LIBERA Brilliance+

Electron Beam Position Processor

User manual

Revision history

Date/version	Author	Description
July 2010, 1.0	Agata Slivka	Original
June 2011, 1.2	Peter Leban	Major update, new functionalities
July 2011, 1.21	Peter Leban	Minor updates and corrections
November 2012, 2.60	Peter Leban	EvRx description, DSC updates
January 2013, 2.61	Peter Leban	Minor corrections and additions
December 2013, 2.80	Peter Leban	Updated DSC chapter, details about EvRx usage, Single Pass, Statistics functionalities
August 2014, 2.81	Peter Leban	Added note about various configuration options
August 2016, 3.0	Peter Leban	Major update, chapter renumbering
January 2019, 3.1	Peter Leban	New ICB module description, DSC data source selection, SA history
April 2020, 3.2	Peter Leban	Complete review; User manual now for the new BPM module only.
March 2021, 3.3	Peter Leban	Added description of the new spike removal algorithm.
February 2022, 3.4	Peter Leban	Updated the FAT
June 2022, 3.5	Peter Leban	Added FDS and Libera XBS FE descriptions
April 2025, 3.6	Peter Leban	Updates for software release 3.16

Edition
First edition, July 2010
Printed in Slovenia

Instrumentation Technologies, Velika Pot 22
SI-5250 Solkan, Slovenia

Assistance
You can rely on our Technical support. Our core team consists of skilled engineers with full knowledge of the systems. We will help you with hardware, software, or system integration issues throughout the product's life cycle.

Contact us
E-mail: support@i-tech.si
Phone: +386 5 335 2600
Fax: +386 5 335 2601

Technologies licenses
The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Document dependencies

Table of contents

Index of figures

Index of tables

1. Introduction

This document contains description of all available functionalities in Libera Brilliance+ instrument. There are several configuration options (hardware and software) available. Some configuration options do not include all described functionalities.
Technical specifications refer to latest hardware modules.

1.1 Product's evolution

Libera Brilliance+ has been upgraded through its product lifecycle with software and hardware upgrades. They are listed in Table 1.

Table 1: Upgrades of Libera Brilliance+.

Part	Year	Description
	2010	New instrument
Timing module	2012	Event-receiver module replaces the TIM module. Software and hardware backward compatibility maintained.
OS	2016	Operating system upgraded to Ubuntu 14.04 32-bit LTS (supported until 2019)
ICB module	2016	Upgraded ICB module to support new CPU module (not backward compatible)
CPU module	2016	Upgraded CPU from Intel Atom N270 to Intel i5 3230M
ICB module	2018	Upgraded ICB module (major redesign, microSD card support)
OS	2020	Operating system upgraded to Ubuntu 18.04 64-bit LTS (supported until 2023)
CPU module	2020	Upgrade CPU from Intel i5 3230M to Intel i5 7440EQ
BPM module	2020	Upgraded FPGA from Virtex-5 to Kintex Ultrascale+, redesigned analog front-end. Software and hardware backward compatibility maintained.
BPM module	2022	Hardware and software support for the Libera XBS FE
ICB module	2025	New ICB module version with 10 GbE and new CPU support CPU module: i5-13600H
Timing module	2025	Upgraded event-receiver module. Uses Kintex Artix FPGA.

1.2 Capabilities and features overview

Libera Brilliance+ is a BPM signal processing system developed especially for the needs of the electron synchrotron accelerators. It can be used for position measurement of the electron beam both in the booster and in the storage ring.

Libera Brilliance+ can contain up to four electron beam position processor modules, thus enabling the measurement of the electron beam position in up to four points in the synchrotron tube. With such modular design the number of instrumentation boxes is significantly reduced. The compactness of the Libera Brilliance+, however, does not influence its performance at all.

Libera Brilliance+ features accurate electron beam position measurements at various data rates and bandwidths simultaneously, excellent beam current dependence characteristics, sub-micron position measurement resolution and low crosstalk between channels. It is built on proven technology and with extensive experience and knowledge gained from its predecessor, Libera Brilliance and Libera Electron.

Libera Brilliance+ is optimized to work with input signals from button pick-ups. The signal from the pick-ups is processed in the signal processing chain, which is composed of analog signal processing, digitalization on fast ADCs and digital signal processing. Four data paths at different sampling rates with different bandwidth and resolution are available to the user. Acquisitions can be done simultaneously on all four data paths (ADC raw data, turn-by-turn data, fast acquisition, and slow acquisition data).

User can access functions implemented in Libera Brilliance+ unit through a control system interface, named measurement and control interface (MCI). This interface is developed to be easily integrated into the accelerator's control system software.

1.3 List of supplied items

The package contains the following items:

- Libera Brilliance+
- Power cord (2 m)
- Mounting screws set (screws, plastic washers, nuts)
- Ethernet cable RJ45, 3 m
- Set of spare fan filters, 4 pcs
- Warranty and Test record
- USB stick with user documentation and software release

The minimum Libera Brilliance+ configuration consists of 1 ICB module, 1 timing module and 1 BPM module. Libera Brilliance+ can host up to 4 BPM modules.

NOTE: Before removing, adding, or exchanging any of the modules or other components, consult with Instrumentation Technologies support team.

1.4 Abbreviations

Table 2: Abbreviations

Abbreviation	Description
ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
BMC	Baseboard Management Controller
BPM	Beam Position Monitor
CPU	Central Processing Unit
COM	Computer on Module
DDC	Digital Down Conversion
DHCP	Dynamic Host Configuration Protocol
DSC	Digital Signal Conditioning
DVI	Digital Visual Interface
EvRx	timing module (event receiver)
FA	Fast Acquisition data
FIR	Finite Impulse Response filter
FPGA	Field Programmable Gate Array
GDX	Gigabit Data Exchange module
HW	Hardware
ICB	Inter Connection Board
IIR	Infinite Impulse Response filter

Abbreviation	Description
IP	Internet Protocol address
IPMI	Intelligent Platform Management Interface
LED	Light Emitting Diode
LMC	Libera (or Local) Machine Clock
LMT	Libera (or Local) Machine Time
LVDS	Low Voltage Differential Signaling
MAC	Media Access Control address
MC	Machine Clock
MCI	Measurement and Control Instrument interface
NCO	Numerically Controlled Oscillator
OS	Operating System
PC	Personal Computer
PCIe	Peripheral Components Interconnect – Express
PLL	Phase Locked Loop
RAM	Random Access Memory
RF	Radio Frequency
RMS	Root Mean Square
SA	Slow Acquisition data
SC	System Clock
SEL	System Event Log
SFP	Small Form Pluggable slot
SW	Software
TBT	Turn-by-Turn
TDP	Time Domain Processing
TIM	timing module (TIM)
USB	Universal Serial Bus
VCXO	Voltage Controlled Crystal Oscillator

1.5 Safety symbols and terms

The  symbol on the instrument indicates that the user should refer to the operating instructions located in the manual.

The  symbol on the instrument shows that it can source 70 V or more. Use standard safety precautions to avoid personal contact with this voltage.

The **WARNING** heading used in this manual explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading used in this manual explains hazards that could damage the instrument. Such damage may invalidate the warranty.

The **NOTE** heading used in this manual gives important explanations on the usage to avoid misunderstandings.

If the device is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

1.6 Warranty information

The manufacturer guarantees its customers that the products it supplies are free from defects in materials and workmanship for a period of 2 years.

This warranty shall not apply to any defect, failure or damage caused by improper use or inadequate maintenance and care. Instrumentation Technologies shall not be obliged to provide service under this warranty to repair damage resulting from maintenance by personnel other than Instrumentation Technologies representatives.

This warranty is limited to the repair and, if necessary, the replacement of the instrument based solely on the decision of Instrumentation Technologies. Each instrument is subjected to a quality test. Any early failures are detected by this method.

1.7 Deliveries and the responsibility of the manufacturer

- The compliance with all detailed requirements presented in this document.
- The design, procurement of material, fabrication, factory testing, cleaning, packing and delivery of product, according to the present specification.
- The supply of the Test Record for each processor.
- The supply of the User Manual and other related documents with full description of interfaces and connection recommendations.
- The supply of the software for complete installation of the instrument.
- The compliance with all the relevant IEC safety codes, recommendations, and standards.
- The supply of an effective warranty against equipment failure.

1.8 Command-line examples and abbreviations

The Libera Brilliance+ contains several modules. Due to hardware upgrades, the modules have unique labels that are visible in the registry tree. This document includes several examples for parameter control and signal readout. Majority of examples refer to a module, e.g. BPM or timing. To keep examples simple, these abbreviations apply:

Table 3: Abbreviations for example code.

Abbreviation	Options
bpm	raf3, raf4, raf5, raf6 … for 1st generation BPM module kraf3, kraf4, kraf5, kraf6 … for 2nd and 3rd generation BPM module kraf23, kraf24, kraf25, kraf26 … for sensors' readout from the KRAF2 module through libera-bmc
tim	tim2 … for original timing module (TIM) with SMA connectors evrx2 … for original event-receiver module (EVRX) evrxa2 … for the 2nd generation EVRX module
evrx	evrx2 … for original event-receiver module (EVRX) evrxa2 … for the 2nd generation EVRX module

In cases where certain functionality is supported only in certain module generation, this is additionally explained with the example.

2. Specifications

2.1 General specifications

Table 4: Platform specifications

Power supply	85 – 265 VAC (47 - 63Hz) or 120 – 330 VDC, supply current 8.1 A (100 VAC) or 3.9 A (200 VAC), fuse T,10A
Power consumption (chassis with 4 BPM modules)	Power-up: Imax = 12 A, Usup = 223 V, P = 2676 W Operation: I = 0.91 A, Usup = 223 V, P = 203 W
Operating environment	Temperature: 0-30°C* Relative Humidity: 0 – 80%
Warm up time	1 hour
Pollution degree	2
Installation category	II

* Environmental temperature from 0-40°C is allowed with special fans. Fan filters must be examined once per month and replaced if significant dirt is noticed. Fan rotation speed must be always set to maximum and fixed. Temperature sensors must be monitored and actions taken in case they exceed 80°C.

2.2 General target performance specifications

The following performance specification assumes kx = ky = 10 mm. Please refer to convention in Chapter 3.2.1.
Other important Libera Brilliance+ parameters are setup dependent and cannot be presented in a single table.

Please refer to Chapter for detailed information on standard Testing Setup and Testing Criteria.

Please contact support@i-tech.si for details.

Table 5: General target performance

Parameter	**Δ / $\sum$**	**Δ / $\sum$ x 10mm**
RMS uncertainty 2 kHz bandwidth, Pin = –20 dBm	-94 dB	0.2 µm
RMS uncertainty 500 kHz bandwidth, Pin = –20 dBm	-70.5 dB	3 µm
8-hour stability (ambient temp. = T±1°C)	-80 dB	1 µm
Temperature drift (ambient temp. range: 10°C to 35°C)	-94 dB/°C	0.2 µm /°C

2.3 External interfaces and schematics

Majority of interfaces to external systems are located in the front panel of Libera Brilliance+ (see Figure 1).

Figure 1: Libera Brilliance+ front panel

The power supply socket and ON/OFF switch are located in the back panel of Libera Brilliance+ (see Figure 2).

Figure 2: Libera Brilliance+ back panel

List of modules and their labeling is shown in Table 6 and in Figure 3.

Table 6: List of modules in Libera Brilliance+.

Module	Internal name	Description
ICB	icb0 / icb20	Inter-connection board with CPU and OS
GDX	gdx1	Gigabit data exchange module
EvRx	evrx2 / evrxa2	Timing module
BPM1	raf3 / kraf3 / kraf23	Beam position processing module
BPM2	kraf4 / kraf4 / kraf24	Beam position processing module
BPM3	kraf5 / kraf5 / kraf25	Beam position processing module
BPM4	kraf6 / kraf6 / kraf26	Beam position processing module

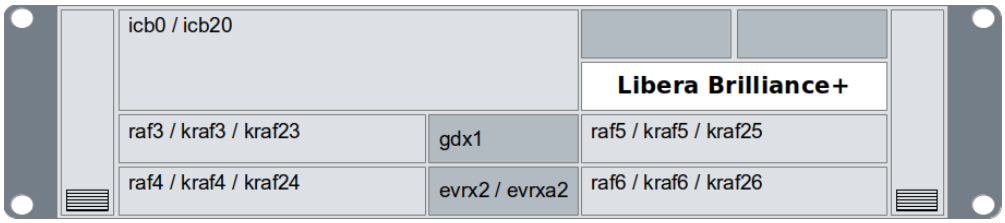

Figure 3: Module labeling and their geographical location.

2.3.1 ICB module

The inter connection board (ICB) contains interfaces that are used for communication with the user/control system. It contains the following port types (see also Figure 4):

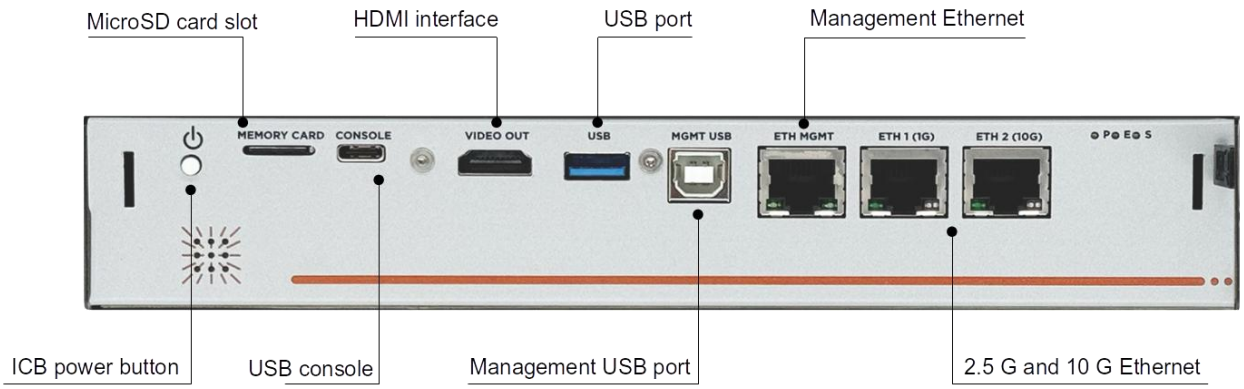

Figure 4: Front panel of the ICB2 module (model year 2025). Older versions feature slightly different interfaces.

- ICB power button: boots-up or shuts down the Linux OS. Always use this button to shut down the Linux OS before powering OFF or rebooting the instrument.
- Micro SD card slot: contains the micro SD card with OS and Libera software
- HDMI interface: it is used to connect an external display to the instrument.
- Management USB port: When the PC is connected to the "MGMT USB", the COM module is "disconnected" from the BMC module. Please contact support@i-tech.si before using it.
- USB port: standard USB port to connect the keyboard / mouse / wi-fi module / USB thumb drive / USB hub.
- Management Ethernet (ETH MGMT): Used for management purpose, e.g. IPMI-over-LAN
- Ethernet (2.5G and 10G): Used for standard network connection.
- USB console: micro USB console connector that has access to the COM module.

The inter connection board (ICB) provides the control actions of all AMC modules inside the Libera Brilliance+ chassis and offers both HW and SW interface toward the user's control system. The ICB communicates with other AMC modules through various protocols, like PCIe, USB, JTAG and I2C. It also offers a wide range of different HW interfaces (connectors) that are used for connecting Libera Brilliance+ to the accelerator control system.

The ICB module hosts the CPU module. Supported CPU modules:

- Intel i5-3230M 2.6 GHz: Installed in the instruments from beginning of year 2016.
- Intel i5-7440EQ 2.9 GHz: Installed in the instruments from beginning of year 2020.
- Intel i5-13600H 2.8 GHz: Installed in instruments with ICB2 module.

2.3.2 Timing module

The timing module contains optical and electrical interfaces that are connected to the accelerator's timing system. The timing module's front panel is shown in Figure 5. Table 7 contains details about the interfaces.

***Figure 5:** Front panel of evrxa2 module.*

Table 7: Timing module interfaces.

Connector	Connector type	Input configuration	Output configuration	Trigger rate and pulse width	Description
MC	LEMO single	LVTTL 3.3 V High-Z (10 kΩ) termination	LVTTL 3.3 V 50 Ω termination	<15 MHz, >20 ns	Machine Clock input
T0				<15 MHz, >20 ns	Custom I/O
T1					Postmortem trigger input
T2				< 20 Hz, >20 ns	Trigger input
ILK	LEMO differential	N/A			Interlock output
SFP	SFP				Rx/Tx optical events

2.3.2.1 MC interface (Machine Clock)

The MC interface is configured to receive the revolution clock (turn-by-turn). The revolution clock is usually derived from the RF. Revolution clock can be daisy chained between multiple platforms.
The PLL reference signal should have a jitter lower than 100 ps for optimum performance of the instrument.

2.3.2.2 T0 interface (custom)

The T0 interface can be customized for user's application. By default, it can transform optical events into electrical pulses.

2.3.2.3 T1 interface (postmortem)

The T1 interface is configured to receive the postmortem trigger. The postmortem trigger is usually connected with critical events, such as beam loss. After trigger arrival, history before the trigger is stored in the postmortem buffer.

2.3.2.4 T2 interface (trigger)

The T2 interface is configured to receive a trigger. Trigger can be used in 2 different modes:

- Acquisition trigger (default): Acquisition of ADC and turn-by-turn data buffers. Acquisition is started after trigger arrival. Typical use case is synchronized acquisitions with external events like beam injections or magnet kicks.
- Set-time trigger: Used after set-time announcement (synchronization). After the trigger, LMT is reset to 0. When synchronization is complete, trigger mode is set to "Acquisition trigger" mode. See also Chapter 3.7.2 for details.

Trigger is intended for triggering rare, typically asynchronous events. For most use cases the trigger repetition rate can be up to 20 Hz. If the repetition frequency is higher, some trigger signals will be ignored.

2.3.2.5 ILK interface (interlock)

The ILK interface is set active when beam position and intensity exceed pre-defined thresholds. The interface requires external circuitry as shown in Figure 6. Figure 7 shows schematics on the timing module.

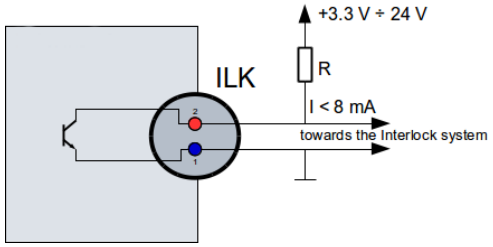

Figure 6: Interlock interface external connection.

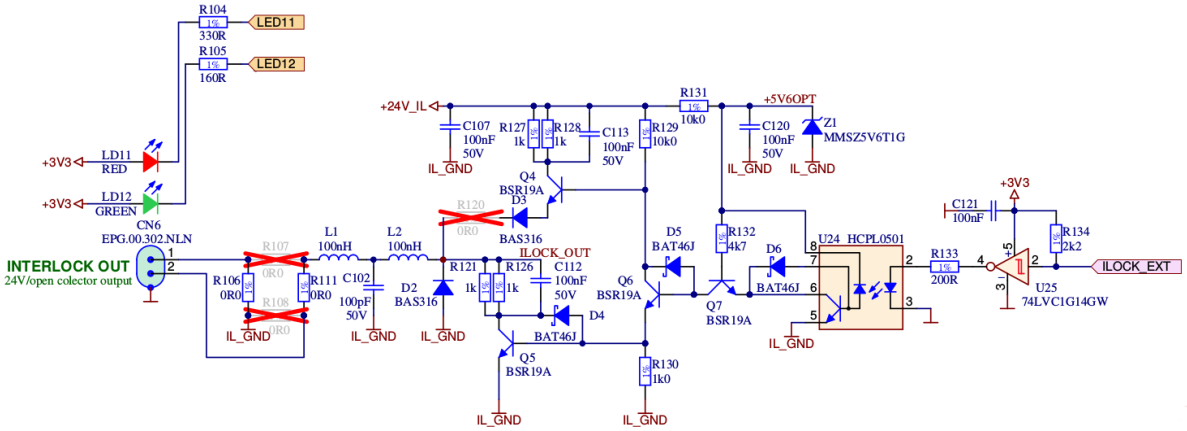

Figure 7: Interlock schematics.

The interlock functionality is described in Chapter 3.5 in details.

NOTE: The interlock output is active during the instrument boot up for approximately 30 seconds. When going to shutdown, it gets activated (if not before).

2.3.2.6 SFP slot

The SFP slot must be used with a compatible optical SFP transceiver module. Its main purpose is receiving optical events from the event generator. To establish the link between the transmitter (generator) and EvRx SFP slot, a proper SFP slot's frequency must be set. Default frequency is 100.625 MHz. To check the value, use:

```
libera-ireg boards.evrx.sfp_freq
boards.evrx.sfp_freq: 100625000
```

The `sfp_freq` parameter is not runtime configurable. It is loaded at `libera-ebpm` daemon start up. To change the value, stop the `libera-ebpm` daemon first, edit the `/var/opt/libera/cfg/libera-ebpm.xml` file and start the `libera-ebpm` daemon again.
Event decoding is done within the timing module's FPGA. It is described in Chapter 3.8.2.

2.3.2.7 LED diodes

Latest generation EVRX module (evrxa2) has 2 LEDs above each of the input LEMO connectors (only 1 LED above the ILK output). Their behavior is described in Table 8.

Table 8: LED status in evrxa2 module.

Interface	
MC	The orange LED is illuminated when the MC signal is present The green LED is illuminated when the MC PLL is locked
T0, T1, T2	The orange LED is illuminated when the interface is set as an output (by default it is input) The green LED blinks when it receives a pulse
ILK	The red LED blinks when Interlock is detected. If it is illuminated, the interlock condition is fulfilled continuously.

2.3.3 BPM module

Inputs to the BPM module are the RF signals from the button pick-ups in the synchrotron tube. RF chains are equipped with appropriate band-pass filters (usually 500 MHz or 352 MHz). Other versions are available, too. Input signal limitations are given in Table 10.
RF signals are connected to the input connectors on the BPM front panel. There are four female SMA connectors, labeled A, B, C and D (see Figure 8). They can be connected directly to the pick-ups (or through additional attenuators).

Figure 8: *BPM module front panel (3rd generation BPM module)*

2.3.4 LED diodes

There are 3 LEDs on the front panel of each module. The LED colours are:

- blue (marked S - status)
- red (marked E - failure)
- green (marked P - FPGA status)

On the ICB module, there are also 3 additional LEDs (marked 1, 2, 3) that are intended for future use.

During the boot-up, a hardware test is performed. All 3 LEDs on all modules blink for 3 times, after that the diodes indicate the status as described in Table 9.

Table 9: *LED diodes status.*

Indication	Description	ON	OFF	Blink
BLUE (S)	Board status	FPGA not running and main power is OFF	FPGA running and main power is ON	**Only on ICB – after shutdown:** FPGA running, but chassis is powered OFF. It is safe to remove/replace the modules

Indication	Description	ON	OFF	Blink
RED (E)	Basic feedback about failures	Failure detected (check SEL for more information)	No failure detected	
GREEN (P)	FPGA status	FPGA running		FPGA not running **Only on ICB2: the system is shutting down or starting up**

Additional LEDs on the ICB module only blink once at startup and then stay turned off.

2.4 Input signal specifications

The input signal specifications for the BPM module and for the timing module are shown in Table 10 and Table 11.

Table 10: BPM module specifications.

	Specification
Input frequency range	< 550 MHz
A/D converters	125 MHz / 16 bit
Input impedance	50 Ω
Maximum input signal	+16 dBm (damage value)
Peak voltage	< 50 V 2-ns-single-bunch*
Input connectors	SMA female (4x)
Variable attenuation	0 dB to 31 dB

* estimated value

Table 11: Timing (EvRx) module specifications.

	Specification
Timing interfaces	single-ended LEMO (electrical pulses for MC, T0, T1, T2) SFP slot (optical interface)
Input impedance	10 kΩ … MC, T0, T1, T2
Input signal level	LVTTL 3.3 V
Interlock	differential LEMO, optoisolated open collector output
Maximum input frequency	20 Hz … (T1, T2), can be modified <15 MHz … MC
Optical event decoding	16 bit
Internal event codes	16 per trigger line (MC, T0, T1, T2)

NOTE: Do not exceed maximum input signal ratings. It may result in severe damage of the instrument.

2.5 Software

Libera software is organized in several layers. The bottom layer contains the Linux Kernel Module and BMC library which communicate with FPGA and Hardware through the IPMI and PCI Express. The BPM application and Platform daemon communicate with the BMC and Linux Kernel Module (towards the hardware modules) as well as with the MCI layer (towards the user).

Several control systems and clients are supported by libera servers that run in the instrument (libera-ioc, libera-ds, etc.). , See Figure 9 for schematic overview.

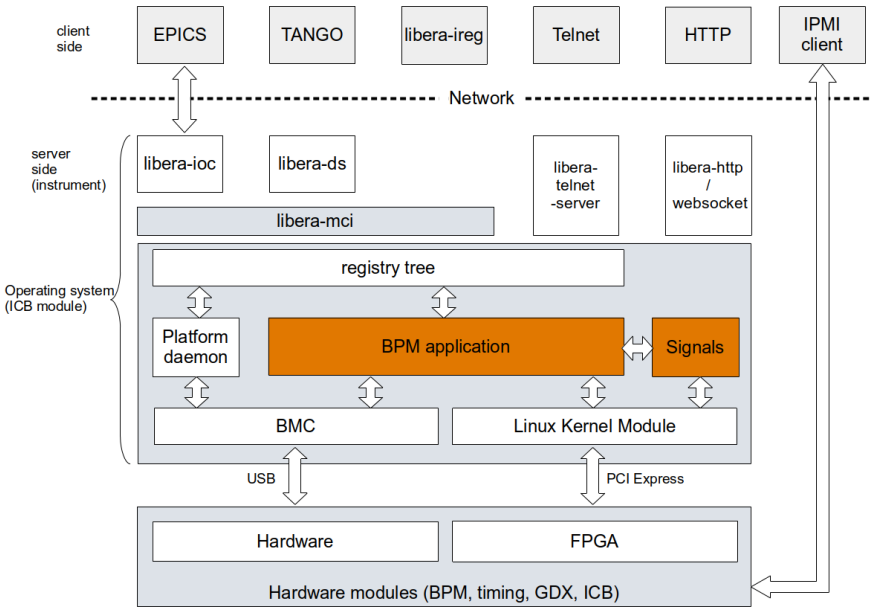

Figure 9: Libera Brilliance+ software structure

The platform daemon, BMC and Linux Kernel Module are part of the Libera Base framework which can host various applications (for other Libera instruments). Servers (libera-ioc, libera-ds, etc.) are generic and not hard-linked to specific application and parameters. Upper software layer is usually insensitive to software upgrades.

NOTE: IPMI over LAN support is available in ICB module produced from year 2025 on (2nd generation ICB module).

2.5.1 Registry tree (MCI)

Parameters, statuses and signals are organized in a tree-like structure. Nodes of the registry represent readable/settable parameters, sensor values or data buffers and streams. The registry tree is built dynamically based on the application and type and number of modules currently present in the platform.

```
module1.
      .parent node1.
      .             .child node1
      .             .child_node2
      .             .child_node3
      .parent_node2.
      .             .child node1
      .             .child_node2
      .parent_node3
```

Property	Description
Node name	Name of the node (e.g. parent_node1)
Value	Current node value; No value, if parent node
Value type	Integer, Floating, Enumeration, Boolean; Undefined, if parent node
Validator expression	Settable limits; Empty, if parent node
Num of values	Number of elements (if array); zero, if parent node
Full path	Full path to the node
Num of children	Number of child nodes (if any)
Flags	Readable, writable, persistent, hidden; none, if parent node
Parent node name	Parent node name
Children	Children node names, if any

Every node has its own pre-defined type:

- integer
- floating point
- enumeration

Some nodes are marked with “persistence” flag. Values of these nodes are stored to the custom .xml file and loaded at next application or instrument restart.

Certain nodes emit notifications when:

- value changes
- value is out of range

The available data streams (signals) are enumerated in the registry and defined by instrument application software. Data stream classes provide access to different types of signals.

2.5.2 Application parameters

Application parameters are organized in the registry tree. The registry tree is generated at the instrument’s boot up dynamically. Registry entries (parameters) are available only for the modules that are present in the platform. All parameters are accessible with a `libera-ireg` command line utility. Detailed description of the `libera-ireg` utility commands and options together with some examples of use can be found in Appendix A.

2.5.3 Platform parameters

Platform parameters include temperature, voltage, current and other sensors and are intended for health monitoring only. Sensors’ values are useful for diagnostics if a module fails.
Parameters can be accessed with `libera-ireg` client using “`-P`” option, see example below. For more details on health monitoring and platform daemon application see Chapter 5.7.

```
libera-ireg dump -P boards -l2
boards
  icb0
    board info
    sensors
  gdx1
    board info
    sensors
  evrx2
    board info
    sensors
  kraf3
    board info
    sensors
  kraf4
    board_info
    sensors
  kraf5
    board info
    sensors
  kraf6
    board_info
    sensors
  os
    board info
    sensors

libera-ireg dump -P fans
fans=5877
  left front=5812
  left middle=5692
  left rear=5659
  right_front=5973
  right_middle=5937
  right_rear=6105
```

2.5.4 Upgrade policy

The software release development process includes extensive QC testing. This is necessary to guarantee the stability of the release, which is of a paramount importance.

Software versions for platform B instruments started with version 2.00. Major versions are marked with even numbers (e.g. 2.20). Minor versions usually contain bugfixes and differ from the major version by build number. The upgrade procedure is documented in the README file that is provided with software packages.

Deliveries included in each major release are:

- Complete install ready SW stack (FPGA, SW) customized to certain machine
- Updated manual and other documents

2.6 Enclosure

The 2U high 19” enclosure is ready for assembly in the rack using four M6 mounting screws.

Height: 88 mm
Width: 448/483 mm
Depth: 310 mm
Weight: 13 kg (fully populated unit)

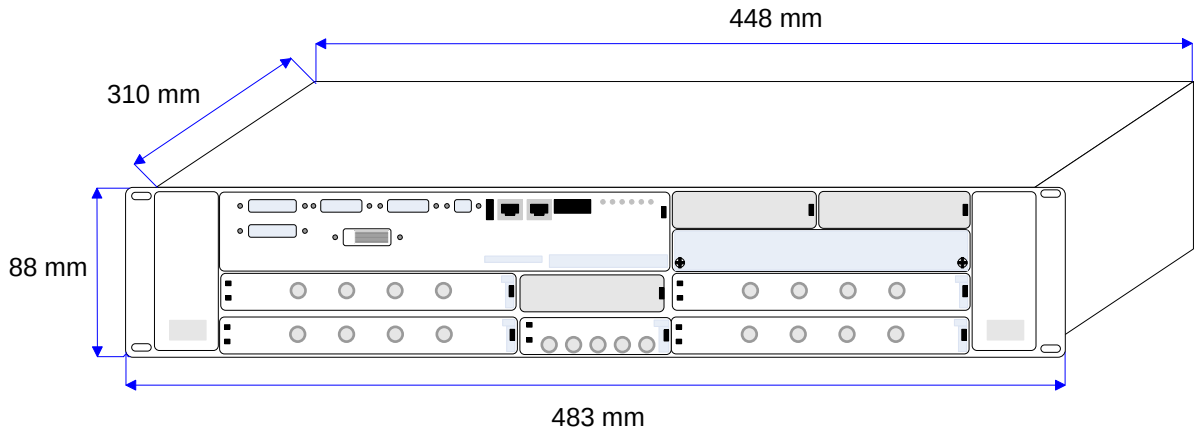

Figure 10: Libera Brilliance+ dimensions.

2.7 Data throughput

The Libera Brilliance+ connects to the network 1 GbE, 2.5 GbE or 10 GbE interfaces. Theoretically, the throughput for 1 GbE is 1 Gbps = 125 MBps.

The following example calculates the data throughput under certain conditions:

- 4 BPMs in a chassis
- Turn-by-turn acquisition on-trigger (10.000 samples)
- ADC acquisition on-trigger (2048 samples)

Turn-by-turn data: 10.000 × 32 bytes × 4 BPMs = 1.22 MB
ADC data: 2.048 × 8 bytes × 4 BPMs = 0.07 MB

Altogether, the example acquisitions require ~2 MB of data on each trigger. In case of 1 Hz trigger, this is 2 MBps data throughput. With 10 Hz trigger frequency, this is 20 MBps only for upper mentioned data acquisitions with predefined buffer lengths.

One must consider that the real 1 GbE throughput is less 125 MBps and the network may get overloaded if other instruments and computers send and exchange the data.

The throughput for 2.5 GbE and 10 GbE interfaces is scaled by 2.5 and 10, respectively.

Turn-by-turn data buffer is taken from the circular buffer, which is filled in continuous mode. If the writing frequency (=revolution frequency) is lower than the read-out speed, the data buffer readout is theoretically unlimited. However, the data on demand acquisition request allocates the memory (RAM), which is limited. By default, the Libera Brilliance+ features 4 GB of RAM. If requested data on demand buffer is too large, it may not be read successfully. Use reasonable buffer sizes or temporarily disable other readings while doing large data on demand acquisitions.

Adjust the buffer lengths and read-out frequencies properly. In case of data traffic congestion (on the client side), not all requested data may arrive in time.

