
4	0	0	1	1000
5	1	0	1	1500
6	0	1	1	2000

7	1	1	1	3000

parameter ‘lfp_gain’	CHxxxG_LFP0	CHxxxG_LFP1	CHxxxG_LFP2	Gain
0	0	0	0	50
1	1	0	0	125
2	0	1	0	250
3	1	1	0	500
4	0	0	1	1000
5	1	0	1	1500
6	0	1	1	2000
7	1	1	1	3000

The function also updates the gain correction factors as applied in the BSC FPGA to the neural data stream (ref. chapter 4.3).

The function checks whether the input parameters ‘slotID’ and ‘port’ are valid and updates the base configuration register variable in the API. The function does not transmit the base configuration shift register to the memory map of the probe.

Function
setGain (unsigned char slotID, signed char port, unsigned int channel, unsigned char ap_gain, unsigned char lfp_gain)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
channel: the channel number (valid range: 0 to 383)
ap_gain: the AP gain value (valid range: 0 to 7)
lfp_gain: the LFP gain value (valid range: 0 to 7)
Returns

7.5.6 Bandwidth

The high-pass corner frequency can be programmed for each AP channel individually.

The high-pass corner frequency setting of the probe is programmed via the probes’ base configuration register (ref. [Base configuration]). The function sets the CHxxxFULL bit in the base configuration register.

The function checks whether the input parameters ‘slotID’ and ‘port’ are valid and updates the base configuration register variable in the API. The function does not transmit the base configuration shift register to the memory map of the probe.

Function

setAPCornerFrequency (unsigned char slotID, signed char port, unsigned int channel, bool disableHighPass)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
channel: the channel number (valid range: 0 to 383)
disableHighPass: enables/disables the highpass filter of the AP channel
Returns

bool disableHighPass	
True	300 Hz highpass cut-off filter disabled
False	300 Hz highpass cut-off filter enabled

7.5.7 Standby

The user can set each channel individually in stand-by mode. In standby mode the channel amplifiers are disabled. The channel is still output on the PSB data bus.

The standby mode setting of the probe is programmed via the probes' base configuration register (ref. [Base configuration]). The function sets the CHxxxSTDB bit in the base configuration register. If the function input parameter 'standby' is set to '1', the CHxxxSTDB bit is set to 1, and vice-versa.

The function checks whether the input parameters 'slotID' and 'port' are valid and updates the base configuration register variable in the API. The function does not transmit the base configuration shift register to the memory map of the probe.

Function
setStdb (unsigned char slotID, signed char port, unsigned int channel, bool standby)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
channel: the channel number (valid range: 0 to 383)
standby: the standby value to write
Returns

bool standby	
True	Channel in standby
False	Channel active

7.5.8 Writing probe configuration settings to the probe

The probes' shank and base configuration register cannot be written partially, but always need to be set in their entirety. Therefore, to keep the probe configuration time to a minimum, the probe configuration functions as described above do not transmit the new shank and base configuration to the probe. Instead, by calling the functions above the user first sets the shank and base configuration settings in the API variables and subsequently calls the function described hereunder to transmit shank and base configuration settings to the shift registers of the probe.

Function
writeProbeConfiguration (unsigned char slotID, signed char port, bool readCheck)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
readCheck : if enabled, write the configuration registers twice and check the SR_OUT_OK bit.
Returns

The probe has the functionality to compare stored data in the shift register with freshly written data (base configuration registers only). In order to do so the API function needs to write twice the same complete shift configuration register data to the probe. If the data written during the second write cycle is the same as the data shifted out, the data was written correctly, and the probe sets the SR_OUT_OK bit in the STATUS register in the memory map.

If the parameter 'readCheck' is set, the API function writes the shift register twice, and checks the SR_OUT_OK bit. If the parameter 'readCheck' is not set, the API function writes the shift register only once, and ignores the SR_OUT_OK bit.

The user needs to call this function after a change has been made to the electrode selection, reference selection, gain setting, bandwidth setting or standby setting, in order to apply the change to the probe.

7.6 Trigger configuration

7.6.1 'arm' mode

In anticipation of receiving a start trigger, the system is set in 'arm' mode. In 'arm' mode, neural data packets from the probes are not transmitted to the PC memory, and the timestamp is fixed at 0. Upon receiving the start trigger, the system starts the timestamp generator and starts to transmit neural data to the PC.

