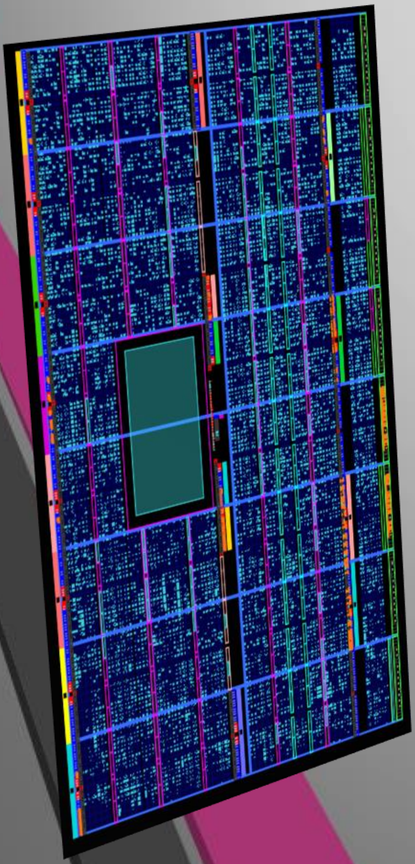
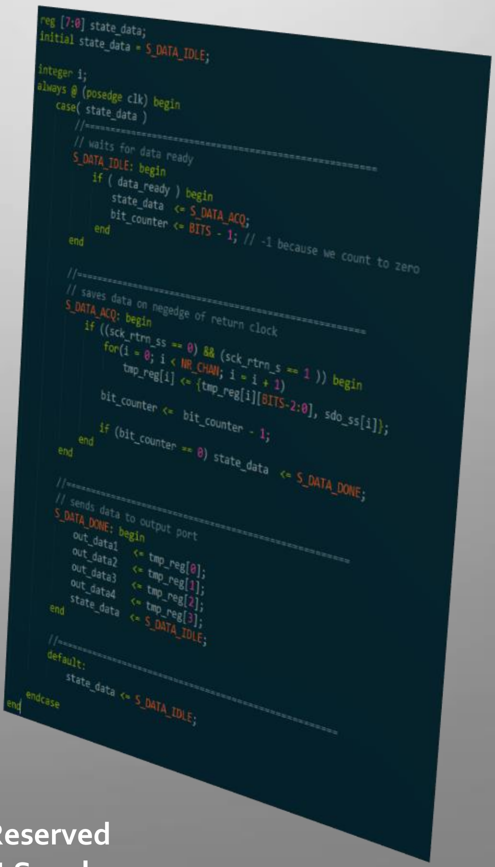
AMC-SOFFOX

4-channel Bipolar 20-bit 4SFP+

Board Support Package Overview





```
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
```

MTCA.4 – MicroTCA for Physics

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

Table of Contents

1. INTRODUCTION 5

2. AMC-SOFFOX BSP OVERVIEW 6

2.1 SPI MODULES 7
2.2 SIGNAL PROCESSING 7
2.3 EEPROM INTERFACE 7
2.4 IIR FILTER 7
2.5 RING BUFFER 8
2.6 PICO FASTPS 8
2.7 FASTPS IF 4SFP 8

3. ADDRESS MAP 10

3.1 PICO SLAVE (ADDRESS 0X0000 0000) 10
3.2 PICO SIGNAL PROCESSING (ADDRESS 0X0000 0100) 11
3.3 EEPROM (0x0000 0200) 13
3.4 IIR_FILTER (0x0000 0300) 14
3.5 ID_MODULE (0x0000 0700) 17
3.6 FASTPS_IF_4SFP (0x0000 0800) 18
3.7 PICO_FASTPS (0x0000 0C00) 18

Document Revisions

Document Revision	Date	Comment
1.0	July 25th 2018	First Release
1.1	June 12th 2019	Added saturation limits Corrected IIR filter bug

1. Introduction

This document describes the AMC-SOFFOX Board Support Package (BSP). This solution is based on the CAEN ELS DAMC-FMC25 AMC carrier board and the AMC-SOFFOX BSP extends the standard BSP of its carrier board.

