UniMon
Gateware Documentation

Version 1.0

February 9, 2023

René Geißler
r.geissler@gsi.de

GSI Helmholtzzentrum für Schwerionenforschung GmbH

Contents

Documentation formats

There are three available documentation formats:

- Markdown: `https://git.gsi.de/BEA_HDL/UniMon_Gateware/-/blob/master/README.md`
- DokuWiki: `https://www-bd.gsi.de/dokuwiki/doku.php?id=ds:projects:unimon::gateware`
- LaTeX (PDF): `https://www-bd.gsi.de/dokuwiki/lib/exe/fetch.php?media=ds:projects:unimon:gateware:UniMon_Gateware_Documentation.pdf`

All documentation is based on the content of `README.md`, which is the most up to date format. The DokuWiki and the PDF version have been generated automatically, but will not be updated automatically.

Up to date DokuWiki and PDF versions can be downloaded from the CI/CD section:
`https://git.gsi.de/BEA_HDL/UniMon_Gateware/-/pipelines`

Resources

The code of this project and also the source of this documentation are under version control in a Git repository whose upstream is:
`https://git.gsi.de/BEA_HDL/UniMon_Gateware`.
The relevant branch is *master*.

Common code is included as a Git submodule of the main Git repository. The upstream of the submodule is:
`https://git.gsi.de/BEA_HDL/FPGA_Common`.
The relevant branch is *master*.

Installation scripts to set up a Gitlab runner for continuous integration including all necessary software to build the gateware can be found in a Git repository whose upstream is:
`https://git.gsi.de/BEA_HDL/Gitlab_Runner_Setup_Centos_7`.
The relevant branch is *master*.

1 Introduction

This document describes the gateware (= FPGA firmware) implementation of the UniMon Tank Signal Monitor.

The system uses 4 IOxOS ADC_3117 FMC cards which are mounted on two OpenHardware AFC v4.0 FMC carriers. Each of the FMCs provides 20 ADCs with a resolution of 16 bits. The sampling rate of the ADCs is adjustable between 1 kHz and 5 MHz. Each ADC input is individually configurable as a differential or single ended and different gain settings are available.

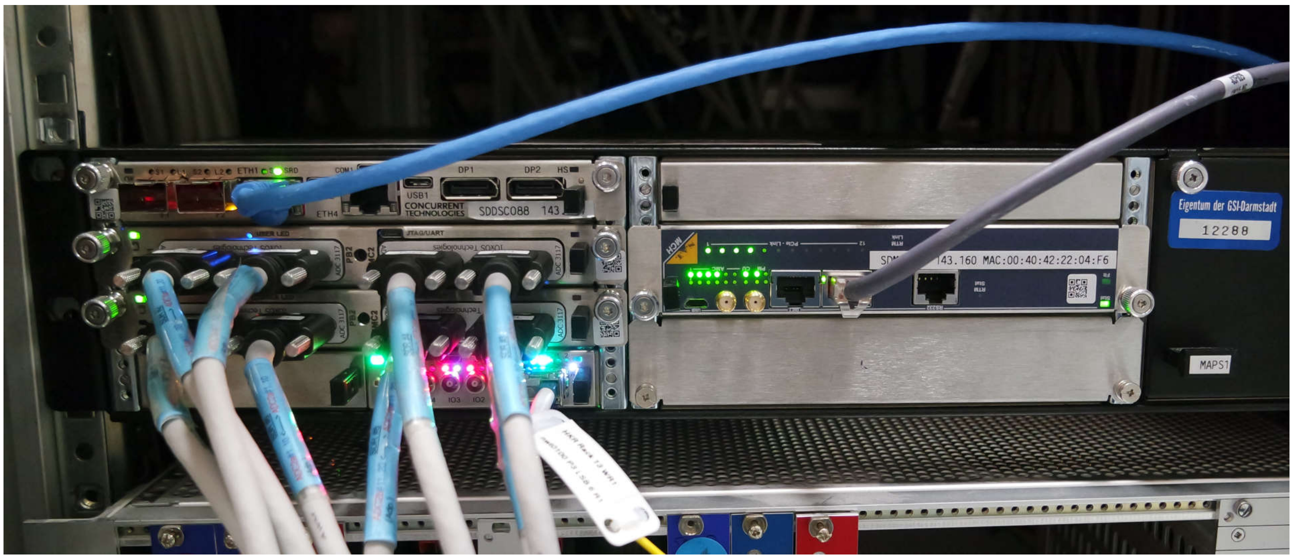

Figure 1.1: UniMon crate in the HKR

The gateware of each AFC stores the data stream of respectively 40 ADCs to a ring buffer, from which it can be read via PCIe.

