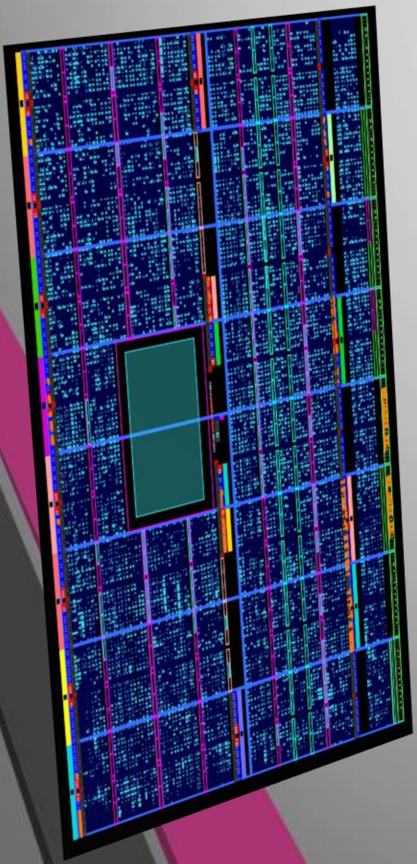
AMC-PICO8-MPS

8-channel Bipolar 20-bit

CAENels
Gear For Science

Board Support Package Overview

MTCA.4 – MicroTCA for Physics



```
reg [7:0] state_data;
initial state_data = S_DATA_IDLE;

integer i;
always @ (posedge clk) begin
    case( state_data )
        //==========================================================
        // waits for data ready
        S_DATA_IDLE: begin
            if ( data_ready ) begin
                state_data  <= S_DATA_ACQ;
                bit_counter <= BITS - 1; // -1 because we count to zero
            end
        end

        //==========================================================
        // saves data on negedge of return clock
        S_DATA_ACQ: begin
            if ((sck_rtrn_ss == 0) && (sck_rtrn_s == 1 )) begin
                for(i = 0; i < NR_CHAN; i = i + 1)
                    tmp_reg[i] <= {tmp_reg[i][BITS-2:0], sdo_ss[i]};

                bit_counter <=  bit_counter - 1;

                if (bit_counter == 0) state_data  <= S_DATA_DONE;
            end
        end

        //==========================================================
        // sends data to output port
        S_DATA_DONE: begin
            out_data1   <= tmp_reg[0];
            out_data2   <= tmp_reg[1];
            out_data3   <= tmp_reg[2];
            out_data4   <= tmp_reg[3];
            state_data  <= S_DATA_IDLE;
        end

        //==========================================================
        default:
            state_data <= S_DATA_IDLE;
    endcase
end
```

CAEN ELS s.r.l.
SS14, km 163.5
34149 Basovizza (TS) – Italy
Mail: info@caenels.com
Web: www.caenels.com

Table of Contents

Document Revisions

Document Revision	**Date**	**Comment**
1.0	July 03rd 2019	First Release

1. Introduction

This document describes the AMC-PICO8 Machine Protection System (AMC-PICO8-MPS) Board Support Package (BSP). This solution is based on the CAEN ELS DAMC-FMC25 AMC carrier board equipped with 2 FMC-PICO-1M4 (4-channel bipolar 20-bit) and the AMC-PICO8-MPS BSP extends the standard BSP of its carrier board.

In this document the AMC-PICO8-MPS specific BSP part is described. For additional information regarding the standard DAMC-FMC25 BSP, please refer to the "*DAMC-FMC25 Board Support Package Overview*" document.

2. AMC-PICO8-MPS BSP Overview

The following block diagram shows application logic on AMC-PICO8-MPS. Some connections (such as range selection, EEPROM interface, mux/demux selector) are omitted for clarity.