In this document the AMC-SOFFOX 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-SOFFOX BSP Overview

The following block diagram shows application logic on AMC-SOFFOX. Some connections (such as range selection, EEPROM interface, ports 17-20, mux selector) are omitted for clarity.

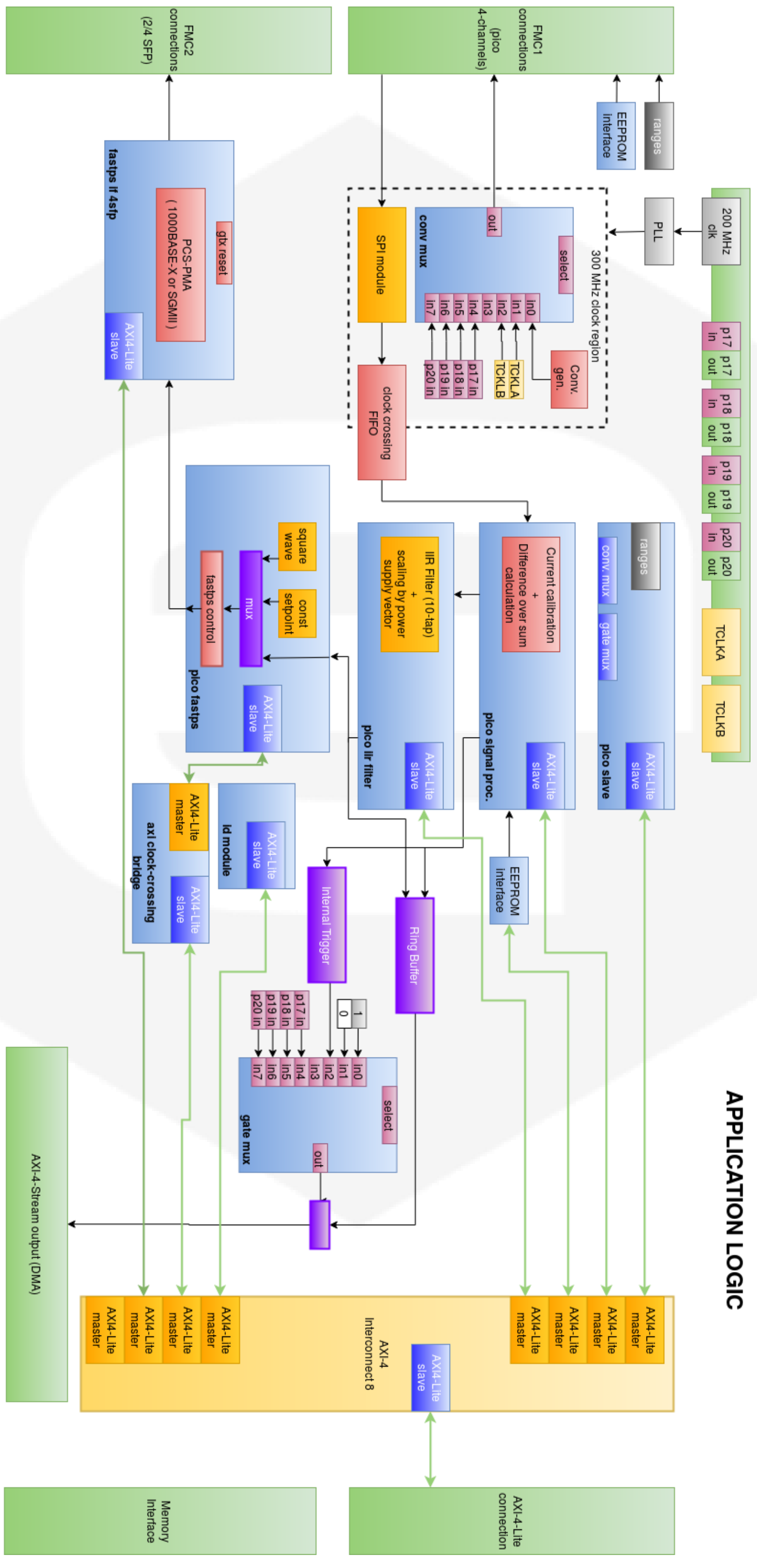


Figure 1: AMC-SOFFOX Application logic block diagram

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 4 channels 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 16 parameters in total (gain and offset, 2 different input ranges for each of the 4 channels).