2 Common FPGA based projects documentation

This project incorporates the code from the *FPGA_Common* Git repository which is used in multiple projects. The documentation of the common features can be found here:

2.1 Common monitoring and control features

Documentation about the register bank, the architecture information storage and the observer can be found here:
https://git.gsi.de/BEA_HDL/FPGA_Common#2-common-monitoring-and-control-features

2.2 FPGA Observer

There is a expert GUI that can be used together with multiple projects:
https://git.gsi.de/BEA_HDL/FPGA_Common#3-fpga-observer

2.3 Build flow and simulation

You can find instructions on how to build and simulate the gateware here:
https://git.gsi.de/BEA_HDL/FPGA_Common#4-build-flow-and-simulation

2.4 Helper scripts

You can find usefull scripts here:
https://git.gsi.de/BEA_HDL/FPGA_Common#5-helper-scripts

2.5 Continuous integration environment

Information about the continuous integration setup can be found here:
https://git.gsi.de/BEA_HDL/FPGA_Common#6-continuous-integration-environment

2.6 Programming and hardware configuration

You can find instructions on how to program the FPGA and configure other hardware here:
https://git.gsi.de/BEA_HDL/FPGA_Common#7-programming-and-hardware-configuration

3 Peripheral devices

The gateware communicates with the following devices on each of the IOxOS ADC_3117 FMC cards:

- 10 * Analog Devices LTC2323-16 dual ADC
- 10 * EXAR XRA1405 GPIO expander

3.1 Analog Devices LTC2323-16 ADC

Each LTC2323-16 consists of two ADCs that are sampled in parallel.

The sampling frequency is adjustable in the range between 1 kHz and 5 MHz.

Each ADC provides a differential input pair and is directly controlled by the FPGA without any external clock.

The FPGA has to read the samples at a clock speed of 105 MHz. The sampling rate of the ADCs is controlled by the distance between the so called conversion pulses that have to be generated by the FPGA [1].

Due to this clock generation logic of the ADCs, the sampling frequency can be set with a certain granularity described in chapter ??.

3.2 EXAR XRA1405 GPIO expander

The GPIO expanders are used for multiple purposes. Among these is the control of the inputs to each ADC.

3.2.1 ADC input selection

There are dedicated switch devices in front of each of the differential ADC inputs, which can be controlled independently.

The following input settings are available:

- differential p / n input pin
- ground
- configurable calibration voltage (not implemented yet)
- fixed calibration voltage of + 4.128 V

3.2.2 Gain selection

The gain of each differential ADC input pair can be controlled by a dedicated fixed gain amplifier device which can be programmed to an amplifications of 1, 2, 5 or 10.

In front of each ADC input there is an attenuator with a fixed gain of 0.4.

In total the following gains are available [2]:

amplifier gain	input range	resolution per LSB
1	+/- 10.24 V	312.5 μV
2	+/- 5.12 V	156.25 μV
5	+/- 2.048 V	62.5 μV
10	+/- 1.024 V	31.25 μV

4 Gateware implementation

4.1 Clocking

The gateware uses only one primary clock.

4.1.1 PCIe reference clock

The PCIe reference clock has a nominal frequency of 100 MHz and feeds the reference clock input of the PCIe IP core by Xilinx, which contains a PLL producing a 125 MHz output clock for the AXI interface named *clk_125_pcie_axi*.

clk_125_pcie_axi drives a MMCM to generate:

- *clk_100*, 100 MHz, this is the main processing clock of the design
- *clk_200*, 200 MHz, used for the SDRAM interface IP core by Xilinx
- *clk_50*, 50 MHz, used for generating a 210 MHz clock for the ADC interface

The SDRAM interface IP core contains a MMCM which generates a 100 MHz clock named *clk_100_sdram* for the AXI interface.