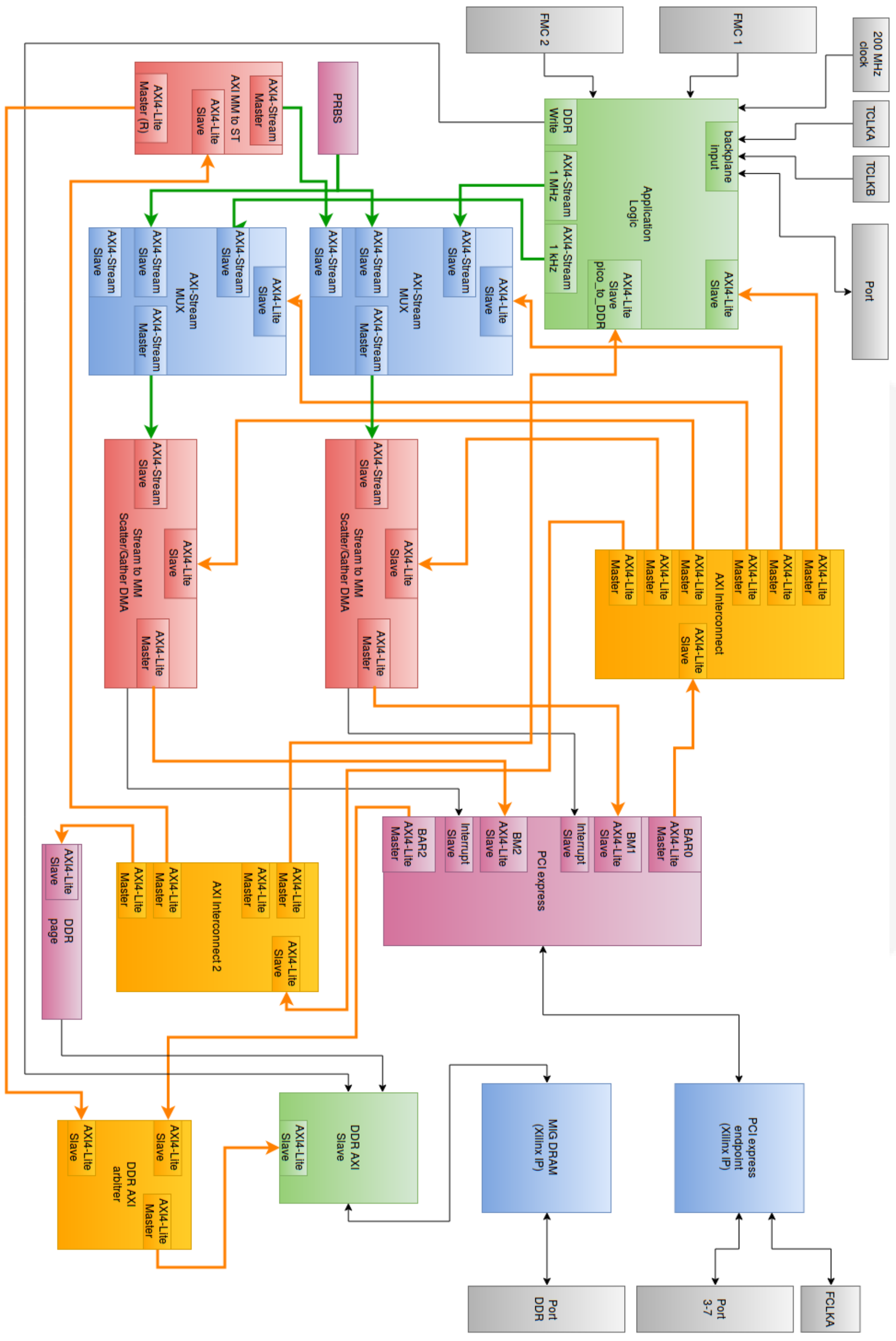


Figure 1: AMC-PICO8-MPS system block diagram

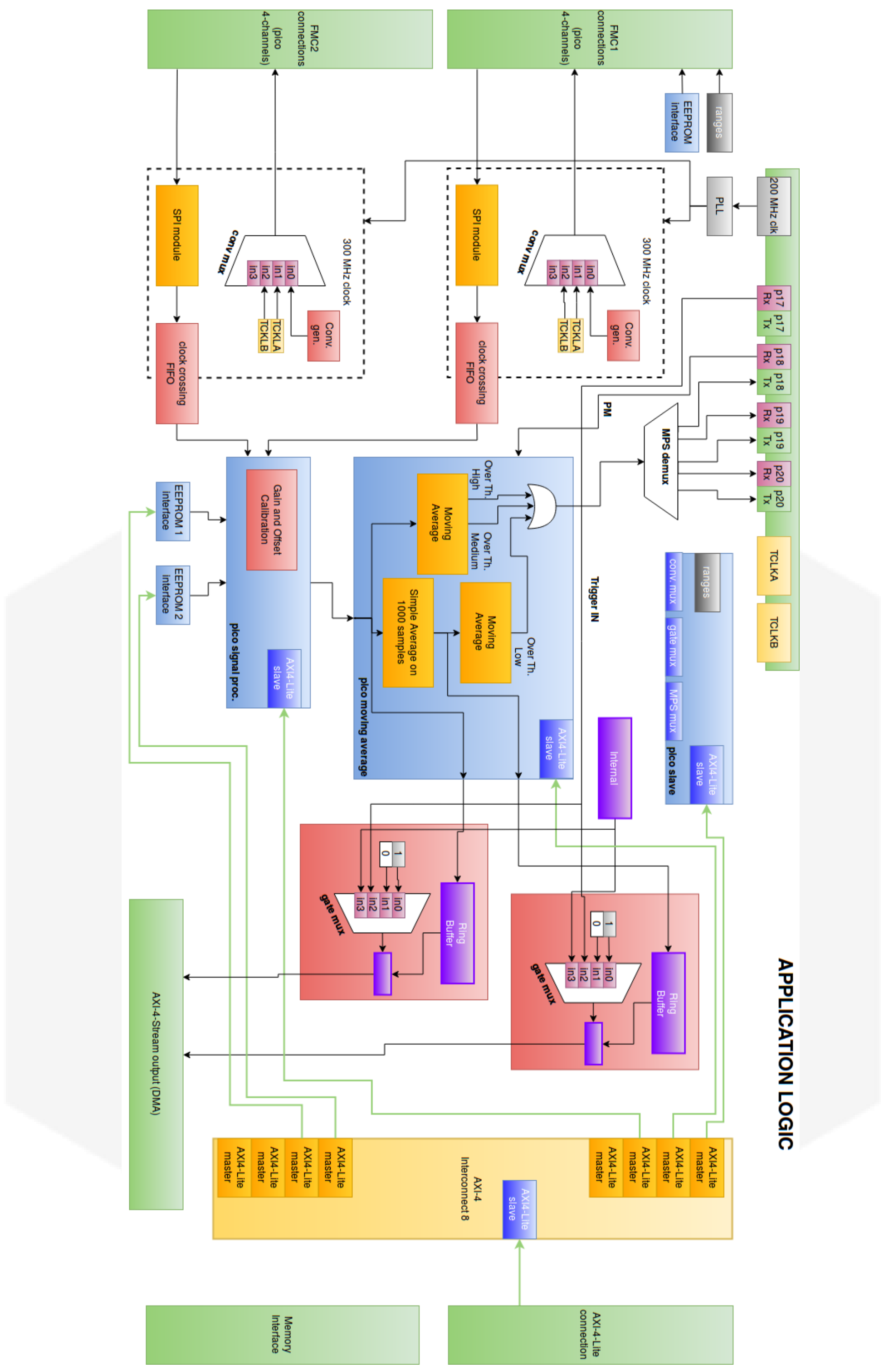


Figure 2: AMC-PICO8-MPS Application logic block diagram

2.1 SPI modules

The *SPI* module communicate with ADCs over SPI at 75 MHz and forwards the captured data to clock-crossing FIFOs.

2.2 Signal Processing

The *pico signal processing* module wraps around *calibration* module. The *calibration* is a HLS module, written in C. The *pico signal processing* is a wrapper module and provides all the necessary infrastructure (state machines) to interface with HLS two-way handshake signals.

The signal processing module accepts 8 channels (4 from the FMC1 and other 4 from the FMC2) of captured data in parallel from clock-crossing FIFO. The "raw" measurements are then feed into the calibration module. The calibration also requires a *gain* and *offset* parameters for each individual FMC-PICO-1M4 channel. There are 32 parameters in total (gain and offset, 2 different input ranges for each of the 8 channels).

After the calibration, the currents (now represented in floating precision) are sent to the *pico moving average* module. Thus, the output of this module are 8 calibrated currents at 1 MHz.

The BSP includes the output product from HLS, so it can be recompiled even without Xilinx HLS.

2.3 EEPROM interface

At the power-up, the EEPROM Interfaces reads from a predefined address in EEPROM memory. If the magic number read from EEPROM memory are valid, the calibration parameters are downloaded to *signal processing module* (see FMC-Pico-1M4 User's Manual for detailed description of EEPROM content). The EEPROM interface modules also allow accessing EEPROM memory from PCI express.