After the calibration, the currents are used for the beam position calculation (X and Y positions). The output of this module are then the 4 calibrated currents and the X/Y positions of the beam.

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 IIR Filter

As for the calibration module, the *pico iir filter* module is a wrapper for the *iir filter* module. The IIR digital filter is implemented in FPGA logic to handle 10 taps. The following formula is implemented:

$$y_n = \sum_{i=0}^{N} b_i x_{n-i} - \sum_{i=1}^{N} a_i y_{n-i}$$

where N = 9 and $a_0 = 1$.

The IIR filter is applied only to the Y position. The IIR output signal is then multiplied by a vector of 4 coefficients, also called *power supply vector* ([psv1, psv2, psv3, psv4]), and sent to the *pico_fastps* module.

Since the IIR may become instable for some configuration, to avoid the generation of very large numbers that at the end are sent to the FAST-PSs, we limit the output of this module. The limits are stored in 8 R/W registers (see 3.4.51-3.4.58) and can be configured by the user.

IIR filter has a control register that is used to:

- Disable the IIR calculations, thus IIR input = IIR output;
- Enable the IIR calculations;
- Reset the IIR filter output buffer. This option is useful when the IIR filter saturates.

All the parameters (control register, IIR coefficients, power supply vector and saturation limits) are controlled through AXI-Lite.

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 4 calibrated currents, the X/Y positions and the output of the iir filter.

2.6 Pico Fastps

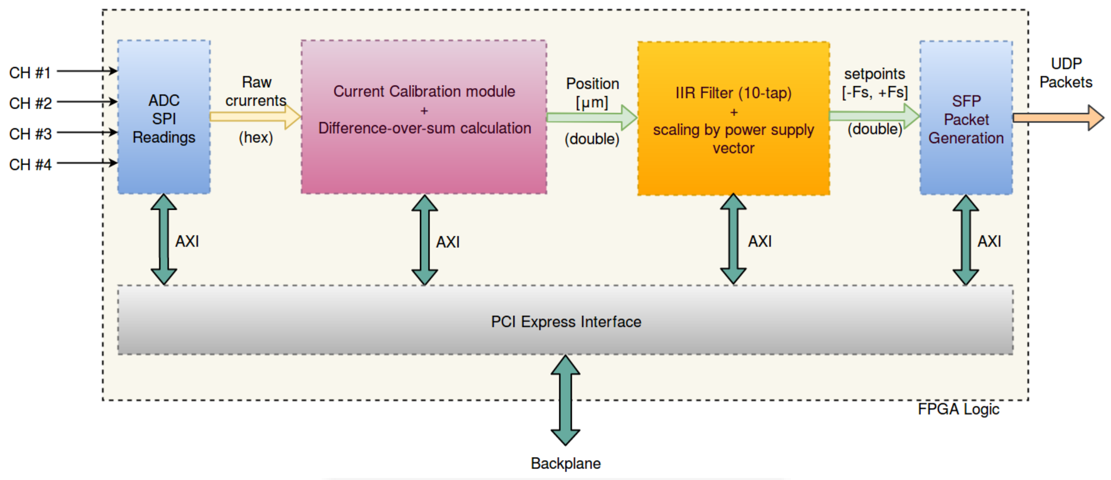
This module contains essentially a mux and the *fastps_control* module. The *fastps_control* module takes the setpoints and ids of the 4 fastps modules and builds up the SFP packet. Before the *fastps_control* module there is a mux that selects the setpoints to be encapsulated in the SPF packet. There are three possible configurations:

1. IIR output,
2. Fixed setpoint (setpoint constant value),
3. Square wave (2 setpoints constant values and a prescaler to set the frequency, duty 50%). Period in [s] is prescaler/125 MHz.