3. Functionalities and features

3.1 Libera Brilliance+ block diagram

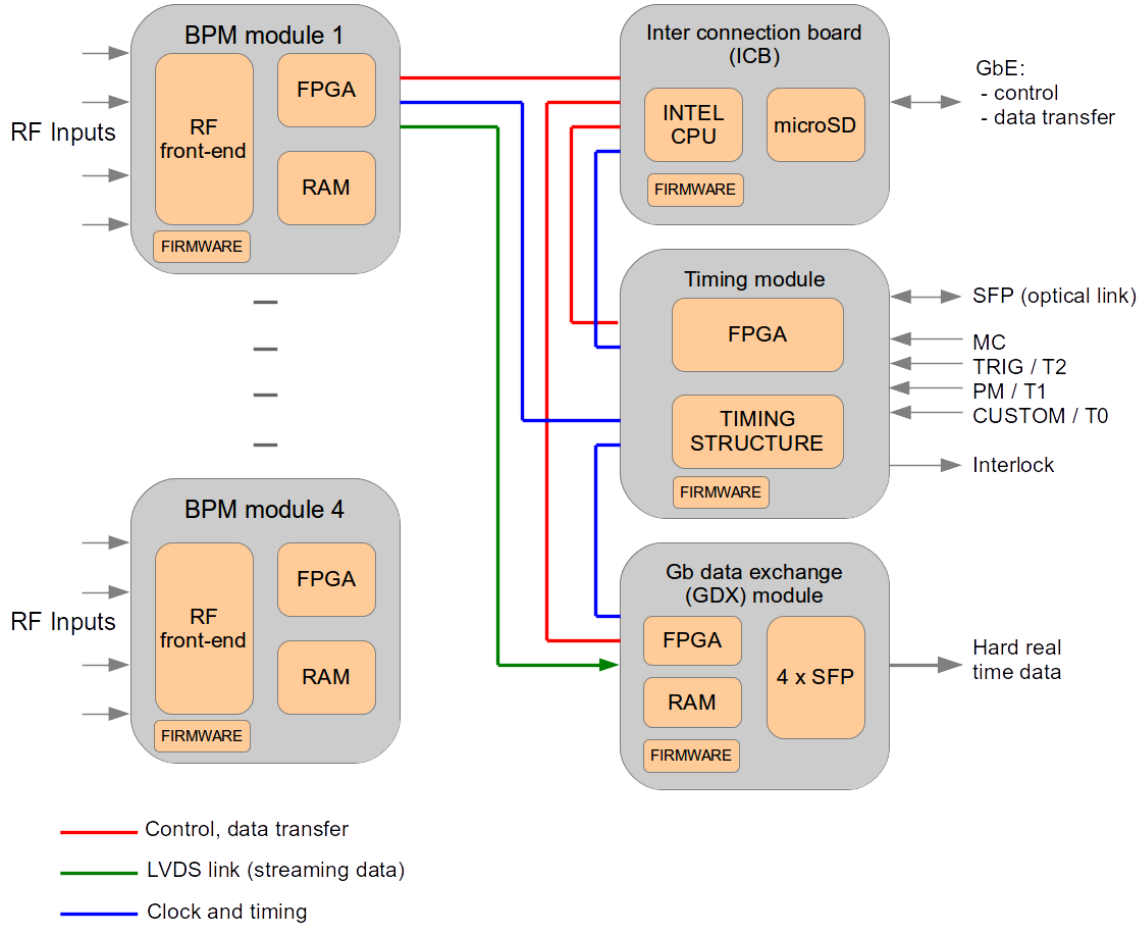

Figure 11: *Libera Brilliance+ block diagram*

Libera Brilliance+ consists of four types of modules:

- BPM module receives the signal from the button pick-ups, does analog signal processing and conversion to digital. The module also contains the FPGA for digital data processing and RAM for storing the data that is available to the user on demand.
- The inter connection board (ICB) provides both HW and SW interface towards the user's control system. It performs the control actions of all boards in the Libera Brilliance+ chassis and enables the data transfer to the user.
- The functionality of the timing module is the frequency locking of the MC signal by the means of the PLLs and the distribution of clock and trigger signals to all application boards inside the Libera Brilliance+ chassis.
- Gigabit data exchange (GDX) module offers fast links (SFP connectors) through which the fast data is transferred in hard real time. This data can be used for building a fast feedback application.

3.2 Signal processing

The input signals enter the crossbar switch matrix where each signal passes each analog path. Input signals are digitized by two 2-channel A/D converters with 16-bit resolution and sampling rate typically between 100 and 125 MHz. Raw ADC data (amplitudes A, B, C, D) is used for digital signal processing.
Digital signal processing (DSP) implemented in Libera Brilliance+ provides several data paths at programmable bandwidth. This is the foundation on which a single hardware device can facilitate all required position measurements: pulsed, first turn(s), turn-by-turn, and regular closed orbit.
Core DSP consists of four identical channels. Each channel consists of a digital downconverter (DDC) and time domain processing (TDP) method followed by parallel processing in wideband and narrowband paths.

General overview of the signal processing and data paths is shown in Figure 12.

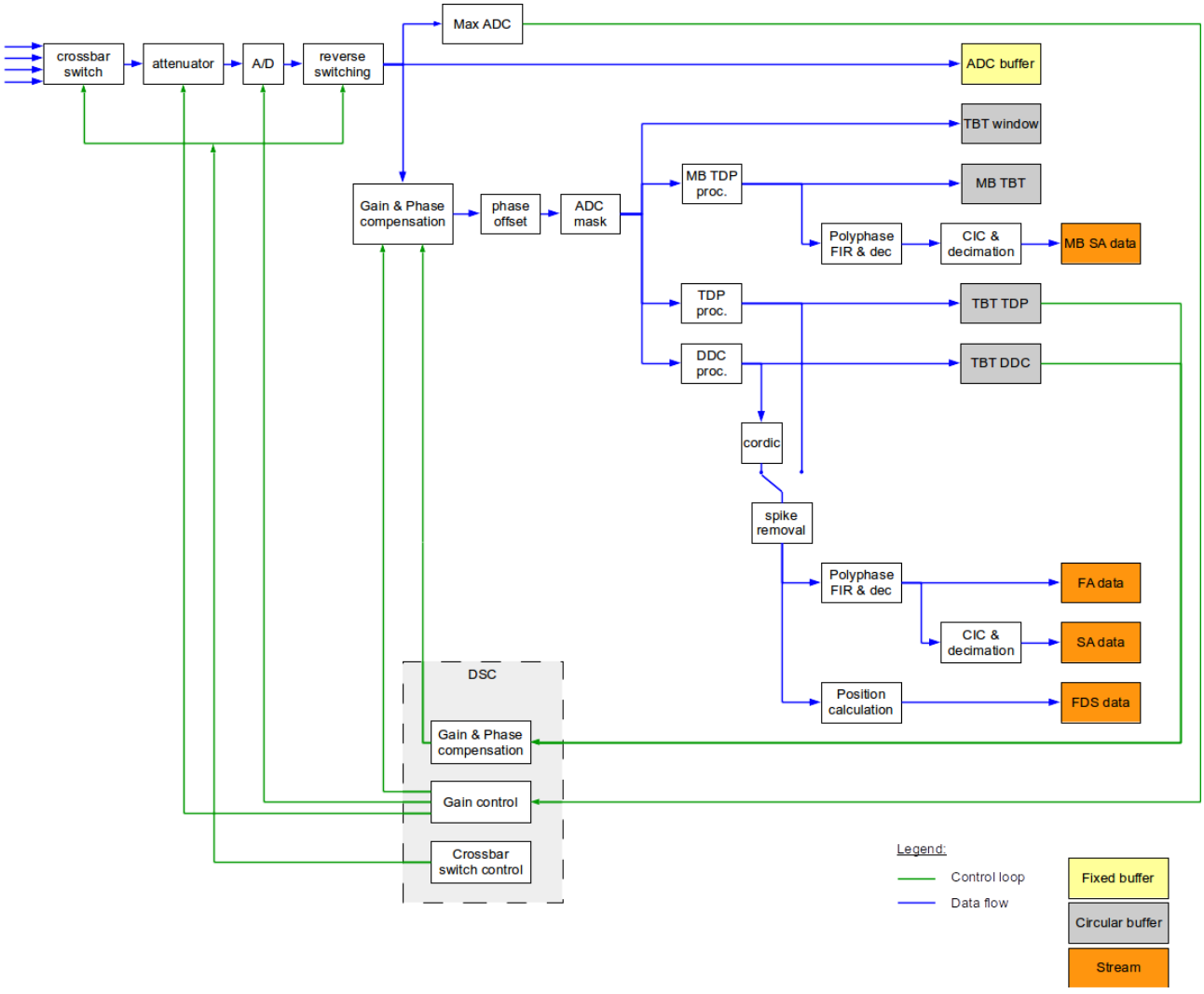

Figure 12: Signal processing overview.

3.2.1 Position calculation

The beam position is calculated from the input RF signals amplitudes Va, Vb, Vc and Vd considering the coordinate system in Figure 13. Calculation of the amplitudes is described in Chapters 3.2.2 and 3.2.4.

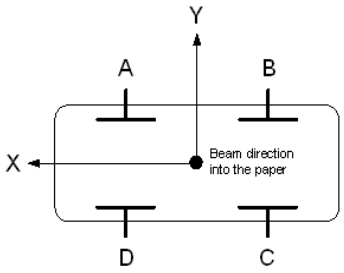

Figure 13: *Coordinate system*

$$X = K_X \frac{(V_a + V_d) - (V_b + V_c)}{V_a + V_b + V_c + V_d} - X_{OFFSET}$$

$$Y = K_Y \frac{(V_a + V_b) - (V_c + V_d)}{V_a + V_b + V_c + V_d} - Y_{OFFSET}$$

$$Q = K_X \frac{(V_a + V_c) - (V_b + V_d)}{V_a + V_b + V_c + V_d} - Q_{OFFSET}$$

$$SUM = K_S * 2^{-28} * (V_a + V_b + V_c + V_d) + S_{OFFSET}$$

Table 12: Position calculation parameters.

Parameter	Minimum value	Default value	Maximum value	Unit
X offset, Y offset, Q offset	-536,870,912	0	536,870,911	nm
Kx, Ky	1	10,000,000	536,870,911	nm
S offset	-536,870,912	0	536,870,911	/
Ks	1	67,108,864	536,870,911	/

Parameters can be read using:

```
libera-ireg dump boards.bpm.signal_processing.position
position
  Kx=10000000
  Ky=10000000
  Ks=67108864
  off_x=0
  off_y=0
  off_q=0
  off_s=0
```

Examples how to set custom values to parameters:

```
libera-ireg boards.bpm.signal_processing.position.Kx=13000000
libera-ireg boards.bpm.signal_processing.position.Ky=11000000
libera-ireg boards.bpm.signal processing.position.Ks=268435456
libera-ireg boards.bpm.signal processing.position.off x=135000
libera-ireg boards.bpm.signal_processing.position.off_y=-215000
libera-ireg boards.bpm.signal_processing.position.off_q=-50000
libera-ireg boards.bpm.signal_processing.position.off_s=1000000
```

3.2.2 Undersampling

The RF frequency of the accelerator (typical cases are ~352 MHz, ~500 MHz) is divided by harmonic number which represents number of buckets in a turn) and gives the revolution frequency. Signals from the are sampled by A/D converter at ~108 MHz (for 352 MHz RF) or ~117 MHz (for 500 MHz RF) sampling rate. Exact sampling rate is controlled by the VCXO (custom for each accelerator) and matches an exact integer multiple of the accelerator revolution frequency. The (under)sampling is possible thanks to two facts:

1. The band-pass filters in each analog channel with the passband between 15-19 MHz around the central frequency (352 MHz or 500 MHz) filter out unwanted spectra.
2. Beam information is in about 19 MHz bandwidth around the 500 MHz (or 352 MHz, depending on accelerator RF frequency) signal.

From the digital signal processing point of view, the result is a signal with about 28-32 MHz (352 MHz – 3x108 MHz; 500 MHz - 4x117 MHz), mirrored from the 7th (352 MHz RF) or 9th (500 MHz RF) Nyquist zone sampled with 108 MHz or 117 MHz sample rate.

3.2.3 Digital down conversion (DDC)

DDC principle is known from the digital signal processing theory. Implementation in Libera Brilliance+ is shown in Figure 14. A numerically controlled oscillator (NCO) with controlled frequency drives two multipliers in quadrature. The multipliers multiply signals coming from the ADC converter with a signal from the NCO. Typically, a series of programmable low pass filters extract the required baseband component. Infinite impulse response (IIR) and Cascaded Integrator Comb (CIC) filters are used. Two quadrature paths (I and Q) recombine in a Cartesian to polar (C2P) converter yielding magnitude of the baseband signal. Cordic algorithm is used.

$$V = \sqrt{I^2 + Q^2}$$

Four signal amplitudes (Va, Vb, Vc, Vd) are available directly at the accelerator revolution rate. A typical configuration of the filters provides low pass bandwidth that is approximately ½ of the accelerator revolution frequency, with sampling rate that equals the accelerator revolution frequency.

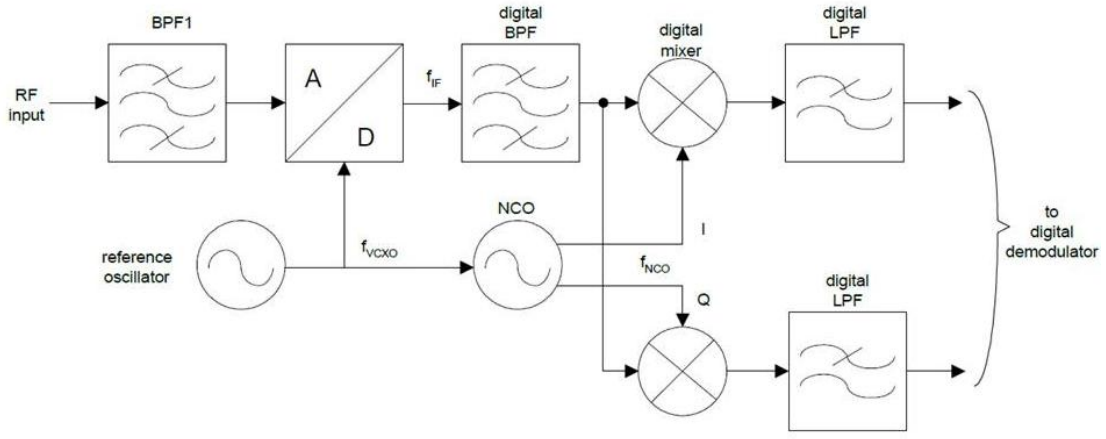

Figure 14: *DDC in Libera Brilliance+*

3.2.4 Time domain processing (TDP)

The time domain processing calculates the amplitudes of the acquired signals at turn-by-turn rate (e.g. from channel A) using the following equation:

$$V = \sqrt{\sum_{i=1}^{D} x_i^2}$$

where D is the turn-by-turn decimation (number of ADC samples in one turn) and x_i is ADC sample from a channel. The decimation D is pre-set for every accelerator design individually.
Amplitudes of signals from all four channels are calculated as described above. The beam position is calculated from these amplitudes as described in Chapter 3.2.1.

3.2.5 Multibunch processing

The multibunch processing uses multiple masks and time-domain processing principle to calculate positions from these ADC masks. Up to 4 independent ADC masks over a turn can be selected. This is useful to measure position from individual bunches that are separated by at least 300-360 ns (depending on the band-pass filter ringing time).
The incoming data is synchronous to other two turn-by-turn processing blocks (DDC and TDP). Data is processed in time domain (see Chapter 3.2.4). Once the masks are processed, the output amplitudes (four amplitudes from each ADC mask) are filtered through CIC and FIR filters that reduce the bandwidth and data rate to FA and SA data rates.
See Chapter 3.2.8 for details how to set the multiple ADC masks.

NOTE: A common ADC mask (Chapter 3.2.6) must be set to full (1) for proper multibunch feature operation.

3.2.6 Common ADC mask

The turn-by-turn data is processed from the ADC data. One turn is described with fixed number of ADC samples (decimation “D”). Decimation varies between accelerator designs and can range from 10 to 1200.
Accelerator designs with larger decimation values (>40) are able to see clear waveforms of single-bunch fill pattern that do not overlap between the turns. Even larger decimations (>200) can distinguish partial fill patterns or hybrid fill patterns.
For special (non-uniform) fill patterns, an ADC mask can be applied to the processing window. The ADC mask is a bit-mask with length of exactly 1 turn. The mask specifies which part of the fill pattern is used for the turn-by-turn processing.
To review the ADC data that is used for the turn-by-turn processing, a TBT window data buffer is used. The TBT window contains masked ADC data (see Figure 12). The first data sample in the TBT window is aligned to the start-of-turn data point.

TBT window data is read with:

```
libera-ireg signal boards.bpm.tbt.tbt_window -s400
```

Example waveform (blue) is shown in Figure 15.

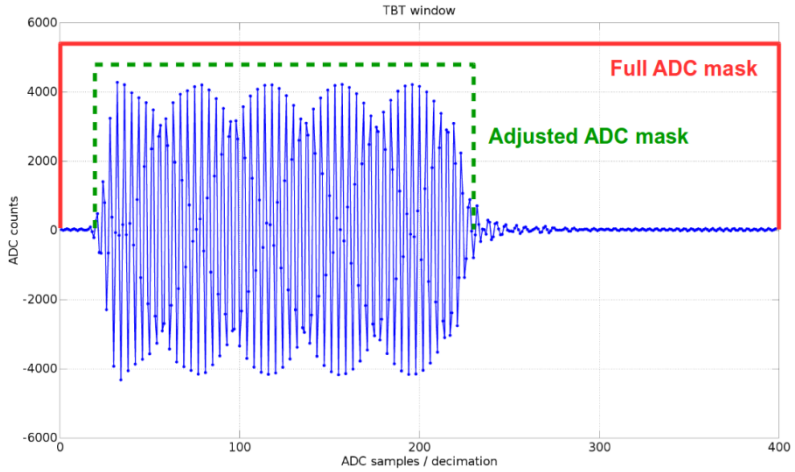

Figure 15: Common ADC mask adjustment.

Example how to read the ADC mask:

```
libera-ireg boards.bpm.tbt.adc_mask
boards.bpm.tbt.adc_mask: [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
```

To set the ADC mask for samples 20 – 230, use the following commands (set all to “0” then set samples from 20 to 230 to “1”):

```
libera-ireg boards.bpm.tbt.adc_mask=0
libera-ireg boards.bpm.tbt.adc_mask[19:229]=1
```

Full ADC mask is used by default (Figure 15, red line). Adjusted ADC mask is shown with green dashed line in Figure 15.

An incorrectly set ADC mask results in bad RMS and non-reliable position calculation. It is better to keep the ADC mask default.

NOTE: The ADC mask setting affects the whole processing chain (TBT, MBTBT, FA, SA).

3.2.7 Phase offset

The phase offset adjusts the first bucket to the start-of-turn sample in the TBT window. Adjustment must be done with a single bunch or partial fill pattern where the first bucket is filled.
Assuming all BPMs in the ring have equal RF cable lengths, the phase difference between the first and last BPM is 1 turn which corresponds to decimation “D” (see Figure 16). Setting range for the phase offset is 0 to D-1 with setting resolution of 1 ADC sample.
Theoretically, BPM1 will have the maximum phase offset (D-1) and BPM N (the last BPM) will have phase offset 0.

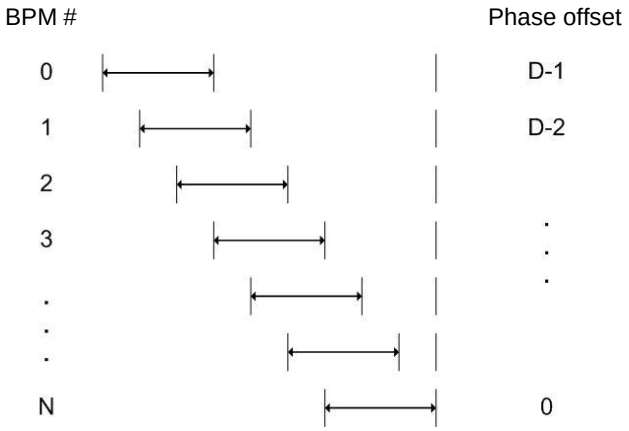

Figure 16: Phase differences between the BPMs (example).

Phase offset is set using:

```
libera-ireg boards.bpm.tbt.phase_offset=20
```

This will delay the data in the TBT window by 20 ADC samples.

Figure 17 shows few cases how the fill pattern moves with different phase offset settings. Once the phase offset is set in multiple BPMs, all platforms must be synchronized. Synchronization trigger must be derived from the RF to keep the synchronism.

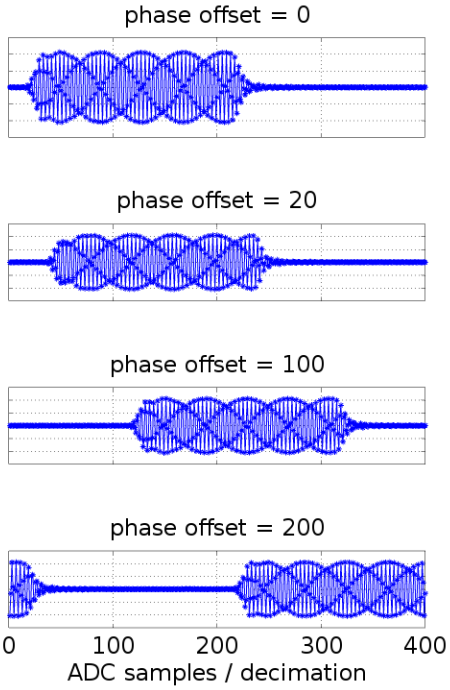

Figure 17: Phase offset adjustment.

NOTE: While setting the phase offset value the DSC algorithm is paused for 1 millisecond (ca. 5 turn-by-turn samples).

3.2.8 Multiple ADC masks

Accelerator designs with large decimations (>200) can use multiple ADC masks. The masks can be set throughout 1 turn. Up to 4 masks can be defined with an offset and window lengths, see Figure 18.

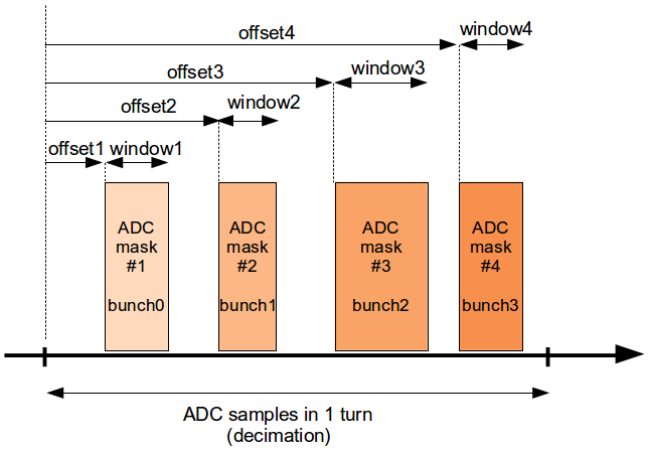

Figure 18: Multiple ADC masks over one turn.

Parameters are configured in nodes (bpm=kraf3,kraf4,kraf5,kraf6):

```
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_0.offset
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_0.window
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_1.offset
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_1.window
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_2.offset
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_2.window
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_3.offset
libera-ireg boards.bpm.tbt.multi_bunch_positions.bunch_3.window
```

Use TBT window buffer data for adjustment.

NOTE: Common ADC mask must be set to 1 (full mask). Check phase offset.

3.3 Signal conditioning

3.3.1 Gain control

Each analogue channel (A, B, C, D) in a BPM module contains one 5-bit programmable attenuator. The attenuation can be controlled in the range from 0 to 31 dB with steps of 1 dB.

Automatic gain control (AGC) reads the second max ADC value and automatically adjusts the attenuation according to the input RF signal power and gain scheme. Refresh period is 0.1 second. If the calculated input power is higher than the highest value defined in the gain scheme (0 dBm), the highest value from the gain scheme is used and vice versa.
Automatic gain control can be enabled or disabled. When enabled, power level is set automatically with no manual access. When disabled, power level is set manually by the user.

NOTE: When AGC is disabled, power level must be controlled by the user. The second max ADC value must be monitored to detect a possible ADC saturation and increase the attenuation properly.

Figure 19 shows the AGC scheme.

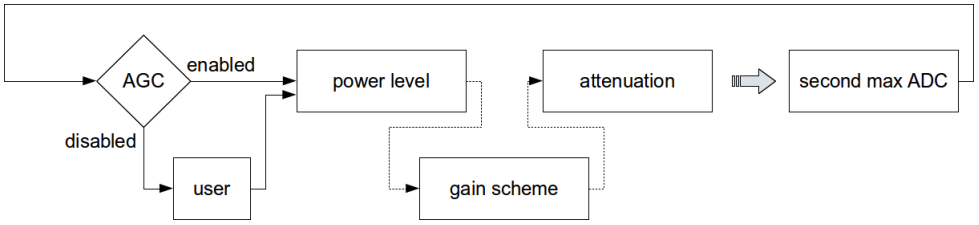

Figure 19: AGC block scheme.

Enabling/disabling the Automatic gain control can be done with the following command:

```
libera-ireg boards.bpm.conditioning.tuning.agc.enabled=true

libera-ireg boards.bpm.conditioning.tuning.agc.enabled=false
```

Reading the Automatic gain control mode and the current power level:

```
libera-ireg boards.bpm.conditioning.tuning.agc.enabled

libera-ireg boards.bpm.conditioning.tuning.agc.power_level
```

Manual power level parameter setting is only allowed when Automatic gain control is disabled. Valid settings are from -80 to 0 dBm in 1 dBm steps. In case of invalid manual setting, last good value is kept. To set the power level:

```
libera-ireg boards.bpm.conditioning.tuning.agc.power_level=0
```

To read the current attenuation value (in dB), do:

```
libera-ireg boards.bpm.conditioning.att
```

NOTE: The attenuator value parameter is intended for read-only mode. It can be used to manually set the attenuation value but this will override other mechanisms (AGC, power level reading, etc.). To ensure proper operation of the instrument, do not attempt to set this registry node but use it only for read-out.

The gain scheme is a table that contains relation between the input signal power and attenuators' settings. There are 4 pre-defined attenuation values that cover the range from 25 dB to 3 dB. The gain scheme was optimized for best

linearity. Configuration file is located in the /`var/opt/libera/cfg` folder for each BPM module independently (gain_kraf3.conf, gain_kraf4.conf, gain_kraf5.conf and gain_kraf6.conf). Gain scheme is used for both, automatic and manual gain control.
Parameters that describe the gain control functionality are gathered in Table 13.

Table 13: Gain control parameters

Registry node* / file	Description
libera-ireg boards.*bpm*.conditioning.tuning.agc.enabled	Enable / disable Automatic Gain Control
libera-ireg boards.*bpm*.conditioning.tuning.agc.power_level	Set / read the power level [dBm]
libera-ireg boards.*bpm*.conditioning.att	Read the current Attenuation value [dB].
/var/opt/libera/cfg/gain_*bpm*.conf	Table with relation between the power_level and Att value.

* *bpm* = kraf3, kraf4, kraf5, kraf6

Gain scheme is graphically shown in Figure 20.

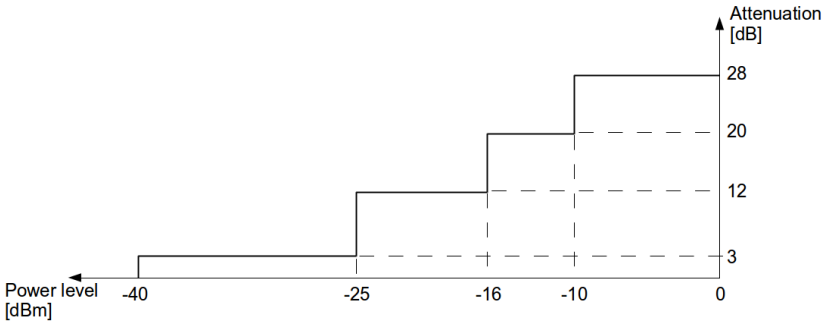

Figure 20: Gain scheme.

An excerpt from the gain scheme (file) is shown in the output below. It shows the points where the attenuation changes (at -11 dBm, -17 dBm, -26 dBm and -40 dBm).

```
# P[dBm] Att h
#
   0     28  0.5
  -1     28  0.5
  -2     28  0.5
...
  -10    28  0.5
  -11    20  0.5
  -12    20  0.5
...
  -16    20  0.5
  -17    12  0.5
  -18    12  0.5
...
  -25    12  0.5
  -26    3   0.5
  -27    3   0.5
...
  -40    3   0.5
  -41    0   0.5
  -42    0   0.5
...
```

The first column (P) determines the input signal level in dBm, the second (Att) is the attenuator setting [dB], the third (h) is a hysteresis [dB]. Hysteresis was introduced to avoid unwanted switching between two adjacent power levels (see Figure 21). If the difference between currently set and actual level reading is less than 3 dBm, the new power level setting is applied directly. If the difference is more than 3 dBm, the new power level is set in few steps.

For proper operation, all BPM modules must use same gain scheme.

NOTE: For a small signal-to-noise ratio, AGC can not determine the input power level with high accuracy. Accurate signal power measurement is guaranteed only down to -60 dBm.

In Figure 21 there is a graphical presentation how transitions between two consecutive power_level are performed within AGC and how the hysteresis value affects these transitions. In the case below, the hysteresis of 0.25 dB is used.

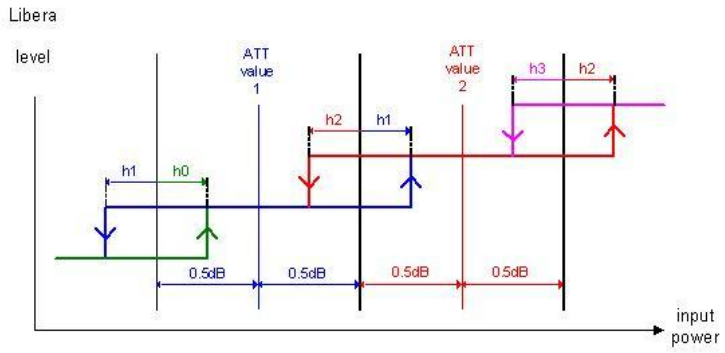

Figure 21: Gain scheme with hysteresis.

Attenuation values for the default gain scheme were chosen for best linearity under all conditions (off-centered and centered).
A lot of attention was put to have proper signal levels on various components on the analogue part of the BPM module. Various attenuators' settings have strong influence to linearity ("beam current dependence"). Users are encouraged to do further tests and customize the default gain scheme, but keep the default configuration as a backup.

NOTE: libera-ebpm daemon restart is required so that gain scheme changes take effect.

3.3.2 Crossbar switch

Four RF inputs enter the crossbar switch and can be redirected to any of the four RF channels (RF A, B, C, D) followed by ADC converters (ADC A, B, C, D). Switching through four switch positions was chosen. This means that each signal goes once through each of the four analog channels. The digital crossbar switch redirects any of the four inputs back to their original channels. After signals exit the digital crossbar switch, they pass through four signal optimization modules (M 1, 2, 3, 4).
The RF and the digital crossbar switches are synchronized so that input channels A, B, C and D are always processed on digital signal processing channels A, B, C and D respectively. Applying a low pass filter at the end of the digital signal processing automatically yields a signal, which amplitude is proportional to the input signal multiplied by an average transfer function of all four RF channel and ADC pairs.

$G = (G1 + G2 + G3 + G4)/4$

G1, G2, G3, G4 are transfer functions of individual RF channels and ADCs. Every input signal is thus processed by the same transfer function G, which eliminates the effect of gain drifts of individual channels to measurement result.
Switching frequency is derived from machine clock (revolution frequency), see Chapter 3.7.1 for details. The division factor is chosen to have the single position switching frequency around 13 kHz (example of the Diamond Light Source storage ring: 533818 Hz divided by 40 results in 13345 Hz). When using a sequence of 4 switch combinations, each combination is repeated with the frequency of 3336 Hz.

There are a total of 16 combinations of the crossbar switch, but only 4 are used for switching as an optimal pattern.

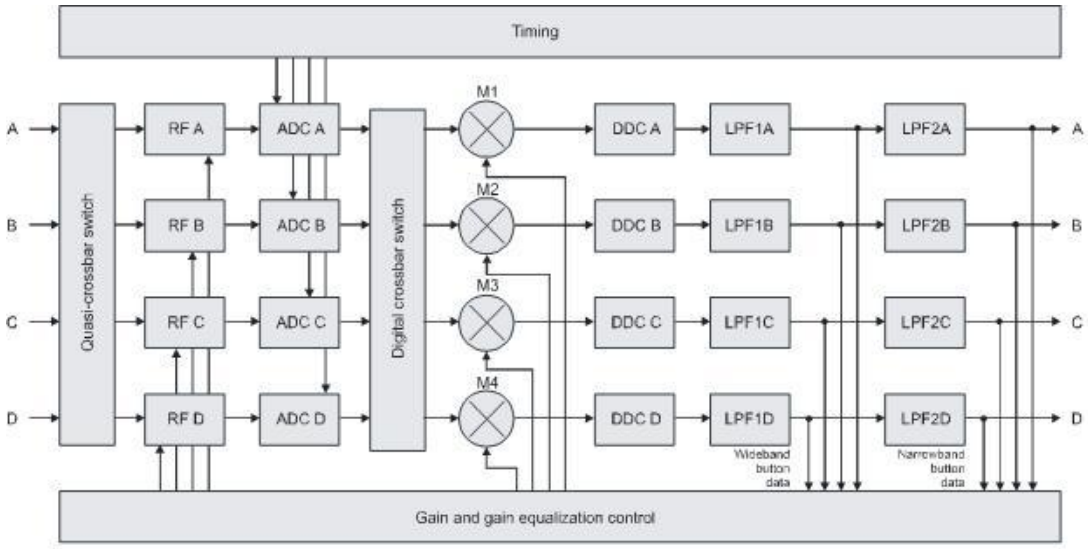

Figure 22: Architecture based on the Instrumentation Technologies patented method and device.

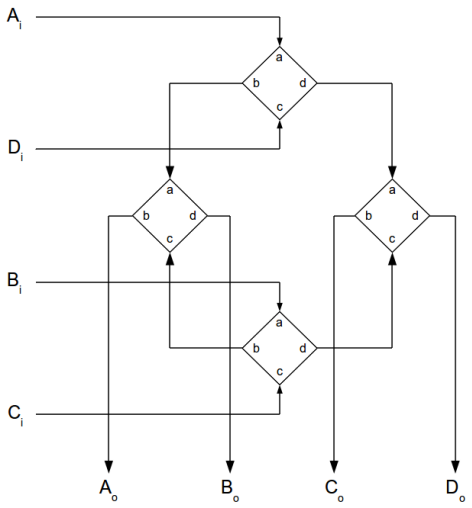

Figure 23: Crossbar switch

There are 16 switch position combinations possible. The chosen switching pattern consists of 4 switch combinations: 0, 9, 6, 15, respectively. Channel mapping is detailed in Table 14.

Table 14: Switching pattern

Switch position / Input channel	A	B	C	D
0	D	A	B	C
9	A	D	C	B
6	C	B	A	D
15	B	C	D	A

When enabled, the switching introduces glitches in the turn-by-turn data. Glitches are minimized by the phase compensation algorithm and by using the notch filter. They can be filtered out further by using spike removal functionality (Chapter 3.3.4).
Switching can be turned ON (true) or OFF (false, default setting). The setting is done with the following command:

```
libera-ireg boards.bpm.conditioning.switching=true
```

3.3.2.1 Switching clock source

The switching period/frequency originates from the internal oscillator (VCXO) by default. The VCXO is controlled by the SW PLL. The crossbar switch is linked to the VCXO (“internal” switching source). Alternatively, it is possible to link the switching clock to the MC directly (“external” switching source).

The crossbar switch changes position on switching source clock rising edge. To minimize the glitches due to switching, the switch timing can be delayed by several ADC samples. This way the switching can be moved to an “empty” part of the fill pattern (if there is any).
However, if the offset tune is used, the VCXO and consequently the switching frequency is changed therefore it is not possible to set a constant switching delay according to the fill pattern. Switching frequency is not an integer sub-harmonic of the revolution frequency. In this case, the external switching source clock should be used.

NOTE: If switching source is set to external and a valid MC is not connected, the switching and DSC will not work.

Switching source is set in registry node:

```
libera-ireg boards.bpm.conf.switching_source=External
libera-ireg boards.bpm.conf.switching_source=Internal
```

When in partial fill operation, the switching delay parameter can be adjusted to move the switching time to an “empty” part of the turn. See Figure 24.

