LUND UNIVERSITY

BPM system for ESS - Specification

Implementation specification

Fredrik Kristensen

6/7/2016

A general overview of the BPM system is given here along with a detailed description of the used control algorithm and its hardware implementation. The SW interface, register and memory map, is specified and expected SW usage is described.

1 Version history

Version	Name & Date	Note
0.1	Fredrik Kristensen, 7 Jun -16	Initial draft
0.2	Fredrik Kristensen, 9 Aug-16	Phase Ref moved from ADC7 to ADC4
0.3	Fredrik Kristensen, 9 Aug-16	BPM_ID updated to 0x03. Wait for DAQ done before User IRQ added.
0.4	Fredrik Kristensen, 10 Aug-16	BPM_ID updated to 0x04. User IRQ based on daq_done completed.
0.5	Fredrik Kristensen, 17 Aug-16	BPM_ID updated to 0x05. Position IRQ mask added in BPM_BOARD_SETUP. DAC output inverter option added in BPM_BOARD_SETUP. Input interlocks removed. PMS state and PMS signals removed.
0.51	Fredrik Kristensen, 19 Aug-16	Only document changes. Removed PMS signals from registers.
0.6	Fredrik Kristensen, 6 Sept-16	BPM_ID updated to 0x06. FIR filter added. Second BPM added. Registers extended for 2nd BPM Memory storage changed.
0.7	Fredrik Kristensen, 21-Mar-17	Reference channel moved to ADC 10, BPM1 moved to adc channel 2-5 and BPM2 remains on channel 6-9. Channel 1 is not used.
0.8	Fredrik Kristensen, 25-Mar-17	Reference magnitude and phase are added to the antenna mag and arg memory channels, see 6.3.4.1.
0.9	Fredrik Kristensen, 25-Mar-17	FW fix, sum-phase corrected.
0.10	Fredrik Kristensen, 26-Mar-17	FW fix, gated start trigger with STRUCK ARM register.
0.11	Fredrik Kristensen, 27-Mar-17	FW fix, allowed to use smaller struck sample count to store to memory than actual samples between start and end trigger.
0.12	Fredrik Kristensen, 30-Mar-17	Added beta version of Self-triggering, see

Contents

2 Abbreviations

BPM	**Beam Position Monitor**
EPICS	**Experimental Physics and Industrial Control System**
ESS	**European Spallation Source**
FPGA	**Field Programmable Gate Array**
FW	**FirmWare**
GPS	**Global Positioning System**
KISS	**Keep It Simple Stupid**
MPS	**Machine Protection System**
LLRF	**Low Level RF**
LPS	**Local Protection System**
PMS	**Post Mortem System**
SEL	**Self-Excited Mode**

3 Introduction

"*ESS, the European Spallation Source, will be a major user facility at which researchers from academia and industry will investigate scientific questions using neutron beams. Neutron methods provide insights about the molecular building blocks of matter not available by other means. They are used for both basic and applied research. European nations are working together in order to build, in southern Scandinavia, this slow neutron source of unparalleled power and scientific performance. ESS will deliver its first protons to a solid, rotating tungsten target in 2019, which will in turn generate neutrons for delivery to an initial suite of seven neutron scattering research instruments. ESS will reach its full design specifications in 2025, with a suite of 22 research instruments.*" Extract from Chapter 1, Introduction: The evolving story [1].

In **Table 1** the high level parameters and requirements of ESS are shown. Worth noticing is that the facility should be operational 5000 hours per year (a year consists of 8760 hours) with a reliability of 95%. Since ESS consists of thousands of subsystems and it's the sum of all these subsystems that makes up the 95% system reliability, each subsystem needs to have a much higher reliability. Hence the KISS design strategy should be used to the extent possible.

Table 1: High level parameters, Chapter 1 [1].

Parameter	Unit	Value
Average beam power	MW	5
Number of target stations		1
Number of instruments in construction budget		22
Number of beam ports		48
Number of moderators		2
Separation of ports	Degrees	5
Proton kinetic energy	GeV	2.5
Average macro-pulse current	mA	50
Macro-pulse length	ms	2.86
Pulse repetition rate	Hz	14
Maximum accelerating cavity surface field	MV/m	40
Annual operating period	H	5000
Reliability	%	95

ESS consists of four major parts: source, accelerator, target and instruments. In the ion source protons are created. In the accelerator these protons are gathered in bundles and accelerated towards the target. The target is a rotating block of Tungsten that when the protons hit will emit neutrons. Around the target instrument stations are placed, these will use the neutrons for different scientific experiments.