All the configuration parameters (control register, setpoints and prescaler) are accessible through AXI-Lite.
The *fastps_control* module has been written in HLS (2015.4).

2.7 Fastps if 4sfp

The *fastps if 4sfp* module is a wrapper for the PCS-PMA. An AXI-Lite connection provides access to the status_vector of the 4 SFP transceivers.


CH #1
CH #2
CH #3
CH #4
ADC SPI Readings
Raw crurrents
(hex)
Current Calibration module + Difference-over-sum calculation
Position [μm]
(double)
IIR Filter (10-tap) + scaling by power supply vector
setpoints [-Fs, +Fs]
(double)
SFP Packet Generation
UDP Packets
AXI
AXI
AXI
AXI
PCI Express Interface
FPGA Logic
Backplane

3. Address map

3.1 Pico slave (address 0x0000 0000)

3.1.1 CONTROL [R/W] (0x00)

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

3.1.2 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

3.1.3 RING_BUF_CTRL [R/W] (0x0C)

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

3.1.4 RAW_CH0 [R] (0x20)

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

3.1.5 RAW_CH1 [R] (0x24)

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

3.1.6 RAW_CH2 [R] (0x28)

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

3.1.7 RAW_CH3 [R] (0x2C)

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

3.1.8 CALIB_CH0 [R] (0x30)

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

3.1.9 CALIB_CH1 [R] (0x34)

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

3.1.10 CALIB_CH2 [R] (0x38)

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

3.1.11 CALIB_CH3 [R] (0x3C)

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

3.1.12 MUX_START [R/W] (0x40)

Bit	Name	Access	Description
2:0	MUX_START	R/W	0: tlast_pkt_gen 1: iir_out_valid 2: iir_in_valid 3: positions_valid

3.1.13 VERSION [R] (0x58)

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

3.1.14 TIMESTAMP [R] (0x5C)

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

3.2 Pico Signal Processing (address 0x0000 0100)

3.2.1 RNG0_GAIN_CH0 (address 0x00)

IEEE754 floating point number (32-bit) R/W.

3.2.2 RNG0_GAIN_CH1 (address 0x04)

IEEE754 floating point number (32-bit) R/W.

3.2.3 RNG0_GAIN_CH2 (address 0x08)

IEEE754 floating point number (32-bit) R/W.

3.2.4 RNG0_GAIN_CH3 (address 0x0C)

IEEE754 floating point number (32-bit) R/W.

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

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

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

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

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

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

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

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

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

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

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

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

3.2.17 DETECTOR_MODE (address 0x58)

Bit	Name	Access	Description
1:0	DETECTORE_MODE	R/W	00: 90° 01: 45° 10: debug

3.2.18 CUR2POS GAIN X (address 0x5C)
IEEE754 floating point number (32-bit) R/W.

3.2.19 CUR2POS GAIN y (address 0x60)
IEEE754 floating point number (32-bit) R/W.

3.2.20 CONFIG_TOP (address 0x64)
IEEE754 uint32_t (32-bit) R/W.

3.2.21 CONFIG_RIGHT (address 0x68)
IEEE754 uint32_t (32-bit) R/W.

3.2.22 CONFIG_BOTTOM (address 0x6C)
IEEE754 uint32_t (32-bit) R/W.

3.2.23 CONFIG_LEFT (address 0x70)
IEEE754 uint32_t (32-bit) R/W.

3.2.24 OFFSET_POSX (address 0x74)
IEEE754 uint32_t (32-bit) R/W.

3.2.25 OFFSET_POSY (address 0x78)
IEEE754 uint32_t (32-bit) R/W.

3.3 EEPROM (0x0000 0200)

3.3.1 Status (0x00)

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

3.3.2 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.