After the system has received a start trigger and is buffering incoming neural data, calling the API 'arm' function again stops the buffering of neural data packets, flushes all remaining datapackets in the datapath, stops the timestamp generator and resets the timestamp to 0.

Function
arm (unsigned char slotID)

Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

7.6.2 Trigger matrix configuration

The start trigger starts the buffering of neural data on the BSC FPGA and starts the timestamp generator, on the condition that the system was set to ‘arm’ mode prior to receiving the start trigger. The system can select between different trigger input sources and can also generate a trigger signal on different lines.

The user can select from several possible sources to generate a start trigger event:

- A software command
- The start trigger SMA connector on the CR board.
- The PXI_Trigger<0:6> bus.
- A user defined trigger. The BSC FPGA holds a defined space where the user can implement a trigger algorithm.

7.6.2.1 Function setTriggerInput/setTriggerOutput

Use the enum triggerInputline_t to select the trigger input signal:

Input parameter ‘input’	**Selected trigger source**
TRIGIN_SW	Software trigger
TRIGIN_SMA	CR SMA connector
TRIGIN_PXI0	PXI_TRIGGER<0>
TRIGIN_PXI1	PXI_TRIGGER<1>
TRIGIN_PXI2	PXI_TRIGGER<2>
TRIGIN_PXI3	PXI_TRIGGER<3>
TRIGIN_PXI4	PXI_TRIGGER<4>
TRIGIN_PXI5	PXI_TRIGGER<5>
TRIGIN_PXI6	PXI_TRIGGER<6>
TRIGIN_SHAREDSYNC	*SYNC clock shared among PXIe acquisition modules: for SYNC configuration only*
TRIGIN_SYNCCLOCK	*Internal SYNC clock: for SYNC configuration only*
TRIGIN_USER0	User implemented
TRIGIN_USER1	User implemented
TRIGIN_USER2	User implemented
TRIGIN_USER3	User implemented
TRIGIN_USER4	User implemented
TRIGIN_USER5	User implemented
TRIGIN_USER6	User implemented

TRIGIN_USER7	User implemented
TRIGIN_USER8	User implemented
TRIGIN_USER9	User implemented

Function
setTriggerInput (unsigned char slotID, triggerInputline_t input)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
input: the trigger source signal (range 0 to 20)
Returns

The system can also generate a trigger pulse. The output line to which trigger is sent is configurable. Only one signal line can be selected to be used as output to generate a trigger pulse. The signal line which triggers the generation of an output trigger pulse is also configurable. Only one signal line can be selected to be used as input to generate a trigger pulse.

Use the enum triggerOutputline_t to select the trigger output signal:

Input parameter 'output'	**Selected trigger source**
TRIGOUT_NONE	None
TRIGOUT_SMA	CR SMA connector
TRIGOUT_PXI0	PXI_TRIGGER<0>
TRIGOUT_PXI1	PXI_TRIGGER<1>
TRIGOUT_PXI2	PXI_TRIGGER<2>
TRIGOUT_PXI3	PXI_TRIGGER<3>
TRIGOUT_PXI4	PXI_TRIGGER<4>
TRIGOUT_PXI5	PXI_TRIGGER<5>
TRIGOUT_PXI6	PXI_TRIGGER<6>

Function
setTriggerOutput (unsigned char slotID, triggerOutputline_t output, triggerInputline_t inputline)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
output: the signal line over which the trigger pulse is transmitted (range 0 to 8)
inputline: the signal which triggers generating the output pulse (range 0 to 20)
Returns

7.6.2.1 Function set setTriggerBinding/getTriggerBinding

An alternative and more intuitive way to set the trigger matrix with a function that uses a mask as input parameter. The enum signalline_t represents the inputs and output of the triggermatrix.