2.4 Moving Average

The *pico moving average* module is responsible for the generation of the *over threshold* signals in Infinite Mode operation.

The input of this module are the 8 channels data samples at 1 MHz. Additional inputs and configuration parameters are read from AXI4-Lite and are reported in the table.

For each channel, 3 moving averages are calculated:

- Moving average with n_samp_h (4.3.4) sample window and threshold th_chX_h (from 4.3.7 to 4.3.14) on the data at 1 MHz,
- Moving average with n_samp_m (4.3.5) sample window and threshold th_chX_m (from 4.3.15 to 4.3.22) on the data at 1 MHz, and
- Moving average with n_samp_l (4.3.6) sample window and threshold th_chX_l (from 4.3.23 to 4.3.30) on the data at 1 kHz

The output of the module is a vector of 24 bits (8 channels * 3 windows) indicating which channel on which window has exceeded the correspondent threshold. This vector can be read from AXI4-Lite (see 4.3.35, 4.3.36 and 4.3.37). Once a channel exceeds the threshold, the correspondent over threshold bit is raised and kept high even if the

moving average of that channels goes back below threshold. The over threshold vector can be cleared by resetting a specific register.
Finally, the MPS signal (one-bit signal) is built by performing a bit-by-bit OR of the over threshold vector. Each AMC-PICO8-MPS board must configure the MPS demux (such that there are no conflicts between multiple boards on the same M-LVDS line) and then enable the MPS generation.