3.3.3 Address (0x08)

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

3.3.4 Write data (0x0C)

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

3.3.5 Read data (0x10)

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

3.4 IIR_FILTER (0x0000 0300)

3.4.1 CTRL (0x00)

Value	Access	Description
0x00	R/W	Disable IIR. The IIR outputs are the IIR inputs (current readings) multiplied by the respective fastps coefficients.
0x01	R/W	Enable IIR. The IIR filter is applied to the IIR inputs (current readings) and finally multiplied by the respective fastps coefficients.
0x02	R/W	Reset Output Buffer. With this option it is possible to reset the IIR output buffer, useful if the filter saturates.

3.4.2 COEFF_A0_LSB (0x18)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.3 COEFF_A0_MSB (0x1C)
uint32_t (32-bit) R/W.

3.4.4 COEFF_A1_LSB (0x20)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.5 COEFF_A1_MSB (0x24)
uint32_t (32-bit) R/W.

3.4.6 COEFF_A2_LSB (0x28)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.7 COEFF_A2_MSB (0x2C)
uint32_t (32-bit) R/W.

3.4.8 COEFF_A3_LSB (0x30)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.9 COEFF_A3_MSB (0x34)
uint32_t (32-bit) R/W.

3.4.10 COEFF_A4_LSB (0x38)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.11 COEFF_A4_MSB (0x3C)
uint32_t (32-bit) R/W.

3.4.12 COEFF_A5_LSB (0x40)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.13 COEFF_A5_MSB (0x44)
uint32_t (32-bit) R/W.

3.4.14 COEFF_A6_LSB (0x48)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.15 COEFF_A6_MSB (0x4C)
uint32_t (32-bit) R/W.

3.4.16 COEFF_A7_LSB (0x50)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.17 COEFF_A7_MSB (0x54)
uint32_t (32-bit) R/W.

3.4.18 COEFF_A8_LSB (0x58)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.19 COEFF_A8_MSB (0x5C)
uint32_t (32-bit) R/W.

3.4.20 COEFF_A9_LSB (0x60)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.21 COEFF_A9_MSB (0x64)
uint32_t (32-bit) R/W.

3.4.22 COEFF_B0_LSB (0x68)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.23 COEFF_B0_MSB (0x6C)
uint32_t (32-bit) R/W.

3.4.24 COEFF_B1_LSB (0x70)
uint32_t (32-bit). LSB has to be written before the MSB for data integrity.

3.4.25 COEFF_B1_MSB (0x74)
uint32_t (32-bit) R/W.

3.4.26 COEFF_B2_LSB (0x78)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.27 COEFF_B2_MSB (0x7C)
uint32_t (32-bit) R/W.

3.4.28 COEFF_B3_LSB (0x80)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.29 COEFF_B3_MSB (0x84)
uint32_t (32-bit) R/W.

3.4.30 COEFF_B4_LSB (0x88)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.31 COEFF_B4_MSB (0x8C)
uint32_t (32-bit) R/W.

3.4.32 COEFF_B5_LSB (0x90)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.33 COEFF_B5_MSB (0x94)
uint32_t (32-bit) R/W.

3.4.34 COEFF_B6_LSB (0x98)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.35 COEFF_B6_MSB (0x9C)
uint32_t (32-bit) R/W.

3.4.36 COEFF_B7_LSB (0xA0)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.37 COEFF_B7_MSB (0xA4)
uint32_t (32-bit) R/W.

3.4.38 COEFF_B8_LSB (0xA8)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.39 COEFF_B8_MSB (0xAC)
uint32_t (32-bit) R/W.

3.4.40 COEFF_B9_LSB (0xB0)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.41 COEFF_B9_MSB (0xB4)
uint32_t (32-bit) R/W.

3.4.42 COEFF_FASTPS0_LSB (0xB8)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.43 COEFF_ FASTPS0_MSB (0xBC)
uint32_t (32-bit) R/W.

3.4.44 COEFF_ FASTPS1_LSB (0xC0)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.45 COEFF_ FASTPS1_MSB (0xC4)
uint32_t (32-bit) R/W.