The accelerator part is divided into a number of sections handling the protons at an increasing energy level. Each section use an electric field to accelerate the protons and these fields are generated in cavities, which can be seen as big barrels. In order for the protons to be accelerated

while passing through the cavities, the electric fields must be at the correct amplitude and phase when the protons enter the field. The LLRF system is used to control the amplitude and phase of each field and to generate the phase reference for the accelerator. The BPM checks that the beam is centred and not deviating from the accelerator central line. A deviating beam will hit something else than the target, potentially causing a lot of damage. For more details on the accelerator see Chapter 4 [1]

4 BPM System functionalities

An overview of the BPM functionality is shown in **Figure 1**. **PICTURE NEEDS TO BE UPDATED TO INCLUDE NEAR-IQ.**

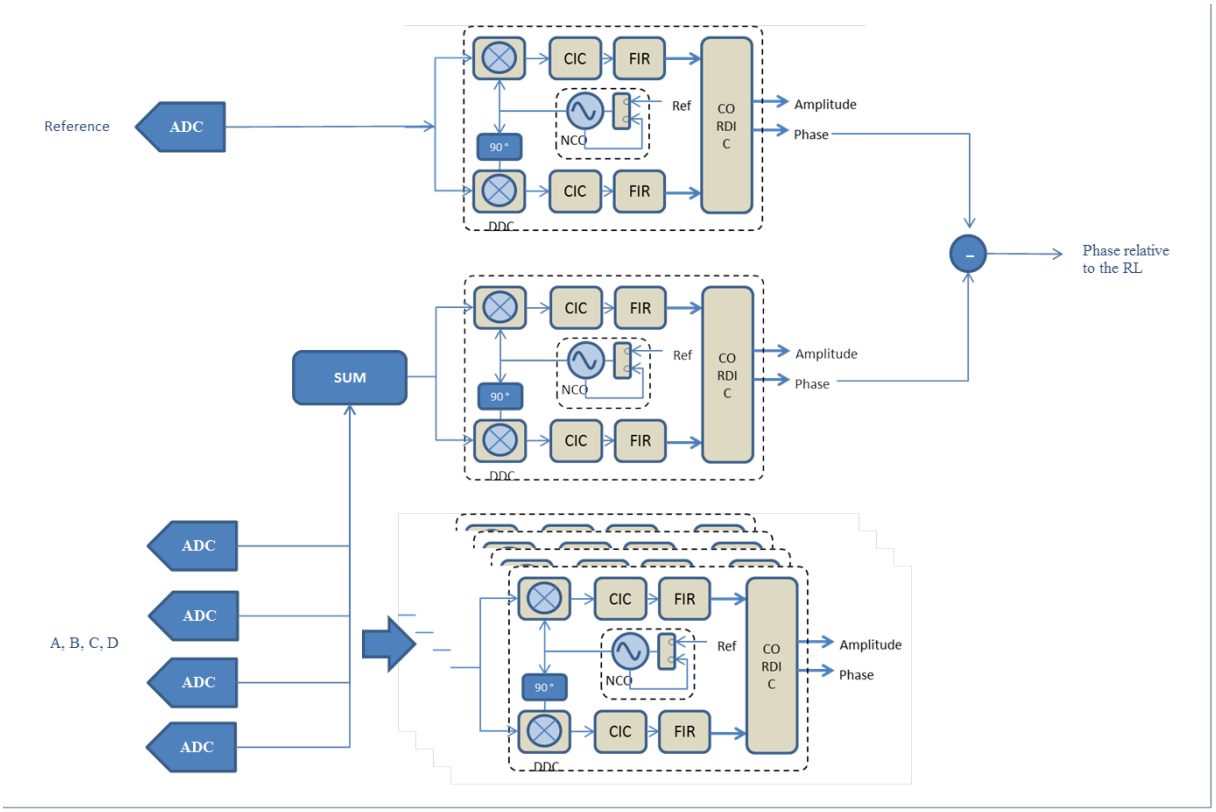

Figure 1, BPM overview.