Enum signalline_t	Description
SIGNALLINE_NONE	Used for disconnecting all trigger inputs/outputs
SIGNALLINE _PXI0	PXI_TRIGGER<0>
SIGNALLINE _PXI1	PXI_TRIGGER<1>
SIGNALLINE _PXI2	PXI_TRIGGER<2>
SIGNALLINE _PXI3	PXI_TRIGGER<3>
SIGNALLINE _PXI4	PXI_TRIGGER<4>
SIGNALLINE _PXI5	PXI_TRIGGER<5>
SIGNALLINE _PXI6	PXI_TRIGGER<6>
SIGNALLINE _SHAREDSYNC	PXI_TRIGGER<7> used exclusively to share the SYNC clock among Neuropixels modules
SIGNALLINE _LOCALTRIGGER	Signal to the BSC where the neural data acquisition is triggered
SIGNALLINE _LOCALSYNC	Signal to the BSC which samples the SYNC clock and adds to the neural data packets
SIGNALLINE _SMA	SMA connector on the carrier
SIGNALLINE _SW	Trigger signal generated by software
SIGNALLINE_LOCALSYNCCLOCK	The SYNC clock generator on the module

This corresponds with the signals on the triggermatrix:

parameter 'output'		SIGNALLINE _PXI0	SIGNALLINE _PXI1	SIGNALLINE _PXI2	SIGNALLINE _PXI3	SIGNALLINE _PXI4	SIGNALLINE _PXI5	SIGNALLINE _PXI6	SIGNALLINE _SHAREDSYNC	*SIGNALLINE _LOCALTRIGGER*	*SIGNALLINE _LOCALSYNC*	SIGNALLINE _SMA	SIGNALLINE _SW	SIGNALLINE_LOCALSYNCCLOCK
	parameter 'inputs'	From PXI backplane								NA	NA	From SMA CR	From software	From local SYNC clock
To PXI backplane	SIGNALLINE _PXI0	▨							▨	▨	▨			▨
	SIGNALLINE _PXI1		▨						▨	▨	▨			▨
	SIGNALLINE _PXI2			▨					▨	▨	▨			▨
	SIGNALLINE _PXI3				▨				▨	▨	▨			▨
	SIGNALLINE _PXI4					▨			▨	▨	▨			▨
	SIGNALLINE _PXI5						▨		▨	▨	▨			▨
	SIGNALLINE _PXI6							▨	▨	▨	▨			▨
	SIGNALLINE _SHAREDSYNC	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		▨	
To BSC	SIGNALLINE _LOCALTRIGGER								▨	▨	▨			▨
	SIGNALLINE _LOCALSYNC	▨	▨	▨	▨	▨	▨	▨		▨	▨		▨	
To SMA CR	SIGNALLINE _SMA									▨	▨	▨		
NA	*SIGNALLINE _SW*	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
NA	*SIGNALLINE_LOCALSYNCCLOCK*	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

▨ = connection not allowed

Function	
setTriggerBinding (unsigned char slotID, signalline_t outputline, signalline_t inputlines)	
Arguments	
slotID: which slot in the PXI chassis (valid range depends on the chassis)	
outputline: which line to be connected to the ‘inputlines’	
inputlines: 1 or more inputlines (can be ORed) which can source a trigger	
Returns	

A function is available to read out which inputlines are connected to an outputline:

Function	
getTriggerBinding (unsigned char slotID, signalline_t outputline, signalline_t* inputlines)	
Arguments	
slotID: which slot in the PXI chassis (valid range depends on the chassis)	
outputline: which outputline to check	
inputlines: the inputlines that are connected to the outputline.	
Returns	

7.6.3 Trigger edge

The functionality of this function depends on the setting made with the function ‘setTriggerInput’/‘setTriggerOutput’ or ‘setTriggerBinding’:

- In case the start trigger is defined as any of the hardware inputs, the trigger edge sensitivity can be configured to falling or rising edge with the function below. In case the rising edge is selected, a rising edge on the input signal will pulse the start trigger flag in the data packet.

- If the user has selected a start trigger source any other than the BS SMA start trigger, the SMA start trigger connected on the BS becomes an output port. For example, when a software start trigger is transmitted through the API, the BS generates a 1 ms pulse on the BS SMA connector when the start trigger is actually received. The function below now sets the pulse polarity of this output pulse:

    - Rising edge: the trigger output is idle low. The trigger pulse is a positive going pulse of 1 ms width.

    - Falling edge: the trigger output is idle high. The trigger pulse is a negative going pulse of 1 ms width.