4.2 Resets

4.2.1 PLL not in lock

As long as the PLL in the MMCM producing the main processing clock *clk_100* is not yet in lock, the design is held in reset. After this the lock should be stable until the next power cycle.

4.2.2 PCIe reset

The design will be reset whenever the PCIe connection is re-initialized. This happens e.g. when the FEC is rebooted.

4.2.3 Reset button

There is a push button labeled *RST* at the center of the AFC front panel which is connected to the microcontroller for the MMC firmware. The OpenMMC firmware forwards a button press with a duration of at least two seconds to the FPGA pin AG26 as an active low signal to initiate a reset of the gateware.

Resetting the FPGA leads to the loss of the PCIe connection. To re-enable the connection, the FEC has to be rebooted.

4.3 Data flow diagram

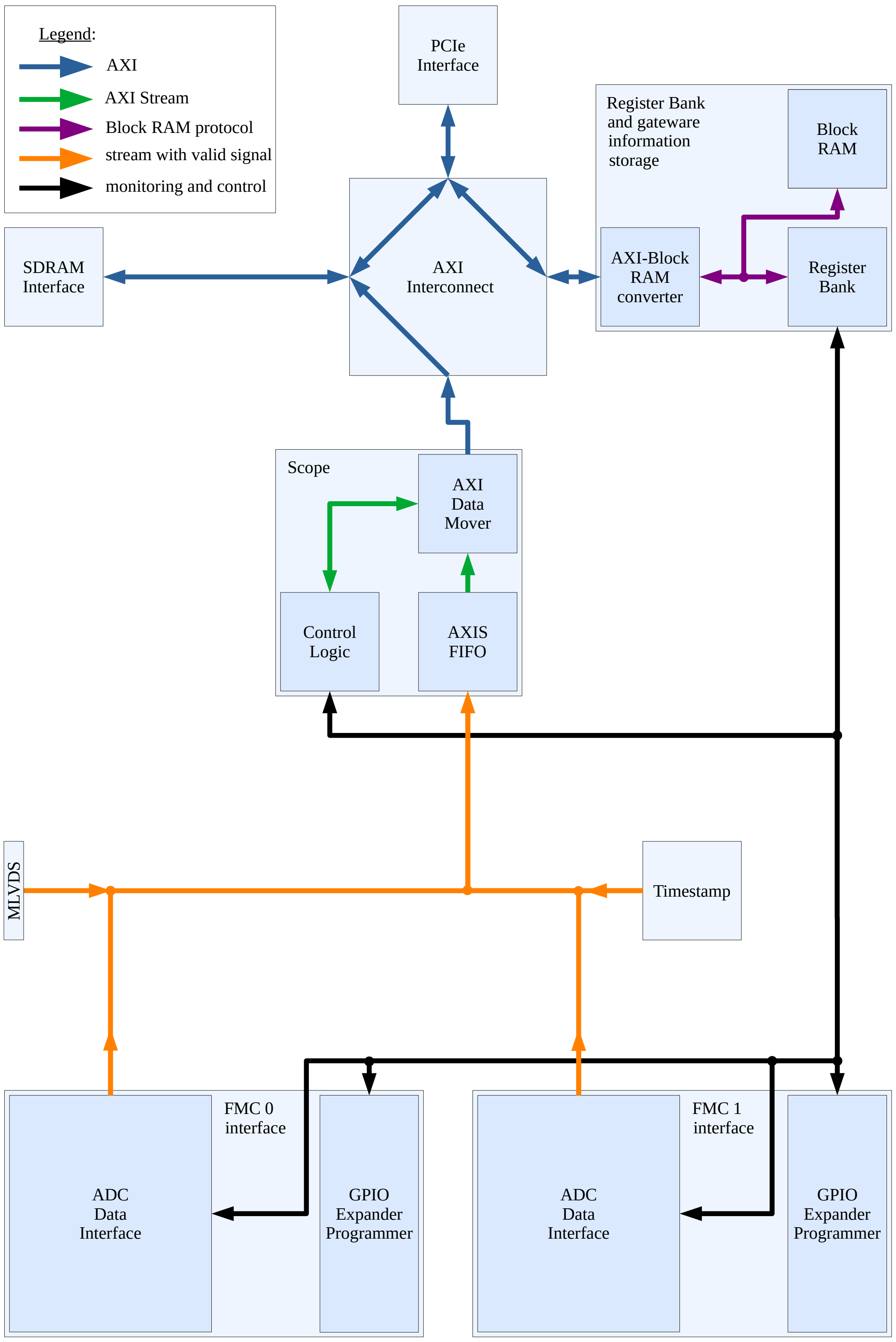

Figure 4.1: Simplified data flow diagram

Figure 4.1 shows a simplified data flow diagram. For simplicity, some features are not included in the diagram:

- processing clocks and clock domain crossings
- resets
- LEDs
- read out logics of FPGA serial number and build time stamp
- data width conversions of AXI connections

4.4 PCIe Interface

This gateware uses the Xilinx IP core *DMA/Bridge Subsystem for PCI Express* with the following configuration:

- PCIe speed: 5 GTransfers/s
- AXI clock frequency: 125 MHz
- reference clock frequency: 100 MHz

The PCIe reference clock is routed to the FPGA via the MMC firmware and is configured to be driven by the 100 MHz *FCLKA* clock coming from the AMC connector.

4.5 SDRAM interface

For the communication with the SDRAM, an IP core by Xilinx is used. The clock frequency of the SDRAM interface's AXI bus is 100 MHz, but has a different source so that an AXI clock converter is used to connect it to the rest of the AXI infrastructure which is clocked at the main processing clock of 100 MHz.

4.6 Observer

This project incorporates the observer interface from the *FPGA_Common* Git submodule (see `https://git.gsi.de/BEA_HDL/FPGA_Common#13-observer`).

The following signals are connected to the observer inputs:

value	input vector(64 bits)	valid signal
0	corrected ADC data of ADCs 0 - 3	corrected ADC data valid (0)
1	corrected ADC data of ADCs 4 - 7	corrected ADC data valid (0)
2	corrected ADC data of ADCs 8 - 11	corrected ADC data valid (0)
3	corrected ADC data of ADCs 12 - 15	corrected ADC data valid (0)
4	corrected ADC data of ADCs 16 - 19	corrected ADC data valid (0)
5	corrected ADC data of ADCs 20 - 23	corrected ADC data valid (1)
6	corrected ADC data of ADCs 24 - 27	corrected ADC data valid (1)
7	corrected ADC data of ADCs 28 - 31	corrected ADC data valid (1)
8	corrected ADC data of ADCs 32 - 35	corrected ADC data valid (1)
9	corrected ADC data of ADCs 36 - 39	corrected ADC data valid (1)
10	value of the eight MLVDS inputs	1

The rest of the multiplexer inputs are connected to zero.

5 Gateware software interface

The communication between the gateware and the software takes place via a PCIe driver by Xilinx called XDMA. There is only one PCIe Bar in use in the gateware which maps the memory space to different physical memories on the AMC board.

The following mapping is applied:

address	size	memory type	description
0x00000000	2 kiB	Flip Flops	inside FPGA, for registers
0x00004000	16 kiB	Block RAM	inside FPGA, for architecture information
0x00010000	64 kiB	Block RAM	inside FPGA, for observer
0x80000000	2 GiB	SDRAM	external, for scope data

Table 5.1: Memory mapping

The *architecture information* and the *observer* are documented here:
https://git.gsi.de/BEA_HDL/FPGA_Common#2-common-monitoring-and-control-features

5.1 PCIe Driver

Read and write accesses are mapped to virtual file accesses:

- /dev/xdma0_c2h_0 for read accesses
- /dev/xdma0_h2c_0 for write accesses

5.1.1 Reading from a register

Example in C:

```c
uint32_t address = 0x00000000;
int fd = open("/dev/xdma0_c2h_0", O_RDWR);
lseek(fd, address, SEEK_SET);
uint64_t value;
read(fd, &value, sizeof(uint64_t));
```

5.1.2 Writing to a register

It is important to write the whole register width of 64 bits. If a register has less than 64 bits, the unused MSBs have to be written to any value. 32 bit write accesses will not have any effect.