4.1 IQ sampling

The most straight forward way to get from an analog RF signal to digital IQ samples is to sample four times faster than the RF frequency, i.e. with 90 degrees between each sample, since it's 90° between I and Q. A drawback with this is that RF signals tend to have such a high frequency that the digital logic can't cope. The solution is either to down sample, to have a constant*360+90 degrees between two samples, or to down convert the RF signal with an analog mixer and then sample the down converted signal (f_{IF}) with 90° between the samples.

In LLRF the RF input is either 352.21 or 704.42 MHz and the required operating speed of the PI-ctrl is around 10 MHz. To get from the RF signal to 10 Msamples/s both down sampling and averaging was used in the first version of the LLRF, as shown in **Figure 2**. To do averaging of the IQ samples they need to be rotated to the same place in the IQ plane, in case of 90° between sample rotation is simply changing sign and place of the I and Q components, e.g. the IQ point a+ib rotated 90° is –b+ia.

The upside with sampling with 90° in between is the easy rotation, the downside is that it's sensitive to dc-offset in the RF signal, non-linearities in ADC and mixers, and harmonic distortion. These effects can be lessened by choosing a non 90° angle between samples. Taking N samples in M RF-periods, i.e. 360*M/N degrees between each sample, I and Q can be calculated as:

$$I = \frac{2}{N}\sum_{i=0}^{N-1} s_i \cdot \sin\left(i \cdot 2\pi \cdot \frac{M}{N}\right)$$

$$Q = \frac{2}{N}\sum_{i=0}^{N-1} s_i \cdot \cos\left(i \cdot 2\pi \cdot \frac{M}{N}\right)$$

, where s_i is the i_{th} sample. This is referred to as near-IQ sampling or non-IQ sampling. It's no longer easy to rotate samples but if the sin and cos factors are pre-calculated it still allows for an efficient digital implementation. Note that this is a general solution and also works for 90° between samples, e.g. using M=17 and N=4 will produce the same result as shown in **Figure 2**, with the same downsides.

All of this about IQ sampling and non-IQ sampling and some more is very nicely presented in [2].

The latest version of LLRF uses near-IQ sampling with arbitrary N and M up to 255. The final numbers for N and M is still to be decided.