Function
setTriggerEdge (unsigned char slotID, bool rising)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
rising: the trigger edge (false = falling, true = rising)
Returns

7.6.4 Software start trigger

The following function generates a software start trigger which is transmitted to the BS module.

Function
setSWTrigger (unsigned char slotID)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

7.7 Data acquisition

7.7.1 *Streaming to file*

The API can stream the neural data to files on the disc drive. The following function specifies the filename of the binary file:

Function
setFileStream (unsigned char slotID, const char* filename)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
filename: to which file the data will be saved
Returns

The following function enables/disables the streaming to file:

Function
enableFileStream (unsigned char slotID, bool enable)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
enable: enable/disable the saving of neural data to file
Returns

7.7.2 *Reading from RAM FIFO*

The data is transferred from the PCIe interface to the PC RAM memory. The user can read data directly from the RAM, for example for visualization purposes. This is a non-blocking read.

Separate functions read AP or LFP data. These functions fill a buffer with data interleaved for 384 channels. The number of samples is specified by the user.

Function
readAPFifo (uint8_t slotid, uint8_t port, uint32_t* timestamps, int16_t* data, int samplecount)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
timestamps: optional timestamps buffer (NULL if not used), size should be ‘samplecount’
data: buffer of size samplecount*384
samplecount: the number of data packets to fetch
Returns
the number of samples received

Function
readLFPFifo (uint8_t slotID, uint8_t port, uint32_t* timestamps, int16_t* data, int samplecount)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4).
timestamps: optional timestamps buffer (NULL if not used), size should be ‘samplecount’
data: buffer of size samplecount*384
samplecount: the number of data packets to fetch
Returns
the number of samples received

The data can also be read out in electrodePacket format.

Function
readElectrodeData (uint8_t slotID, uint8_t port, struct electrodePacket* packets, size_t* actualAmount, size_t requestedAmount)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4).
packets: packets of data in format electrodePacket
actualAmount: the amount of data packets received
requestedAmount: the requested amount of data packets
Returns

The electrodePacket structure format is defined in NeuropixAPI.h. Each electrodePacket contains:

- uint32_t timestamp[12]: 12 samples of the 100 kHz timestamp clock.
- int16_t apData[12][384]: 12 samples for each of the 384 AP channels.
- int16_t lfpData[384]: 1 sample for each of the 384 LFP samples.
- uint16_t Trigger[12]: 12 samples of the STATUS byte as described in Table 2. Trigger signal is on bit 0, SYNC clock is on bit 6.
- uint16_t SYNC[12]: not used

The filling percentage of the RAM FIFO for electrode data packets can be checked using the following function:

Function
getElectrodeDataFifoState (uint8_t slotID, uint8_t port, size_t* packetsavailable, size_t* headroom)
Arguments
filename: specifies an existing file with probe acquisition data
port: for which BS port (valid range 1 to 4)
packetsAvailable: returns the amount unread of packets in the FIFO. NULL allowed for no return
headroom: returns the amount of space left in the FIFO. NULL allowed for no return

Returns

7.7.3 Reading binary files

The probe data which is saved in the binary files by the API during data acquisition can be extracted using a set of functions:

The following function opens an acquisition stream from an existing binary file:

Function
streamOpenFile (const char* filename, int8_t port, bool lfp, np_streamhandle_t* pstream)
Arguments
filename: specifies an existing file with probe acquisition data
port: for which BS port (valid range 1 to 4)
lfp: specifies whether AP or LFP data will be extracted (true = LFP)
pstream: pointer to the stream handle that will receive the handle to the opened stream
Returns

The following function moves the stream pointer to a given position:

Function
streamSeek (np_streamhandle_t stream, uint64_t filepos, uint32_t* actualtimestamp)
Arguments
stream: the stream handle
filepos: the file position to navigate to
actualtimestamp: returns the timestamp at the stream pointer (NULL allowed)
Returns

The following function sets the stream pointer to a given position:

Function
streamSetPos (np_streamhandle_t sh, uint64_t filepos)
Arguments
stream: the stream handle
filepos: the file position to navigate to
Returns

The following function gets the current file position of the stream pointer:

Function
streamTell (np_streamhandle_t stream)