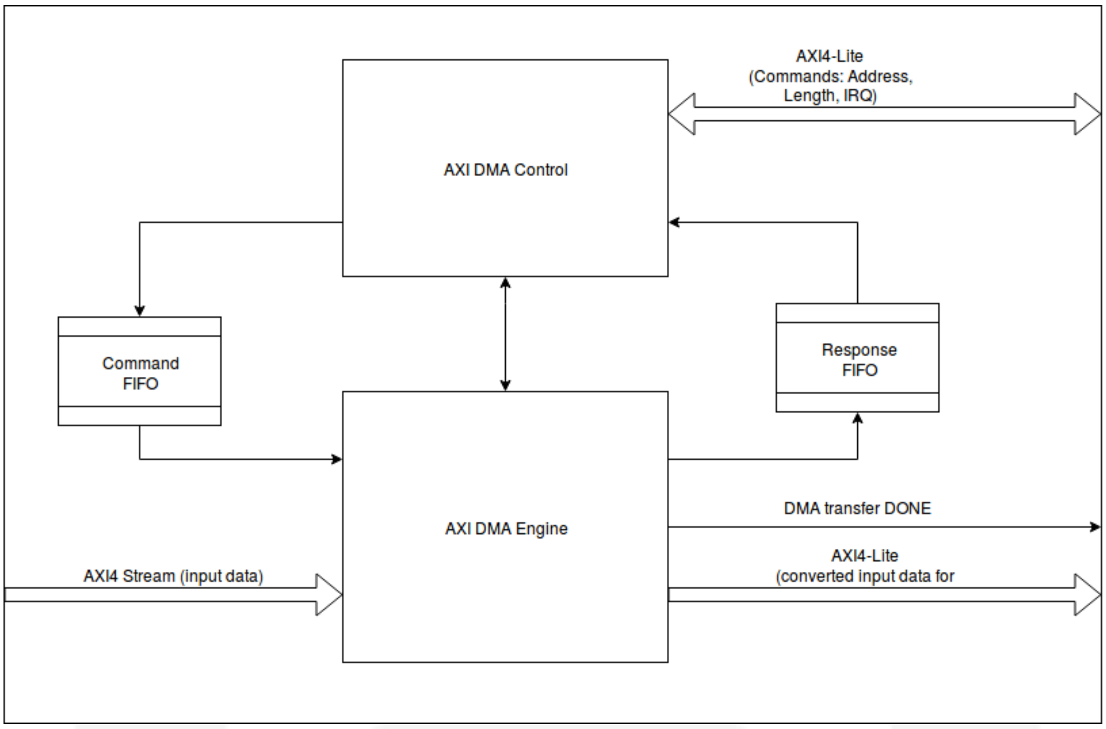
	THRESHOLD HIGH	THRESHOLD MEDIUM	THRESHOLD LOW
CH0 (CH1 FMC 2)	**th_ch0_h**	**th_ch0_m**	**th_ch0_l**
CH1 (CH2 FMC 2)	**th_ch1_h**	**th_ch1_m**	**th_ch1_l**
CH2 (CH3 FMC 2)	**th_ch2_h**	**th_ch2_m**	**th_ch2_l**
CH3 (CH4 FMC 2)	**th_ch3_h**	**th_ch3_m**	**th_ch3_l**
CH4 (CH1 FMC 1)	**th_ch4_h**	**th_ch4_m**	**th_ch4_l**
CH5 (CH2 FMC 1)	**th_ch5_h**	**th_ch5_m**	**th_ch5_l**
CH6 (CH3 FMC 1)	**th_ch6_h**	**th_ch6_m**	**th_ch6_l**
CH7 (CH4 FMC 1)	**th_ch7_h**	**th_ch7_m**	**th_ch7_l**
N_SAMP	**n_samp_h**	**n_samp_m**	**n_samp_l**

In Trigger Mode operation, only n_samp_h and n_samp_l are used and set to 1000. A trigger module generates the necessary trigger signals, for the operation of the moving average modules, upon a trigger event from the MLVDS line (or from an internal register for debugging purposes, see 4.3.3). The trigger signal is sensible to the rising edge. Once the trigger window ends, the moving average internal accumulators are frozen and saved in the AXI4-Lite registers (from 4.3.40 to 4.3.55). Furthermore, two counters that represent the number of 1 MHz and 1 kHz samples sampled in the acquisition window are saved as AXI4-Lite registers (see 4.3.56 and 4.3.57). The Delay time and Acquisition Time parameters can be set using two registers (see 4.3.58 and 4.3.59)

2.5 Ring Buffer

The ring buffer enables a user to save a history for the signals. The number of samples in ring buffer can be set dynamically from *application logic* module (PCIe interface) from 0 to 1023. The maximum value for the number of samples can be set in as a module parameter before the FPGA synthesis. The FPGA resources allows up to 16384 samples (composed of 8 floating point numbers) to be stored in ring buffer. In the ring buffer we store the 8 calibrated currents, and additionally is possible to store a counter (if needed).