The switching delay is set in registry node:

```
libera-ireg boards.bpm.conf.switching_delay
```

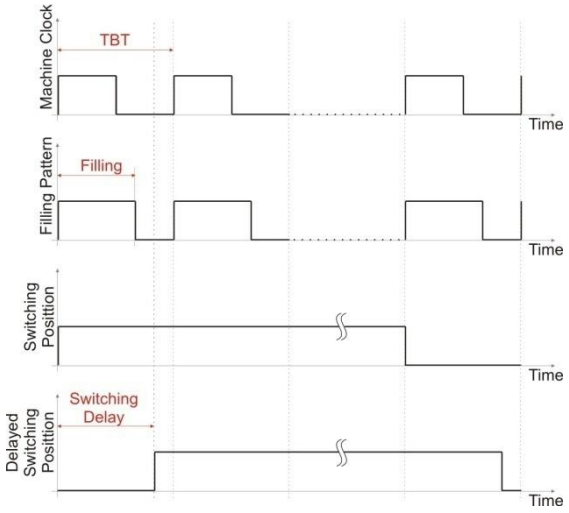

Figure 24: *The switching delay functionality.*

3.3.2.2 Notch filter

The turn-by-turn data is filtered and decimated through a FIR filter with bandwidth up to approximately 2 kHz (at -3 dB). Additional dedicated filter (notch type) is placed at a sub-harmonic of the switching frequency (at ~3.3 kHz). Switching artefacts are effectively removed, see Figure 25.

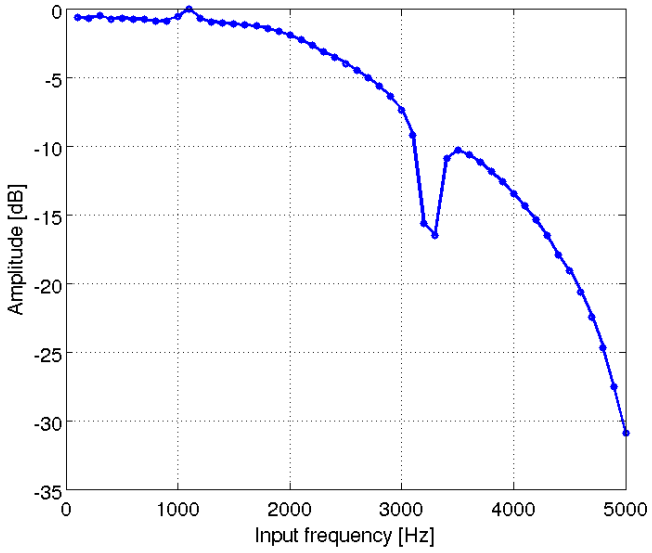

Figure 25: *Example polyphase FIR filter with notch filter added*

FA polyphase FIR and FA notch filter coefficients are stored in configuration files (see Table 15).

Table 15: FA notch and FIR coefficients.

Parameter name	Description	Full path to the file
FA notch filter coefficients	The setting of notch coefficients consists of five parameters. Note that in Libera Brilliance+ two notch filters per each BPM module are implemented.	`/var/opt/libera/cfg/libera-ebpm_sr.xml` (if used in the storage ring) `/var/opt/libera/cfg/libera-ebpm_bo.xml` (if used in the booster ring)
FA polyphase FIR coefficients	These settings are essential for proper operation of Libera Brilliance+, so be careful when changing them. The number of coefficients must be equal to the number of filter taps.	`/var/opt/libera/cfg/fircoef_sr.conf` (if used in the storage ring) `/var/opt/libera/cfg/fircoef_bo.conf` (if used in the booster ring)

3.3.2.3 Switching frequency

The default switching frequency is chosen for every accelerator design individually and is in range of approximately 13 kHz. For special purposes, the switching frequency can be modified. The switching frequency is defined:

$$f_{sw} = \frac{MC}{SW\_DEC}$$

The switching decimation (SW_DEC) corresponds to the number of turns between two switch rotations. Based on experience, the SW_DEC must be equal or higher than 10. If there are less than 10 turns between two switch rotations, the spike removal and DSC may not have sufficient data for correct operation.

NOTE: The switching introduces strong frequency components to the frequency spectrum of the turn-by-turn data. The notch filters are calculated for default switching frequency components and are applied to the FA data only. If the switching frequency is modified, the original notch filters may not eliminate the switching frequency components properly.

The switching decimation is set for all BPM modules in the crate. The setting procedure:

1. Create the custom configuration file:

```
root@libera:~# libera-ireg system.persistence.save{}
system.persistence.save{}: done
```

2. Stop the libera-ebpm application:

```
root@libera:~# /etc/init.d/libera stop
```

3. Edit the custom configuration file and modify the switching decimation

```
root@libera:~# nano /var/opt/libera/cfg/libera-ebpm.xml
```

The parameter is located in the .xml structure: application → clock_info → decimation → sw . See example where the switching decimation is set to 20 (row " `<sw value="20"/>` ".

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noName ...
  <application>
    <clock info>
      <decimation>
        <sw value="20"/>
     ...
```

Save the file and exit.

4. Start the libera-ebpm application

```
root@libera:~# /etc/init.d/libera start
```

The DSC adapts to the new switching decimation automatically.

3.3.2.4 External switching matrix (Libera XBS FE)

The 3rd generation of the BPM module (production from end 2022) supports the Libera XBS FE. It is functionally equivalent to internal crossbar switches but it can be installed closer to the BPM pickup. The RF cables from the Libera XBS FE to the BPM module become part of the analog processing chain and disturbances get compensated by the DSC. The only non-compensated path remaining are the cables between the BPM pickup and the Libera XBS FE.

Registry nodes that control the crossbar switch:

```
root@libera:~# libera-ireg dump boards.kraf3.crossbar_switch.
crossbar_switch
  current=0.0136875
  source=External
  current ok=false
  power supply fault=false
  external_switch_led=Off
  current_thr_low=0.09
  current_thr_high=0.12
```

The crossbar switch source:

- Internal: uses the on-board crossbar switch
- External: uses the external switching module (Libera XBS FE)

The Libera XBS FE is connected with the BPM module via standard Cat.7 S/FTP cable and RJ-45 connectors (see Figure 26). The BPM module provides the power supply and switching control lines to the Libera XBS FE. Pinout is presented in Figure 26 and represents:

- 1,2,7,8: control pins
- 3,6: ground
- 4,5: power

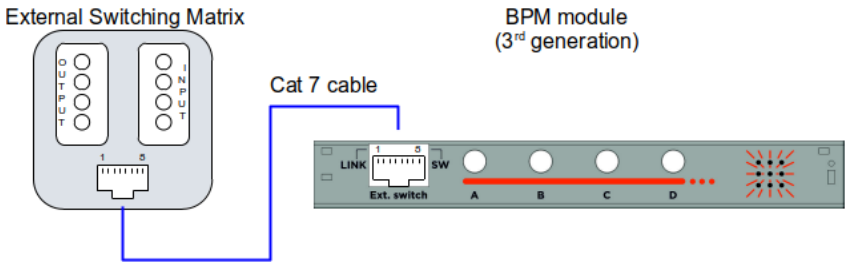

Figure 26: Connection to the Libera XBS FE.

For stable operation, Cat.7 S/FTP cable is recommended. It has a better shielding and lower voltage drop due to lower resistance over longer distance compared to Cat.5.

Once the cable is connected, the user must set the crossbar switch source to »External«. Then, the internal crossbar switch is stopped/bypassed automatically. The power supply current for the Libera XBS FE gets monitored and the BPM module checks whether the current consumption is within expected boundaries. Based on consumption, the »LINK« LED indicates power OK (solid state) or current out of boundaries (blinking state). The »SW« LED is lit when the switching is active. See Table 16 for details.

Table 16: Status LEDs for Libera XBS FE.

LED	solid	blinking	Off
LINK (green)	OK	UTP cable or external switching module issue	Power OFF
SW (yellow)	External switching active		External switching not active

The software interface provides a current readout and option to set the low and high current thresholds which are later considered for reporting the power status. The current consumption varies with Cat.7 cable length. Typical current consumptions are presented in Table 17.

Table 17: Current consumption of the Libera XBS FE.

Cat.7 S/FTP cable length	Switching enabled	Switching disabled
10 m	114 mA	108 mA
100 m	104 mA	98 mA
205 m	91 mA	85 mA

The change in current consumption value can indicate a possible failure in the Cat.7 cable. Interruption (far end termination not seen by the driver) in one control pair results in approximately 25 mA lower current consumption. If in shortcut, the consumption exceeds the threshold. Current on the BPM module is limited to 125 mA by internal current limiting circuit.

Current readout and low/high thresholds (all values in A) are accessible in registry nodes (bpm=kraf3, kraf4, kraf5, kraf6):

```
root@libera:~# libera-ireg dump boards.bpm.crossbar_switch.current
current=0.014075
root@libera:~# libera-ireg dump boards.bpm.crossbar switch.current thr low
current thr low=0.09
root@libera:~# libera-ireg dump boards.bpm.crossbar_switch.current_thr_high
current_thr_high=0.12
```

The RF signal path from the internal crossbar switches to the A/D converters has a fixed length. With external switching module, the signal path is variable and can be different from module to module. The parameter that adjusts the delay from analog to digital domains is then a combination of a fixed delay (in the BPM module) and additional delay introduced by the RF and UTP cables.

Example: Testing setup

- Short RF cable, length L1=0,5 m
- Cat.7 cable, length L2=[10, 100, 205] m

Delays:

- fixed delay (3rd generation BPM module): 350 ns
- fixed delay (Libera XBS FE): 60 ns
- the RF: 4.17 ns/m
- the Cat.7 cable ~4.6 ns/m

The total delay:

350 ns + 60 ns + (0.5 m * 4.17 ns/m) + (100 m * 4.6 ns/m) = ~ 872 ns

The sampling clock is accelerator dependent. In our test case, the fs = ~ 109 MHz. The delay, expressed in the sampling clock units:

$$812\ ns * fs\ [1/s] = (8{,}12*10^{-9})\ [s] * (109*10^{6})\ [1/s] = 95\ units$$

For optimal performance, the parameter needs to be adjusted for the actual installation. Before the tuning procedure starts, it is important to enable the switching.

Tuning procedure: Connect the CW signal to the remote switching module to channels A, C and D and keep channel B disconnected. Set DSC coefficients to unity. Monitor the B amplitude in the SA. Adjust the parameter to have the lowest amplitude in channel B:

```
root@libera:~# libera-ireg boards.bpm.conf.analog to dsp delay
boards.bpm.conf.analog_to_dsp_delay: 95
```

Example results (measured) for different Cat.7 cable lengths and fixed RF cable length (0,5 m):

- 10 m: 52
- 100 m: 97
- 205 m: 152

NOTE: Make sure to revert the parameter to its default value (25 or 38) when using internal crossbar switch.

3.3.3 Digital signal conditioning (DSC)

Digital signal conditioning is a software calibration method that improves the Libera Brilliance+ longterm readout performance. It improves the long-term stability as it minimizes the measured position drift, mainly due to temperature variation.

NOTE: DSC is active only if the switching is enabled. The DSC requires the MC PLL locked.

3.3.3.1 The principle

According to the switching pattern (Table 14), each input signal passes through each channel in the BPM module. The switching pattern runs continuously with a switching frequency (around 13 kHz). There are fixed number of turn-by-turn data points between two switch positions. This is called a “switching decimation” parameter.
The DSC reads fresh raw turn-by-turn amplitudes for every DSC period. The buffer length required for the DSC calculations must cover several switching patterns. Default buffer length is described as:

$$DSC_{buffer} = SWdec * pattern\_size * num\_of\_patterns + SWdec * pattern\_size$$

Parameters of the DSC buffer are described in Table 18.

Table 18: Parameters for DSC buffer length.

Parameter	Value	Description
SWdec	Accelerator dependent	Switching decimation, number of turns between 2 switch positions
Pattern size	4	Number of different switch positions (it is fixed)
Number of patterns	8	Number of consecutive switching patterns. Larger number ensures better reliability.

The first position in the switching pattern contains a switching marker. It marks the LSB in the I_A amplitude for the whole switch position. If switching marker is not found, the current DSC cycle is aborted.

It is possible to switch between the DDC and TDP data sources for the FA and SA data. This selection has an impact to behavior of the DSC algorithms:

- When the DDC source is selected, the DSC uses the amplitudes processed from the DDC (frequency domain) processing block. The I & Q amplitudes contain amplitude and phase information.
- When the TDP source is selected, the DSC uses the amplitudes processed from the TDP (time domain) processing block. The TDP amplitudes are useful to calculate optimum amplitude compensation coefficients. When in this mode, the phase compensation must be disabled manually (see Table 19).

Table 19: DSC setting for the TDP and DDC data sources.

Data source for FA & SA	DSC setting

DDC
`boards.bpm.tbt.data_type=DDC`
	Amplitude and phase compensations enabled (default set to 8 patterns)
	`boards.bpm.conditioning.tuning.dsc.averaging_gain_patterns=8`
	`boards.bpm.conditioning.tuning.dsc.averaging_phase_patterns=8`

TDP `boards.bpm.tbt.data_type=TDP`	Phase compensation disabled (must be set to 0 patterns) `boards.bpm.conditioning.tuning.dsc.averaging_phase_patterns=0`

* *bpm* = kraf3, kraf4, kraf5, kraf6

NOTE: DSC coefficients are not applied to the raw ADC data.

3.3.3.2 Amplitude compensation algorithm

The amplitude compensation algorithm is used to apply proper gains to the input channels to 4 switch positions so that the overall gain is equalized. This suppresses the effect of aliasing, which scales with gain difference between individual transfer functions. 20% of the samples (turns) at start and end of the switch position are excluded from the calculation algorithm to avoid glitches between two switch positions. This value is hardcoded.

In the first step, the algorithm calculates the sum of 8 average amplitude-sums (8 is the number of patterns). Then, it calculates the geometric average per channel and looks for the minimum geometric average amplitude. If the minimum geometric average amplitude is lower than the threshold, the current DSC cycle is aborted.

The new amplitude correction coefficients are a ratio between the geometric average of the channel and the sum of average amplitudes per switch position per channel. Result is a 4x4 matrix (= switch positions x channels). Before applying the new coefficients, they are normalized with geometric average of coefficients over all switch positions. The new coefficients are multiplied with existing ones.

User can continuously monitor the amplitude coefficients in the registry tree. This is done with the following command:

```
libera-ireg dump
boards.bpm.conditioning.coefficients.{channel_0,channel_1,channel_2,channel_3}.gain
gain=1.0000028328263335
  pattern 0=0.99870426656068356
  pattern_1=0.99719356124273639
  pattern_2=1.0035834486484914
  pattern_3=1.0005300548534222
gain=1.0000066721663932
  pattern 0=1.0047113561711594
  pattern 1=1.0022918265010949
  pattern_2=0.997406672028189
  pattern_3=0.99561683396512979
gain=1.0000083419351908
  pattern 0=1.004149427977566
  pattern 1=1.0032745564498531
  pattern 2=0.99869247184585597
  pattern_3=0.99391691146748873
gain=1.0000022132843498
  pattern 0=1.0015174458732006
  pattern 1=0.99638233745968663
  pattern_2=1.0012340310074652
  pattern_3=1.0008750387970466
...
```

3.3.3.3 Phase compensation algorithm

The phase compensation algorithm compensates the differences in phases between signals, regardless where the differences originate from – different analogue data paths or outside of the BPM module. The algorithm is executed in the same loop as the amplitude compensation algorithm and both use same input data.
In the first step, the algorithm calculates angles of each sample over the DSC buffer. It normalizes the angles and calculates the average angle per sample.
For further calculation, it only takes the first switching pattern (4 switch positions) after the switching marker. 20 % of the samples (turns) at start and end of the switch position are excluded from the calculation algorithm to avoid glitches between two switch positions.

The algorithm then generates the array of synthetically generated angles (linearly increased by average angle). Channel A is selected as a reference channel whose angles are compared to the synthetic values. Channels B, C and D are compared to Channel A. Results are normalized to ensure all differences are within ¶ (3.14) range.
Resulting values are subtracted from the current values (coefficients) and again normalized to ensure they are within ¶ (3.14) range.
User can continuously monitor the phase coefficients in the registry tree. This is done with the following command:

```
libera-ireg dump
boards.bpm.conditioning.coefficients.{channel_0,channel_1,channel_2,channel_3}.phase
```

NOTE: When using the TDP data source for the FA & SA data, the DSC automatically uses the TDP turn-by-turn data for amplitude compensation. In this case, the phase compensation must be disabled.

3.3.3.4 Modes of operation

The amplitude and phase compensation coefficients are being calculated continuously, see Chapter 3.3.3.5 for details. This process is called learning or adjusting and it can be enabled (true, default setting) or disabled (false). The setting is done with the following command:

```
libera-ireg boards.bpm.conditioning.tuning.dsc.coefficients.adjust=true
```

The signal processing can use either initial (unity) coefficients' value (0 for phase, 1 for amplitude) or adjusted values (default setting). Selection is done with the following command:

```
libera-ireg boards.bpm.conditioning.tuning.dsc.coefficients.type=adjusted
```

Possible combinations of the DSC parameter settings are presented in Table 20. Combinations that are most frequently used are printed in **bold** letters.

Table 20: Typical DSC operation modes.

	Switching: true		Switching: false	
	Adjust: true	**Adjust: false**	**Adjust: true**	**Adjust: false**
Type: unity	Switches rotate Coeff. being calculated Unity coeff. applied	**Switches rotate** **Coefficients not being calculated** **Unity coefficients applied**	Switches stopped Coefficients not being calculated Unity coefficients applied	**Switches stopped** **Coefficients not being calculated** **Unity coefficients applied**
Type: adjusted	**Switches rotate** **Coefficients being calculated** **Calculated coefficients applied**	Switches rotate Coefficients not being calculated Last good calculated coefficients applied	**Switches stopped** **Coefficients not being calculated** **Last good calculated coefficients applied**	Switches stopped Coefficients not being calculated Last good calculated coefficients applied

3.3.3.5 Learning cycle

The DSC cycle is summarized in Figure 27. The DSC cycle runs in 10-second period. The DSC buffer contains typically 8 switching patterns of turn-by-turn data. The input data is checked for quality (see Chapter 3.3.3.6 for details). If input data is considered bad, the DSC cycle is canceled. Good data is used for the amplitude and phase compensation algorithms. Fresh set of coefficients is then updated to the FPGA and written to the internal table which connects the DSC coefficients to power levels. It is possible to keep DSC coefficients linked to specific power levels or update the complete table with lastly calculated (most recent) DSC coefficients. Selection is shown in the “att dependent” block in the scheme.

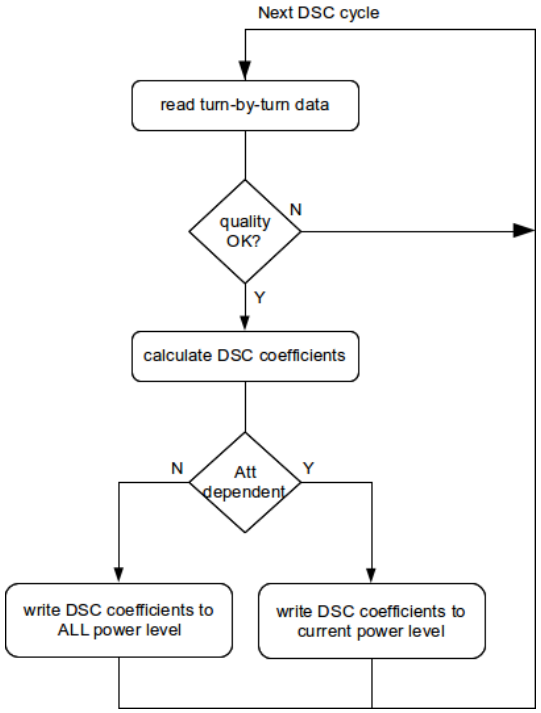

Figure 27: DSC block scheme.

Storage and handling of the DSC coefficients is shown in Figure 28. It can be done in two different modes:

- attenuation (power level) dependent mode
- attenuation (power level) independent mode

When working in attenuation dependent mode, the DSC coefficients had to be calculated and stored for each power level independently. In practice this means, that the full adjustment process shall be done for the whole gain table (for every power level value) and the process is time consuming. The ideal “setup” is to leave the beam to decay slowly with AGC and DSC enabled. The AGC will adjust the power level according to the beam intensity and will eventually cover the complete dynamic range. A faster way is to use an RF signal generator and manually change the power level. It is good to let the DSC run at least 2 cycles (20-25 seconds).
A drawback of this mode is explained from Figure 28. Coefficients in power level -9 have not been calculated yet (they are unity). When power level changes (either manually or by the AGC) from -9 to -10, a jump in position will be noticed. A power level could also contain coefficients that were calculated long ago (example marked as dark orange in Figure 28). When power level changes from -26 to -27, a jump can occur.

Attenuation independent mode addresses the upper mentioned drawbacks. The coefficients will be applied to all power levels at every DSC cycle. This way, jumps due to unity or old coefficients in one particular power level are avoided.

The attenuation dependent mode can be controlled with following commands:

```
libera-ireg boards.bpm.conditioning.tuning.dsc.coefficients.att_dependent=true
libera-ireg boards.bpm.conditioning.tuning.dsc.coefficients.att_dependent=false
```

After hardware or software restart, sets of coefficients are erased.

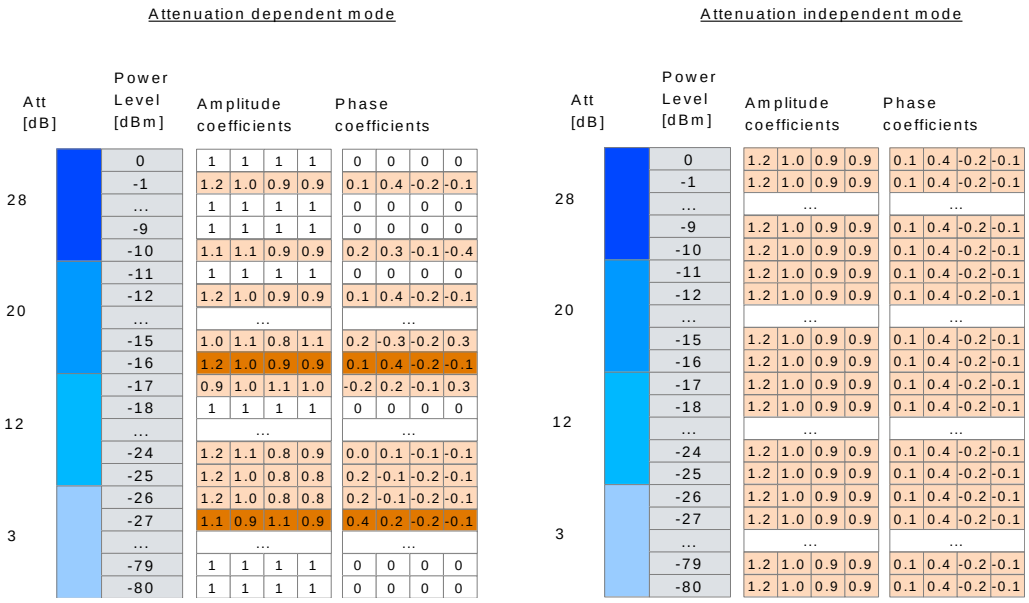

Figure 28: Relation between the DSC coefficients and gain sheme.

3.3.3.6 Input data quality estimation

Quality of calculated coefficients depends on the input data quality. Therefore, if the S/N ratio of the input data is too small, the coefficients' for the DSC cycle do not update. The S/N ratio is low in these typical conditions:

- No beam
- At very low beam current
- During sudden beam loss or abnormal event
- RF and revolution frequencies are not synchronous (typically in the laboratory environment)

The purpose of the functionality is to discard the data that has been captured exactly during beam loss or other abnormal beam behavior and would potentially result in inadequate DSC coefficients.

NOTE: Make sure to enable the compensate tune parameter when using the offset tune different than zero. See Chapter 3.7.5. Even a slight detune between I and Q amplitudes raises the variance and DSC coefficients will not update.

The DSC functionality implements two independent checks on the input data for calculation of new coefficients:

- The learning limit, expressed in ADC counts
- The amplitude variation, expressed in %

The ADC learning limit is set to 400 ADC counts by default. If needed, the limit can be changed. Please contact support@i-tech.si for more instructions.

The amplitude variation check is defined by three parameters, which are accessible as read-only in the registry tree:

```
libera-ireg dump boards.bpm.conditioning.tuning.dsc.quality
quality
  selector=Raw
  tolerance_thr=2
  tolerance_curr=3.9827095724238308
  recent_failures=2
```

Description of parameters:

- selector: Indicates whether the functionality is disabled (Off), uses synthetic amplitudes (Amp) or I/Q amplitudes (Raw) for variation calculation. Default is set to Raw. Parameter is read-only.
- tolerance_thr: Defines the allowed variance [%]. Default is 2.
- tolerance_curr: reports the calculated variance from the last DSC cycle [%]. If it exceeds the “tolerance_thr” new coefficients will not be calculated.
- recent_failures: number of most recent consecutive unsuccessful DSC cycles

The functionality is important for DSC operation. Due to this, the “selector” parameter can not be changed while the application daemon is running (libera-ebpm). To change the “selector” setting, stop the libera-ebpm daemon first, then change the setting in the /var/opt/libera/cfg/libera-ebpm.xml file. Start the libera-ebpm daemon.

3.3.3.7 Storing / loading the DSC coefficients

Storing and loading feature has been kept for backward compatibility with initial software releases. Due to latest update of the DSC functionality (from version 2.8 on) storing the DSC coefficients is not advised.

3.3.4 ADC offset compensation

A/D converters have a DC offset at no input signals (inputs terminated). This offset is called an ADC offset and has no influence to turn-by-turn processing in frequency domain. The offset, however, affects the accuracy of position calculation in time domain.
ADC offset is adjustable for each channel independently:

$$A' = A_{raw} - A_{offset}$$

Offset are accessible through registry node:

```
libera-ireg dump boards.bpm.conf.adc_offset
adc offset
  A=0
  B=0
  C=0
  D=0
```

ADC buffer contains raw data with no compensation applied. Compensation is applied to the processed ADC data just before the turn-by-turn processing blocks (see Figure 12 and Figure 30). Compensated data can be read from the TBT window buffer.

Figure 29 shows an example for setting the offset on one channel. Upper plot shows raw value (non compensated), lower plot shows another data acquisition where the ADC offset was compensated.

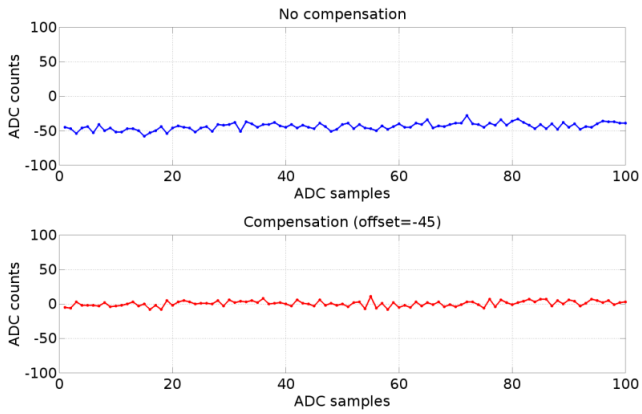

Figure 29: ADC offset compensation.

Depending on the BPM module type, the ADC offset is applied:

- RAF module type: ADC offset applied to the data in the raw ADC buffer already
- KRAF module type: ADC offset applied to the TBT window data

3.3.5 Channel to channel static gain compensation

Due to components' tolerances channel to channel gain may vary up to 1-2 dB. During normal operation (top-up, decay mode), the DSC compensates for the amplitude and phase differences automatically. When the instruments are used to measure trajectory of the first few turns, the crossbar switch and DSC can not be used. In such conditions, the channel to channel gain variation affects the accuracy of measured position.

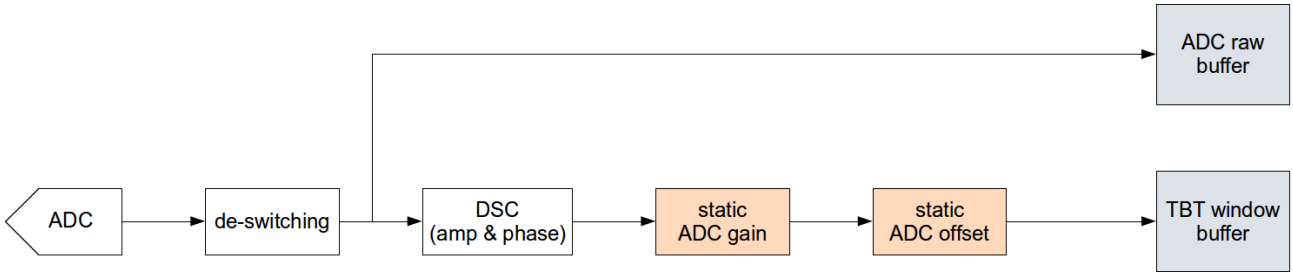

Figure 30: Static ADC gain and offset compensation.

Channel gains are measured during the FAT. The variable attenuator is set to 0 dB. Gain variation is calculated on the channels' standard deviation values.
Gain compensation is applied to each channel individually and is valid only for the attenuation setting where the gain variation was measured. This is a static compensation. Coefficients can be set within $0 \leq k < 2$ range.

Channel gains are accessible through registry node (*bpm* = kraf3, kraf4, kraf5, kraf6):

```
libera-ireg dump boards.bpm.conf.adc_gain
adc_gain
  A=1
  B=1
  C=1
  D=1
```

Gain compensation is applied after the DSC block. Its effect is visible in the TBT window buffer (see Figure 30).

NOTE: When using the crossbar switch and the DSC, set all gain compensation values to 1.

3.3.6 Spike removal functionality

Switching introduces a glitch in the turn-by-turn amplitudes each time the crossbar switch changes the switch position. Glitches are periodical with the switching frequency (~13 kHz). Number of data points (turns) between two glitches varies between different accelerator designs (e.g. 100 samples for accelerator with 1.3 MHz revolution frequency). A glitch affects between 2-4 data points.

To eliminate the spikes, there are two algorithms implemented (see Figure 33 for parameter notations):

- Average: it calculates the average amplitude over the 'average window' and applies the values over the spike ('window').
- Slope: it calculates the average gradient over the 'average window' and applies the gradient values over the spike ('window')

Algorithm selection is controlled by registry node (example for kraf3):

```
libera-ireg boards.kraf3.tbt.spike_removal.mode.type=0
libera-ireg boards.kraf3.tbt.spike_removal.mode.type=1
```

Use "0" for "Average" type or "1" for "Slope" type. Comparison between the algorithms is given in Figure 31 and Figure 32. The black line represents the raw amplitude containing the switching spikes. The red line represents the values corrected by the "Average" algorithm. The blue line represents the values corrected by the "Slope" algorithm. Note the difference in Figure 32, where the "Slope" algorithm corrects the values much better than the "Average" algorithm.

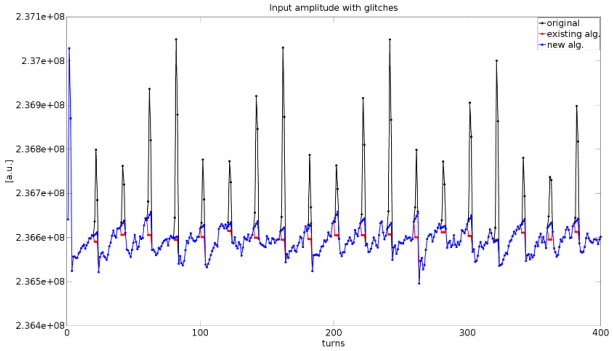

Figure 31: Spike removal algorithm comparison (on static amplitude).

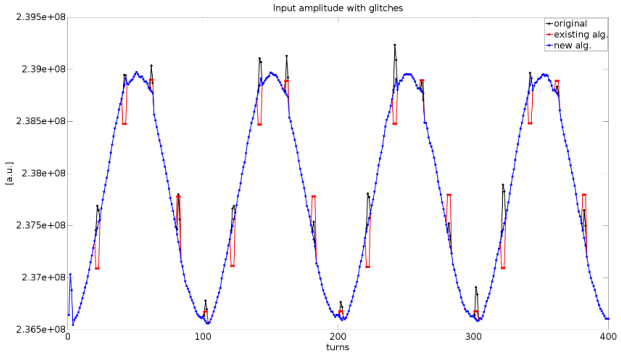

Figure 32: Spike removal algorithm comparison (on dynamic amplitude) .

The functionality is implemented in two levels:

- in FPGA (“hw”): works on the FDS, FA and SA data streams
- in software (“sw”): works on the DDC and TDP turn-by-turn data buffers

The raw DDC buffer (I and Q amplitudes) contains original data.

NOTE: In DDC synthetic and TDP buffers, the first spike may not be suppressed.

Adjustment for averaging/gradient and glitch window is done with four parameters that are shown in Figure 33. These parameters’ values are set with respect to the switching marker position. Switching marker is a status bit that indicates the exact point of the switch position change. The “averaging_stop” parameter defines the point up to which the samples for calculating the average are taken. The number of samples is defined by the “averaging_window” parameter. The parameter “start” defines the point where the application of averaged samples starts. The number of synthetic samples is set with the “window” parameter.
Example commands:

```
libera-ireg boards.bpm.tbt.spike_removal.averaging_stop=-1
libera-ireg boards.bpm.tbt.spike_removal.averaging_window=8
libera-ireg boards.bpm.tbt.spike_removal.start=-2
libera-ireg boards.bpm.tbt.spike_removal.window=6
```

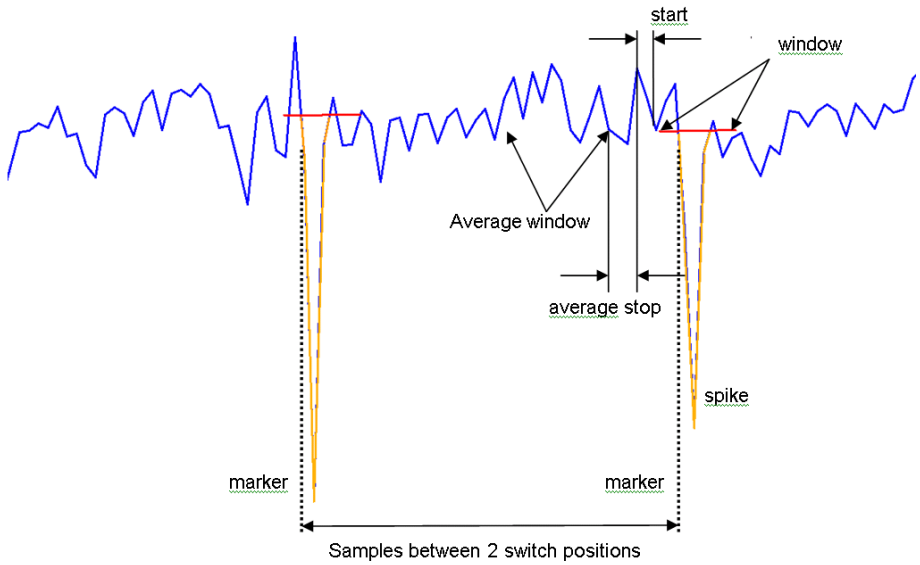

Figure 33: *Spike removal functionality*

Functionality is enabled/disabled for both levels independently. Example command:

```
libera-ireg boards.bpm.tbt.spike_removal.mode.hw_enable=false
libera-ireg boards.bpm.tbt.spike_removal.mode.sw_enable=false
libera-ireg boards.bpm.tbt.spike_removal.mode.hw_enable=true
libera-ireg boards.bpm.tbt.spike_removal.mode.sw_enable=true
```

Default parameters' values are:

- hw_enable = true
- sw_enable = true
- type = 0
- averaging_stop = -1
- averaging_window = 8
- start = -2
- window = 6

The current settings can be checked with the following command (example for kraf3):