Arguments
stream: the stream handle
Returns
the current file position at the stream cursor position

The following function reads a specified number of samples from the file:

Function
streamRead (np_streamhandle_t sh, uint32_t* timestamps, int16_t* data, int samplecount)
Arguments
stream: the stream handle
timestamps: optional timestamps buffer (NULL if not used), size should be ‘samplecount’
data: buffer of size ‘samplecount’*384
samplecount: the requested number of samples
Returns
the number of samples received

The following function closes the acquisition stream from an existing binary file:

Function
streamClose (np_streamhandle_t stream)
Arguments
stream: the stream handle to close
Returns

7.8 BISTS

There are several self-test functionalities implemented in the probe and system. These tests can be enabled from the API.

7.8.1 Basestation platform interconnection test

The function checks the connectivity between PC and BS FPGA board over the PCIe interface. The following subtests are performed:

Function
bistBS (unsigned char slotID)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
Returns

7.8.2 Heartbeat

The heartbeat signal generated by the PSB_SYNC signal of the probe is routed via the SerDes to the BSC FPGA. The BSC analyzes the presence of a 1 Hz clock signal.

The presence of a heartbeat signal acknowledges the functionality of the power supplies on the HS for serializer and probe, the POR signal, the presence of the master clock signal on the probe, the functionality of the clock divider on the probe, and basic communication over the SerDes link.

Function
bistHB (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.8.3 SerDes link

The functionality of the SerDes link can be quantitatively verified using the integrated PRBS test of the SerDes component pair.

The following functions allow the user to start the PRBS test and read back the results:

Function
bistStartPRBS (unsigned char slotID, signed char port) const
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

Function
bistStopPRBS (unsigned char slotID, signed char port, unsigned char* prbs_err)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
prbs_err: the number of prbs errors
Returns

7.8.4 I2C/memory map

This test verifies the functionality of the I2C interface and the probes' memory map. The function reads a register on a memory map address, writes back a modified value, reads this again, and writes again the original value to restore the probe to its settings before test.

The test is successful if the modified value can also be read back, and if a ‘Ack’ byte has been received for each R/W operation.

Function
bistI2CMM (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.8.5 EEPROM

The test verifies the correct access to the EEPROM on flex, HS and BSC. The function writes data to an unused location on the EEPROM and reads back.

The test is successful if the data can be read back, and if a ‘Ack’ byte has been received for each R/W operation.

The function tests the flex, HS and BSC EEPROM consecutively. The function returns an error after the first failed test.

Function
bistEEPROM (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.8.6 Shank and base configuration shift registers

This test verifies the functionality of the shank and base shift registers (SR_CHAIN 1 to 3). The function configures the shift register two times with the same code. After the 2nd write cycle the SR_OUT_OK bit in the STATUS register is read. If OK, the shift register configuration was successful. The test is done for all 3 registers. The configuration code used for the test is a dedicated code (to make sure the bits are not all 0 or 1).

Function
bistSR (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.8.7 PSB bus

A test mode is implemented on the probe which enables the transmission of a known data pattern over the PSB bus. The following function sets the probe in this test mode, records a small data set, and verifies whether the acquired data matches the known data pattern.

This way the PSB bus on the probe and all data lines between probe and serializer are verified.

Function
bistPSB (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.8.8 Pixel signal path

The probe is configured for recording of a test signal which is generated on the HS. This configuration is done via the shank and base configuration registers and the memory map. The AP data signal is recorded, and the frequency and amplitude of the recorded signal are extracted for each electrode. If at least 90% of the electrodes show a signal within the frequency and amplitude tolerance, the function returns a pass value.

Do not load the ADC calibration parameters to the probe prior to calling the bistSignal function. The tolerances on the amplitude test are set for an uncalibrated probe.

The function requires more than 30 seconds to complete**.**

Function
bistSignal (unsigned char slotID, signed char port, bool* pass, float* measuredAmp, float* measuredFreq)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
pass: true if measurement results passed the criteria
measuredAmp: measured amplitude per electrode
measuredFreq: measured frequency per electrode
Returns

7.8.9 Probe noise level

The probe is configured for noise analysis. Via the base configuration registers and the memory map, the channel inputs are shorted to ground. The data signal is recorded, and the noise level is calculated. The function analyses if the probe performance falls inside a specified tolerance range (go/no-go test).