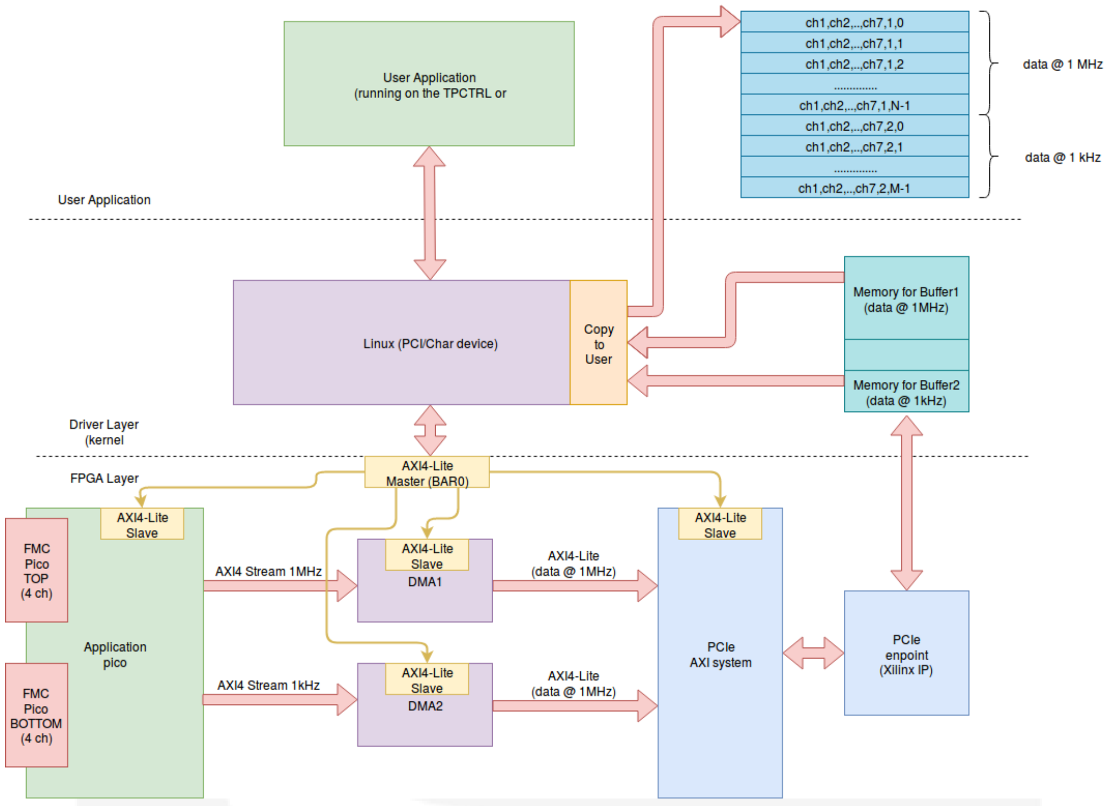
2.6 Scatter-Gather DMA



The DMA (Direct Memory Access) module transfers the data stream from reduced AXI4-Stream slave port (with data, strobe, ready and valid signal) to memory mapped accesses to PCI express module. This advanced implementation of DMA allows queuing of commands, which can be used to write a continuous stream of data to several non-continuous memory location (a common situation with x86 processor with Memory Management Unit).

3. AMC-PICO8-MPS Driver Overview

The following picture reports a schematic overview of the three main players in the MPS system.



4. Address map

4.1 Pico slave (address 0x0000 0000)

4.1.1 CONTROL [R/W] (0x00)

Bit	Name	Access	Description
3:0	RANGE	R/W	Analog frontend input range

4.1.2 CONV_TRG [R/W] (0x04)

Bit	Name	Access	Description
2:0	TRIGGER MUX	R/W	0: fixed 1 1: fixed 0 2: TPCTRL Trigger in (Rx17) 3: Internal Trigger
10:8	CONV SPI MUX	R/W	0: internal OSC (reg CONV_CNTR) 1: TCLKA 2: TCLKB 3: not used

4.1.3 CONV_CNTR [R/W] (0x08)

Bit	Name	Access	Description
11:0	CONV_CNTR	R/W	Counter limit for counter generating the convert signals for ADC. Counter counts from 0 to CONV_CNTR with the frequency of 300 MHz

4.1.4 RING_BUF_CTRL [R/W] (0x0C)

Bit	Name	Access	Description
9:0	RING_BUF_CTRL	R/W	Number of samples in ring buffer.