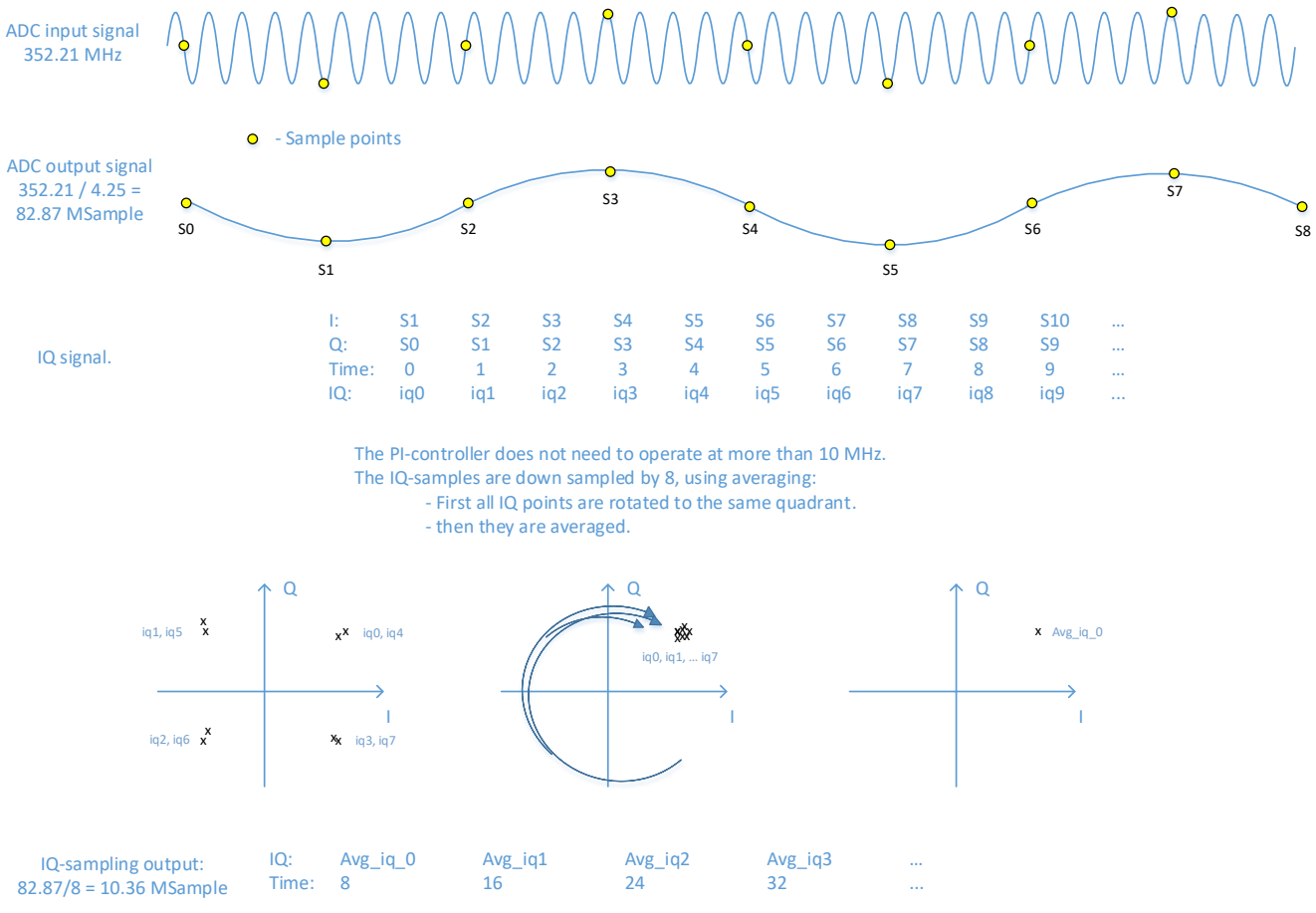

Figure 2, IQ sampling.

4.2 Filters

TBD

4.3 Timing and triggers

In **Table 1** the two major timing parameters for ESS are found; macro-pulse length 2.86 ms and pulse repetition rate 14 Hz. Fourteen times per second a 2.86 ms long proton beam will be accelerated through the cavities. Before the beam can be accelerated by a cavity, there need to be an electrical field in the cavity. This field will be ramped up just before the beam arrives; the ramp up is allowed to take 300 us. The long ramp up time is used to put low stress on the amplifiers.

All BPM timing is controlled by triggers:

- PULSE_START: signals that the beam arrives, time to start beam compensation.
- PULSE_END: signals end of beam, time to turn of the cavity field.

Figure 3 shows a conceptual view of beam and cavity field timing along with the triggers and cavity power. That cavity power decrees during beam is because the beam contributes with power when it pass through the cavity.