```
libera-ireg dump boards.kraf3.tbt.spike removal
spike_removal
  mode
    hw_enable = true
    sw enable = true
    type=0
  averaging stop = -1
  averaging_window = 8
  start = -2
  window = 6
```

Same parameters are used for both the FPGA and software implementation. Default parameters are roughly calculated during design phase. They must be optimized when installed to operating environment and beam signals.

NOTE: Because the calculation duration is different for turn-by-turn DDC and TDP data, spikes' position (regarding the marker) is not the same in both data types. Therefore, the user should adjust the start and averaging_stop parameters when switching between these two data types.

3.3.7 Signal conditioning usage examples

This chapter explains the usage of switching and DSC.
Left plot in Figure 34 shows position reading with switching disabled and unity amplitude and phase coefficients. Then, the switching was enabled (right plot in Figure 34). 4 switch positions are clearly seen. The RMS value in this case is even not relevant.
Figure 35 shows a comparison between unity and adjusted coefficients (switching is always enabled). Position reading with adjusted DSC coefficients has similar RMS value as with switching disabled.

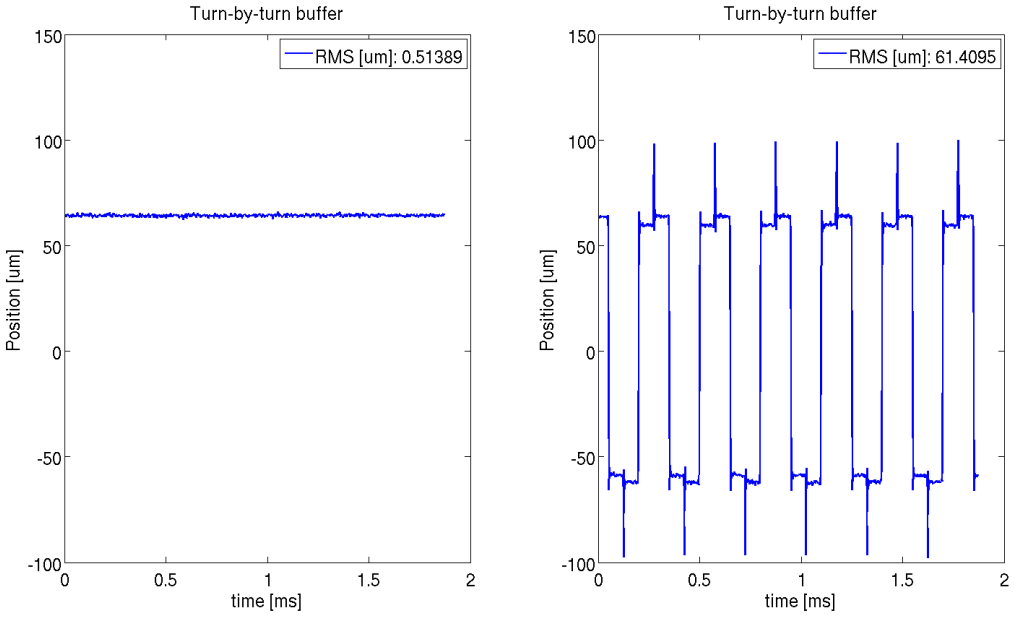

Figure 34: Switching enabled / disabled

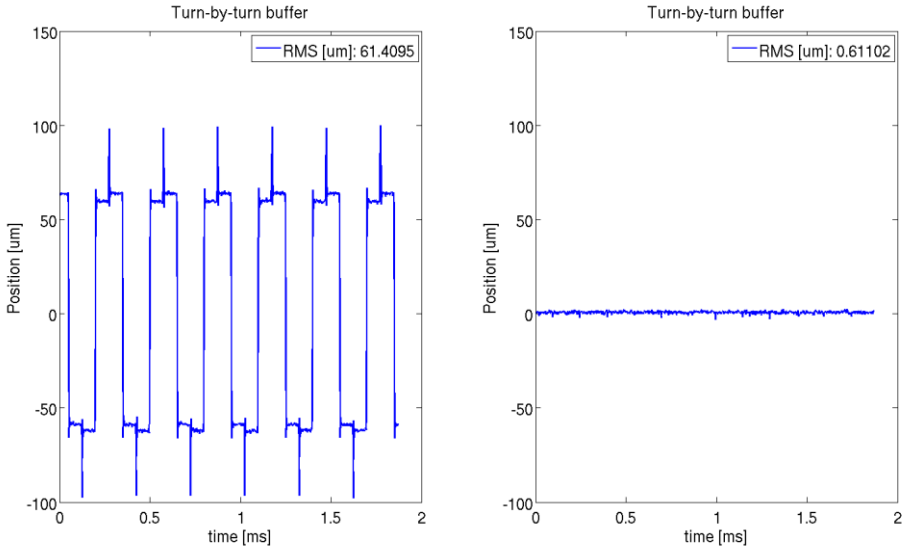

Figure 35: *Unity / adjusted DSC coefficients applied*

Switching and DSC are used to improve the long-term drift and beam current dependence performance of the position readout. Both functions must be enabled during steady conditions (e.g. top-up mode, decay mode).
Due to switching artefacts, the switching can be disabled for short term measurements, such as first-turn measurement, turn-by-turn analyses, injection transient analyses, etc. To achieve best results, follow these steps:

- Let DSC to adjust coefficients
- Disable the switching but keep ‘adjusted’ DSC coefficients

NOTE: Position changes when the switching is enabled. This is due to inherent calibration of the switching method.

3.4 Data paths

There are 6 general data paths available to the Control system as shown in Table 21. All data is accessible in parallel. Data paths differ in the sampling/data rate and bandwidth. Total memory buffer in each BPM module is 4 GB. It is split to 4 parts, each dedicated for buffered data (see Figure 36). The ADC buffer is further split to several segments. Each segment is written with fresh ADC data on trigger. Other buffers are circular-type. Atom sizes for each data path are shown in Table 21.

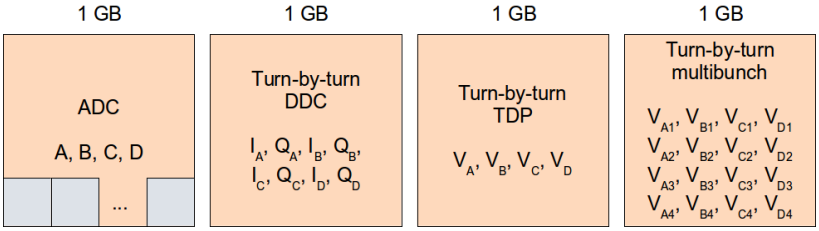

Figure 36: Memory allocation for data paths.

Table 21: Data paths.

Data path	Type	Buffer size	Atom size	Data rate	Bandwidth
ADC raw data	Buffered	1 GB	4x 2 Byte	ADC freq.	~ 15 MHz
Turn-by-turn (DDC)	Buffered	1 GB	8x 4 Byte*	rev.freq.	~ 0.35 × rev.freq.
Turn-by-turn (TDP)	Buffered	1 GB	4x 4 Byte*	rev.freq.	0.5 × rev.freq.
Turn-by-turn (multibunch)	Buffered	1 GB	16x 4 Byte	rev.freq.	0.5 × rev.freq.
FDS**	Stream		16x 4 Byte	rev.freq.	(~ 0.35 – 0.5) × rev.freq.
Fast Acquisition data	Stream		16x 4 Byte	10-30 kS/s	~ 2 kHz
Slow Acquisition data	Stream		16x 4 Byte	10-20 S/s	~ 4 Hz

* actual buffer allocation in the memory
** available in select designs only

Properties of each data path are available in the signal nodes. Example for the synthetic turn-by-turn:

```
libera-ireg dump boards.bpm.signals.ddc synthetic
ddc synthetic
  access_type=DataOnDemand
  atom_size=32
  group_size=1
  components names=[Va,Vb,Vc,Vd,Sum,Q,X,Y]
  components_number=8
  components_type=int32
```

Explanation:

```
access type           buffered data (DataOnDemand)
atom_size             32 [bytes]
group_size            1 [minimum number of atoms to read at a read request]
component_names       Column names
components number     Number of columns
components_type       type of columns' values
```

3.4.1 Data acquisition from buffers and streams

Signals from buffers and streams can be read with libera-ireg utility:

```
libera-ireg signal boards.module.signal.signal_name <behaviour> <size> <offset> <verbose>
```

where:
module … selected module (evrx2, kraf3, kraf4, kraf5, kraf6, gdx1)
signal_name … selected signal (e.g. adc, sa, pll)
< optional switches are listed in Table 22 >

Some exceptions apply (when signals are implemented in another registry tree node – e.g. tbt_window).
There are several acquisition options with 'libera-ireg' client. Buffers can be acquired on-trigger, with offset, user can define the buffer length and much more. All options are listed in Table 22. Some options are not available to all data paths.

Table 22: *Acquisition options*

Option	Description	ADC	TBT	MBTBT	FA	SA
-b	behavior: it specifies when to read the data (Event | Now | Lmt). Default is 'Now'. Event: acquisition on Trigger (external) Now: immediately Lmt: at specific LMT (rarely used)	YES	YES	YES	NO	NO
-s	size: specifies the number of samples to be acquired	YES	YES	YES	YES	YES
-o	offset: sets the offset between the trigger and the start of the acquisition. It is set in MT and can be + or –.	NO	YES	YES	NO	NO
-v	verbose: prints out the header line with columns description. The header line contains also timestamps (LMT, MT) of the acquisition.	YES	YES	YES	YES	YES

Turn by turn data can be read out from the buffer with a user-defined offset. The offset value is set with acquisition command (-o) and is expressed in turns. Use only reasonable values.

The MT timestamp originates from the first sample. If no offset is set, the MT timestamp matches the read-out event timestamp (e.g. trigger timestamp). If positive offset is used, the buffer will be filled with the delay of "offset" number of samples in the future. If negative offset is used, existing data from the buffer will be used. See Figure 37 for all three options.

Postmortem buffer shall not be read with an offset at read request. To apply an offset to the Postmortem data, use a dedicated parameter in the Postmortem functionality.

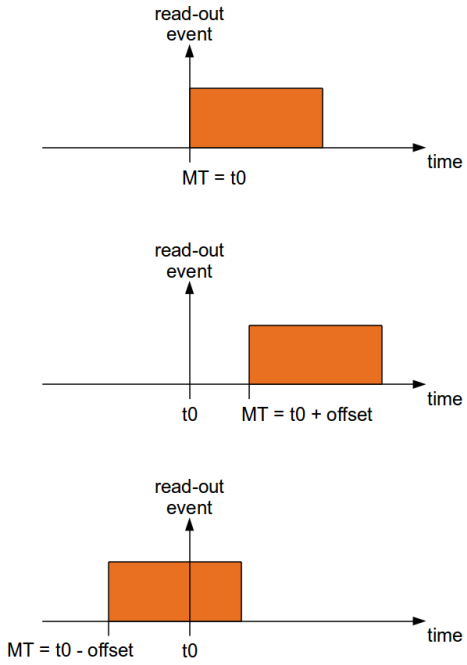

Figure 37: Turn by turn data readout with offset.

3.4.2 ADC data (raw data)

The ADC data is raw and untreated data coming from the A/D converters. The sampling frequency is accelerator dependent but typically in range from 100 MHz to 125 MHz. Data is stored in the segmented buffer where every segment is filled on a trigger. A/D resolution is ±15 bits (±32767 ADC counts).
The measurement bandwidth of the data is approximately 15 MHz. It is determined by two band pass filters on each RF chain. The center frequency of the band pass filters is the RF frequency (e.g. 352 MHz or 500 MHz).
Exact sampling frequency in [Hz] is reported in registry node:

```
boards.bpm.clock_info.adc_frequency
```

Example of ADC data read request:

```
libera-ireg signal boards.bpm.signals.adc -s10 -bEvent -v
```

NOTE: A trigger is required to read the ADC data.

3.4.3 Turn-by-turn data – DDC

The Turn-by-Turn (DDC) data is processed precisely at the accelerator's revolution frequency. Exact sampling frequency in [Hz] is reported in registry node:

```
boards.bpm.clock_info.tbt_frequency
```

The turn-by-turn buffer is a circular buffer which is continuously filled with raw I and Q amplitudes. The data is available in raw and processed format:

- ddc_raw: provides I and Q amplitudes
- ddc_synthetic: provides processed values (V_A, V_B, V_C, V_D, SUM, Q, X, Y) calculated from the I and Q amplitudes, see Figure 38.

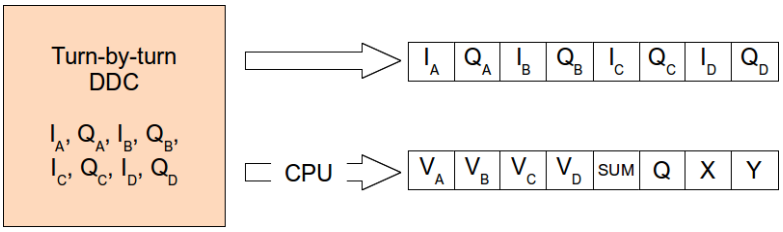

Figure 38: Raw and synthetic data from the turn-by-turn (DDC) buffer.

Example of read request for both data types:

```
libera-ireg signal boards.bpm.signals.ddc_raw -s10 -v
libera-ireg signal boards.bpm.signals.ddc_synthetic -s10 -v
```

Data can be read from the buffer with an offset (positive and negative). The offset is applied at read request with reference to the read request event.
All data acquisition options are listed in Table 22.

Crossbar switch and AGC should preferably be disabled during turn-by-turn data acquisition, see Chapter 3.3.1. Make sure the ADC mask is set accordingly (Chapter 3.2.6).

NOTE: While the synchronization is in progress (see Chapter 2.8.3) the data is not valid, therefore the reading from the buffer is blocked.

The data from the turn-by-turn buffer is averaged to calculate the mean value and standard deviation of both, X and Y positions. The calculation is done on the configurable window length and can be updated in three modes (Now, Periodic, Event). Values and parameters are located in nodes:

```
libera-ireg dump boards.bpm.statistics.ddc
ddc
  mean x=-102166
  mean y=244349
  std_x=2408761
  std_y=2489470
  period=1
  window=10
  mode=Now
```

Explanation of nodes:

- mean_x, mean_y: mean value of X and Y positions (in nanometers)
- std_x, std_y: standard deviation of X and Y positions (in nanometers)
- period: update rate (in seconds). Valid range is [0.05, 10]

- window: calculation window length (in TBT atoms). Valid range is [10, 10000]
- mode: update mode (Now, Period, Event). Now = on demand, Period = with defined period, Event = on trigger

3.4.4 Turn-by-turn – TDP

The Turn-by-Turn (TDP) data is processed precisely at the accelerator's revolution frequency. Exact sampling frequency in [Hz] is reported in registry node:

```
boards.bpm.clock_info.tbt_frequency
```

The turn-by-turn buffer is a circular buffer which is continuously filled with V_A, V_B, V_C and V_D amplitudes. The data is available in processed format:

- tdp_synthetic: provides amplitudes (V_A, V_B, V_C, V_D) and processed values (SUM, Q, X, Y) calculated from the amplitudes, see Figure 38.

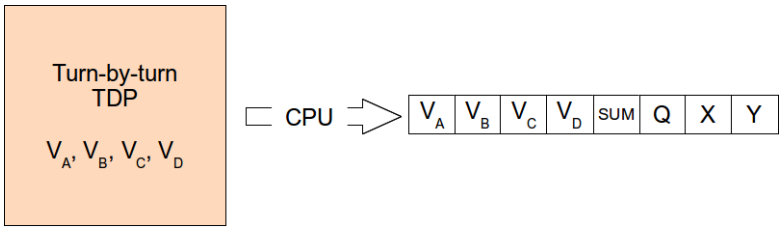

Figure 39: Synthetic data from the turn-by-turn (TDP) buffer

Example of read request for both data types:

```
libera-ireg signal boards.bpm.signals.tdp_synthetic –s10 -v
```

Data can be read from the buffer with an offset (positive and negative). The offset is applied at read request with reference to the read request event.
All data acquisition options are listed in Table 22.

Crossbar switch and AGC should preferably be disabled during turn-by-turn data acquisition, see Chapter 3.3.1. Make sure the ADC mask is set accordingly (Chapter 3.2.6).

NOTE: While the synchronization is in progress (see Chapter 2.8.3) the data is not valid, therefore the reading from the buffer is blocked.

The data from the turn-by-turn buffer is averaged to calculate the mean value and standard deviation of both, X and Y positions. The calculation is done on the configurable window length and can be updated in three modes (Now, Periodic, Event). Values and parameters are located in nodes:

```
libera-ireg dump boards.bpm.statistics.tdp
tdp
    mean x=45584
    mean y=-8913
    std_x=1345
    std_y=904
    period=1
    window=10
    mode=Now
```

Explanation of nodes:

- mean_x, mean_y: mean value of X and Y positions (in nanometers)
- std_x, std_y: standard deviation of X and Y positions (in nanometers)
- period: update rate (in seconds). Valid range is [0.05, 10]
- window: calculation window length (in TBT atoms). Valid range is [10, 10000]
- mode: update mode (Now, Period, Event). Now = on demand, Period = with defined period, Event = on trigger

3.4.5 Multi-bunch Turn-by-turn – MBTBT

The multibunch Turn-by-Turn data is processed precisely at the accelerator's revolution frequency. Exact sampling frequency in [Hz] is reported in registry node:

```
boards.bpm.clock_info.tbt_frequency
```

The multi-bunch turn-by-turn buffer is a circular buffer which is continuously filled with V_A, V_B, V_C and V_D amplitudes and position data from 4 ADC masks (Chapter 3.2.8). The data is available in processed format:

- mbtbt_synthetic: provides amplitudes (V_A, V_B, V_C, V_D) and processed values (SUM, Q, X, Y) calculated from the amplitudes, see Figure 38.

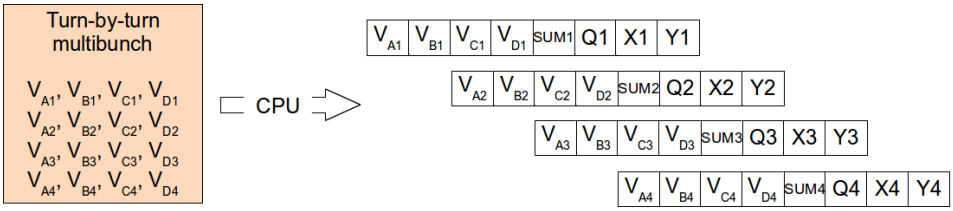

Figure 40: Synthetic data from the multi-bunch turn-by-turn buffer.

Example of read request for both data types (bpm=kraf3, kraf4, kraf5, kraf6):

```
libera-ireg signal boards.bpm.signals.mbtbt_synthetic -s10 -v
```

Data can be read from the buffer with an offset (positive and negative). The offset is applied at read request with reference to the read request event.
All data acquisition options are listed in Table 22.

Crossbar switch and AGC should preferably be disabled during turn-by-turn data acquisition, see Chapter 3.3.1. Make sure the multibunch ADC mask is set accordingly (Chapter 3.2.8).

NOTE: While the synchronization is in progress (see Chapter 2.8.3) the data is not valid, therefore the reading from the buffer is blocked.

3.4.6 Fast acquisition (FA) data

The Fast Acquisition (FA) data is a continuous data stream. Source of the data is either Turn-by-turn DDC or TDP (selectable) which is then filtered and decimated (polyphase FIR filter) to reduce the data rate to approximately 10 kS/s and bandwidth to approximately 2 kHz. The decimation factor between the turn-by-turn and FA data rate is a hard-coded integer value (FA decimation) and is accelerator dependent.

The crossbar switch and DSC can be enabled since this improves the performance. The switching harmonics (every ~3.3 kHz) are suppressed by a notch filter and they are out of the FA data bandwidth anyway.
The FA data is used for the interlock detection and can also serve the fast feedback applications. The step response latency depends on the FIR filter's coefficients and can range from approximately 270 µs to 500 µs.

Example of read request for the FA data:

```
libera-ireg signal boards.bpm.signals.fa -s10 -v
```

Data can not be read on-trigger nor with an offset. All data acquisition options are listed in Table 22.

Besides the amplitudes and position data, the FA data atom additionally contains a 64-bit timestamp and status bits:

- LMT_l … 32 LSB of the 64-bit LMT
- LMT_h … 32 MSB of the 64-bit LMT
- Status bits … reported in a decimal format. Convert to binary and see Table 23.

Table 23: Status bits in FA data

Status bit	Description
[31:16]	16-bit counter
[15]	Interlock (0 … no Interlock, 1 … Interlock active)
[14]	ADC overflow (0 … no ADC overflow, 1 … ADC overflow active)
[0]	MC PLL status (0 … unlocked, 1 … locked)

The default turn-by-turn data source for the FA data is DDC. To switch between the data sources, use:

```
libera-ireg boards.bpm.tbt.data_type=DDC
```

or switch to turn-by-turn TDP data source:

```
libera-ireg boards.bpm.tbt.data_type=TDP
```

NOTE: Same turn-by-turn data source is used for the FA and SA data.

NOTE: While the synchronization is in progress (see Chapter 2.8.3) the data is not valid.

3.4.7 Slow acquisition (SA) data

The Slow Acquisition (SA) data is a continuous data stream. Source of the data is the FA data which is filtered and decimated to reduce the data rate to approximately 10 S/s and bandwidth to approximately 4 Hz. The decimation factor between the FA data and SA data rate is a hard-coded integer value (SA decimation) which is typically 1024.
The amplitudes may differ between the FA and SA data. The difference comes from CIC filters on the SA data. Position data is not affected.

The SA data is typically used for monitoring the beam at slow update rate. With the crossbar switch and DSC enabled an excellent RMS, beam current dependence, fill pattern dependence and temperature dependence makes the SA data an ideal input for slow feedback applications.

Example of read request for the SA data:

```
libera-ireg signal boards.bpm.signals.sa –s10 -v
```

Data can not be read on-trigger nor with an offset. All data acquisition options are listed in Table 22.

The SA data atom is same as the FA data atom. Besides the amplitudes and position data, the SA data atom contains a 64-bit timestamp and status bits:

- LMT_l … 32 LSB of the 64-bit LMT
- LMT_h … 32 MSB of the 64-bit LMT
- Status bits … reported in a decimal format. Convert to binary and see Table 23.

When performing the synchronization, the SA stream is stopped for up to 2.

NOTE: While the synchronization is in progress (see Chapter 3.7.2) the data is not valid. The SA data is stopped for up to 2 seconds.

The SA data is averaged to calculate the mean value and standard deviation of both, X and Y positions. The calculation is done on a sliding window with configurable length (in SA data atoms) and is updated with a given step (in SA data atoms). Values and parameters are located in nodes:

```
libera-ireg dump boards.bpm.statistics.sa
sa
  mean x=-154258
  mean y=-26571
  std_x=10984
  std_y=13094
  step=10
  window=100
```

Explanation of nodes:

- mean_x, mean_y: mean position of X and Y positions (in nanometers)
- std_x, std_y: standard deviation of X and Y positions (in nanometers)
- step: update rate (in SA atoms). Valid range is [10, 100]
- window: calculation window length (in SA atoms). Valid range is [10, 600]

The last 8,192 SA data samples are buffered in the SA data history buffer (last ~13 minutes). Data atom is same as the SA data atom.

Example of read request for the SA data history buffer:

```
libera-ireg signal boards.bpm.signals.sa.history -s10 -v
```

The SA data history buffer can not be read on-trigger nor with an offset. All data acquisition options are listed in Table 22.

3.4.8 Single Pass (SP)

The single pass functionality is intended for position calculation of a single bunch. It is most suitable for injection studies as it supports relatively slow trigger repetition rates, below 20 Hz. The source data is taken from the ADC buffer.

The calculation requires a trigger and three parameters:

- threshold: specifies the threshold for start of calculation (in ADC counts)
- pre-trigger (n_before): specifies the number of samples to take before the threshold (in ADC samples)
- post-trigger (n_after): specifies the number of samples to take after the threshold (in ADC samples)

After the trigger event, 1024 ADC samples (~9 µs) are analysed. Once the first ADC amplitude (A, B, C or D) exceeds the threshold (T0), a calculation window is set:

$$W = n\_before + n\_after$$

The window starts a (T0 - n_before) as shown in Figure 41. Threshold detection time (T0) is logged and provided in the data atom.

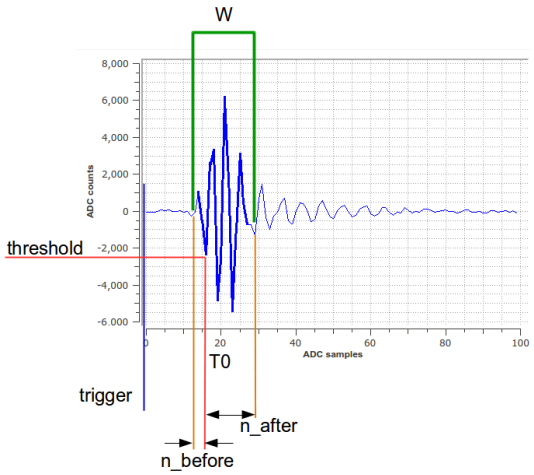

Figure 41: Single pass calculation parameters.

Amplitudes from ADC data are calculated using the following equation:

$$Va\,(Vb, Vc, Vd) = \sqrt{\sum_{i=1}^{W} x_i^2}$$

where W is window length and x_i is ADC sample from a channel. Bunch position is calculated from these amplitudes as described in Chapter 3.2.1.

Nodes for the single pass functionality:

```
libera-ireg dump boards.bpm.single_pass
pickup_pos=Diagonal
threshold=1000
n before=1
n after=200
signal
  access_type=DataOnDemand
  atom size=32
  group size=1
  components names=[Va,Vb,Vc,Vd,Sum,Thr Idx,X,Y]
  components_number=8
  components_type=int32
```

Example of read request for the single pass data:

```
libera-ireg signal boards.bpm.single_pass.signal -bEvent -v
```

The single pass data atom contains:

- amplitudes (V_A, V_B, V_C, V_D, SUM)
- positions (X, Y)
- Thr_Idx value which marks the T0 (Figure 41)

Position can be calculated for diagonal or orthogonal pickup orientation. Default is diagonal orientation with calculation equation as shown in Chapter 3.2.1. Orthogonal orientation is rotated by 45° and calculation equation changes to:

$$X = Kx * \frac{Va - Vc}{Va + Vc} - Xoffset$$
$$Y = Ky * \frac{Vb - Vd}{Vb + Vd} - Yoffset$$

To switch between the calculation equations, use:

```
libera-ireg boards.bpm.single_pass.pickup_pos=Orthogonal
```

or switch to default:

```
libera-ireg boards.bpm.single_pass.pickup_pos=Diagonal
```

3.4.9 Fast data stream (FDS data)

The Fast Data Stream (FDS data) is a continuous data stream. Source of the data is either Turn-by-turn DDC or TDP (selectable). The incoming data is optionally filtered and decimated. Following the filtering and decimation blocks, the position is calculated and FDS data atom is generated. There is a selector to switch between the FDS data and original FA data for the GDX module. The selection must be done through a configuration file and when the 'libera-bpm' daemon is stopped. Processing branch is presented in Figure 42.

The data source for the GDX module must set for all BPM modules in the crate to same value. The setting procedure:

1. Create the custom configuration file:

```
root@libera:~# libera-ireg system.persistence.save{}
system.persistence.save{}: done
```

2. Stop the libera-ebpm application:

```
root@libera:~# /etc/init.d/libera stop
```

3. Edit the custom configuration file and modify the switching decimation

```
root@libera:~# nano /var/opt/libera/cfg/libera-ebpm.xml
```

The parameter is located in the .xml structure: <module> → signal_processing → data_source → lvds . See example where the data source is set to FDS (choices: FDS ... 'tbt', FA ... 'fa')

```
      <signal_processing>
        <data_source>
          <lvds value="tbt"/>
        </data_source>
      ...
```

Save the file and exit.

4. Power cycle the instrument

```
root@libera:~# libera-bmc --power --cycle --set
```

NOTE: The FDS data must be set to same turn-by-turn data source (DDC or TDP) in all BPM modules.

NOTE: Slot 3 must always be populated (raf/kraf3).

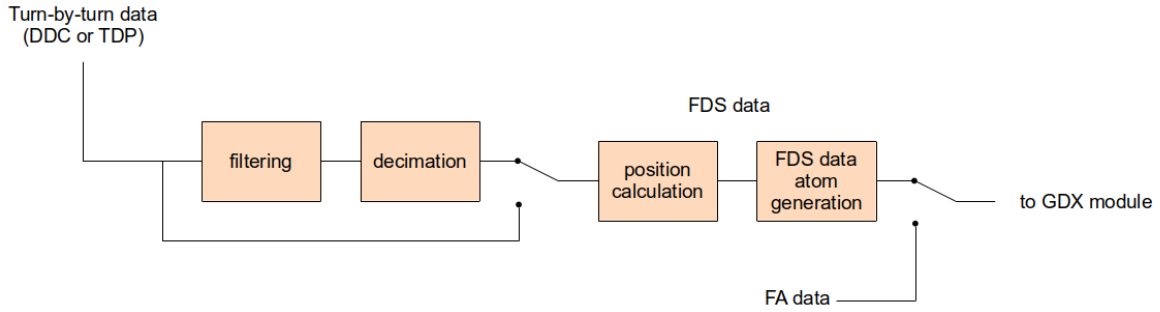

Figure 42: FDS data processing branch.

The filter is a standard low-pass buiquad IIR filter implementation. Filter coefficients are defined in the configuration file and can only be modified through the same file and when the libera-ebpm daemon is stopped. Parameter values are readable in libera registry tree (bpm=kraf3, kraf4, kraf5, kraf6):

```
root@libera:~# libera-ireg dump boards.bpm.signal_processing.fds_iir_coefficients
fds_iir_coefficients
  iir 1
    c=124347
    d=-60982
    o=21121
  iir_2
    c=116846
    d=-53351
    o=68866
  iir_3
    c=111690
    d=-48105
    o=94383
  iir 4
    c=109085
    d=-45455
    o=99855
```

Settings of the decimation factor (picking every D sample) and a general bypass option are readable in registry nodes (bpm=kraf3, kraf4, kraf5, kraf6):

```
root@libera:~# libera-ireg dump boards.bpm.signal_processing.fds_control
fds_control
  fds iir decimation=10
  iir_and_decimation_bypass=false
```

Both parameters can be modified through the configuration file and when the ‘libera-ebpm’ daemon is stopped. Decimation D can be set in the [1, 1024] range.

NOTE: Change of the FDS decimation factor requires a power cycle.

The FDS data atom contains the amplitudes and position data, and timestamps (LMT and MT). The data atom is shown in Figure 43. The data atom is streamed from the BPM module to the GDX module at the (turn-by-turn / D) data rate.

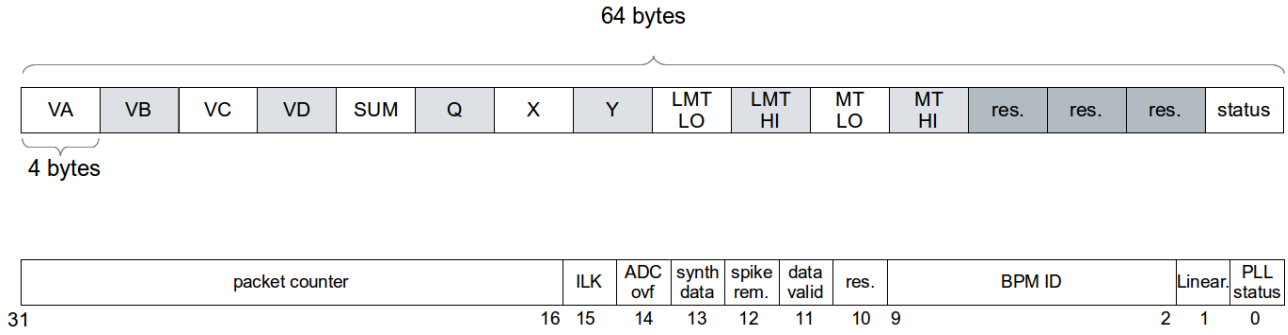

Figure 43: FDS data atom.

In the GDX module, the FDS data is handled in the same way as the standard FA data atom (depending on the GDX userblock).

NOTE: The Libera Grouping+ may not support the FDS data concentration if the data rate exceeds 10 kS/s.

3.5 Interlock functionality

The Interlock mechanism continuously monitors the beam position and intensity. The purpose of the Interlock function is to enhance the safety in the accelerator, preventing the beam to cause any damage. Different modes of operation are available.

Immediately after the interlock condition is detected, Libera Brilliance+ outputs the digital signal through the timing module. The interlock output signal can be generated also by transmitting a pre-defined optical event from the timing system.

Interlock can be detected by three scenarios (see Figure 44):

- Beam position exceeds X / Y limits. Limits are defined as minimum X / Y and maximum X / Y and are set in nanometers. Detection is based on the FA data (by default, 10 kS/s). Details are available in Chapter 3.5.1.
- Signal level saturates the ADCs. Saturation limit is defined in the ADC counts. Detection is based on the ADC data. Details are available in Chapter 3.5.2. Position and overflow conditions are monitored on each BPM module.
- Optical event with pre-defined ID is receiver on the SFP interface. See Chapter 3.5.4

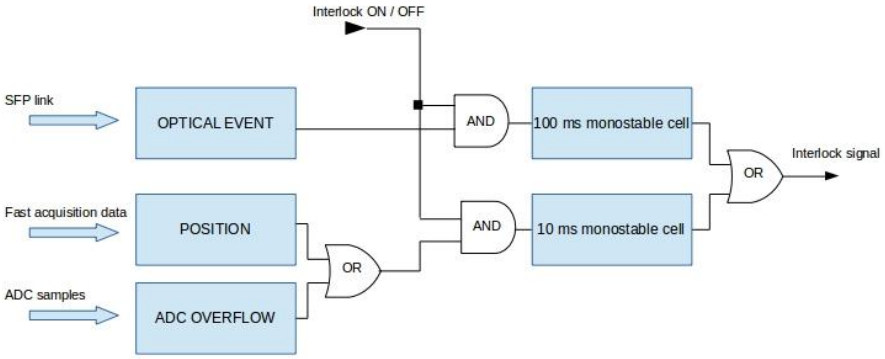

Figure 44: *Interlock signal generation*

Special cases:

- interlock detection can depend on the input signal level. If the signal level is below the threshold, interlock is not triggered. Details are available in Chapter 3.5.2.2.
- interlock detection can be disabled.

To enable/disable the interlock detection, issue the following commands:

```
libera-ireg boards.bpm.interlock.enabled=true
libera-ireg boards.bpm.interlock.enabled=false
```

When interlock detection is enabled, all mechanisms are active (position and ADC overflow monitoring, optical event stream monitoring). Interlock output is active for all the time the interlock conditions are fulfilled. The interlock output remains active for 10 ms after conditions (position, saturation) have been cleared (mono-stable cell). 10 ms time has been chosen to ensure the minimum length of the Interlock trigger signal. For optical event, the interlock output remains active for 100 ms.

In addition to interlock output signal (electrical connector), an optical event can be sent upstream the SFP link. The event with a pre-defined ID will be sent when interlock is detected either in position or saturation conditions. Functionality is disabled when ID is set to 0.
To enable and set the upstream optical event (example ID=4444), use these commands:

```
libera-ireg boards.evrx.rtc.mgt_out=connectors
libera-ireg boards.evrx.rtc.sfp_tx.interlock.id=4444
```

Schematically, the interlock mechanism is presented in Figure 45.

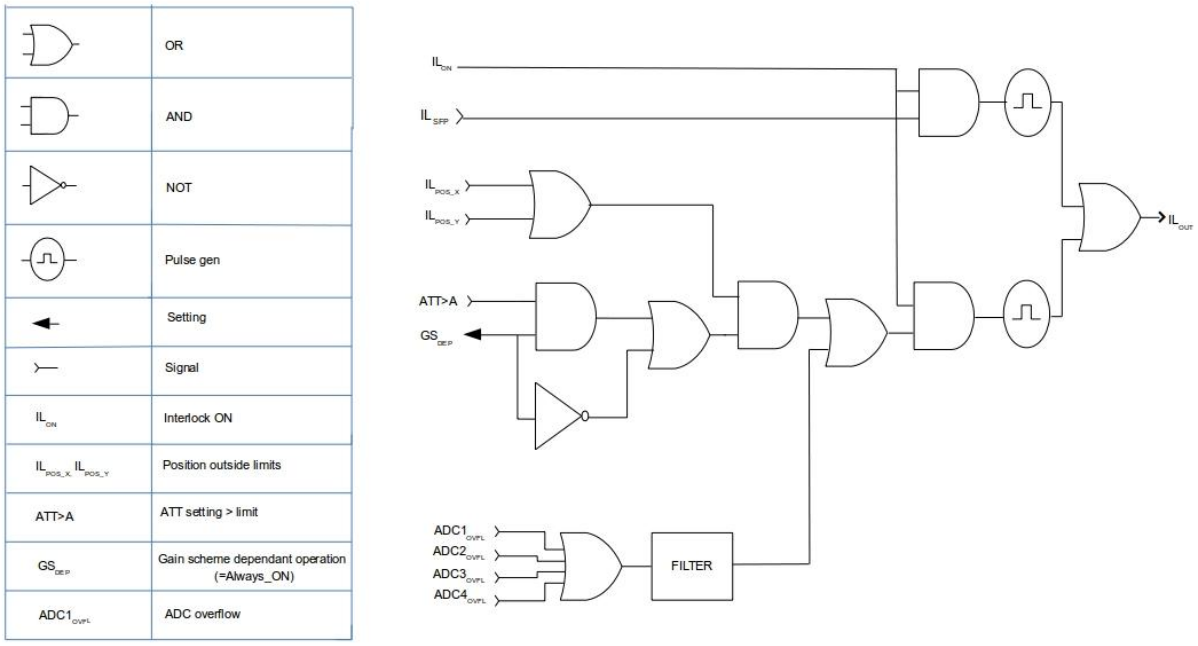

Figure 45: *Interlock block diagram*

3.5.1 Beam position Interlock

Beam position is monitored on the FA data at ~10 kS/s data rate (by default). See Figure 46. Limit values for X and Y position are set in nanometers. Minimum value must not exceed the maximum value (or the opposite) otherwise, there is no limitation in the absolute values (limits can be all set to positive or all to negative values).

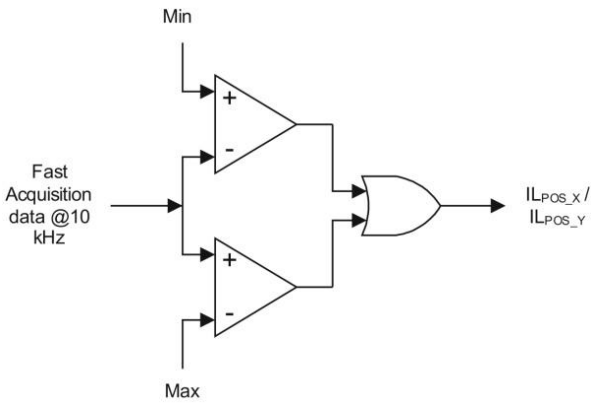

Figure 46: *Single component for X or Y*

To check position limits, issue the following command:

```
libera-ireg dump boards.bpm.interlock.limits.position
position
  min
    x=-1000064
    y=-1000064
  max
    x=999936
    y=999936
```

The next example shows how to set specific limits (-1 mm < X < 1.75 mm and -0.8 mm < Y < 2.344 mm).