Example in C:

```c
uint64_t value = 42;
uint32_t address = 0x00000400;
int fd = open("/dev/xdma0_h2c_0", O_RDWR);
lseek(fd, address, SEEK_SET);
write(fd, &value, sizeof(uint64_t));
```

5.1.3 Reading of scope data

Example in C:

```
uint32_t address = 0x80000000;
int fd = open("/dev/xdma0_c2h_0", O_RDWR);
lseek(fd, address, SEEK_SET);
char data[1024];
read(fd, data, 1024);
```

This example reads 1024 bytes of data to an array. For bigger data blocks, instead of using an array, you will probably prefer a dynamically allocated memory region.

5.2 Scope memory

The complete SDRAM is used for the scope memory. The ADC data is stored in the following format:

address	bits	radix	description
0x80000000	16	signed	ADC 0 data (time = 0)
0x80000002	16	signed	ADC 1 data (time = 0)
...	...	...	...
0x8000004E	16	signed	ADC 39 data (time = 0)
0x80000050	64	unsigned	timestamp (time = 0)
0x80000058	8	none	state of the MLVDS lines (time = 0)
0x80000059	312	none	unused
0x80000080	16	signed	ADC 0 data (time = 1)
...	...	...	...

Table 5.2: ADC data storage format

The scope is a free running ring buffer. The write pointer is incremented with each incoming ADC sample vector by the vector size of 128 bytes.

The scope memory can hold up to 2^{24} samples. At the maximum sampling frequency of 5 MHz this corresponds to a maximum capture duration of 3.36 seconds.

5.3 Register map

5.3.1 Status registers

The following status registers can be read by software:

index	address	bits	radix	description
0	0x00000000	16	signed	latest ADC 0 data
1	0x00000008	16	signed	latest ADC 1 data
...	...	...	...	...
39	0x00000138	16	signed	latest ADC 39 data
48	0x00000180	32	unsigned	scope next write address
111	0x00000788	1	binary	SDRAM initial calibration complete
124	0x000003E0	32	unsigned	build timestamp
125	0x000003E8	57	unsigned	FPGA serial number
126	0x000003F0	64	unsigned	module ID
127	0x000003F8	64	unsigned	magic number

Table 5.3: List of status registers

0 - 39: latest ADC {0 - 39} data

This register holds the latest ADC data sample.

48: Scope next write address

Address where the next data sample will be stored during the scope's capturing process. The recently stored ADC data can be found below this address, or - in case of a ringbuffer wraparound - also below the end of the memory region.

111: SDRAM initial calibration complete

The communication to the SDRAM is controlled by an IP core by Xilinx which performs a timing calibration at start up. The value of this register will be `1` after completion of the initial calibration. If this registers reads `0`, this might indicate that the SDRAM is damaged.

124: build timestamp

Time when the bitstream was created. This information can be used to identify the gateware version (together with the Git commit information documented in `https://git.gsi.de/BEA_HDL/FPGA_Common#122-gateware-information`).

Format:

bits 31 - 27	bits 26 - 23	bits 22 - 17	bits 16 - 12	bits 11 - 6	bits 5 - 0
day	month	year (last two decimal digits)	hours	minutes	seconds

125: FPGA serial number

The XDMA PCIe driver by Xilinx numbers the devices randomly and is not able to identify the slot number of an AMC board. This register holds the FPGA's unique serial number and can be used to identify an AMC board.

126: module ID

The module ID can be used to identify the type of the current bitstream.
The module ID of the UniMon gateware is `0x0001010300010002`.

The fields are defined as follows:

bits 63 - 56	bits 55 - 48	bits 47 - 40	bits 39 - 32	bits 31 - 16	bits 15 - 0
minor gateware version	major gateware version	minor board version	major board version	developer ID	project ID

Here is an incomplete list of project IDs:

project ID	project name
`0x0001`	Cryring BPM
`0x0002`	UniMon
`0x0003`	Rate Divider
`0x0004`	BLoFELD
`0x0005`	Resonant Transformer
`0x8001`	Red Pitaya

127: magic number

The magic number can be used to determine if the gateware uses the expected register format.
The value of this register is the same for all module IDs: `0xBADEAFFEDEADC0DE`.