4.1.5 Register (0x10)

Reserved.

4.1.6 Register (0x14)

Reserved.

4.1.7 Register (0x18)

Reserved.

4.1.8 Register (0x1C)

Reserved.

4.1.9 RAW_CH0 [R] (0x20)

Bit	Name	Access	Description

31:0	RAW_CH0	R/W	2's complement of value read from AD

4.1.10 RAW_CH1 [R] (0x24)

Bit	Name	Access	Description
31:0	RAW_CH1	R/W	2's complement of value read from ADC

4.1.11 RAW_CH2 [R] (0x28)

Bit	Name	Access	Description
31:0	RAW_CH2	R/W	2's complement of value read from ADC

4.1.12 RAW_CH3 [R] (0x2C)

Bit	Name	Access	Description
31:0	RAW_CH3	R/W	2's complement of value read from ADC

4.1.13 CALIB_CH0 [R] (0x30)

Bit	Name	Access	Description
31:0	CALIB_CH0	R/W	Stores value from signal processing module (float)

4.1.14 CALIB_CH1 [R] (0x34)

Bit	Name	Access	Description
31:0	CALIB_CH1	R/W	Stores value from signal processing module (float)

4.1.15 CALIB_CH2 [R] (0x38)

Bit	Name	Access	Description
31:0	CALIB_CH2	R/W	Stores value from signal processing module (float)

4.1.16 CALIB_CH3 [R] (0x3C)

Bit	Name	Access	Description
31:0	CALIB_CH3	R/W	Stores value from signal processing module (float)

4.1.17 VERSION [R] (0x58)

Bit	Name	Access	Description
31:0	VERSION	R/W	FPGA version.

4.1.18 TIMESTAMP [R] (0x5C)

Bit	Name	Access	Description
31:0	TIMESTAMP	R/W	Unix timestamp of FPGA compilation.

4.2 Pico Signal Processing (address 0x0000 0100)

4.2.1 RNG0_GAIN_CH0 (address 0x00)
IEEE754 floating point number (32-bit) R/W.

4.2.2 RNG0_GAIN_CH1 (address 0x04)
IEEE754 floating point number (32-bit) R/W.

4.2.3 RNG0_GAIN_CH2 (address 0x08)
IEEE754 floating point number (32-bit) R/W.

4.2.4 RNG0_GAIN_CH3 (address 0x0C)
IEEE754 floating point number (32-bit) R/W.

4.2.5 RNG0_OFFS_CH0 (address 0x10)
IEEE754 floating point number (32-bit) R/W.

4.2.6 RNG0_OFFS_CH1 (address 0x14)
IEEE754 floating point number (32-bit) R/W.

4.2.7 RNG0_OFFS_CH2 (address 0x18)
IEEE754 floating point number (32-bit) R/W.

4.2.8 RNG0_OFFS_CH3 (address 0x1C)
IEEE754 floating point number (32-bit) R/W.

4.2.9 RNG1_GAIN_CH0 (address 0x20)
IEEE754 floating point number (32-bit) R/W.

4.2.10 RNG1_GAIN_CH1 (address 0x24)
IEEE754 floating point number (32-bit) R/W.

4.2.11 RNG1_GAIN_CH2 (address 0x28)
IEEE754 floating point number (32-bit) R/W.

4.2.12 RNG1_GAIN_CH3 (address 0x2C)
IEEE754 floating point number (32-bit) R/W.

4.2.13 RNG1_OFFS_CH0 (address 0x30)
IEEE754 floating point number (32-bit) R/W.

4.2.14 RNG1_OFFS_CH1 (address 0x34)
IEEE754 floating point number (32-bit) R/W.

4.2.15 RNG1_OFFS_CH2 (address 0x38)
IEEE754 floating point number (32-bit) R/W.

4.2.16 RNG1_OFFS_CH3 (address 0x3C)
IEEE754 floating point number (32-bit) R/W.

4.3 PICO_MOVING_AVERAGE (0x0000 0200)

4.3.1 ID_MODULE (0x00)
(uint32_t) ID module.

4.3.2 TIMESTAMP (0x04)
(uint32_t) Timestamp.

4.3.3 CTRL (0x08)
uint32_t (32-bit). R/W

Value	Access	Description
[0:0]	R/W	Reset active High. This is the only way to clear the over threshold vector and reset the accumulators used for calculating the moving averages.
[1:1]	R/W	0: Infinite Mode 1: Trigger Mode

In order to keep the MPS signal low when in Trigger Mode the CTRL register should be set to 3_h