```
libera-ireg boards.bpm.interlock.limits.position.min.x=-1000000
libera-ireg boards.bpm.interlock.limits.position.max.x=-1750000
libera-ireg boards.bpm.interlock.limits.position.min.y=-800000
libera-ireg boards.bpm.interlock.limits.position.max.y=-2344000
```

Parameter values are in nanometers for position with 128 nm setting resolution. The value, set by user is rounded to the closest possible value. See example below.

```
libera-ireg boards.bpm.interlock.limits.position.min.x=1000000
libera-ireg boards.bpm.interlock.limits.position.min.x
x=999936
```

The FA data could contain glitches (e.g. due to gain changes). To prevent erratic interlock triggering, additional filtering on position data can be applied. By default, filtering is disabled.
For detailed explanation, see the next example.

The simulation foresees two FA samples above the position limit. There are 3 examples of different filter settings with their response:

- No filtering, setting 255, (Figure 47)
- Moderate filtering, setting 63, (Figure 48)
- Substantial filtering, setting 15, (Figure 49)

The red line presents the actual FA position fluctuation. The blue line shows the input (after filtering) for the interlock comparator.

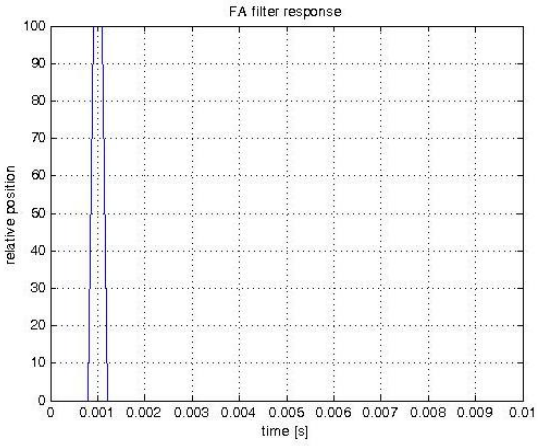

Figure 47: *No filtering (setting 255)*

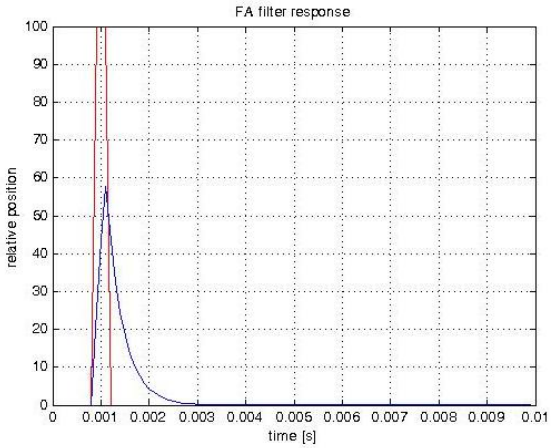

Figure 48: *Moderate filtering (setting 63)*

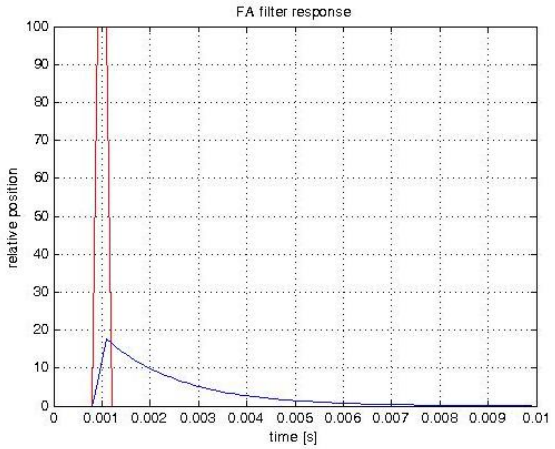

Figure 49: *Substantial filtering (setting 15)*

Using at least some filtering on position data efficiently reduces glitches. Interlock detection speed is still in the range of milliseconds.
If no position filtering is used, the interlock detection speed is limited with the following processing elements:

- Latency in the FA processing chain (~200-500 µs)
- FA data rate (100 µs resolution)
- FPGA => ILOCK connector transfer time (negligible)

Setting range of the filter is from 0 (strongest) to 255 (no filtering). Logic is inverted. To set the filter value to 200, issue the following commands:

```
libera-ireg boards.bpm.interlock.filter.position=200
```

The interlock response on step change is estimated to be around 300-500 µs.

3.5.2 ADC saturation Interlock

Functionality continuously monitors the ADC data for the saturation (overflow) continuously. The interlock event is detected based on 2 parameters:

- The ADC **overflow threshold**, which represents the maximum safe signal level for the ADCs. Full scale of the ADCs in Libera Brilliance+ is 32,767 ADC counts. The default ADC overflow threshold is set to 30,000. Setting unit is 1 ADC count.
- The ADC **overflow duration**, which defines allowed duration of the overflow. The duration value represents: (1) ADC samples or (2) turns, depending on the detection mode chosen. By default, it is set to 5.

The **mode** selector switches between two detection modes: adc and tbt (see Figure 50). Both are described in the next two chapters. The **mode** can not be changed runtime but it is loaded at libera-ebpm daemon start. To change the mode, one must stop the libera-ebpm daemon and edit the `/var/opt/libera/cfg/libera-ebpm.xml` file. Default is **adc** mode.

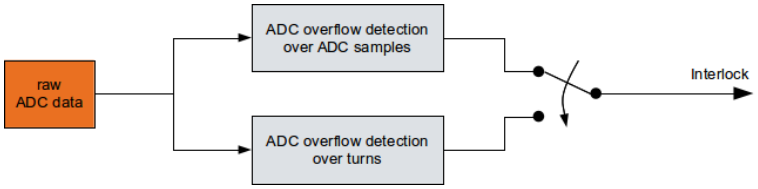

Figure 50: ADC saturation detection modes.

To check current settings for both parameters, issue the following command:

```
libera-ireg dump boards.bpm.interlock.limits.overflow
threshold=30000
duration=5
mode=adc
```

Interlock could trigger in this scenario:

- No beam or low beam current
- Power level is low (< -25 dBm), attenuation is low
- AGC is disabled
- Injection, beam current increasing
- Depending on final beam current/attenuation value, ADCs could
- Interlock is activated to make the operation fail safe

3.5.2.1 ADC saturation detection – adc mode

A comparator compares the current amplitude on ADCs with the threshold amplitude. It is further led to a counter which counts the number of ADC samples within certain period. If the comparator determines that the amplitude is over the threshold limit and the counter exceeds the overflow duration parameter, interlock is activated (see Figure 51). Configurable overflow duration parameter was added to prevent unwanted interlock triggering due to random glitches.

Attenuation changes cause glitches with approximately 100-200 ns duration. To prevent unwanted interlock triggering, certain dead time after the attenuators' change was introduced. At each attenuation change, the interlock mechanism is disabled for a period of 200 ADC samples (approximately 1.5 µs).

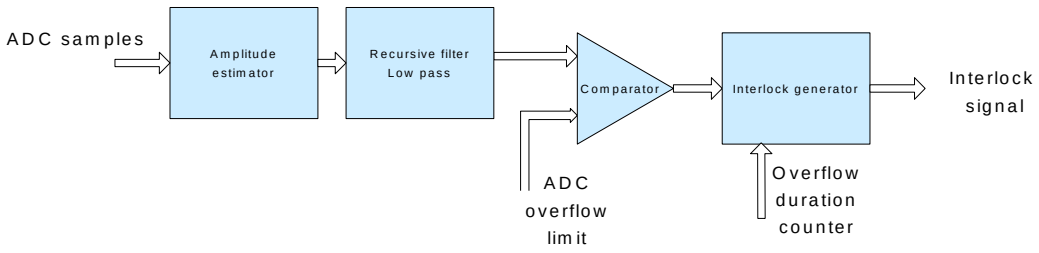

***Figure 51:** ADC saturation detection (adc mode)*

Additional filtering on the ADC data can be applied in order to prevent unwanted interlock triggering. Filter formula is:

$$Y_n = Y_{n-1} * (1 - 2^{-n}) + X_n * 2^{-n}$$

Output (Y_n) depends on previous output (Y_{n-1}) and current input (X_n). By default, “n” is set to “0”, no filtering. The maximum filtering is applied when set to 6 (setting resolution is 1).
To check current setting for ADC filtering, issue the following command:

```
libera-ireg boards.bpm.interlock.filter.overflow
overflow=0
```

3.5.2.2 ADC saturation detection – tbt mode

The **tbt** mode uses MaxADC detector that figures out if any ADC sample within a turn exceeded the threshold. If yes, the turn is counted as overflown. The turn which is not overflown resets the counter. If the number of consecutive overflown turns exceeds the **duration** parameter, the Interlock is detected.

Functionality is described with example settings (shown also in Figure 52):

- Allowed number of turns in saturation = 3
- ADC count threshold = 80%
- Interlock is triggered after the third turn reports saturation

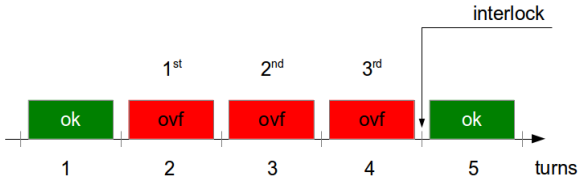

Figure 52: ADC saturation detection over turns.

3.5.3 Gain dependent operation

At low beam current, position reading is not as accurate as at high beam currents. Gain dependent mode of operation prevents interlock triggering at low beam currents.
The gain dependent mode requires a threshold value. It is given as a power level (0 to -80 dBm). The power level is related with attenuation settings through the gain scheme, see Figure 53. The interlock is disabled at power level which is linked to lower attenuation than the attenuation at gain dependent threshold power level.

Default gain dependent threshold is set at -32 dBm. According to default gain scheme, the attenuation at -32 dBm is 3 dB. Attenuation changes from 3 dB to 0 dB between power levels -40 dBm and -41 dBm (see Figure 53). Despite the default gain dependent threshold set at -32 dBm, the interlock remains active until power level -40 dBm.

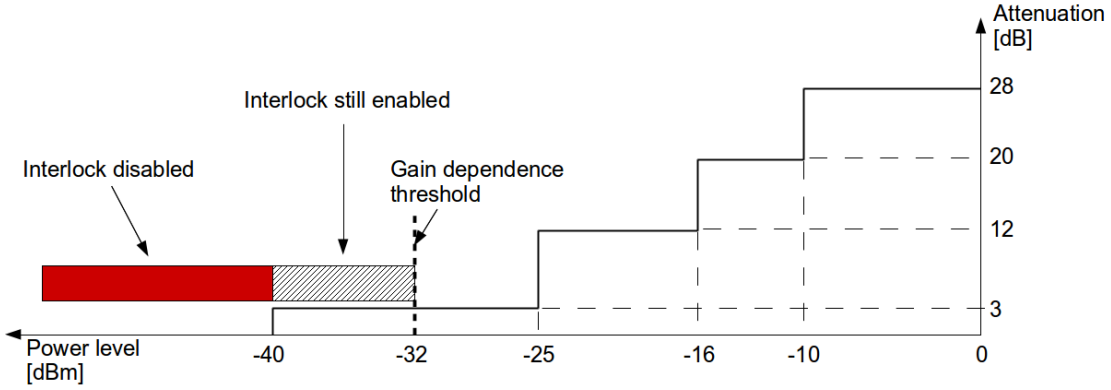

***Figure 53:** Gain scheme and gain dependence threshold*

If the gain dependent threshold is set to the power level where the attenuator is already set to 0 dB or the attenuation does not change any more, the interlock detection will remain enabled.

Gain dependence mode of operation is set in node:

```
libera-ireg boards.bpm.interlock.gain_dependent.enabled=true
```

Gain dependence threshold is set in node:

```
libera-ireg boards.bpm.interlock.gain_dependent.threshold=-32
```

The attenuation at selected gain dependence threshold is a read-only node:

```
libera-ireg boards.bpm.interlock.gain_dependent.threshold_dB=3
```

3.5.4 Optical event-generated interlock

An optical event received through the SFP in the timing module can trigger the interlock. Interlock duration is < 110 ms.

NOTE: While optical event interlock is active, notification of BPM-related interlock over optical stream will not be sent.

To send the optical event upstream the SFP link, the interlock functionality must be enabled in the first of installed BPM modules (kraf3, kraf4, kraf5, kraf6). Position limits must be set appropriately to beam position.

Following commands demonstrate how to enable and configure the functionality. Interlock will be enabled on raf3 module.

```
libera-ireg boards.kraf3.interlock.enabled=true
```

Enable decoding of the optical event stream:

```
libera-ireg boards.evrx.rtc.decoder_switch=on
```

Configure the RTC table (example for event ID 44444):

```
libera-ireg dump boards.evrx.rtc.sfp_2_connectors.interlock
   in mask=[65535,65535,65535,65535]
   in function=[65535,65535,65535,65535]

libera-ireg boards.evrx.rtc.sfp_2_connectors.interlock.in_function[0]=44444

libera-ireg dump boards.evrx.rtc.sfp_2_connectors.interlock
   in_mask=[65535,65535,65535,65535]
   in_function=[44444,65535,65535,65535]
```

See Chapter 3.8.2 for details on RTC usage.

Upon optical event reception (configured for Interlock), the Interlock status will report:

```
root@ubuntu-libera:~# libera-ireg dump boards.kraf3.interlock.status.il_status.
reset
x=true
y=true
attenuator=true
adc_overflow_filtered=true
adc_overflow=false
```

Consult support@i-tech.si for more information.

3.5.5 Interlock status and cause

The cause of the Interlock event is latched in the Interlock status register. There are 3 possible causes:

- X position exceeded the limit values
- Y position exceeded the limit values
- ADCs in saturation

Additionally, the Interlock status contains the following information:

- If the attenuator's value is higher than gain dependence threshold ("Attenuator")
- The ADC overflow status (filtered/non filtered)

The Interlock status latches all conditions that caused and interlock event. It is not possible to read the first condition only. To check and then reset the interlock status, issue the following commands:

```
libera-ireg dump boards.bpm.interlock.status
status
  il status=0
    reset
    x=false
    y=false
    attenuator=false
    adc overflow filtered=false
    adc overflow=false

libera-ireg boards.bpm.interlock.status.il_status=0
```

The Interlock status is set also in turn-by-turn and Fast & Slow data streams. Exact location is shown in Table 24.

Table 24: Interlock status bits in data signals

Interlock status	FA & SA data streams (status register)	raw turn-by-turn (I & Q)
Active	bit 15	N/A
ADC overflow set	bit 14	LSB of Qa
X position out of range	N/A	LSB of Ib
Y position out of range	N/A	LSB of Qb

For example, if the interlock is triggered because of the ADC overflow, the LSB of Qa data will become active. In this way one can precisely define the moment and the cause of the interlock on the BPM module.

3.6 Postmortem functionality

The Postmortem (PM) allows reviewing the causes of critical events, such as sudden beam loss. After the PM event is generated (external or other condition), the turn-by-turn data before the PM event is acquired and stored into a separate fixed buffer.
After the PM functionality is enabled, it waits for the first PM event and then disables automatically. This mechanism provides the user enough time to read PM data from the buffer.
To enable the PM functionality, use the following command:

```
libera-ireg boards.bpm.postmortem.capture=true
```

Source of the PM event can be one of the following events:

- external PM trigger, received through the PM connector on the timing module
- Interlock condition
- exceeded position limits (X, Y) and/or ADC overflow, set only for PM detection

When one source is selected, other two are ignored. To set the selected source, issue the following command:

```
libera-ireg boards.bpm.postmortem.source_select=external
```

Altogether there are three options:

- external
- interlock
- limits

Once the buffer is filled, it ignores all other possible PM events. Only the first PM event triggers the PM buffer.

When “limits” source is selected, user can set the limits for position (min/max) and ADC overflow/duration. To check the currently set values, issue the following command:

```
libera-ireg dump boards.bpm.postmortem.limits
limits
  position
    min
      x=896
      y=-1000064
    max
      x=2944
      y=999936
  overflow
    threshold=30000
    duration=5
```

To set the minimum X position limit to (-550 µm), issue the following command:

```
libera-ireg boards.bpm.postmortem.limits.position.min.x=-550000
```

Similar commands apply for other parameter settings.

Position is monitored **on the FA data** (~10 kS/s). Position is given in nanometers with setting resolution 128 nm.

The ADC overflow detection is monitored **at ADC rate**. As soon as the first ADC count exceeds the threshold, this moment is considered as a starting point. The duration of the overflow is compared to duration setting and if it is

exceeded, the PM event is generated. Parameter value for ADC overflow is given **in ADC counts** (up to 32,766). Parameter value for ADC overflow duration is given **in ADC samples**.

PM buffer capacity is adjustable up to 524,288 samples (default is 65,536). After the PM event, complete PM buffer is filled. If user is interested only in the last part of the PM buffer (just before the PM event), the read-out buffer can be shorter than complete PM buffer. Data orientation in the read-out PM buffer is exactly the same as it was filled-in. The last sample in the buffer is always the sample at t0 (with offset, if any). Please see Figure 54 for more details.
PM buffer capacity can be set using the following command:

```
libera-ireg boards.bpm.postmortem.capacity=65535
```

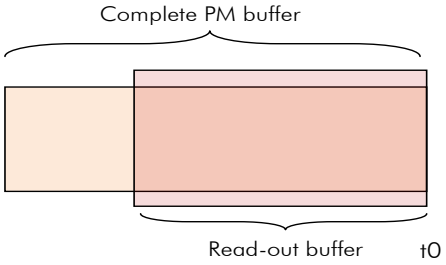

Figure 54: *Complete and read-out PM buffer*

It is possible to off-set the PM event to capture the data more in the history or after the PM event, see Figure 55.
A positive offset will virtually move the PM event 1024 to the future. To set the PM offset to 1024 turns, issue the following command:

```
libera-ireg boards.bpm.postmortem.offset=1024
```

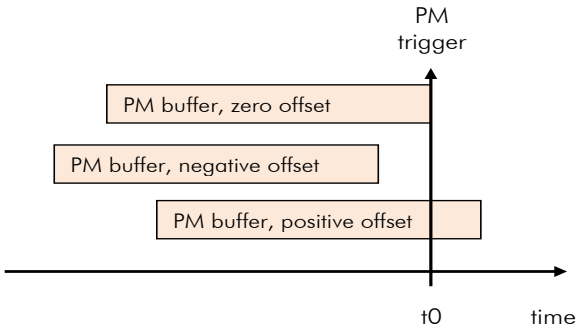

Figure 55: *Postmortem offset*

The PM event is marked with two timestamps (not offset or buffer length dependent):

- “timestamp”: presented in the MT
- “os_time”: presented in the absolute time. This time is reported by the operating system. It may vary from instrument to instrument as it depends on the jitter and NTP synchronization. It is intended for rough estimation of the PM event in absolute time.

Timestamps' values are empty until the first PM event.
To read the timestamps, use the following commands:

```
libera-ireg boards.bpm.postmortem.timestamp
boards.bpm.postmortem.timestamp: 450472007512

libera-ireg boards.bpm.postmortem.os_time
boards.bpm.postmortem.os_time: 2020-03-17 06:38:51.437270000
```

The PM buffer is filled with raw turn-by-turn data (ddc_raw). To read the buffer (ddc_raw or ddc_synthetic), issue the following command:

```
libera-ireg signal boards.bpm.postmortem.signals.ddc_synthetic
```

There is an additional parameter "length" (read-only), which reports how much data is available in the PM buffer. Usually, "capacity" and "length" values match. In case that the circular buffer did not contain enough turn-by-turn data at the time of the PM event, the "length" value can be lower than "capacity".

Full set of PM settings:

```
libera-ireg dump boards.bpm.postmortem
postmortem
  capture=false
  source select=external
  limits
    position
      min
        x=-1000064
        y=-1000064
      max
        x=999936
        y=999936
    overflow
      threshold=30000
      duration=5
  timestamp=450472007512
  os_time=2020-03-17 06:38:51.437270000
  offset=0
  capacity=65536
  length=65536
  signals   …
```

3.7 Timing and synchronization

3.7.1 Clock domain

A clock domain is a set of frequencies, linked with an integer (or rational) ratio. The clock domain in Libera Brilliance+ is based on Machine Clock (MC). One period of the MC signal equals the time the electron (bunch) needs for one turn in the ring. MC signal must be distributed to the timing module via electrical or optical event.
Frequencies derived from the MC signal are presented in Table 25.

Table 25: Clock domain frequencies.

Clock / frequency	Relation to MC	Value range	Description
f_{LMC}	f_{MC} x D_{TBT}	95 – 125 MHz	Libera (or Local) Machine Clock, also known as ADC sampling clock f_{ADC}
f_{MC}	1	100 kHz – 12 MHz	Machine Clock, also known as Turn-by-turn clock f_{TBT}
f_{FDS}	$f_{MC} \div D_{FDS}$	depends on D_{FDS}	Fast Data Stream
f_{FA}	$f_{MC} \div D_{FA}$	~10 kS/s	Fast Acquisition data rate
f_{SA}	$f_{FA} \div D_{SA}$	~10 S/s	Slow Acquisition data rate
f_{SW}	$f_{MC} \div D_{SW}$	~13 kHz	Switching Clock
f_{PLL}	$f_{MC} \div D_{PLL}$	~10 Hz	PLL loop execution rate

The decimation factors used to obtain the clock domain frequencies are listed in Table 26. Graphical presentation of the frequencies is given in Figure 56.

***Table 26:** Clock domain decimation factors.*

Decimation factor	Description
D_{TBT}	Turn-by-turn decimation
D_{FDS}	Fast Data Stream (FDS) decimation
D_{FA}	Fast Acquisition (FA) decimation
D_{SA}	Slow Acquisition (SA) decimation
D_{SW}	Switching decimation
D_{PLL}	PLL decimation

Acquired data is equipped with various timestamps.

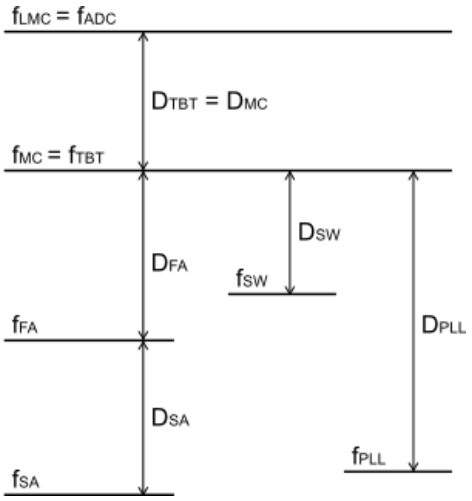

***Figure 56:** Clock domain frequencies.*

The MC signal should be supplied at exact revolution frequency. The VCXO's frequency (ADC sampling frequency) is locked to the MC frequency with the SW PLL. Normally, MC PLL is locked within a few seconds after connecting the correct MC signal. If there is no MC signal present, the sampling clock frequency will be somewhere in the VCXO's

tuning range (±90 ppm). Its exact frequency is an integer multiplier of the revolution frequency and varies from one accelerator to another. A nominal frequency value can be checked in the node:

```
libera-ireg boards.tim.clock_info.adc_frequency
```

The timing module distributes the clock signals to the BPM modules and a GDX module. The absolute synchronization of the local (BPM) clock signals to the external trigger event is done in the BPM modules. Synchronization scheme is shown in Figure 57.

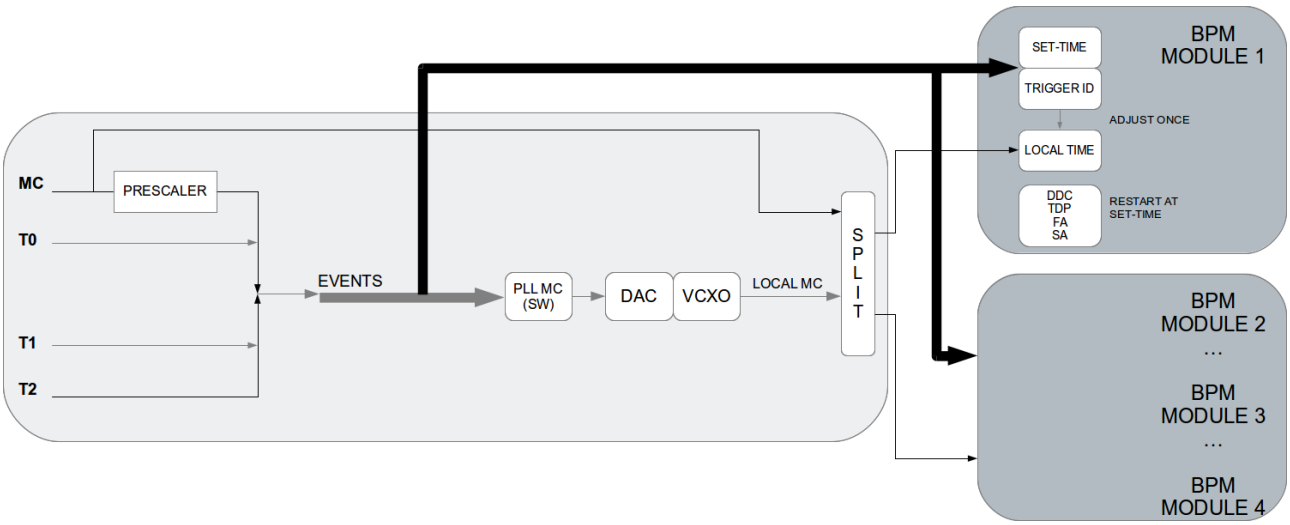

Figure 57: Synchronization scheme

The instrument is usually configured for the storage ring frequency. Reconfiguration is possible at any time by using the `/opt/libera/bin/libera-ebpm-configure` script. Contact support@i-tech.si for details.

NOTE: Information about the user specific clock domain frequencies can be found in the <ACCELERATOR.pdf> file.

3.7.2 Synchronization state machine

The synchronization state machine enables the control application to easily monitor the local and global synchronization state of all modules.

After the instrument's boot-up, each BPM module and GDX module in each Libera Brilliance+ is in "NoSync" state. In this state, timestamps of the BPM modules and the GDX module in a single instrument are aligned (synchronous). Timestamps across multiple platforms are not synchronous.

The synchronization procedure starts with synchronization announcement. The synchronization announcement will automatically change the status from "NoSync" to "Tracking" on all modules (BPM modules and GDX). This is marked as circle 1 in Figure 58. In this state, it is possible to check if the synchronization announcement was set to all.

After the synchronization trigger arrives, the state will automatically change from “Tracking” to “Synchronized” (circle 2) if synchronization was successful. If synchronization was not successful, the state will change back to “NoSync” (circle 4).

The “Synchronized” state is the final state where the synchronization was successful. The state “Synchronized” can be manually changed to “Tracking” (not recommended, unless required for specific purpose – circle 5) or it will automatically change to “NoSync” if the MC PLL gets unlocked (circle 3). The MC PLL gets temporarily unlocked if the offset tune has changed or if the MC signal is invalid.

See Figure 58 for graphical presentation of the Synchronization State Machine functionality.

The Synchronization State Machine is important for fast feedback operation or machine studies, where synchronization state is essential.

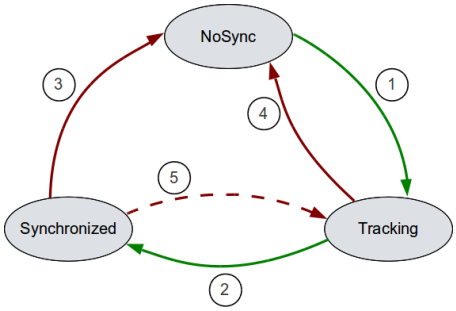

Figure 58: *Synchronization state diagram*

Synchronization state machine status is reported in node (module=gdx1, kraf3, kraf4, kraf5, kraf6):

```
libera-ireg boards.module.local_timing.sync_state_machine
```

3.7.3 Synchronization procedure

When instrument (or libera-ebpm application) starts, it automatically synchronizes the LMTs of all local BPM modules in the platform to LMT=0 by using internal trigger. To synchronize the processing and data acquisition timestamps across multiple platforms, a proper global synchronization procedure must be done.
Pre-requisites:

- MC PLL must be locked
- Trigger (T2) must arrive to all platforms synchronously

During the synchronization procedure, monitor the synchronization state on all modules (see Chapter 3.7.2).

The synchronization procedure:

- Turn the Trigger OFF (in the timing system)
- Announce the synchronization

```
libera-ireg application.synchronize_lmt=0
```

The application is waiting for the Trigger now. The Synchronization State Machine will set “Tracking” state to all modules (BPM and GDX). Check the state on all modules. It takes less than a second to change the state but due to network load, the synchronization announcement could take a few seconds.

- Turn the Trigger ON. The first trigger signal after the synchronization announcement is automatically considered as the synchronization trigger. It starts the internal synchronization process which takes few seconds (see explanation below for more details).
- After synchronization is successful, the Synchronization State Machine will be automatically set to “Synchronized” state. The LMT is reset on all modules.

NOTE: If clients are reading the data (e.g. turn-by-turn) continuously on trigger, the data will not refresh on the synchronization trigger but will refresh on the first next trigger again.

Details about the synchronization process:

- All processing chains are stopped: turn-by-turn, FA and SA data
- Clear (invalidate) SW LMT associations with ADC partitions
- Set the initial LMT to 0
- Schedule event generation to start turn-by-turn, FA and SA data simultaneously.

Restart of data processing chains is essential for alignment of the phase of all the chains involved (see Figure 59). This takes about 400-500 milliseconds.

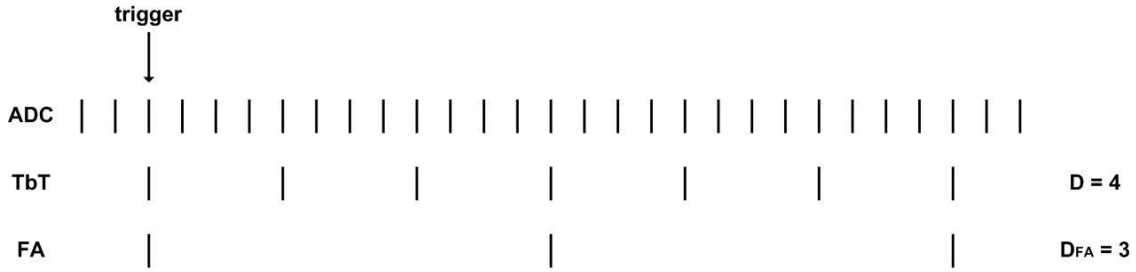

Figure 59: *Phase of ADC, turn-by-turn and FA transmissions relative to the trigger.*

NOTE: Since the processing chains are restarted during the synchronization process the data is temporarily not available.

3.7.4 Trigger delay

Trigger pulse is used to trigger data acquisitions and for synchronization of multiple modules and platforms. Due to different cable lengths and physical location of the platforms in the ring, the trigger arrival to multiple platforms may be different.

Trigger arrival can be delayed in every BPM module individually. It is set in steps of 1 ADC sample (1/adc_frequency), which is approximately 8.5 ns.
Example how to set the trigger delay:

```
libera-ireg boards.bpm.local_timing.trigger_delay=10
```

The trigger delay works as a real physical delay (e.g. longer cable) of the input trigger pulse.

NOTE: Trigger delay affects all functionalities of the trigger.

3.7.5 Offset-tune

The ADC sampling clock is a multiple of the revolution frequency. The sampling clock is locked to the revolution frequency (MC input) through a software PLL. The drawback is that the input signals are sampled in limited points. By offsetting the sampling clock for a few Hz, the sampling clock is not locked to the input signals anymore but it slowly drifts. The benefit is that the sampling is done in multiple points over the input signal.

In perfectly locked condition, the DDC brings the input signals into baseband to DC (for detail see Chapter 3.2.2 and 3.2.3). In slightly unlocked condition, the input signals are slightly offset from the DC. The offset from the DC is the “offset tune”. The actual turn-by-turn data rate also changes according to the offset tune value.

Offset tune is a user-configurable value. The software PLL tunes the sampling clock with a precision of about 1 Hz. An offset tune value sets the PLL to a different sampling clock. Maximum offset tune value is ±500 with step of 1.

The equation for the exact frequency change in the baseband is:

$$df_{BB} = -\frac{K}{D * P} f_{RF0}$$

where:
df_{bb} … final offset-tune in baseband [Hz]
K … offset tune value (units)
D … decimation from ADC to TBT
P … MC prescaler (= round (turn-by-turn frequency [Hz]/10[Hz]))
f_{RF0} … the RF frequency
Example for decimation 398 and offset tune value 25:

$$df_{BB} = -\frac{25}{398 * 27165} \times 352.055\ MHz = 814\ Hz$$

Offset tune is set in node:

```
libera-ireg boards.tim.pll.vcxo_offset
```

NOTE: The timing module provides the sampling clock to all BPM modules in the Libera Brilliance+ chassis. All BPM modules are offset-tuned for the same amount. Setting of the offset-tune value for each BPM module individually is not possible and even not reasonable.

With an offset tune applied, the data rates will slightly differ from the nominal value. However, to move the signal to the DC (as with offset tune=0), the calculation of the carrier on the NCO can be automatically adjusted. This function is called a double offset tune or “compensate offset”. It is set in node:

```
libera-ireg boards.tim.pll.compensate_offset=true
```

Usually, the compensate offset is enabled (set to »true«).

To present the effect of the offset tune and double offset tune, raw turn-by-turn data was acquired. Results are presented in Figure 60. The upper plot shows I and Q for channel A. Then, the offset tune was set to 25. This made I and Q detuned for approximately 1 kHz (see middle plot in Figure 60). In the lower plot, double offset tune was enabled. I and Q were tuned as in the first plot, but now with 1 kHz offset tune applied.

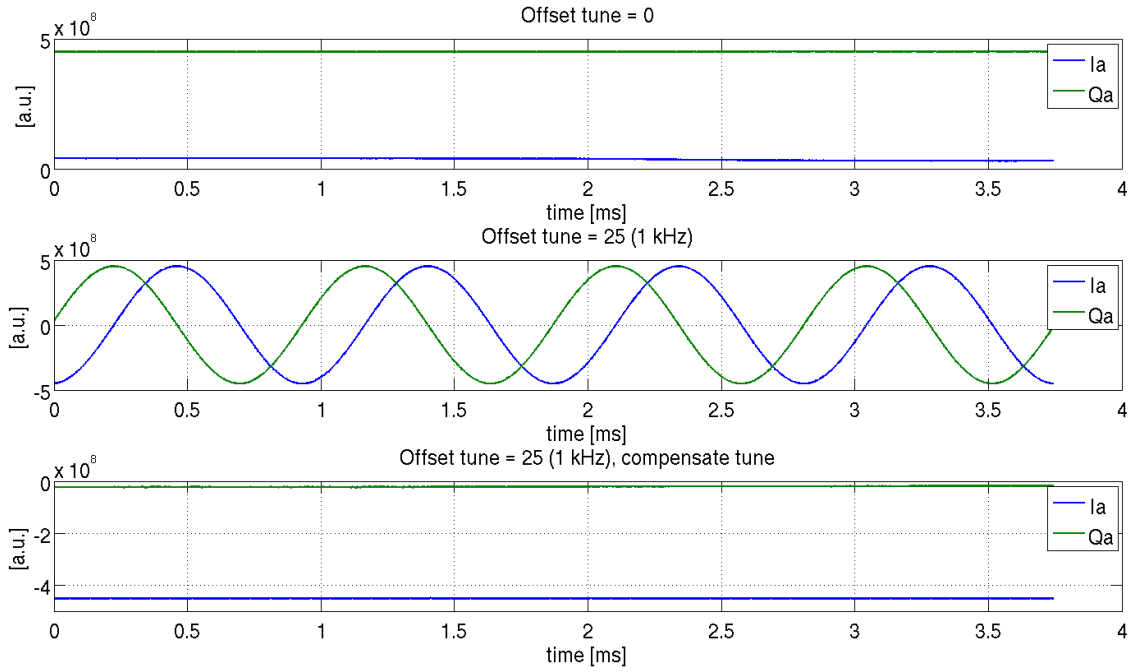

Figure 60: *Offset tune, double offset tune effect*

3.7.6 Signals and Events

The timing module receives up to 4 input signals from the electrical interfaces:

- MC – machine clock
- T0 – custom input/output
- T1 – postmortem
- T2 – trigger

These signals are distributed to satellite modules (BPM, GDX). The BPM module receives 3 (MC, T1, T2), the GDX receives all 4. Events that come over each of the input signals are uniquely equipped with event ID, count number and timestamp. Internally, the Libera Brilliance+ generates other events, such as Interlock, which are handled in the same matter.

Each of installed modules contains signals.event node which is a stream-type signal. The data rate of the stream is conditioned by the incoming events which are either periodical (e.g. injection trigger) or one from time to time (manual triggers).
Timestamps are all presented in LMT units.

Example how to read the event stream from the timing module (EvRx) and kraf3 module:

```
libera-ireg signal boards.tim.signals.event –s100 -v

libera-ireg signal boards.kraf3.signals.event –s100 –v
```

The event IDs are explained in Table 27.

***Table 27:** Event IDs*

event ID	Description
0	Event timestamp synchronization event
1	Postmortem event (PM or T1)
2	Trigger event (TRIG or T2)
3	Interlock event
4	Current time event
16	Machine clock event

3.7.7 MC PLL diagnostics

For large installations that use multiple Libera Brilliance+ platforms it is essential to keep the MC PLL locked with no interruption. Any interruption is detected by the Synchronization State Machine (see Chapter 3.7.2). The PLL runs as a software thread that receives prescaled MC events with approximately 10 Hz rate and corrects the sampling clock at same rate. During normal PLL operation and constant environmental temperature, the phase error value is usually in range of 2 clocks (absolute value). The PLL loop (PI) operates in 2 modes, coarse and fine, depending on the current error value.

There are 4 possible causes for the PLL unlock:

- PLL timeout
- Error > thr1
- Error > thr2
- Offset tune changed

There is a special signal stream available in the timing module that reads the status of the PLL:

```
libera-ireg signal boards.tim.signals.pll
```

It outputs 8 columns:

- Err: phase error
- Dac: current value applied to DAC
- Lock: PLL status (1 = locked, 0 = unlocked)
- ClkGood: (1=locked, 0 = error over threshold, 10 = PLL timeout, 20 = Err > thr1, 21 = Err > thr2, 30 = offset tune changed)
- Freq: current sampling frequency
- P: P-term of the PI controller
- I: I-term of the PI controller
- IsCoarse: PI controller mode (0 = fine, 1 = coarse)

The PLL loop latches the absolute maximum Err value since last Locked status:

```
libera-ireg boards.tim.pll.max_err
```

It must be reset manually:

```
libera-ireg boards.tim.pll.max_err.reset{}
boards.tim.pll.max_err.reset{}: done
```

In case the PLL unlock happens, the absolute (OS) time of unlock is reported in:

```
libera-ireg boards.tim.pll.os_unlock_time
```

It is reset manually:

```
libera-ireg boards.tim.pll.os_unlock_time.reset{}
```

3.8 Other timing module's functionalities

Timing module supports other functionalities, such as optical event reception and generation, and generation of electrical pulses. General overview of timing module’s functionalities is shown in Figure 61.

NOTE: The SFP link frequency must be set properly.

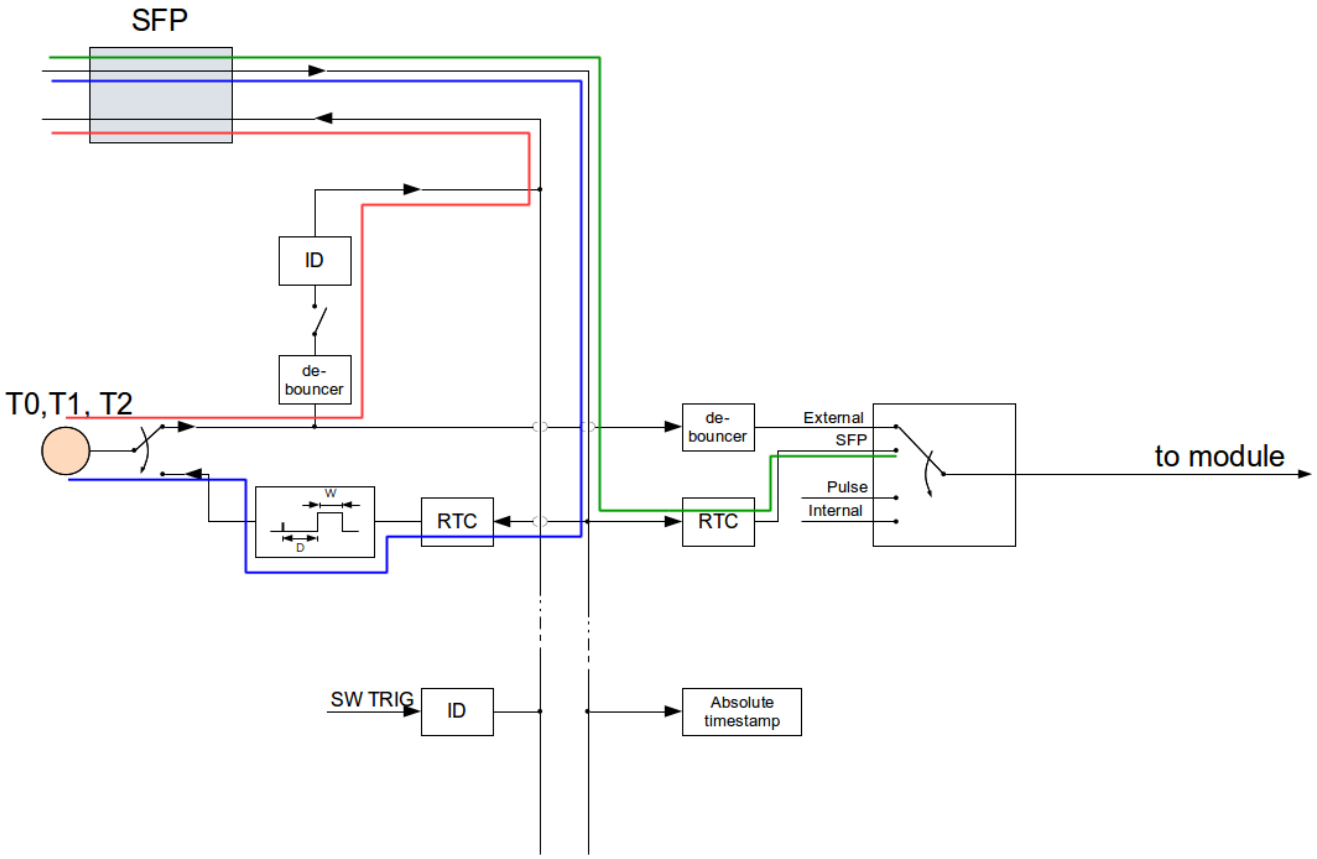

Figure 61: EvRx functionalities overview.

3.8.1 Trigger line source selection

MC, Trigger, Postmortem and T0 triggers can be provided through various sources as shown in Figure 62. Source can be selected for each of the triggers individually.

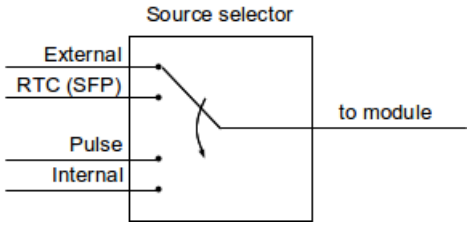

Figure 62: Trigger source selector.

Current configuration can be read from the registry:

```
libera-ireg boards.tim.triggers.{mc,t0,t1,t2}.source
boards.tim.triggers.mc.source: External
boards.tim.triggers.t0.source: Off
boards.tim.triggers.t1.source: External
boards.tim.triggers.t2.source: External
```

To switch the source of the MC, Trigger and Postmortem trigger from hardware signal to optical events:

```
libera-ireg boards.evrx.triggers.t2.source=RTC
```

To switch back to hardware signals:

```
libera-ireg boards.tim.triggers.t2.source=External
```

3.8.2 Event decoding

The decoding functionality is configurable for each trigger type (MC, T0, T1, T2) and for other functionalities (interlock by optical event, hardware signal generation) via Relevant Trigger Coding (RTC) tables – one per trigger type. The RTC consist of 3 arrays:

- Masking array
- Function array
- Code array

For triggers, each array contains 16 entries. For hardware signal generation and interlock events (see Chapters 3.5.4 and 3.8.6), arrays contain 4 entries and have no Code array.

The Masking array (**in_mask**) contains 16-bit entries that select the relevant bits from the 16-bit accelerator's timing system. The Function array (**in_function**) contains 16-bit entries that define the value of masked bits. The Code array contains 4-bit entries that define the internal (within the instrument) **code**.
The first entry in arrays have the highest priority in case of conflicts with other entries.

The transform function between the in_mask, in_function and code is:

```
trig_code[n] = S(n) if(M&MGT)==(M&F)

MGT = incoming event ID
M = in_mask
F = in_function
```

```
libera-ireg dump boards.evrx.rtc.t2
t2
  in_mask=[2,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65...
  in_function=[2,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65...
  code=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

To enable decoding, enable decoder switch:

```
libera-ireg boards.evrx.rtc.decoder_switch=on
```

Example
The following example explains how to link event ID=2 to Trigger signal.

First, set the T2 source to RTC:

```
libera-ireg boards.evrx.triggers.t2.source=RTC
```

Then, set the in_mask and in_function arrays to decode event ID:

```
libera-ireg boards.evrx.rtc.t2.in_mask[0]=2
libera-ireg boards.evrx.rtc.t2.in_function[0]=2
```

Leave the code array to 0. Enable decoder switch.

3.8.3 Reading raw event stream

Raw event stream is available to identify all events that are coming to the SFP. The in_mask and in_function arrays for the raw stream are available in stream_filter node:

```
libera-ireg dump boards.evrx.rtc.stream_filter
stream_filter
  in mask=[65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553...
  in_function=[65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553...
```

Default values (65535) will read all incoming events. When not used, the in_function array elements must be set to 0 to reduce the load of the event processing module.

Read the raw event stream:

```
libera-ireg signal boards.evrx.signals.evrx
```

The first column represents the event's ID. The second column is event's timestamp expressed in clock cycles.

3.8.4 Upstream Optical Event Generation

Timing module can be configured to send upstream optical events through SFP link upon electrical trigger detection (inputs T0, T1, T2). Input signal edge detection on trigger inputs T0-T2 (when configured as inputs) and MC is used as trigger for optical event sending. Events are logged in the event stream and equipped with LMT timestamp. Software switch on mgt_out controls the sending of the event. By default, it is enabled.
See Figure 61, the red line.
See procedure below as an example how to use the functionality.

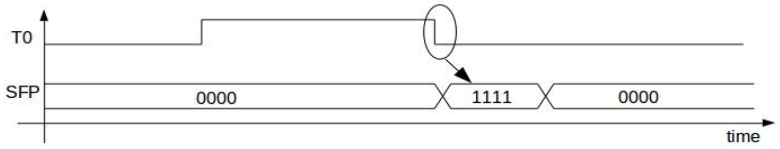

Figure 63: Send optical event with ID 0x1111 on T0 falling edge.

Configure T0 (T1, T2) as input:

```
libera-ireg boards.evrx.connectors.t0.direction=Input
```

Specify edge detection and upstream event ID:

```
libera-ireg boards.evrx.rtc.connectors.t0.edge.falling=true
libera-ireg boards.evrx.rtc.mgt_out=connectors
libera-ireg boards.evrx.rtc.connectors.t0.id=1111
```

Functionality is disabled by setting edge detection both to »falling«:

```
libera-ireg boards.evrx.rtc.connectors.t0.edge.falling=false
libera-ireg boards.evrx.rtc.connectors.t0.edge.rising=false
```

Arbitrary event ID triggered by a software command can be sent upstream through SFP link. Once the ID is set, event is sent immediately:

```
libera-ireg boards.evrx.rtc.sfp_tx.arbitrary.id=1001
```

Use the exectuable node to resend the event with the same ID again:

```
libera-ireg boards.evrx.rtc.sfp_tx.arbitrary.resend{}
```

In case of simultaneous triggers on T0-T2 inputs, priority is set by their numbers (0 is highest). Arbitrary event is sent out last.

NOTE: Setting node mgt_out to off disables all optical event sending, including interlock and arbitrary events. Use with caution.

3.8.5 Absolute timestamp

A reference timestamp can be set by pre-defined optical events from the event system. Event IDs are coded in hexadecimal. 32-bit shift register is loaded with 1 and 0 based on events received:

- event ID 0x60 loads a 0
- event ID 0x61 loads a 1
- event ID 0x6D resets register contents to 0

Event IDs are hard-coded and not configurable.

Shift register can only be read once it is full. After reset, reading will produce last full value until it fills again. Node 'state' monitors the filling.

Register fills continuously at LSB, so once it is full and new events keep coming, MSB will shift out. This does not influence state node, so shifting can be observed in ts_timestamp node:

```
libera-ireg  boards.evrx.rtc.{ts_timestamp,ts_timestamp.state}
boards.evrx.rtc.ts_timestamp: 0
boards.evrx.rtc.ts_timestamp.state: progress
```

3.8.6 Hardware signal generation

Connectors T0, T1 and T2 can be configured as an output. When an optical event with a pre-defined ID is received over the optical link, a hardware signal is generated on selected connectors. The electrical pulse has configurable properties as shown in Table 28.
Received optical events are decoded with a 16-bit mask. See Chapter 3.8.2 for more details on event decoding. Functionality is presented also in Figure 61, the blue line.

Table 28: Output signal properties.

Property	Value/Description
Signal active	selectable High/Low
Delay (optical to electric event)	0s <= t <= 1 s, configurable in f_{SFP} cycles
Duration of signal active pulse	100 ns <= t <= 1 s, configurable in f_{SFP} cycles
Maximum repetition rate	Apropriate regarding delay and duration
Optical event ID	0-65535
Decode mask	0-65535

Example how to set connector T0 as an output and enable optical event reception for signal generation:

```
libera-ireg boards.evrx.connectors.t0.direction=Output
libera-ireg boards.evrx.connectors.t0.out_type=SFP
libera-ireg boards.evrx.rtc.decoder_switch=on
```

Node 'direction' sets connector as output, 'out_type' sets it to transmit at optical event on SFP link, and 'decoder_switch' enables optical events decoding.

To set or check the values, issue:

```
libera-ireg dump boards.evrx.rtc.sfp_2_connectors.t0
   duration=0
   delay=0
   state=low
   in_mask=[65535,65535,65535,65535]
   in_function=[65535,65535,65535,65535]

libera-ireg boards.evrx.rtc.sfp_2_connectors.t0.duration=1005
libera-ireg boards.evrx.rtc.sfp_2_connectors.t0.delay=100
libera-ireg boards.evrx.rtc.sfp_2_connectors.t0.state=high
libera-ireg boards.evrx.rtc.sfp_2_connectors.t0.in_mask[0]=4444
libera-ireg boards.evrx.rtc.sfp_2_connectors.t0.in_function[0:1]=22222,11111

libera-ireg dump boards.evrx.rtc.sfp_2_connectors.t0
duration=1005
delay=100
state=high
in_mask=[4444,65535,65535,65535]
in_function=[22222,11111,65535,65535]
```

Repetition rate of the optical event should be chosen appropriately to delay and duration parameters. In extreme case (1 second delay, 1 second duration) it must be > 2 s. Any events received while the previous one is still in effect will be ignored.

3.9 Synthetic data generator

The fast data streams (FA, FDS) which are used for the fast orbit feedback purposes are sent to the GDX module over the LVDS links. The fast data stream contains standard fields (Va, Vb, Vc, Vd, SUM, Q, X, Y). For testing purposes, it is possible to replace the standard fields with synthetically generated values. The synthetic values can be either constants or a pre-defined waveform. The constant values are set for each field individually. Setting range is $\pm 2^{31}$.
Example how to set the constant value for the X position:

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.set_points.X=10000
```

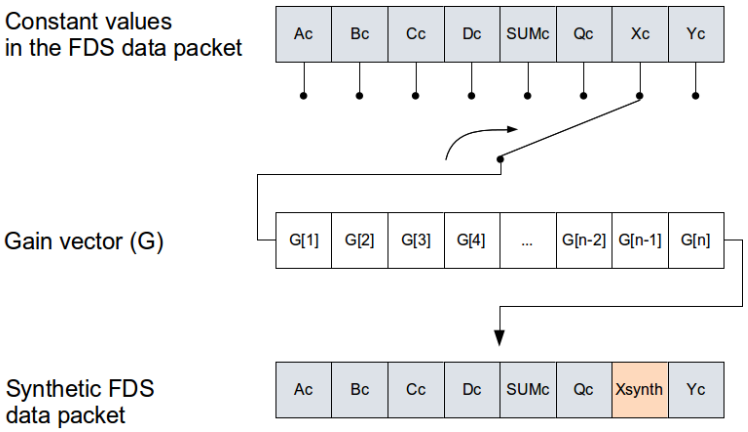

Figure 64: Synthetic data generator.

The waveform is defined with the gain vector (G). It contains 4096 elements. Each element (G0, G1, ..., G4095) contains a value in [-2048, 2047] integer range. The constant value (e.g. Xc) is multiplied by the gain vector elements.
The waveform can be applied to a single selected field as shown in Figure 64. The field is selected with a “mask” parameter which is given in 2^n values (bits). Values for specific fields:

Field	Va	Vb	Vc	Vd	Sum	Q	X	Y
Value	1	2	4	8	16	32	64	128

To set the waveform for position X:

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.mask=64
```

Example how to set the gain vector (must set all elements):

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.gain_coefs=100,101,105,120,90,200,…
```

Result is a synthetic data packet with a select field (e.g. X position) that follows the gain vector setting, see example in Figure 65.

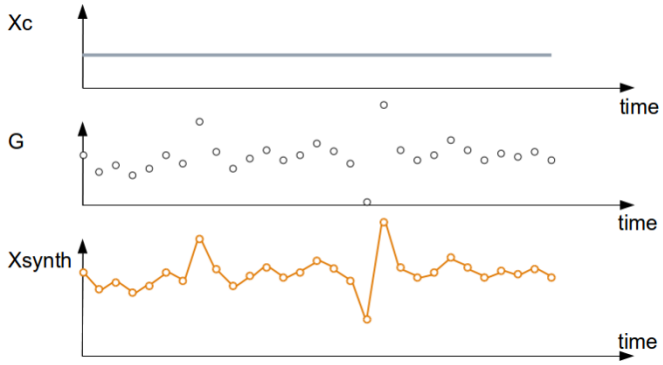

Figure 65: Synthetic waveform.

Once enabled, fields in the fast data stream contain the constant values (default is 0). When the waveform generation is enabled, it waits for the first trigger (T2). If no trigger is available, the waveform can be started manually. It is possible to roll-out the waveform once or repeat it for up to 4095 times. When the waveform generation is finished, constant values are kept.

The synthetic data is enabled with:

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.enable=true
```

Waveform generation is enabled with:

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.enable=true
```

To start the waveform generation without the trigger (T2):

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.apply{}
```

The usage example is shown in Figure 66. Constant value was set to 10000. The upper plot shows the gain vector (e.g. sine wave). The middle plot shows the output data when the gain vector rolls-out only once. It starts with constant value (10000), the gain vector then rolls-out once after the trigger/manual request and ends with the constant value. The lower plot shows an example when the gain vector repeats four times.
To set the number of repetitions (<4096):

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.repetitions=4
```

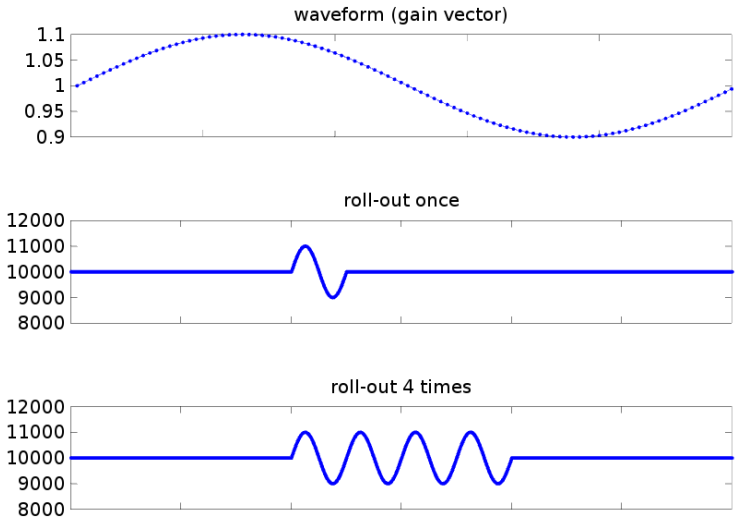

Figure 66: Synthetic data generation example.

Maximum gain vector length is 4096 elements. It is possible to limit the length of the gain vector. The example is shown in Figure 67. The original gain vector elements are marked in blue. The actually selected gain vector elements are marked with red points.
To limit the gain vector to a custom value (<4095):

```
libera-ireg boards.kraf3.lvds_tx.synth_generator.shaping.gain_coefs.size=1500
```

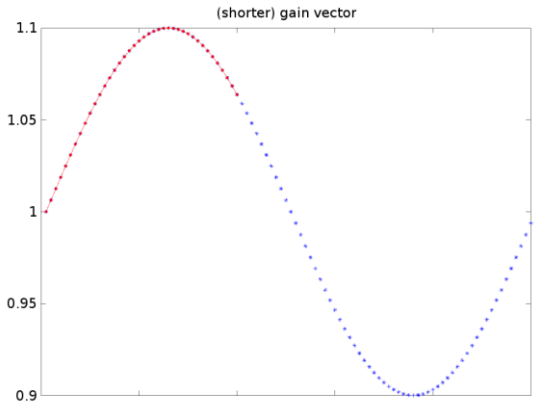

Figure 67: Shorter gain vector.

The complete list of parameters is shown below.

```
root@libera:~# libera-ireg dump boards.kraf3.lvds_tx.synth_generator.
synth_generator
  enable=false
  set points
    ChA=0
    ChB=0
    ChC=0
    ChD=0
    Q=0
    Sum=0
    X=0
    Y=0
  shaping
    enable=false
    apply
    repetitions=0
    mask=0
    gain coefs=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
      size=1
```

4. Usage instructions and diagnostics

4.1 Installation requirements

Libera Brilliance+ is intended to be mounted into a 19” rack. For temporary operation it can be also used as a desktop unit.

Rack installation requirements:

- Use the rack cabinet that allows proper ventilation. Incoming cool air enters the chassis through the left perforated panel with three fans. Warm air is leaving the chassis through the right perforated panel. Make sure at least 7 cm of free space is available at the sides (see Figure 68). A cabinet with no ventilation is not appropriate for continuous run. Outgoing air on the right panel needs to be properly guided out of the cabinet.
- The unit must always be fastened with all four screws. It is recommended that plastic washers are also used to protect the front panel from scratching.
- Multiple units can be stacked one above another into a rack cabinet with no additional spacing. In such case additionally check whether the cabinet ventilation is sufficient. The temperature in the cabinet must never exceed highest operating temperature. For optimal performances it is recommended that the temperature is not significantly higher than normal conditions room temperature.

Fully populated Libera Brilliance+ (4 BPM modules, timing module, GDX module) requires 30°C maximum environment temperature to keep all temperatures within non critical limits. Environment temperature of 25°C or less is recommended.

Temperatures apply for elevation up to 500 m (1600 ft). At higher elevation, environment temperatures must be lower. Consult Instrumentation Technologies support team at support@i-tech.si.

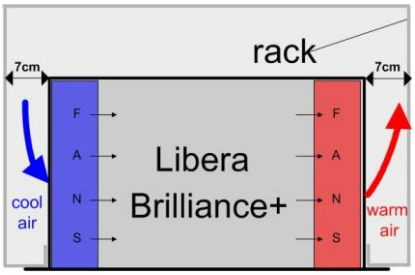

Figure 68: *Libera Brilliance+ mounted in a rack cabinet*

WARNING: Mains plug should be easily accessible to the user.

WARNING: The instrument must be grounded by using the PE terminal at the back panel of the device.

The minimum measurement setup should consist of the following:

- Libera Brilliance+
- RF input signals
- Machine Clock signal
- Ethernet connection

Schematically, the setup is shown in Figure 69. Such setup provides the user to test basically all data buffers and streams, except trigger-type buffers.

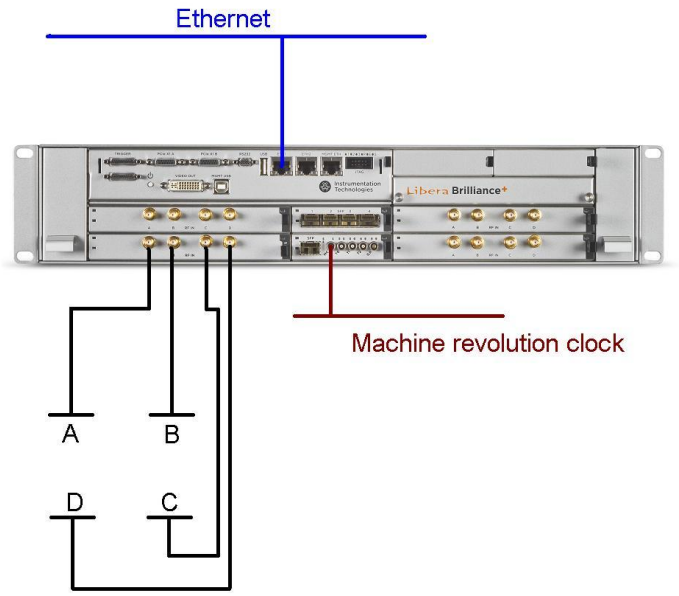

Figure 69: *Libera Brilliance+ minimum measurement setup*

For more advanced use, Trigger and Postmortem inputs can be connected. Trigger signal enables reading trigger-type buffers (ADC buffer, triggered turn-by-turn buffer) and do the synchronization.

4.2 Application daemon

The application daemon consist of the customer-specific FPGA design and customer-specific libera-ebpm application. Both are configured to run with machine-specific parameters such as RF and harmonic number. Versions of the FPGA and SW packages must be compatible to run properly. To check the application daemon version, do:

```
libera-ireg version
version: 3.6-250-r28892 thor
```

libera-ebpm application keeps customer-specific parameters in the registry. To check these parameters, do:

```
libera-ireg dump boards.bpm.clock_info.
clock_info
  adc frequency=119365611
  decimation
    tbt=86
    fa=138
    sa=1024
    sw=105
  harmonic number=360
  rf_frequency=499669999.53488374
  tbt_frequency=1387972.2209302327
```

Explanation of each node:

- adc_frequency: nominal ADC sampling frequency expressed in Hz.
- decimation.tbt: relation between the ADC and TBT data rate.
- decimation.fa: relation between the TBT and FA data rate.
- decimation.sa: relation between the FA and SA data rate.
- decimation.sw: relation between the TBT and switching frequency.
- harmonic_number: accelerator's harmonic number (HN).
- rf_frequency: accelerator's RF expressed in Hz.
- tbt_frequency: accelerator's revolution frequency (RF/HN) expressed in Hz.

4.3 Multi-user and multi-application simultaneous access

Digital signal processing and the software structure allow parallel access all data paths at the same time. User can monitor the beam position (Slow Acquisition), while the Fast Feedback is running (Fast Acquisition). At the same time, another user from another computer can do physics studies with turn-by-turn data. Third user can simultaneously monitor the ADC buffer. The only limitation is the network bandwidth and CPU and memory load.

4.4 List of parameters

List of important application specific parameters is shown in Table 29, Table 30, Table 31, Table 32 and Table 33. Parameters are grouped by function (signal conditioning, signal processing, interlock, etc.). Instructions how to set the parameters is described in Chapter 7.
There also parameters not listed in the tables. For more information please contact support@i-tech.si.

Table 29: Signal conditioning parameters.

Parameter	Chapter	Registry path (prefix: `boards.bpm.`) *bpm=kraf3,kraf4,kraf5,kraf6*	Properties	Auto save
Switching	3.3.2	`conditioning.switching`	r/w Bool	yes
Switching delay	3.3.2.1	`conf.switching_delay`	r/w ULong [0,decimation]	yes
Switching source	3.3.2.1	`conf.switching_source`	r/w Enum [Internal,External]	yes
DSC adjust	3.3.3.4	`conditioning.tuning.dsc.` `.coefficients.adjust`	r/w Bool	yes
DSC coefficients type	3.3.3.4	`conditioning.tuning.dsc.` `.coefficients.type`	r/w Enum [unity,adjusted]	yes
Att.dependent DSC coefficients	3.3.3.5	`conditioning.tuning.dsc.` `.coefficients.att_dependent`	r/w Bool	yes
DSC tolerance threshold	3.3.3.6	`conditioning.tuning.dsc.` `.quality.tolerance_thr`	r/w Double [0,100]	yes
AGC	3.3.1	`conditioning.tuning.` `.agc.enabled`	r/w Bool	yes
Power level	3.3.1	`conditioning.tuning.` `.agc.power_level`	r/w Long [-80,0]	yes

Table 30: Signal processing parameters.

Parameter	Chapter	Registry path (prefix: `boards.bpm.`) *bpm=kraf3,kraf4,kraf5,kraf6*	Properties	Auto save
FA data source	0	`tbt.data_type`	r/w Enum [DDC,TDP]	yes
Position calculation: Kx	3.2.1	`signal_processing.position.` `.Kx`	r/w ULong [1, 536870911]	yes
Position calculation: Ky	3.2.1	`signal_processing.position.` `.Ky`	r/w ULong [1, 536870911]	yes
Position calculation: Xoff	3.2.1	`signal_processing.position.` `.off_x`	r/w Long [-536870912, 536870911]	yes
Position calculation: Yoff	3.2.1	`signal_processing.position.` `.off_y`	r/w Long [-536870912, 536870911]	yes
Position calculation: Qoff	3.2.1	`signal_processing.position.` `.off_q`	r/w Long [-536870912, 536870911]	yes
Trigger delay	3.7.4	`local_timing.trigger_delay`	r/w ULong [0,32767]	yes
ADC mask	3.2.6	`tbt.adc_mask`	r/w ULong, Array [0,1]	yes
Phase offset	3.2.7	`tbt.phase_offset`	r/w ULong [0,decimation]	yes

Multiple ADC masks - offsets	3.2.8	`tbt.multi_bunch_positions.`
		`.bunch_0.offset`
		`tbt.multi_bunch_positions.`
		`.bunch_1.offset`
		`tbt.multi_bunch_positions.`
		`.bunch_2.offset`
		`tbt.multi_bunch_positions.`
		`.bunch_3.offset`
			r/w
			ULong
			[0,decimation-1]
				yes

Multiple ADC masks - window	3.2.8	`tbt.multi_bunch_positions.`
		`.bunch_0.window`
		`tbt.multi_bunch_positions.`
		`.bunch_1.window`
		`tbt.multi_bunch_positions.`
		`.bunch_2.window`
		`tbt.multi_bunch_positions.`
		`.bunch_3.window`
			r/w
			ULong
			[1,decimation]
				yes

Parameter	Chapter	Registry path (prefix: `boards.bpm.`) *bpm=kraf3,kraf4,kraf5,kraf6*	Properties	Auto save

ADC offset	3.3.4	`conf.adc_offset.A`
		`conf.adc_offset.B`
		`conf.adc_offset.C`
		`conf.adc_offset.D`
			r/w
			Long
			[-32768, 32767]
				yes

Spike removal (FA, SA)	0	`tbt.spike_removal.` `.mode.hw_enable`	r/w Bool	yes
Spike removal (DDC,TDP,FDS)	0	`tbt.spike_removal.` `.mode.sw_enable`	r/w Bool	yes
Spike removal start	0	`tbt.spike_removal.start`	r/w Long [-16,15]	yes
Spike removal window	0	`tbt.spike_removal.window`	r/w ULong [0,127]	yes
Spike removal averaging stop	0	`tbt.spike_removal.` `.averaging_stop`	r/w Long [-16,15]	yes
Spike removal averaging window	0	`tbt.spike_removal.` `.averaging_window`	r/w ULong [0,1,2,4,8,16]	yes

Table 31: Interlock parameters.

Parameter	Chapter	Registry path (prefix: `boards.bpm.`) *bpm=kraf3,kraf4,kraf5,kraf6*	Properties	Auto save
Interlock enable	3.5	`interlock. .enabled`	r/w Bool	No
Gain dependent mode	3.5.3	`interlock.gain_dependent. .enabled`	r/w Bool	Yes
Gain dependent threshold	3.5.3	`interlock.gain_dependent. .threshold`	r/w Long [-80,0]	Yes
Position limits	3.5.1	`interlock.limits.position. .min.x` `interlock.limits.position. .min.y` `interlock.limits.position. .max.x` `interlock.limits.position. .max.y`	r/w Long [-16777088, 16776960]	Yes
Saturation limits : threshold	3.5.2	`interlock.limits.overflow. .threshold`	r/w ULong [0,32766]	Yes
Saturation limits : duration	3.5.2	`interlock.limits.overflow. .duration`	r/w ULong [0,65535]	Yes
Saturation limits : mode	3.5.2	`interlock.limits.overflow. .mode`	r Enum [adc,tbt]	Yes
Filter on overflow	3.5.2	`interlock.filter.overflow`	r/w ULong [0,6]	Yes
Filter on position	3.5.2	`interlock.filter.position`	r/w ULong [0,255]	Yes

Table 32: Postmortem parameters.

Parameter	Chapter	Registry path (prefix: `boards.bpm.`) `bpm=kraf3,kraf4,kraf5,kraf6`	Properties	Auto save
Postmortem enable	3.6	`postmortem .capture`	r/w Bool	No
Postmortem source	3.6	`postmortem. .source_select`	r/w Enum [external,interlock,limits]	Yes
Buffer capacity	3.6	`postmortem. .capacity`	r/w ULong [0,524288]	Yes
Buffer written	3.6	`postmortem. .length`	r ULong	N/A
Postmortem offset	3.6	`postmortem. .offset`	r/w Long [-524288,524288]	Yes
Postmortem OS timestamp	3.6	`postmortem. .os_time`	r String	N/A
Position limits	3.6	`postmortem.limits.position. .min.x` `postmortem.limits.position. .min.y` `postmortem.limits.position. .max.x` `postmortem.limits.position. .max.y`	r/w Long [-16777088, 16776960]	Yes
Saturation limits : threshold	3.6	`postmortem.limits.overflow. .threshold`	r/w ULong [0,32766]	Yes
Saturation limits : duration	3.6	`postmortem.limits.overflow. .duration`	r/w ULong [0,4095]	Yes

Table 33: Common timing parameters.

Parameter	Chapter	Registry path	Properties	Auto save
Synchronization announcement	3.7.3	application.synchronize_lmt=0	r/w always set 0	N/A
Offset tune	3.7.5	boards.*tim*.pll.vcxo_offset	r/w Long [-500,500]	Yes
Compensate offset tune	3.7.5	boards.*tim*.pll.compensate_offset	r/w Bool	Yes
SFP link frequency	2.3.2.6	boards.*evrx*.sfp_freq	r [100000000, 125000000	Yes
Optical event decoding switch	3.8.2	boards.*evrx*.rtc.decoder_switch	r/w Enum [off,on]	No

4.5 Factory defaults and custom configuration

When the application daemon starts up, custom configuration is loaded to the vital parameters. If no custom configuration exists, factory defaults will be loaded instead. Some application parameters are essential for correct instrument's operation and must not be modified by the user runtime but only when application daemon is stopped. For these reasons, application parameters are marked with different properties:

- Readable: Runtime modification not allowed. Value can be modified only through custom configuration file.
- Writable: Runtime modification is allowed.
- Persistent: Parameter's value can be saved to customer configuration file.
- Hidden: Parameter is not shown in the registry tree.

After virgin installation (or when custom configuration file does not exist yet), custom configuration must be created in order to modify parameters marked as readable&persistent. Custom configuration file can be created:

- Automatically (on application exit)
- Manually (on user's request)

Automatic custom configuration file creation is enabled/disabled as shown below:

```
libera-ireg system.persistence.save_on_exit=true
libera-ireg system.persistence.save_on_exit=false
```

Custom configuration file is created manually as shown below. Application daemon can be running normally.

```
libera-ireg system.persistence.save{}
system.persistence.save{}: done
```

Table 34 shows file locations.

Table 34: Factory and custom configuration files.

Configuration type	File location
Factory default	`/opt/libera/sbin/cfg/`
Custom	`/var/opt/libera/cfg`

In case the custom configuration file gets corrupted or contains parameters that cause application daemon crash, factory defaults can be used. To use factory defaults:

- Stop the application daemon (if it runs)
- Erase `/var/opt/libera/cfg/libera-ebpm*` files
- Start the application daemon. It will load factory default settings.

Create custom configuration file (automatically or manually). Edit the parameters of interest.

5. Platform management

5.1 Power supply

Mains power supply connector is placed on the back panel of Libera Brilliance+. Libera Brilliance+ features standard power supply. The supply voltage and frequency are country dependent. Please check marking label on the rear panel for your supply type.

5.2 Power up and power down procedure

The instrument is powered up with the main power ON/OFF switch located in the back panel (see Figure 70). After turning the switch ON, the instrument starts booting up. Boot-up sequence:

- Boot of FPGA images in connected hardware modules: during booting-up, the green LEDs are blinking. After successful boot, the green LEDs remain lit.
- OS boot up: it starts booting up immediately after the FPGA images are boot-up. One can notice the activity in LEDs in the ICB. ETH interfaces start the connections.
- Start of Libera software: libera daemons are loaded after the OS is ready.

Complete boot-up procedure lasts between 2 and 3 minutes.

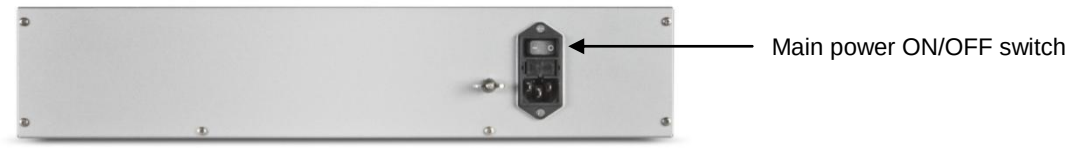

Figure 70: Main power ON/OFF switch

The power down procedure consists of 2 important steps that must follow in correct sequence:

- ICB shut down
- instrument shut down

The ICB must be shut down using the ICB power button, circled green in Figure 71. This will safely shut down the Libera software and OS. The ICB power OFF takes some seconds, typically under 10 seconds. After ICB power OFF, the blue LED on ICB starts blinking.

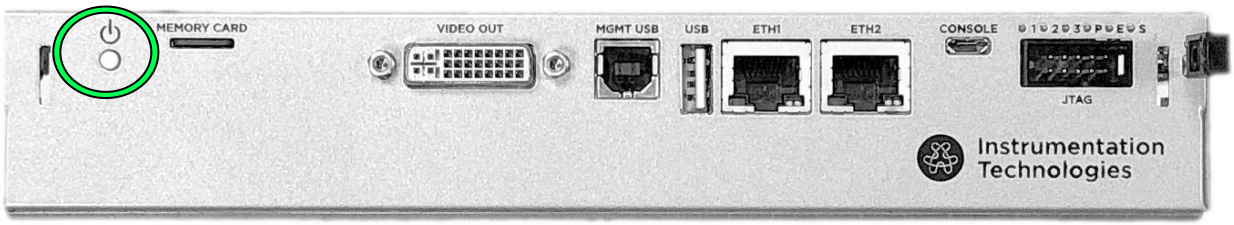

Figure 71: ICB power button

After successful ICB power OFF (blue LED on ICB is blinking), the instrument can be completely shut down with main power switch (see Figure 70).

5.3 Hardware power cycle and remote power cycle

Libera Brilliance+ can be power cycled with ICB power button (see Figure 71). To shut the Libera software and OS down, press the button once and wait for the blue LED in the ICB to start blinking.
To boot the OS and the Libera software up, press the ICB power button once and wait for the boot procedure to finish (2-3 minutes).
Libera Brilliance+ can be power cycled also remotely. To power cycle the instrument, use the following command:

```
libera-bmc --power --cycle --set
```

Remote power cycle is equivalent to the hardware power cycle.

Note: Issuing “reboot” command (Linux) does not power cycle the instrument with its modules but only the OS. After Linux reboot, the instrument will not operate normally.

5.4 Connecting to Libera Brilliance+

Libera Brilliance+ can be accessed through Ethernet connection or directly with keyboard and external display (console connection).

5.4.1 Login information

There is one full-access and one standard-access user enabled in the OS. Log in information is given in Table 35.

Table 35: Login information.

User	Password
root	Jungle
test	libera1

The “root” has full access to complete software structure. Be careful when modifying basic settings as this may result in a crash of the instrument.

5.4.2 Console connection

Console connection is mainly used for troubleshooting in case of network configuration problems. Use a standard micro USB cable. Connection can be established with standard console utilities (e.g. minicom, gtkterm, etc.), see connection parameters in Table 36.

Table 36: Console connection parameters.

Parameter	Value
Speed	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	None

5.4.3 Local connection (display, keyboard)

Local connection is mainly used for troubleshooting in case of network configuration problems. To access the instrument directly, a USB keyboard and display is required (see Figure 72). The instrument doesn't require a network connection.

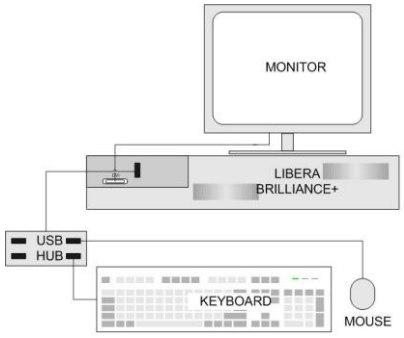

Figure 72: *Local connection scheme.*

5.4.4 Ethernet connection

Libera Brilliance+ is a network attached device and requires a properly configured network interface (IP address and network mask). From this point of view, the instrument is equivalent to any PC running Ubuntu Linux distribution.

Libera Brilliance+ should be connected to the network through "ETH1" interface in the ICB's front panel.

Note: By default, Libera Brilliance+ attempts to configure its network interface using the DHCP. If no DHCP server is found in 60 seconds, it assigns a localhost (127.0.0.0) and is accessible through a local or console connections only.

MAC address of the network interface is labeled on a sticker on a Libera Brilliance+ chassis. User's system administrator should map the MAC address in the DHCP server.
Static IP address can be set manually. The Libera Brilliance+ has all the usual interfaces, commonly found in PCs. Libera Brilliance+ uses standard Ubuntu Linux software structure.

5.4.5 Adding a new user

In addition to 'root' and 'test' users, a new user can be added with the following command:

```
useradd newuser
```

where newuser is the user name. This user will login without a password. The password can always be added using "passwd" command.

5.4.6 Setting the network environment

Network settings are in located in the `/etc/netplan/01-netcfg.yaml` file. Default network settings are shown below. By default, the instrument will try to configure the network interface using the DHCP.

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      dhcp4: yes
      dhcp-identifier: mac
```

For static IP configuration the file must be properly edited. Example below shows how to configure the IP address to 192.168.1.100 with gateway 192.168.1.1 and DNS server as 8.8.8.8 and 8.8.4.4:

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
     dhcp4: no
     addresses: [192.168.1.100/24]
     gateway4: 192.168.1.1
     nameservers:
       addresses: [8.8.8.8,8.8.4.4]
```

After the file is edited and save, apply changes with:

```
sudo netplan apply
```

In case of issues, check the status with:

```
sudo netplan --debug apply
```

5.5 OS system backup and restore

The OS and other software in Libera Brilliance+ is installed in the microSD card. Backup and restore procedure will be shown for Linux (Ubuntu) OS. In other operating systems, please use standard utilities for creating and restoring microSD card image.

To create a backup image:

- Power off the Libera Brilliance+ and unplug the microSD card
- Plug in the memory card to computer

- Unmount the memory card (example for `mmcblk0`):

```
umount /dev/mmcblk0p1
```

- Create an image from the memory card to “backup_image.img”:

```
sudo dd if=/dev/mmcblk0 of=backup_image.img
```

- Unmount the memory card and unplug from computer.

To restore an existing backup image:

- Plug in the microSD memory card to the computer. Unmount the memory card (example for `mmcblk0`):

```
umount /dev/mmcblk0p1
```

- Transfer the image from computer to memory card (it takes several minutes):

```
sudo dd if=backup_image.img of=/dev/mmcblk0
```

- Unmount and unplug the memory card
- Plug it into Libera Brilliance+ and power up the instrument

Empty OS image (with no Libera software) is available on request. Please contact support@i-tech.si.

5.6 IPMI

The Intelligent Platform Management Interface (IPMI) is an open standard for monitoring, logging, recovery, inventory, and control of hardware. It is implemented independently of the operating system. The service processor (or BMC) is the brain behind platform management and its primary purpose is to handle the autonomous sensor monitoring and event logging features.

A simple command-line interface, called “libera-bmc” is implemented in Libera Brilliance+. It is based on the IPMI standard and enables the user to monitor the boards’ and sensors’ status and to monitor and configure the SEL and FPGA image.

The list of commands and examples of usage can be found in Appendix B: “libera-bmc” command line utility.

5.7 Platform daemon

The platform daemon is an application independent daemon that enables monitoring and setting the hardware specific data. These data are read through the MCI interface (see Chapter 2.5.1). These exposed resources can be used by remote applications and they can be shared among many applications at the same time.

The platform daemon provides the following functionalities:

- Health (temperature, voltage, current, memory, CPU) monitoring
- Temperature damage protection
- Persistent logging and notification mechanism

- Regular update of time in the firmware
- Remote network interface to platform information

For all options check the content of the `libera-platformd.xml` configuration file, which is located in `/var/opt/libera/cfg` folder.

5.7.1 Health monitoring

Sensors in the platform's modules can be continuously monitored and logged to files. Monitoring period is configured in the `libera-platformd.xml` configuration file and is 60 seconds by default.
Configuration options:

- Enable/disable: Logging can be enabled or disabled. Default setting is enabled.
- Include/exclude list of sensors: There is the list of sensors that are included or excluded for monitoring. By default, all sensors are monitored.
- Absolute value logging: sensor values are logged only when one of the tracked sensors changes above the HNC or below the LNC value. One entry only is logged also when this sensor goes back to the normal operating value. This filter does not work for sensors which do not have HNC and LNC limits defined. By default, absolute value logging is enabled.
- Relative value logging: The sensor values are logged only when the difference between new and previous value of the tracked sensor is more than X% of its nominal value. This filter does not work for sensors which do not have nominal value defined. By default, relative value logging is disabled.

When any sensor exceeds the HNC, HC or HNR or LNC, LC, or LNR a warning is sent to the Status & Errors log file (see Chapter 5.7.5.1). When the sensor value returns to its normal value this information is sent to the same log file.

5.7.2 Fans operation

There are 2 sets of 3 fans used to deliver fresh air to and extract warm air from Libera Brilliance+ chassis. Fresh air enters the chassis on its left side (3 fans) and exits the chassis on its right side (3 fans). The rotation speed is controlled by the platform daemon which monitors the temperature sensors.

Fans are labeled according to their position in the chassis: left/right, front/middle/rear. To read their rotation speed, issue the following commands:

```
libera-ireg -P fans.left_front
fans.left_front: 3982
libera-ireg -P fans.left_middle
fans.left_middle: 4023
libera-ireg -P fans.left_rear
fans.left_rear: 4032
libera-ireg -P fans.right_front
fans.right_front: 3990
libera-ireg -P fans.right_middle
fans.right_middle: 3998
libera-ireg -P fans.right_rear
fans.right_rear: 4015
libera-ireg -P fans
fans: 4009.5
```

The last in the series of commands returns the average rotation speed. To change the average rotation speed, issue the following command:

```
libera-ireg -P fans=4500
```

The minimum rotation speed is limited to 2500 rpms. The maximum value is approximately 5500 rpms.

NOTE: If the platform daemon shuts down for any reason, rotation speed is set to maximum value

NOTE: If the temperature control is enabled (=true) the fan speed setting will be overridden by the platform daemon. To set the fan speeds manually, the temperature control must be set to false.

5.7.3 Temperature control

Temperature can be automatically controlled by a health daemon which is part of the platform daemon and monitors the temperature sensors of installed boards/modules. It can monitor all or just selected temperature sensors.
Temperature control is done by controlling the rotation speed of the fans. The control is done manually by the user or automatically by the platform daemon.
Configuration of the temperature control is set through the look-up table:

- 4000 rpm at temperatures below 57 °C
- 5000 rpm at temperatures between 65 °C and 70 °C
- Full speed at temperatures above 70 °C

There is a certain hysteresis defined for each temperature level. Please see Figure 73 for graphical representation.

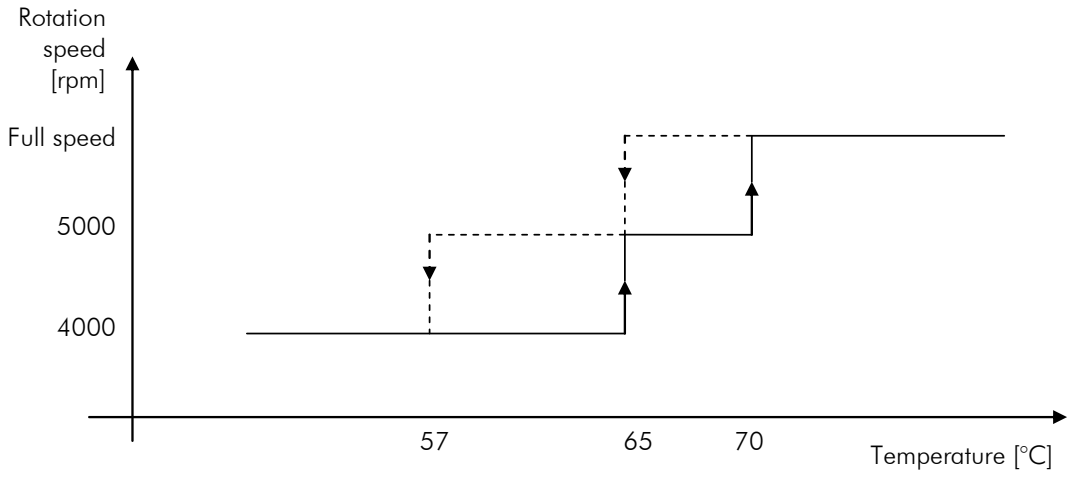

Figure 73: *Temperature limits with hysteresis*

Manual control of the fan rotation speeds is only allowed when the temperature control is disabled in the `/var/opt/libera/cfg/libera-platformd.xml` file. By default, the health daemon is **disabled**.

5.7.4 Temperature damage protection

The platform daemon implements a damage protection mechanism that prevents the damage of the boards caused by the overheating. When the temperature of any sensor exceeds the HNR value, the platform daemon shuts down the module on which the sensor in question is located. If the temperature measured on the ICB is too high, the whole instrument shuts down. The HNR temperature at which the board shuts down is 85°C. Other limit temperatures are:

- HNR = 85°C, shut down the board (the whole instrument)
- HNC = 70°C, set fans to full speed
- HC = 80°C
- LNC = 10°C
- LC = 5°C
- LNR = 0°C

Before shutting down any of the modules or the whole instrument, a warning is written to the Status & Errors log file (see Chapter 5.7.5.1).

Following options are available for the damage protection:

- Enable/disable: damage protection can be enabled or disabled. It is enabled by default.
- Include/exclude list of sensors: There is the list of sensors that are included or excluded for monitoring. By default, all sensors are monitored.

NOTE: If any sensor is detected broken, it should be put to the exclude list, so the daemon will not track its value. In this way, unwanted shutting down of the boards is prevented. User must understand if the instrument can operate safely with this broken sensor.

5.7.5 Sensors

The platform daemon uses an IPMI standard feature SEL to perform the monitoring and logging of the values from all sensors. The purpose of persistent logging is the support of the instrument as it enables the identification of unusual situations and the reasons of potential instrument's problems.

5.7.5.1 Logging of persistent sensors

The sensor values are written to two different types of log files:

Status & Errors
General status of the daemon, warnings and errors are written here. These logs are located in /var/opt/libera/log/ in the file libera-platformd.log.
The entries in the Status & Error log file have one of the following prefixes that enable easy searching inside the file:

- Info
- Warning
- Error
- Critical

The user application has the possibility to subscribe to notification mechanism, which notifies about the changes in certain sensors or about monitored sensor reaching the critical value.

Sensor values
Sensor values from all boards are written here. Each board has its own log file <board Id>.log located in /var/opt/libera/log/sensors/.

Sensor values are written in the csv format (columns delimited by semicolon) – one line for all sensors per board at a given time. Sample output of sensor data logging:

[TIME];[0]MAX6698;[1]PLX;[2]POWER.TEMP1;[3]POWER.TEMP2;[4]AIR.FLOWIN
[11:59:03.843022123] ;31;39;35;32;31
[12:00:04.523010090] ;31;39;35;31;31
[12:01:05.199035458] ;31;39;35;31;31
[12:02:05.873002470] ;31;39;35;32;31
...

The log rotation mechanism is handled by log rotate daemon. This is a standard approach for Linux log file handling.

5.7.5.2 Sensors in the modules

Every module contain several temperature, voltage and current sensors. Each sensor has its own short enumeration in the registry tree. Description of temperature sensors is presented in Table 38, Table 39, Table 40 and Table 41. Description of voltage and current sensors is not given in this chapter. Please contact support@i-tech.si to get more information.

The read full list of sensors and their values, use this command:

```
libera-bmc --sensor --read --board=X -v
```

X is the module identification number:

- 0: ICB
- 1: GDX
- 2: EvRx
- 3-6: BPM

The output of the reading provides information (in columns) presented in Table 37:

Table 37: Sensor read-out information.

Column	Description
ID	Sensor ID
NAME	Sensor name
TYPE	Sensor type (temperature, voltage, current, other)
UNIT	Unit of a sensor value (deg.C, V, A, other)
NOM	Nominal value
LNC	Low non-critical value
LC	Low critical value
HNR	High non-recoverable value
HC	High critical value
HNC	High non-critical value
TIME	Absolute time of readout
CONV(RAW)	Converted (raw) value

Table 38: Sensors in ICB

Sensor ID	Sensor name	Description
0	MAX6698.LOCAL	Internal chip temperature. This chip controls sensors.
1	PLX	Temperature of the PCI Express switch.
2	POWER.TEMP1	Temperature around highly loaded power supplies.
3	POWER.TEMP2	Temperature around highly loaded power supplies.
4	AIR.FLOWIN	Temperature of the (board) intake airflow
5	AIR.FLOWOUT	Temperature of the (board) outtake airflow
6	BACKPLANE	Temperature of the exhaust airflow from the instrument's power supply.

Table 39: Sensors in the GDX module

Sensor ID	Sensor name	Description
1	MAX6698.LOCAL	Internal chip temperature. This chip controls sensors.
2	FPGA	Temperature inside FPGA chip.
3	POW.SUPP.TEMP	Temperature around highly loaded power supply.
4	CENT.BOARD TEMP	Temperature in central part of the board (no specific chip).
5	MGT SUPP.TEMP	Temperature of the SFP power supplies.
6	AIRFLOW	Temperature of the airflow passing through the board.
7	DDR3 MODULE	Temperature of the DDR3 memory (placed below the memory chip).

Table 40: Sensors in the timing module

Sensor ID	Sensor name	Description
1	MAX6698.LOCAL	Internal chip temperature. This chip controls sensors.
1	SFP CAGE	Temperature around the SFP cages.
2	SI571	Temperature of the Si571 VCXO.
3	VX-5011	
4	POW.SUPP.	Temperature around the power supplies.
5	CDCE7201	Temperature around the PLL clock synthesizer.
7	AIRFLOW	Temperature of the airflow passing through the board.

BPM module contains 8 high precision temperature sensors (ts1-8, PT-1000 type) that are placed nearby the RF chains. Other sensors report temperatures of the FPGA, airflow and others. See Figure 74 for details.

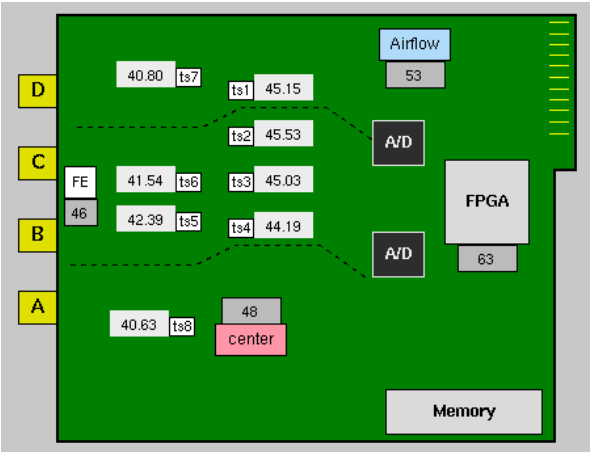

Figure 74: Temperature sensors in the BPM module.

Table 41: Standard temperature sensors in the BPM module.

Sensor ID	Sensor name	Description
1	MAX6698.LOCAL	Internal chip temperature. This chip controls sensors.
2	TempPS2	Temperature nearby power supplies.
3	TempPS1	Temperature nearby power supplies.
4	FPGA	Temperature inside FPGA chip.
5	AirFlow	Temperature of the airflow passing through the board.
6	FarEnd	Temperature at the front panel.
7	BrdCenter	Temperature in the board center (approximately).

5.7.6 Absolute time synchronization in satellite modules

Every module (ICB, GDX, EvRx, BPM) runs its own module time. The module's absolute time is not as accurate as the OS time which is controlled by a NTP. Platform daemon issues modules' times synchronization periodically every 7 days (default). SEL log entries are timestamped with modules' times.

6. Standard testing procedure

6.1 Factory acceptance test (FAT)

Each Libera Brilliance+ goes through routine production testing (at the hardware manufacturer) and through final performance tests. Each software release passes the functional and regression tests. A standard testing procedure is routinely performed on each instrument before shipping. The testing setup is standardized and located in stable environment for best reproducibility. Test procedures are automatic to exclude human errors. As a result, a standard test record is issued and archived. Content of the test record is shown in Chapters 6.2.

6.2 Testing setup

Test setup is shown in Figure 75.

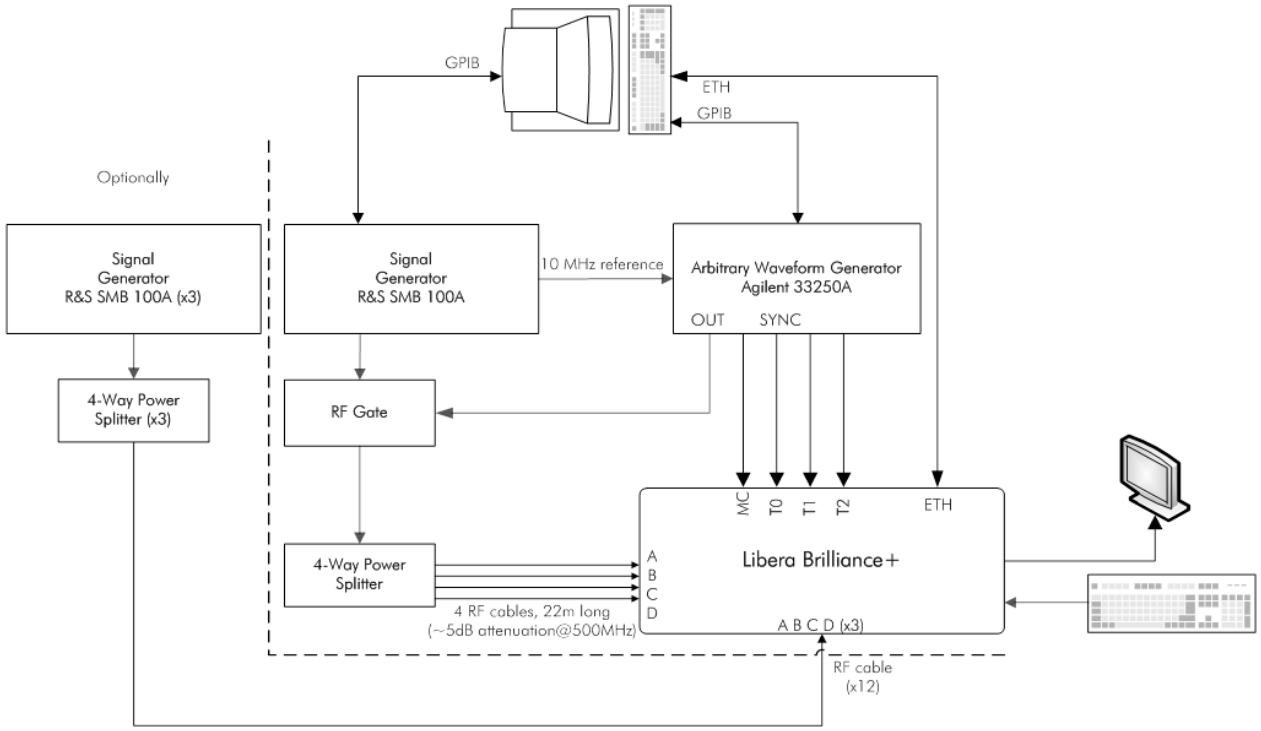

Figure 75: Test setup block diagram.

6.2.1 List of standard tests with testing criteria

Standard test uses pass/fail logic. Instruments typically outperform pass criterion by a factor of 2. The RMS performance depends on the bandwidth of the turn-by-turn data. Table 42 presents the specified RMS limit values for the FA data (~10 kS/s) and TBT DDC data. Specified limits for the TBT DDC data are split to three ranges that depend on the revolution frequency.

Table 42: Measurement uncertainty / noise.

Data type	FA data fs=~ 10 kS/s, BW=2 kHz	TBT DDC data fs=MC, BW=~0.3*fs		
Settings	DSC=ON AGC=ON	DSC=OFF AGC=OFF		
		MC < 2 MHz	2 MHz < MC < 6 MHz	fs > 6 MHz
Power [dBm]*	**RMS limit [µm]**	**RMS limit [µm]**		
0	0.25	2	8	10
-2	0.25	2	8	10
-4	0.25	2	8	10
-6	0.25	2	8	10
-8	0.25	2	8	10
-10	0.25	2	8	10
-12	0.25	2	8	10
-14	0.25	2	8	10
-16	0.25	2	8	10
-18	0.25	2	8	10
-20	0.25	2	8	10
-24	0.4	2	8	10
-28	0.4	2	8	10
-32	0.8	4	12	20
-36	0.9	4	12	20
-40	1.5	8	20	40
-44	2	10	40	50
-50	4	20	60	100
-56	8	40	120	200
-62	20	100	300	500
-68	30	200	600	1000
-74	60	400	1200	2000
-80	120	1000	2000	5000

* Indicates the power at the input, CW

If any measured parameter does not meet pass criterion, the instrument is non-compliant and requires hardware repair or replacement.

6.2.2 Site acceptance tests (SAT)

There are typically two types of SAT foreseen:

- Visual inspection, switch-on inspection (within 10 days after receipt of the goods).
- Repetition of test no.1: Measurement uncertainty/noise, using test setup described in Figure 75 (without the gate).

7. Maintenance

7.1 Hardware maintenance

7.1.1 Effect of dirty fan filters

Instrument is cooled by the environmental air. The airflow is generated by 3 inlet and 3 outlet fans. To prevent the dust accumulating on the electronics, the inlet fans contain a filter. Eventually, this filter will fill up with dust and reduce the inlet and outlet airflow. Consequently, the cooling will be less effective. Figure 76 shows the average fan rotation speed with a clean and a dirty fan filter. The reference picture of a dirty fan filter is given in Figure 77.

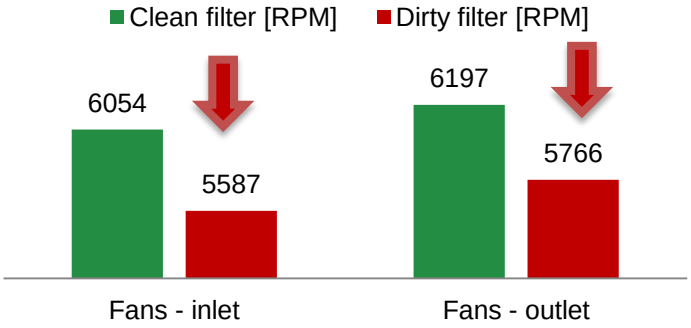

Figure 76: Degradation of fans' RPMs.

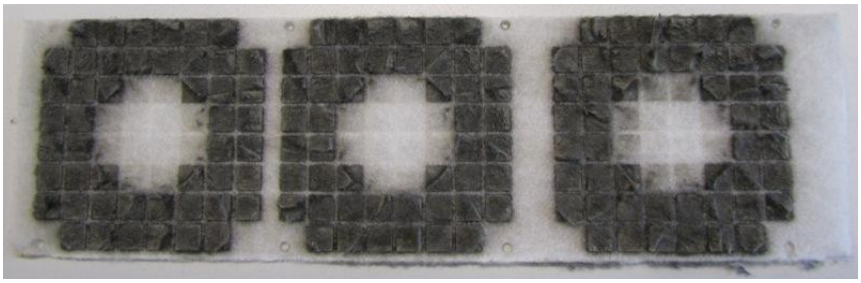

Figure 77: Dirty fan filter used for comparison test.

The test system consisted of Libera Brilliance+ with 4 BPM modules and a timing module. The system was running at 25° C environment temperature (in the temperature chamber) with full fan rotation speed.

The effect of a dirty fan filter on temperature is significant, see Figure 78. Chart shows the average temperature of selected temperature sensors on 4 BPM modules. Highest temperatures are always expected in FPGA and ADC regions (first four in the chart). Temperature increase was measured around 5° C – 8° C.

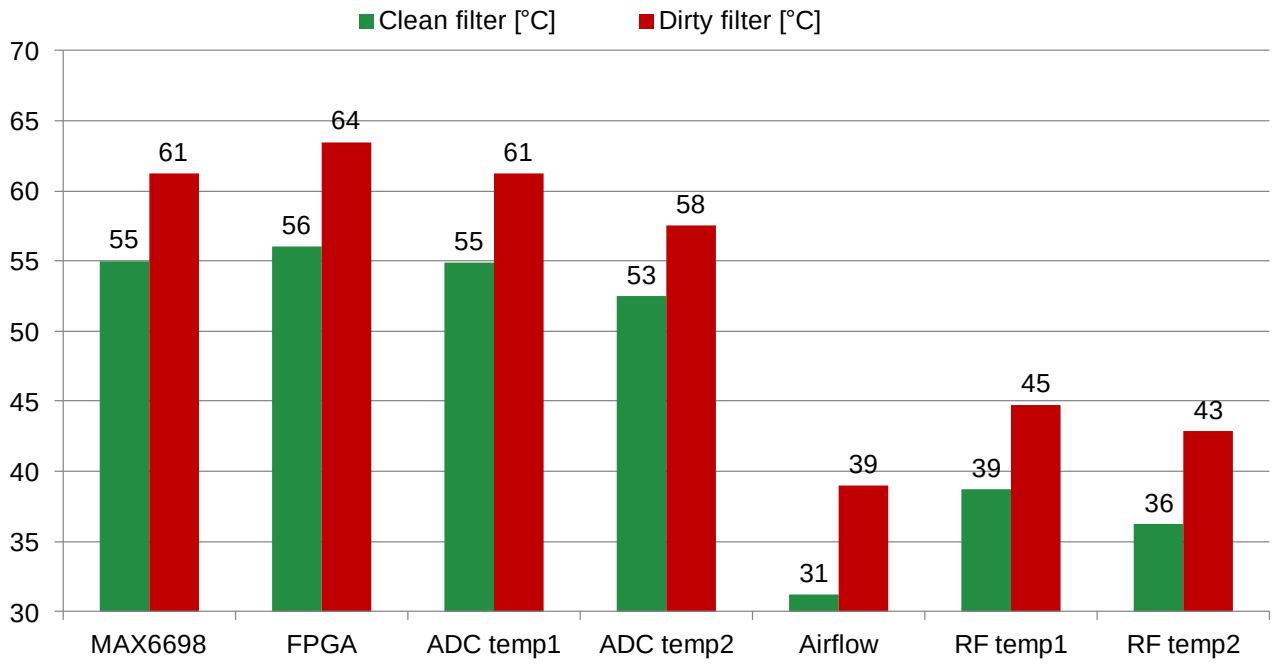

Figure 78: Temperature increase with dirty fan filter.

It is very important to do a regular check of the fan filter cleanliness. Once it shows a visible traces of dirt, it must be replaced by a clean one. For replacement procedure, see Chapter 7.1.2.

7.1.2 Replacing the fan filters

WARNING: Before replacing the fan filters, make sure Libera Brilliance+ is switched off and the power cord.

It is necessary to regularly check the fan filters. They should be inspected (and replaced) within 3 months after the first installation. Later, they should be replaced depending on their cleanliness, which depends mainly on the ambient air properties. Fans are located on both sides of Libera Brilliance+ if viewed from the front (see Figure 79).

***Figure 79:** Location of fans*

First, make sure that Libera Brilliance+ is switched off. The filter is located only in the left set of fans – that is where the fresh air enters into the instrument. To pull the fans out, use practical handles.

Figure 80: *Pull the left fans out*

Pull the left set of fans out completely and put it on a solid surface (Figure 80). Remove eight screws that hold the filter to the fans (Figure 81). Should any of the fans itself be wasted, it can be easily removed by just unscrewing it and fixing a new one in its place. Each fan is fixed with four screws.

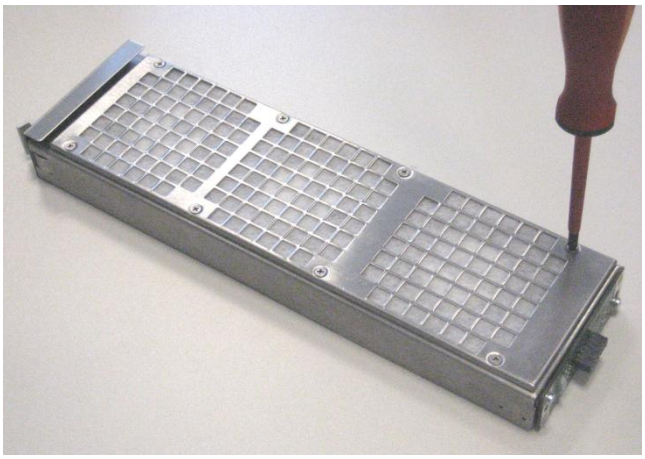

Figure 81: *Remove the screws*

After removing the screws, the old filter has to be replaced with a new one (Figure 82). A couple of new clean filters are enclosed with the instrument. The filter material is oil-wetted polyurethane foam, the type that is commonly used in various fans. Please use only enclosed filters, as the performance is guaranteed.

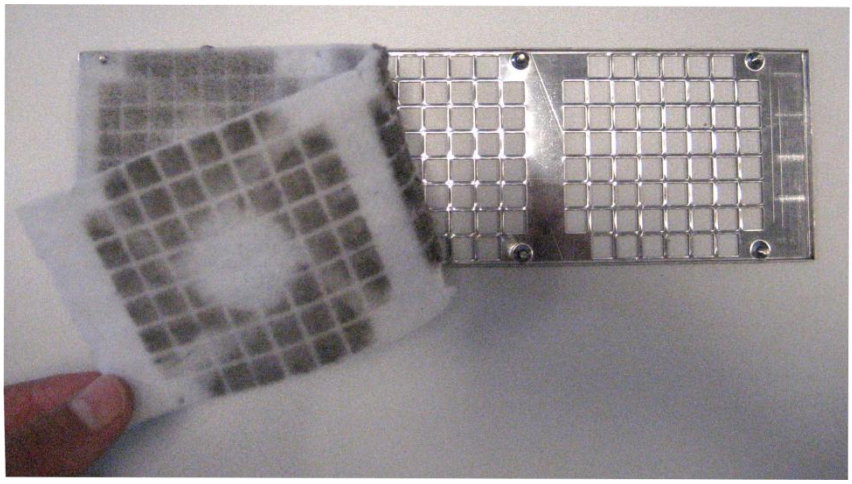

Figure 82: Remove the dirty filter

After the dirty filter was removed, place the new clean one to the same position. It must fit well. Fix the filter with screws.

The fan set has a connector at its back (Figure 83), which must fit to its proper place when inserting the fans back into the chassis.

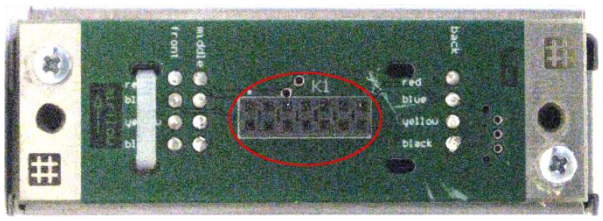

Figure 83: Back connector on a fan set.

7.1.3 Cleaning

WARNING: Before cleaning, make sure Libera Brilliance+ is switched off and the power cord is disconnected.

Cleanliness is important for proper operation of the instrument. Before cleaning, the instrument must be switched off and disconnected from mains supply.

The dust from the chassis may be removed using a softcloth or a paper towel dampened with water. Instead, 75% isopropyl alcohol solution can be used for more efficient cleaning. Alcohol should be used in liquid rather than spray form. If a spray must be used, always spray the alcohol onto a cloth, never directly on the instrument.

The dust from the hardware modules (shielding) may be removed using a softcloth or compressed air.

To avoid the damage, do not expose the instrument or its hardware modules to any sprays, liquids or solvents.

7.1.4 Connector care

WARNING: Before cleaning, make sure Libera Brilliance+ is switched off and the power cord is disconnected.

Visual inspection and, if necessary, cleaning of the connectors should be done every time a connection is made. This is must be done for:

- RF inputs on BPM modules
- Connectors on the timing module

Special care is required particularly for the RF input channels on the BPM modules. Use isopropyl or ethyl alcohol on a swab to clean the connectors. Clean the conductive surfaces carefully and avoid wetting the plastic parts inside the connector. After cleaning, be sure the connector is blown dry before reassembly of the cable.

7.2 Software maintenance

The OS that runs in the Libera Brilliance+ is Linux Ubuntu. It is installed on and runs from a memory card. There is no graphical interface installed but just the terminal. The installation is done in the factory – the Linux image is maintained by Instrumentation Technologies. If applications are added by the user, the missing packages/libraries must be added manually.

7.2.1 Checking the list of installed software

To check the list of installed packages:

```
dpkg --list libera-*
```

To display details about the individual package:

```
dpkg -s [name of package]
```

7.2.2 Log files

There are several files that log the boot-up procedure and other activities. The more important ones are listed in this chapter.

7.2.2.1 Boot-up log file

To check the installed drivers and observe the boot process, one can check the log:

```
tail -f /var/log/messages
```

If the instrument does not boot-up properly, this file contains important information and shall be sent to Instrumentation Technologies for consultation.

Same applies for

```
dmesg
```

output. Redirect the output to a file (`dmesg >file`) for easier check.

7.2.2.2 Presence of Libera devices

To check the PCI Express devices, type:

```
lspci | grep Xilinx
lspci | grep Lattice
```

This will list all PCI Express devices that have been detected and loaded.

For the first information:

- Xilinx device ... the number of these devices must match the number of processor modules, the GDX module and the EvRx module. Maximum number is 6.
- Lattice device ... present on the TIM module and ICB. Must be 1 or 2 (depends if the TIM or EvRx module is used).

Check the device drivers by

```
ll /dev/libera*
```

This will list all "libera" devices. They differ by the number, which corresponds to their position in the chassis:

- libera0 ... ICB
- libera1 ... GDX module
- libera2 ... TIM module / EvRx module
- libera3 – libera6 ... BPM modules

7.2.2.3 Libera daemons' log files

Each of the Libera daemons generates its log file(s). They are located in the `/var/opt/libera/log` directory and differ by name. Should there be any crash or failed start-up, please check these files.

Regularly check the content of the directory and backup the files if needed.

7.2.2.4 Firmware log

Firmware log is stored in the firmware and contains low level information. This log is intended only for Instrumentation Technologies support purpose. To read the log, do

```
libera-bmc --sel --read --board=0
```

Once the log is full, the RED light will illuminate in the ICB. To clear the log, do

```
libera-bmc --sel --clear --board=0
```

The RED light will be cleared after next complete power cycle (with ON/OFF switch).

7.2.3 Configuration files – backing up

7.2.3.1 Libera configuration files

Values of parameters are visible in the registry tree and most of them (if not read-only) can be changed using `libera-ireg` utility or EPICS. To save the modified values, the `libera-ebpm` daemon must be stopped properly.

```
/etc/init.d/libera-ebpm stop
```

After successful stop, the complete registry tree content (parameter values) will be saved into the `libera-ebpm.xml` file, which is located in the `/var/opt/libera/cfg` directory. When the `libera-ebpm` is started again (power cycle, reboot, restart), the content of the `libera-ebpm.xml` file will be loaded to the registry tree.

When configuration has been modified and is essential, please back it up to external server. In case of any failure, you can simply copy the file to the original location (`libera-ebpm` must be stopped first).

The same directory contains also:

- `libera-platform.xml` file, which contains the configuration of the platform daemon functionality (health monitoring and logging, excluded sensors monitoring, temperature control). **Do not modify this file unless consulted with Instrumentation Technologies support team (support@i-tech.si).**
- Gain schemes (for each BPM module), which **can be edited** according to the specific needs (attenuation value at certain power_level settings). Please consult with Instrumentation Technologies support team prior any changes.
- A file with FIR coefficients. **Do not modify this file.**

7.2.3.2 EPICS configuration files

EPICS configuration is set in the `/opt/libera-ioc/` directory. The prefix is set in the `IOCname` file and can be edited by the user.

The initial database is located in the `/opt/libera-ioc/db` directory. It is suggested to backup the configuration.

7.2.4 Software upgrade

The software release includes certain functionalities, related to the release version.
The software upgrade can be done by the user. It is provided by Instrumentation Technologies over the FTP or other electronic media. The installation procedure is described in the README files in the upgrade directories.

The instrument must be excluded from the operation when upgrading the software. Sometimes, the power cycle is requested in order to complete the upgrade. During the upgrade, all libera packages will be reinstalled. It is suggested to make a backup of the custom configuration files before the upgrade.

Typical time, needed for the upgrade, should not exceed 10 minutes.

7.2.5 Firmware upgrade

The firmware of the modules is installed at Instrumentation Technologies and cannot be done by the user. It requires special equipment (cable, adapter) and the firmware code. Except in rare cases, the firmware does not need to be upgraded or reinstalled.

7.2.6 Selecting storage ring / booster design

The accelerator is typically built with booster ring and storage ring. For this reason, there must be 2 different designs developed for Libera Brilliance+. The Libera Brilliance+, as new, is usually configured to work in the storage ring.

If the unit is used in the booster ring, the booster design has to be selected / activated. The following procedure describes how to select/activate the booster/storage ring design:

- Connect to the instrument via terminal window.
- Run the following script.

```
/opt/libera/bin/libera-ebpm-configure
```

Select the VCXO:

- (1) for storage ring
- (2) for booster ring

Select (1) as an internal SC clock source. **Do not select (2).**
Confirm the upgrade of the FPGA images. This will transfer the proper (booster or storage ring) image to module's flash memory. The FPGA loads the image from flash memory at power cycle.

- Wait for some minutes and confirm power cycle. If upgrading the FPGA image in the GDX module, this will take approximately 18 minutes to finish.

7.2.7 Maintaining the installation configuration

It is good to keep the information about the Libera Brilliance+ in one place. The important data is, e.g.:

- Serial number
- MAC address
- IP address
- EPICS prefix
- Number of BPM modules in one chassis
- Location information (sector/cell)

7.3 Transportation

The instrument (chassis and modules) must be packed in its original package for any kind of transportation. The shipments via mail, the packaging must be marked as fragile.

7.4 Storage

The instrument shall be kept in its original package and stored in a dry and dust-free environment. The temperature may vary between 0°C and 50°C.

8. Appendix A: “libera-ireg” command line utility

In this chapter the commands available via command line tool `libera-ireg` are described by using the `help` command available for each command type. Some examples of use are added for further understanding.

NOTE: Node names depend on the type and number of the application modules in the chassis. Therefore, names and paths to the nodes the user can see in the registry tree may differ from the ones listed in the examples below.

Options for `info`:

```
Help for command: info

Valid Options:
  -m [ --max-length ] arg Node's value is truncated to this size
                          (default=80, 0=unlimited)

Global Options:
  -d [ --debug ] arg    Debug level
                         --debug=[fileName or -] [level: 1...4]
  -h [ --host ] arg     Host of the top node (The default is 127.0.0.1)
                         --host [IpAddress]
  -P [ --platform ]     Access platform registry.
  -A [ --application ]  Access application registry (default).
  -n [ --path ] arg     Path of the node
  -v [ --verbose ]      Prints additional information.
  -a [ --app-name ] arg Application on the host
                        (default is libera-app)
                         --application [application name]
  -p [ --port ] arg     Port endpoint to the host application

Show Examples:
  Print info for the root node
    libera-ireg info
  Print the info for the node specified by the path
    libera-ireg info --path=[nodeFullPath] [nodeFullPath] ... [nodeFullPath]
  Print the list of nodes can be typed without '--path'
    libera-ireg info [nodeFullPath] [nodeFullPath]
```

Examples of usage:
Displaying the information for an application specific node:

```
libera-ireg info boards.bpm
-------------------------------------------------
   Registry hostname    : IP_127-0-0-1
   Node name            : krafX
   Value                : No value for this node
   Value type           : Undefined
   Validator expression :
   Num of values        : 0
   Full path            : boards,krafX
   Num of children      : 13
   Flags                :  none
   Root                 : false
   Parent node name     : boards
   Children          : info, conditioning, clock_info, conf, tbt, local_timing, interlock,
postmortem, signal_processing, average_sum, beam, events, signals
-------------------------------------------------
```

Options for `dump`:

```
Help for command: dump

Valid Options:
  -l [ --level ] arg      How many levels are printed out
  -f [ --full-path ]      Prints full path of registry nodes.
  -m [ --max-length ] arg Node's value is truncated to this length
                          (default=80, 0=unlimited)

Global Options:
  -d [ --debug ] arg    Debug level
                         --debug=[fileName or -] [level: 1...4]
  -h [ --host ] arg     Host of the top node (The default is 127.0.0.1)
                         --host [IpAddress]
  -P [ --platform ]     Access platform registry.
  -A [ --application ]  Access application registry (default).
  -n [ --path ] arg     Path of the node
  -v [ --verbose ]      Prints additional information.
  -a [ --app-name ] arg Application on the host
                        (default is libera-app)
                         --application [application name]
  -p [ --port ] arg     Port endpoint to the host application

Show Examples:
  Dump the whole tree
    libera-ireg dump
  Dump the tree from root for a specified levels (1 ...)
    libera-ireg dump --level=[level]
  Dump the tree below the specified path
    libera-ireg dump --path=[nodeFullPath] [nodeFullPath] ...
  Dump the tree from the specified path for specified levels
    libera-ireg dump --path=[nodeFullPath] ... --level=[level]
  Dump all the above commands can include the IP option
    libera-ireg dump --host=[IpHost] --......
  Dump the list of nodes can be typed without '--path'
    libera-ireg dump [nodeFullPath] [nodeFullPath]
  Dump the list of nodes expressed with full path name
    libera-ireg dump --full-path [nodeFullPath] [nodeFullPath]
```

Examples of usage:
Dumping the registry tree to the level 2 (may vary depending on the installed modules and application features):

```
root@libera:~# libera-ireg dump -l2
app
  app-name=libera-ebpm
  version=3.6-253-r28981 thor
  boards
    icb0
    evrx2
    kraf6
  application
    plugins
    http
    synchronize lmt=0
  system
    persistence
```

Options for `access`:

```
Help for command: access

Global Options:
  -d [ --debug ] arg    Debug level
                         --debug=[fileName or -] [level: 1...4]
  -h [ --host ] arg     Host of the top node (The default is 127.0.0.1)
                         --host [IpAddress]
  -P [ --platform ]     Access platform registry.
  -A [ --application ]  Access application registry (default).
  -n [ --path ] arg     Path of the node
  -v [ --verbose ]      Prints additional information.
  -a [ --app-name ] arg Application on the host
                        (default is libera-app)
                         --application [application name]
  -p [ --port ] arg     Port endpoint to the host application

Show Examples:
  Set nodes values
    libera-ireg access --path=[nodeFullPath]=[value] [nodeFullPath]=[value]
  Print out node values
    libera-ireg access --path=[nodeFullPath] [nodeFullPath]
```

Examples of usage:
Two ways of accessing the registry node:

```
libera-ireg access fans.left front -P
fans.left_front: 4207
libera-ireg fans.left_front -P
fans.left_front: 4207
```

Two ways of setting the value of the registry node:

```
libera-ireg access boards.bpm.signal_processing.position.Kx=1010000
libera-ireg boards.bpm.signal_processing.position.Kx=1000000
```

Options for `listen`:

```
Help for command: listen

Valid Options:
  -w [ --wait ] arg     Time period for listening modifications
                        (default is 5 seconds)
                         --wait [seconds]

Global Options:
  -d [ --debug ] arg    Debug level
                         --debug=[fileName or -] [level: 1...4]
  -h [ --host ] arg     Host of the top node (The default is 127.0.0.1)
                         --host [IpAddress]
  -P [ --platform ]     Access platform registry.
  -A [ --application ]  Access application registry (default).
  -n [ --path ] arg     Path of the node
  -v [ --verbose ]      Prints additional information.
  -a [ --app-name ] arg Application on the host
                        (default is libera-app)
                         --application [application name]
  -p [ --port ] arg     Port endpoint to the host application

Show Examples:
  Wait 1 minute for notifications on the specified nodes
    libera-ireg listen --wait=60 [nodeFullPath] [nodeFullPath] ...
```

Example of usage:

```
libera-ireg listen -w 30 fans.left_front fans.left_middle fans.left_rear -P
Listen following nodes:
       left_front
       left_middle
       left_rear
```

Options for `signal`:

```
Help for command: signal

Valid Options:
  -l [ --lmt ] arg          LMT for data which needs to be retrieved.
  -s [ --size ] arg         How many atoms to read.
  -S [ --groups ] arg       How many groups to read.
  -o [ --offset ] arg       Offset in atoms from the read position
  -b [ --behavior ] arg     Behavior specifies when to read data
                            (Event|Now|Lmt) Default is 'Now'
  -r [ --read-size ] arg    How many elements (atoms or groups) are read at
                            once
  -L [ --loop ] [=arg(=-1)] Repeat the acquisition n times (infinite by
                            default)
  -H [ --header ]           Show header (signal component names)
  -B [ --binary ]           Display signal in raw binary format
  -c [ --csv ]              Display signal as comma separated list

Global Options:
  -d [ --debug ] arg    Debug level
                         --debug=[fileName or -] [level: 1...4]
  -h [ --host ] arg     Host of the top node (The default is 127.0.0.1)
                         --host [IpAddress]
  -P [ --platform ]     Access platform registry.
  -A [ --application ]  Access application registry (default).
  -n [ --path ] arg     Path of the node
  -v [ --verbose ]      Prints additional information.
  -a [ --app-name ] arg Application on the host
                        (default is libera-app)
                         --application [application name]
  -p [ --port ] arg     Port endpoint to the host application

Show Examples:
  Get 10 atoms from the signal now (at current LMT)
    libera-ireg signal --size=10 --path=[signalNodeFullPath]
  Get signal data at next trigger event, skip first 10 atoms (DOD)
    libera-ireg signal --behavior=Event --offset=10 [signalNodeFullPath]
  Get signal data at specific LMT (DOD)
    libera-ireg signal --behavior=Lmt --lmt=176000000 --path=[signalNodeFullPath]
  Get 100 atoms from signal data, reading one by one (STRM)
    libera-ireg signal --size=100 --read-size=1 --path=[signalNodeFullPath]
  Get 10 groups from signal data, reading by 2 groups (STRM)
    libera-ireg signal --groups=10 --read-size=2 --path=[signalNodeFullPath]
  Acqure signal on event 10 times (DOD)
    libera-ireg signal --behavior=Event --loop=10 [signalNodeFullPath]
```

See Chapter 4.1 for examples.

9. Appendix B: “libera-bmc” command line utility

In this Appendix, commands available via command line tool `libera-bmc` are described by using the `help` command available for each command type. Some examples of usage are shown for better understanding.

NOTE: All the parameters and settings listed here are also available through the MCI interface (`libera-ireg` command line tool). We highly recommend the usage of `libera-ireg` utility instead of `libera-bmc`. The `libera-bmc` command line utility should be used only for troubleshooting in case that `libera-ireg` tool does not work for whatever reason.

Commands for Board:

```
Board available commands:
  --help                Produce a help message

  --list                list board ids present in the Libera
                        --list --board

  --info                display info for the given board (--verbose displays
                        more details)
                        --info --board
                        --info --board --verbose
                        --info --board=[Uri] [Uri] ...

  --time                read the time stamp for the given board
                        --read --time
                        --read --time --board=[Uri] [Uri] ...
                        write the time stamp for the given board
                        --set --time
                        --set --time=[yyyy-mm-dd] [hh:mm:ss]
```

Commands for Sensors:

```
Sensor available commands:
  --help                Produce a help message

  --list                List of the sensors:
                        --list --sensor
                        --list --sensor --board=[Uri] [Uri]

  --info                Info for the sensors:
                        --info --sensor
                        --info --sensor --board=[Uri] [Uri] ...
                        --info --sensor=[sensorId] [sensorId] ...
                        --board=[Uri] [Uri] ...

  --read                Read sensors
                        --read --sensor
                        --read --sensor --board=[Uri] [Uri] ...
                        --read --sensor=[sensorId] [sensorId] ... --board=[Uri]
                        [Uri] ...
                        --read --sensor --board=[Uri]  --sampling_freq=1000
```

Commands for System Event Log (SEL):

```
SEL available commands:
  --help                Produce a help message

  --info                Read the SEL for the given board
                        --info --sel
                        --info --sel --board=[Uri] [Uri] ...

  --clear               Clear the SEL
                        --clear --sel
                        --clear --sel --board=[Uri] [Uri] ...

  --read                Display info for the given board
                        --read --sel --board=[Uri] ... -num=[numOfEntries]
                        --read --sel=[Id] [Id] ... --board=[Uri] ...
                        --read --sel=[Id] ... --board=[Uri] ... --reverse
```

Examples of usage:

```
libera-bmc --list --board
Detected BMC board IDS:
lbr:///chassis/0
lbr:///chassis/2
lbr:///chassis/3
lbr:///os/

libera-bmc --info --sel --board=chassis/0

Requested SEL info for URI: lbr:///chassis/0
|               SEL Information                |
------------------------------------------------
Entries              :   339
Free space           :   65535 (or more) bytes
Last Time Added      :   00:24:39.000000000
Last Time Deleted    :   01:00:00.000000000
```

```
libera-bmc --read --sel --board=chassis/0 --num=10 --reverse

SEL entries for URI: lbr:///chassis/0
| ID |      TimeStamp     | MsgID | MsgSeverity |                    MsgDescription
|      MsgParameter     |
-------------------------------------------------------------------------------------------------------
---------------------------------
339   2011-06-22 00:24:39     6           Info                    Time stamp synchronization.
(Time)               (1308695081)
338   2011-06-21 11:24:38     6           Info                    Time stamp synchronization.
(Time)               (1308648280)
337   2011-06-21 02:24:38     6           Info                    Time stamp synchronization.
(Time)               (1308615879)
336   2011-06-20 05:54:52     6           Info                    Time stamp synchronization.
(Time)               (1308542094)
335   2011-06-19 09:54:51     6           Info                    Time stamp synchronization.
(Time)               (1308470093)
334   2011-06-18 14:54:50     6           Info                    Time stamp synchronization.
(Time)               (1308401692)
333   2011-06-17 18:54:49     6           Info                    Time stamp synchronization.
(Time)               (1308329691)
332   2011-06-16 23:54:48     6           Info                    Time stamp synchronization.
(Time)               (1308261290)
331   2011-06-16 05:54:47     6           Info                    Time stamp synchronization.
(Time)               (1308196489)
330   2011-06-15 15:09:04     6           Info                    Time stamp synchronization.
(Time)               (1308143345)
```

```
libera-bmc --read --time
Board 'chassis:0' has time: 2011-06-22 14:21:47
Board 'chassis:2' has time: 2011-06-22 14:21:46
Board 'chassis:3' has time: 2011-06-22 14:21:47
Board 'os' has time: 2011-06-22 14:21:48
```

```
libera-bmc --read --sensor --board=chassis/2
 0,            2-MAX6698.LOCAL,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:28, 44 (44) ok
 1,               2-POWR.SUPP1,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:28, 58 (58) ok
 2,                     2-FPGA,  Temperature [   degrees C],      40,      10,       5,     127,
85,      65 : 14:22:28, 43 (43) ok
 3,                       2-NC,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:28, not valid na
 4,                  2-INPUTS,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:28, 44 (64) ok
 5,               2-POWR.SUPP2,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:29, 57 (77) ok
 6,                  2-AIRFLOW,  Temperature [   degrees C],      40,      10,       5,     127,
75,      65 : 14:22:29, 53 (73) ok
 7,                   2-3V3MP,      Voltage [       Volts],  3.2928,  3.1017,  2.9547,  3.7485,
3.6456,  3.4986 : 14:22:29, 3.2634 (222) ok
 8,                     2-3V3,      Voltage [       Volts],  3.2928,  3.1017,  2.9547,  3.7485,
3.6456,  3.4986 : 14:22:29, 3.3369 (227) ok
 9,                   2-3V3CLK,      Voltage [       Volts],  3.2928,  3.1017,  2.9547,  3.7485,
3.6456,  3.4986 : 14:22:29, 3.2634 (222) ok
10,                     2-2V5,      Voltage [       Volts],  2.4975,  2.3532,  2.2533,  2.8305,
2.7528,  2.6529 : 14:22:29, 2.442 (220) ok
11,                  2-1V2MGT,      Voltage [       Volts],  1.1956,  1.1466,   1.078,   2.499,
1.323,  1.2544 : 14:22:29, 1.2054 (123) ok
12,                     2-3V7,      Voltage [       Volts],   3.696,  3.5145,  3.2835,  4.2075,
4.1085,   3.894 : 14:22:29, 3.6795 (223) ok
13,                    2-1V2M,      Voltage [       Volts],  1.1956,  1.1466,   1.078,   2.499,
1.323,  1.2544 : 14:22:29, 1.1956 (122) ok
14,                     2-5V,      Voltage [       Volts],   4.995,  4.7475,     4.5,  5.7375,
5.49,  5.2425 : 14:22:29, 4.995 (222) ok
15,                  2-AD GND1,      Voltage [       Volts],       0,       0,       0,       0,
0,       0 : 14:22:29, 0 (0) ok
16,                  2-AD_GND2,      Voltage [       Volts],       0,       0,       0,       0,
0,       0 : 14:22:29, 0 (0) ok
17,                  2-AD GND3,      Voltage [       Volts],       0,       0,       0,       0,
0,       0 : 14:22:29, 0 (0) ok
18,                  2-12V_AMC,      Voltage [       Volts],  11.984,  11.424,  10.808,   14.28,
13.216,    12.6 : 14:22:29, 11.872 (212) ok
```

10. Appendix C: libera-telnet-server interface

In this Appendix, an alternative server interface is explained. It allows socket connections from various clients, e.g. MATLAB or LabVIEW for reading data and setting/reading parameters. Command syntax is same as for the `libera-ireg` client. Following example shows how to connect and work through the 'telnet' connection.

Telnet port is shown in the registry tree (port=5579):

```
root@libera:~# libera-ireg dump application.telnet_server.
telnet_server
  port=5579
  connections=0
```

```
libera@home:~$ telnet 192.168.1.100 5579
Trying 192.168.1.100...
Connected to 192.168.1.100.
Escape character is '^]'.
Hello from libera-ebpm
>version
3.4-29-r27388
>boards.raf3.conditioning.tuning.agc.power level
-66
>boards.raf3.conditioning.tuning.agc.enabled
true
>boards.raf3.conditioning.tuning.agc.enabled=false
ok
>boards.raf3.conditioning.tuning.agc.power level=-30
ok
>boards.raf3.conditioning.tuning.agc.power_level
-30
>signal boards.raf3.signals.sa -s3
>  67211    67111    67090    67025    67109    6349    1566    7797    -294440901    1711    0
0    0    0    0    436144129
  67351    67243    67132    67153    67219    3461    5012    11497    -278642629    1711    0
0    0    0    0    436209665
  67295    67161    67155    67108    67179    6587    3450    7391    -262844357    1711    0
0    0    0    0    436275201
>
```

11. Appendix D: libera-http-plugin interface

The *libera-http-plugin* interface supports management of the instrument over HTTP v1.1 protocol. The instrument acts as a server. Clients must send a HTTP POST to send JSON request. Response is given in JSON format. It is possible to use modern web-browsers or other HTTP clients.

Simple examples are presented in this document (IP address 10.0.3.8, plugin's port 80). More detailed API description is available in a separate document.

Example how to read a parameter's value using a *curl* command line utility:

```
curl -X POST 10.0.3.8:80/api -d '{"path":"boards.tim.pll.locked","cmd":"get"}'
```

Example how to set a parameter's value using a `curl` command line utility:

```
curl -X POST 10.0.3.8:80/api -d \
'{"path":"boards.tim.triggers.t2.source","cmd":"set","value":"Pulse"}'
```

Example how to read a signal using a `curl` command line utility:

```
curl -X POST 10.0.3.8:80/api -d
'{"path":"boards.kraf3.signals.ddc_synthetic","cmd":"signal","size":10,"mode":"Event"}'
```

Basic example how to read parameter's value using `python`:

```
import requests

# This example assumes the libera-http-plugin listens to port 80
r = requests.post('http://10.0.3.8:80/api', \ json={'path':'boards.tim.pll.locked','cmd':'get'})

if r.status code != 200:
    print("\nRead Error.\n")
else:
    print (r.json())
```

12. Appendix E: Zabbix service

Zabbix service is available in Libera Brilliance+ instruments that run Ubuntu 24.04 Operating System (or greater). The service is disabled by default. It must be configured manually by the users.

Main purpose of using Zabbix is to monitor the metrics, such as network utilization, CPU load and disk space, but also sensor values (temperature, voltage, current) from various Libera hardware modules.

Configuration procedure:

- Copy the `zabbix_sensors.sh` script to `/opt/libera/bin/` directory
- Set the script with executable flag:

```
chmod +x /opt/libera/bin/zabbix_sensors.sh
```

- Configure Zabbix agent user parameters. Edit the `/etc/zabbix/zabbix_agentd.conf` file and add the two rows:

```
UserParameter=itech.sensors.discover,sudo /opt/libera/bin/zabbix_sensors.sh sensors
UserParameter=itech.sensors.collect,sudo /opt/libera/bin/zabbix_sensors.sh
```

to the “USER-DEFINED MONITORED PARAMETERS → Option: UserParameter” section.
Set the Timeout value in the “ADVANCED PARAMETERS → Option: Timeout” section.

```
Timeout=12
```

Specify the Zabbix server IP address. Optionally, set the ServerActive IP address:

```
Server=<IP address of the Zabbix server>
ServerActive=<IP address of the Zabbix active server>
```

Set the unique name of the Libera Brilliance+ instrument:

```
Hostname=myLibera
```

- Add sudoers configuration:

```
root@libera:~# cat /etc/sudoers.d/itech
zabbix ALL = (root) NOPASSWD : /opt/libera/bin/zabbix_sensors.sh
```

- Restart zabbix service in Libera Brilliance+:

```
systemctl restart zabbix-agent.service
```

The `zabbix_sensors.sh` script is used to collect sensor data from the Libera hardware modules and make it available to Zabbix agent.
A template (`zbx_libera_instrument.yaml`) for the Zabbix server is provided with the software release. It can be imported to the Zabbix server’s interface directly and later customized for specific use.

More at www.i-tech.si

Visit our website to read more about Libera products, download conferencepapers on the use of Libera at different accelerators around the world, subscribeto the I-Tech Newsletter and learn about the next gathering of the communityat the Libera Workshop.

Technical support

Prompt and reliable. You can ask for on-site support or we can assist you remotely. You are also welcome to join us at the Libera Workshop treningsessions to get the most out of Libera products.

Instrumentation Technologies, Velika pot 22, SI- 5250 Solkan, Slovenia
P: +386 5 335 26 00, **F:** +386 5 335 26 01,
E: info@i-tech.si, **W:** http://www.i-tech.si

INSTRUMENTATION
TECHNOLOGIES