5.3.2 Configuration registers

The following registers can be written by software:

index	address	bits	radix	description	default value
0	0x00000400	18	unsigned	ADC conversion pulse length	0x00000
1	0x00000408	5	unsigned	MLVDS calibration delay	0x00
16	0x00000480	40	none	FMC0 ADC gain	0x0000000000
17	0x00000488	40	none	FMC0 ADC input mux select p	0x0000000000
18	0x00000490	40	none	FMC0 ADC input mux select n	0x0000000000
24	0x000004C0	40	none	FMC1 ADC gain	0x0000000000
25	0x000004C8	40	none	FMC1 ADC input mux select p	0x0000000000
26	0x000004D0	40	none	FMC1 ADC input mux select n	0x0000000000
32	0x00000500	16	signed	ADC 0 offset correction summand	0x0000
...	...	...	...	...	...
71	0x00000638	16	signed	ADC 39 offset correction summand	0x0000
72	0x00000640	16	unsigned	ADC 0 gain correction factor	0x8000
...	...	...	...	...	...
111	0x00000778	16	unsigned	ADC 39 gain correction factor	0x8000
127	0x000007F8	1	binary	reset	0

Table 5.4: List of configuration registers

0: ADC conversion pulse length

This parameter p defines the sampling frequency of all 40 ADCs in parallel.

The sampling frequency is given by:

$f_s = 210\text{MHz} \cdot \frac{1}{42+p}$

The allowed range for the sampling frequency is: $[1\text{kHz}, 5\text{MHz}]$

The default setting of 0 corresponds to a sampling frequency of 5 MHz.

The allowed range for p is: $[0, 209958]$

1: MLVDS calibration delay

The signals from the timing receiver enter the FPGA via the MLVDS lines with virtually no delay, while the corresponding ADC samples will be delayed due to the sampling process.

This register can be used to add an additional delay to the signals from the timing receiver in order to calibrate them to the delays of the ADCs.

The delay can be chosen from the range $[10\text{ns}, 320\text{ns}]$. Each LSB corresponds to an additional delay of 10 ns.

16, 24: FMC {0, 1} ADC gain

There are programmable amplifiers in front of each ADC input, which provide four discrete gains.

The following gains are available:

value	gain	input range	resolution per LSB
0	1	+/- 10.24 V	312.5 μV
1	2	+/- 5.12 V	156.25 μV
2	5	+/- 2.048 V	62.5 μV
3	10	+/- 1.024 V	31.25 μV

This register holds the concatenation of the gain settings of the FMC's 20 ADCs:

bits	gain of ADC
1 .. 0	0
3 .. 2	1
...	...
39 .. 38	19

17, 18, 25, 26: FMC {0, 1} ADC input mux select p / n

The ADCs have differential inputs. The two lanes p and n can be configured separately and individually for each ADC.

The following input settings are available:

value	input
0	differential p / n input pin
1	ground
2	not implemented
3	calibration voltage of +4.128 V

This register holds the concatenation of the input settings of the FMC's 20 ADCs:

bits	settings of input p / n of ADC
1 .. 0	0
3 .. 2	1
...	...
39 .. 38	19

32 - 71: ADC {0 - 39} offset correction summand

Correction summand for a possible offset deviation of the ADC. The offset correction precedes the gain correction.

72 - 111: ADC {0 - 39} gain correction factor

Correction factor for a possible gain deviation of the ADC. The default value 0x8000 corresponds to a multiplication by 1. The possible correction range is $[0, 2[$.

127: Reset

Writing a 1 to this register triggers a reset on the gateware, which also resets all configuration registers to their default values.
The reset will be automatically lifted so that the register does not have to be written to 0 after initiating a reset.

References

[1] Analog Devices LTC2323-16 datasheet, https://git.gsi.de/BEA_HDL/UniMon_Gateware/-/blob/master/doc/datasheets/AD_LTC2323_ADC_datasheet.pdf

[2] IOxOS ADC_3117 datasheet, https://git.gsi.de/BEA_HDL/UniMon_Gateware/-/blob/master/doc/datasheets/ADC_3117_UG.pdf