4.3.4 N_SAMP_H (0x0C)
uint32_t (32-bit). R/W. Number of samples of the moving average window for the HIGH threshold.

4.3.5 N_SAMP_M (0x10)
uint32_t (32-bit). R/W. Number of samples of the moving average window for the MEDIUM threshold.

4.3.6 N_SAMP_L (0x14)
uint32_t (32-bit). R/W. Number of samples of the moving average window for the LOW threshold.

4.3.7 TH_CH0_H (0x18)
Float (32-bit). R/W. Threshold HIGH for channel 0.

4.3.8 TH_CH1_H (0x1C)
Float (32-bit). R/W. Threshold HIGH for channel 1.

4.3.9 TH_CH2_H (0x20)
Float (32-bit). R/W. Threshold HIGH for channel 2.

4.3.10 TH_CH3_H (0x24)
Float (32-bit). R/W. Threshold HIGH for channel 3.

4.3.11 TH_CH4_H (0x28)
Float (32-bit). R/W. Threshold HIGH for channel 4.

4.3.12 TH_CH5_H (0x2C)
Float (32-bit). R/W. Threshold HIGH for channel 5.

4.3.13 TH_CH6_H (0x30)
Float (32-bit). R/W. Threshold HIGH for channel 6.

4.3.14 TH_CH7_H (0x34)
Float (32-bit). R/W. Threshold HIGH for channel 7.

4.3.15 TH_CH0_M (0x38)
Float (32-bit). R/W. Threshold MEDIUM for channel 0.

4.3.16 TH_CH1_M (0x3C)
Float (32-bit). R/W. Threshold MEDIUM for channel 1.

4.3.17 TH_CH2_M (0x40)
Float (32-bit). R/W. Threshold MEDIUM for channel 2.

4.3.18 TH_CH3_M (0x44)
Float (32-bit). R/W. Threshold MEDIUM for channel 3.

4.3.19 TH_CH4_M (0x48)
Float (32-bit). R/W. Threshold MEDIUM for channel 4.

4.3.20 TH_CH5_M (0x4C)
Float (32-bit). R/W. Threshold MEDIUM for channel 5.

4.3.21 TH_CH6_M (0x50)
Float (32-bit). R/W. Threshold MEDIUM for channel 6.

4.3.22 TH_CH7_M (0x54)
Float (32-bit). R/W. Threshold MEDIUM for channel 7.

4.3.23 TH_CH0_L (0x58)
Float (32-bit). R/W. Threshold MEDIUM for channel 0.

4.3.24 TH_CH1_L (0x5C)
Float (32-bit). R/W. Threshold MEDIUM for channel 1.

4.3.25 TH_CH2_L (0x60)
Float (32-bit). R/W. Threshold MEDIUM for channel 2.

4.3.26 TH_CH3_L (0x64)
Float (32-bit). R/W. Threshold MEDIUM for channel 3.

4.3.27 TH_CH4_L (0x68)
Float (32-bit). R/W. Threshold MEDIUM for channel 4.

4.3.28 TH_CH5_L (0x6C)
Float (32-bit). R/W. Threshold MEDIUM for channel 5.

4.3.29 TH_CH6_L (0x70)
Float (32-bit). R/W. Threshold MEDIUM for channel 6.

4.3.30 TH_CH7_L (0x74)
Float (32-bit). R/W. Threshold MEDIUM for channel 7.

4.3.31 READBACK_TH_1MHz (0x78)
Float (32-bit). R/W. Readback.

4.3.32 READBACK_TH_1kHz (0x7C)
Float (32-bit). R/W. Readback.

4.3.33 READBACK_DATA_IN_1MHz (0x80)
Float (32-bit). R/W. Readback.

4.3.34 READBACK_ DATA_IN _1kHz (0x84)
Float (32-bit). R/W. Readback.

4.3.35 READBACK_OVER_TH_H (0x88)
Uint32_t (32-bit). R/W. Readback Over threshold HIGH [7:0].

- [7]: CH7
- ...
- [0]: CH0

4.3.36 READBACK_OVER_TH_M (0x8C)
Uint32_t (32-bit). R/W. Readback Over threshold MEDIUM [7:0].

- [7]: CH7
- ...
- [0]: CH0

4.3.37 READBACK_OVER_TH_L (0x90)

Uint32_t (32-bit). R/W. Readback Over threshold LOW [7:0].

- [7]: CH7
- ...
- [0]: CH0

4.3.38 MANUAL MPS (0x94)

uint32_t (32-bit). R/W.

Value	Access	Description
[0:0]	R/W	Manual MPS bit
[1:1]	R/W	Manual Trigger bit. Trigger sensible on Rising edge.