The probe passes the noise test if less than 20% of the electrodes have their AP noise value out of spec, and if less than 20% of the electrodes have their LFP noise value out of spec.

The API function returns a pass/fail value.

Do not load the ADC calibration parameters to the probe prior to calling the bistNoise function. The tolerances on the noise test are set for an uncalibrated probe.

Function
bistNoise (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.9 HST module functions

The functionality of the HS can be verified with the Headstage Test module. Several tests are implemented on a microcontroller on the HST module and these are activated by calling API functions.

7.9.1 Open HSTest

The ‘openProbeHSTest’ function initializes the SerDes link and HS, similar as the ‘openProbe’ function, but specific for HST module functionality. This function needs to be called once before accessing the HST module API functions.

Function
openProbeHSTest (unsigned char slotID, signed char port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns

7.9.2 Supply test

The function ‘HSTestVDDA1V2’ checks if both pins on the ZIF connector contain an analog 1.2 V supply signal within the required tolerances. If the test fails, the functions returns ‘BISTERROR’, and prints a message with more details on the failure.

Function
HSTestVDDA1V2 (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

The function ‘HSTestVDDA1V8’ checks if both pins on the ZIF connector contain an analog 1.8 V supply signal within the required tolerances. If the test fails, the functions returns ‘BISTERROR’, and prints a message with more details on the failure.

Function
HSTestVDDA1V8 (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

The function ‘HSTestVDDD1V2’ checks if both pins on the ZIF connector contain a digital 1.2 V supply signal within the required tolerances. If the test fails, the functions returns ‘BISTERROR’, and prints a message with more details on the failure.

Function
HSTestVDDD1V2 (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

The function ‘HSTestVDDD1V8’ checks if both pins on the ZIF connector contain a digital 1.8 V supply signal within the required tolerances. If the test fails, the functions returns ‘BISTERROR’, and prints a message with more details on the failure.

Function
HSTestVDDD1V8 (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)

Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.3 MCLK

The HST module can verify whether the MCLK clock signal generated on the HS is present. The function returns ‘BISTERROR’ if the MCLK signal is not detected.

Function
HSTestMCLK (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.4 PSB clock

The HST module can check the functionality of the PSB clock.

Note: The PCLK signal is required by the serializer for communication with the HS. Under normal operations this signal is generated by the probe. If the HS is connected to the HST module, the microcontroller on the HST module provides the PCLK to the serializer. So, if the signal connection for PCLK on the HS between the ZIF connector and the serializer would fail, no communication with the HST module would be possible, since all communication from API to the HST module runs via the HS over the SerDes link. Therefore, this API function is redundant.

Function
HSTestPCLK (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.5 PSB bus

The HST module can check the connectivity of the PSB_DATA<0:6> and PSB_SYNC signal on between the ZIF connector and the serializer on the HS. The function first checks PSB_SYNC and continues with PSB_DATA<0> to PSB_DATA<6> consecutively. The function returns at the first failing signal, without checking the remaining signals.

The function returns ‘BISTERROR’ if one of the 8 PSB signals is not functioning correctly. The function prints a message indicating which signal has failed.

Note: an electrical short-circuit on the input pins of the serializer cannot be detected with this the current version of the HST module.

Function
HSTestPSB (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.6 I2C bus

The HST module can check the functionality of the I2C bus.

Note: The I2C bus is used by all other HST module API functions to communicate with the microcontroller in the API for activating test sequences and reading back the result. Therefore, this API function is redundant.

Function
HSTestI2C (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.7 NRST

The HST module can check the functionality of the NRST signal between the ZIF connector and serializer on the HS.

The function returns ‘BISTERROR’ if the NRST signal is not functioning correctly.

Function
HSTestNRST (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
‘SUCCESS’ if the test passes, ‘BISTERROR’ if the test doesn’t pass the criteria

7.9.8 REC_NRESET

The HST module can check the functionality of the REC_NRESET signal between the ZIF connector and serializer on the HS.

The function returns 'BISTERROR' if the REC_NRESET signal is not functioning correctly.