3.4.46 COEFF_ FASTPS2_LSB (0xC8)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.47 COEFF_ FASTPS2_MSB (0xCC)
uint32_t (32-bit) R/W.

3.4.48 COEFF_ FASTPS3_LSB (0xD0)
uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.4.49 COEFF_ FASTPS3_MSB (0xD4)
uint32_t (32-bit) R/W.

3.4.50 SATURATION_FLAG (0xD8)
unit32_t (32-bit) R/W.

3.4.51 FASTPS0_MAX_LIM (0xDC)
float (32-bit) R/W.

3.4.52 FASTPS1_MAX_LIM (0xE0)
float (32-bit) R/W.

3.4.53 FASTPS2_MAX_LIM (0xE4)
float (32-bit) R/W.

3.4.54 FASTPS3_MAX_LIM (0xE8)
float (32-bit) R/W.

3.4.55 FASTPS0_MIN_LIM (0xEC)
float (32-bit) R/W.

3.4.56 FASTPS1_MIN_LIM (0xF0)
float (32-bit) R/W.

3.4.57 FASTPS2_MIN_LIM (0xF4)
float (32-bit) R/W.

3.4.58 FASTPS3_MIN_LIM (0xF8)
float (32-bit) R/W.

3.5 ID_MODULE (0x0000 0700)

3.5.1 REG_ID (0x00)
uint32_t (32-bit)

3.5.2 REG_VERSION (0x04)
uint32_t (32-bit)

3.5.3 REG_TIMESTAMP (0x08)

uint32_t (32-bit)

3.5.4 REG_GIT_SHA1 (0x0C)

uint32_t (32-bit)

3.6 FASTPS_IF_4SFP (0x0000 0800)

3.6.1 ID (0x00)

uint32_t (32-bit)

3.6.2 SFP_MOD0 (0x04)

uint32_t (32-bit)

3.6.3 SFP_LOS (0x08)

uint32_t (32-bit)

3.7 PICO_FASTPS (0x0000 0C00)

3.7.1 MAC_DEST_LSB (0x00)

uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.7.2 MAC_DEST_MSB (0x04)

uint32_t (32-bit) R/W.

3.7.3 MAC_SRC_LSB (0x08)

uint32_t (32-bit). R/W. LSB has to be written before the MSB for data integrity.

3.7.4 MAC_SRC_MSB (0x0C)

uint32_t (32-bit) R/W.

3.7.5 IP_DEST (0x10)

uint32_t (32-bit) R/W.

3.7.6 IP_SRC (0x14)

uint32_t (32-bit) R/W.

3.7.7 PICO_FASTPS_CTRL (0x18)

uint32_t (32-bit)

Value	Access	Description
0x00	R/W	IIR output
0x01	R/W	Constant output voltage (SP1_1, SP2_1, SP3_1, SP4_1)
0x02	R/W	Square wave output voltage (SP1_1/2, SP2_1/2, SP3_1/2, SP4_1/2). The voltage will jump from SPX_1 to SPX_2 with a frequency equal to 125 MHz/PRESCALER. Max frequency

		limited to 100 KHz (PRESCALER 0x04E2). Duty cycle 50% fixed.

3.7.8 PRESCALER (0x1C)
uint32_t (32-bit) R/W.

3.7.9 SP1_1 (0x20)
uint32_t (32-bit) R/W.

3.7.10 SP1_2 (0x24)
uint32_t (32-bit) R/W.

3.7.11 SP2_1 (0x28)
uint32_t (32-bit) R/W.

3.7.12 SP2_2 (0x2C)
uint32_t (32-bit) R/W.

3.7.13 SP3_1 (0x30)
uint32_t (32-bit) R/W.

3.7.14 SP3_2 (0x34)
uint32_t (32-bit) R/W.

3.7.15 SP4_1 (0x38)
uint32_t (32-bit) R/W.

3.7.16 SP4_2 (0x3C)
uint32_t (32-bit) R/W.