4.3.39 MPS_CONFIG (0x98)

Value	Access	Description
[0:0]	R/W	0: disable MPS generation 1: enable MPS generation
[3:1]	R/W	MPS DEMUX CONFIG 0: MPS OUT on Tx18 1: MPS OUT on Rx19 2: MPS OUT on Tx19 3: MPS OUT on Rx20 4: MPS OUT on Tx20

4.3.40 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch0 (0xA0)

Float (32-bit). R. Accumulator output of moving average on 1 MHz data. This data is the accumulator output, it should be divided by the number of accumulated samples (see accumulator counter register) in order to calculate the average. These registers are updated only in Trigger Mode.

4.3.41 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch1 (0xA4)

Float (32-bit). R.

4.3.42 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch2 (0xA8)

Float (32-bit). R.

4.3.43 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch3 (0xAC)

Float (32-bit). R.

4.3.44 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch4 (0xB0)

Float (32-bit). R.

4.3.45 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch5 (0xB4)

Float (32-bit). R.

4.3.46 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch6 (0xB8)

Float (32-bit). R.

4.3.47 READBACK_ACCUMULATOR_OUTPUT_1MHz_Ch7 (0xBC)

Float (32-bit). R.

4.3.48 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch0 (0xC0)

Float (32-bit). R. Accumulator output of moving average on 1 kHz data. This data is the accumulator output, it should be divided by the number of accumulated samples (see accumulator counter register) in order to calculate the average. These registers are updated only in Trigger Mode.

4.3.49 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch1 (0xC4)

Float (32-bit). R.

4.3.50 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch2 (0xC8)

Float (32-bit). R.

4.3.51 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch3 (0xCC)

Float (32-bit). R.

4.3.52 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch4 (0xD0)

Float (32-bit). R.

4.3.53 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch5 (0xD4)

Float (32-bit). R.

4.3.54 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch6 (0xD8)

Float (32-bit). R.

4.3.55 READBACK_ACCUMULATOR_OUTPUT_1kHz_Ch7 (0xDC)

Float (32-bit). R.

4.3.56 READBACK_ACCUMULATOR_COUNTER_1MHz (0xE0)

Uint32_t (32-bit). R. This register represents the number of data sampled by the moving average module (of 1 MHz samples) in Trigger Mode (in every trigger window). It can be used to calculate the average from the accumulator output registers.

4.3.57 READBACK_ACCUMULATOR_COUNTER_1kHz (0xE4)

Uint32_t (32-bit). R. This register represents the number of data sampled by the moving average module (of 1 kHz samples) in Trigger Mode (in every trigger window). It can be used to calculate the average from the accumulator output registers.

4.3.58 ACQ_NSAMPLES_TRIGGER_MODE (0xE8)

Uint32_t (32-bit). R. This register sets the acquisition window in units of 1 MHz samples, in Trigger Mode.

4.3.59 DELAY_NSAMPLES_TRIGGER_MODE (0xEC)

Uint32_t (32-bit). R. This register sets the delay window in units of 1 MHz samples, in Trigger Mode.

4.4 EEPROM FMC 2 (0x0000 0400)

4.4.1 Status (0x00)

Bit	Name	Access	Description
0	DONE	R/W1C	Done, the data has been read/written. Write 1 to clear.

4.4.2 Control (0x04)

Bit	Name	Access	Description
0	GO	W	Starts the memory access. This bit is automatically cleared.
1	R_WN	R/W	Read/not write. Selects the type of memory access.

4.4.3 Address (0x08)

Bit	Name	Access	Description
12:0	ADDRESS	R/W	Address

4.4.4 Write data (0x0C)

Bit	Name	Access	Description
7:0	WR_DATA	R/W	Data to be written

4.4.5 Read data (0x10)

Bit	Name	Access	Description
7:0	RD_DATA	R	Data read from memory

4.5 EEPROM FMC 1 (0x0000 0500)

4.5.1 Status (0x00)

Bit	Name	Access	Description
0	DONE	R/W1C	Done, the data has been read/written. Write 1 to clear.

4.5.2 Control (0x04)

Bit	Name	Access	Description
0	GO	W	Starts the memory access. This bit is automatically cleared.
1	R_WN	R/W	Read/not write. Selects the type of memory access.

4.5.3 Address (0x08)

Bit	Name	Access	Description
12:0	ADDRESS	R/W	Address

4.5.4 Write data (0x0C)

Bit	Name	Access	Description
7:0	WR_DATA	R/W	Data to be written

4.5.5 Read data (0x10)

Bit	Name	Access	Description
7:0	RD_DATA	R	Data read from memory

4.6 ID_MODULE (0x0000 0700) (To be implemented)

4.6.1 REG_ID (0x00)

uint32_t (32-bit)

4.6.2 REG_VERSION (0x04)

uint32_t (32-bit)

4.6.3 REG_TIMESTAMP (0x08)

uint32_t (32-bit)

4.6.4 REG_GIT_SHA1 (0x0C)

uint32_t (32-bit)