Function
HSTestREC_NRESET (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
'SUCCESS' if the test passes, 'BISTERROR' if the test doesn't pass the criteria

7.9.9 Test signal generator

The HST module can test the functionality of the test signal generator which is located on the HS and which is primarily used for the BIST pixel signal path (API function 'bistSignal').

The test involves two signals: the output signal coming from the HS: CAL_SIGNAL and the input signal going to the HS which is used to enable or disable the test signal generator, and which is normally coming from the probe: OSC_PDOWN.

The function returns 'BISTERROR' if the OSC_PDOWN signal is not functioning correctly or if the CAL_SIGNAL signal is not functioning correctly.

Function
HSTestOscillator (uint8_t slotID, int8_t port)
Arguments
slotID: which slot in the PXI chassis (valid range depends on the chassis)
port: for which BS port (valid range 1 to 4)
Returns
'SUCCESS' if the test passes, 'BISTERROR' if the test doesn't pass the criteria

7.10 NP_ErrorCode enumerator

All API functions return a value from the following table.

NP_ErrorCode	Value	Description
SUCCESS	0	Successful operation
FAILED	1	Unspecified failure

ALREADY_OPEN	2	Connection to BS module is already open
NOT_OPEN	3	Function cannot execute because connection to selected BS module or port is not open
IIC_ERROR	4	Error while accessing devices on the BS I2C bus
VERSION_MISMATCH	5	FPGA firmware and software versions don't match
PARAMETER_INVALID	6	One of the function parameters is invalid
UART_ACK_ERROR	7	UART communication on the SerDes link failed to receive an acknowledgement
TIMEOUT	8	The function did not complete within a restricted period of time
WRONG_CHANNEL	9	Illegal channel number
WRONG_BANK	10	Illegal electrode bank number
WRONG_REF	11	Reference number outside the valid range was specified
WRONG_INTREF	12	Internal reference number outside the valid range was specified
CSV_READ_ERROR	13	Parsing error occurred while reading a malformed CSV file
BIST_ERROR	14	Error in BIST
FILE_OPEN_ERROR	15	The file could not be opened
READBACK_ERROR	16	BIST readback verification failed
READBACK_ERROR_FLEX	17	BIST Flex EEPROM readback verification failed
READBACK_ERROR_HS	18	BIST HS EEPROM readback verification failed
READBACK_ERROR_BSC	19	BIST BSC EEPROM readback verification failed
TIMESTAMPNOTFOUND	20	The specified timestamp could not be found in the stream
FILE_IO_ERR	21	A file IO operation failed
OUTOFMEMORY	22	The operation could not complete due to insufficient process memory
LINK_IO_ERROR	23	SerDes link IO error
NO_LOCK	24	Missing serializer clock
WRONG_AP	25	AP gain number out of range
WRONG_LFP	26	LFP gain number out of range
ERROR_SR_CHAIN_1	27	Validation of SRChain1 data upload failed
ERROR_SR_CHAIN_2	28	Validation of SRChain2 data upload failed
ERROR_SR_CHAIN_3	29	Validation of SRChain3 data upload failed
PCIE_IO_ERROR	30	PCIe data stream IO error occurred
NO_SLOT	31	No Neuropixels BS module found at the specified slot number
WRONG_SLOT	32	Specified slot is out of bound
WRONG_PORT	33	Specified port is out of bound
STREAM_EOF	34	End of file reached

HDRERR_MAGIC	35	Error in MagicID in data packet header
HDRERR_CRC	36	Error in CRC in data packet header
WRONG_PROBESN	37	S/N in csv file does not correspond with connected probe S/N
WRONG_TRIGGERLINE	38	Illegal trigger line connection
PROGRAMMINGABORTED	39	Flash programming was aborted
VALUE_INVALID	40	The parameter value is invalid
NOTSUPPORTED	0xFE	Functionality is not supported
NOTIMPLEMENTED	0xFF	Functionality is not implemented

8 Known bugs and issues

8.1 Hardware

Bug/issue	Action

8.2 BSC FPGA

Bug/issue	Action

8.3 Enclustra FPGA module

Bug/issue	Action

8.4 API

Bug/issue	Action
API functions are not part of their own namespace	Add C++ wrapper around C API