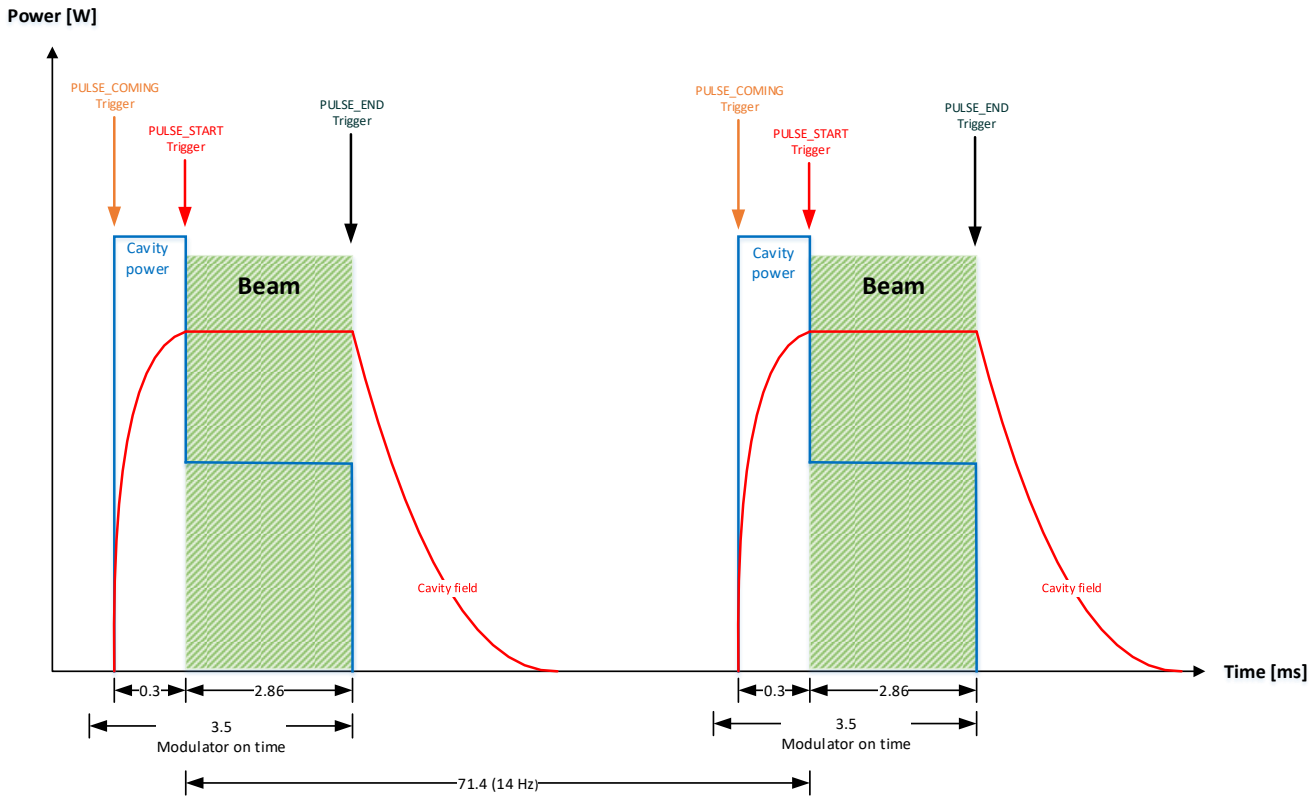

Figure 3, Cavity power and beam timing.

4.3.1 Continues wave (CW) or Pulse-mode

ESS is a pulse-mode facility, hence the BPM system is designed for pulse mode, but in certain circumstances it can be used in CW-mode. Whether the BPM is operating in CW or pulse mode is only depending on the timing triggers, e.g. if BPM only receives the PULSE_START trigger it will continue operation until it's shut down or a PULSE_END is received.

The drawback with CW mode is that memories are not large enough to store all data, the first ~3.5 ms of data will be recorded, counted from PULSE_START trigger.

4.4 Position Monitor

The position monitor use the magnitude value of the four antennas A, B, C and D to determine where the beam is positioned. **Figure 4** shows the antenna positions in the beam pipe. Position is calculated as:

$$Pos_x = \frac{A_m - B_m}{A_m + B_m}$$

$$Pos_y = \frac{C_m - D_m}{C_m + D_m}$$

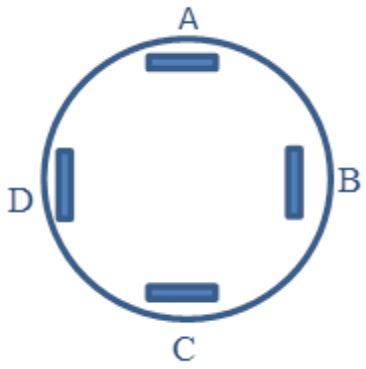

Figure 4, Beam position antennas.

5 RTM

5.1 DWC8VM1 Setup

To be expanded.

Use SIS8300-L register 0x12F to control the below settings. See SW functions for details.

Set interlock0, interlock1 and interlock_enable to ‘1’.

6 Digitizer board

The main function of the digitizer is to sample and store all measured signals from an accelerator section and to provide the output measurements. The digitizer used here is the SIS8300-L MTCA.4 DIGITIZER [3], which features 10 125 MS/s 16-bit ADCs and two 16-bit feedback DACs. The central part of the digitizer is an FPGA that handles sample logic and storage control of the input samples from the ADCs.

In addition the FPGA that can be used for custom implementations. In this design it's used to implement the Beam Position Monitor (BPM) functionality. The custom logic is added to the already existing Struck FW as a sub unit, preserving the original Struck functionality. An overview is shown in **Figure 5**.

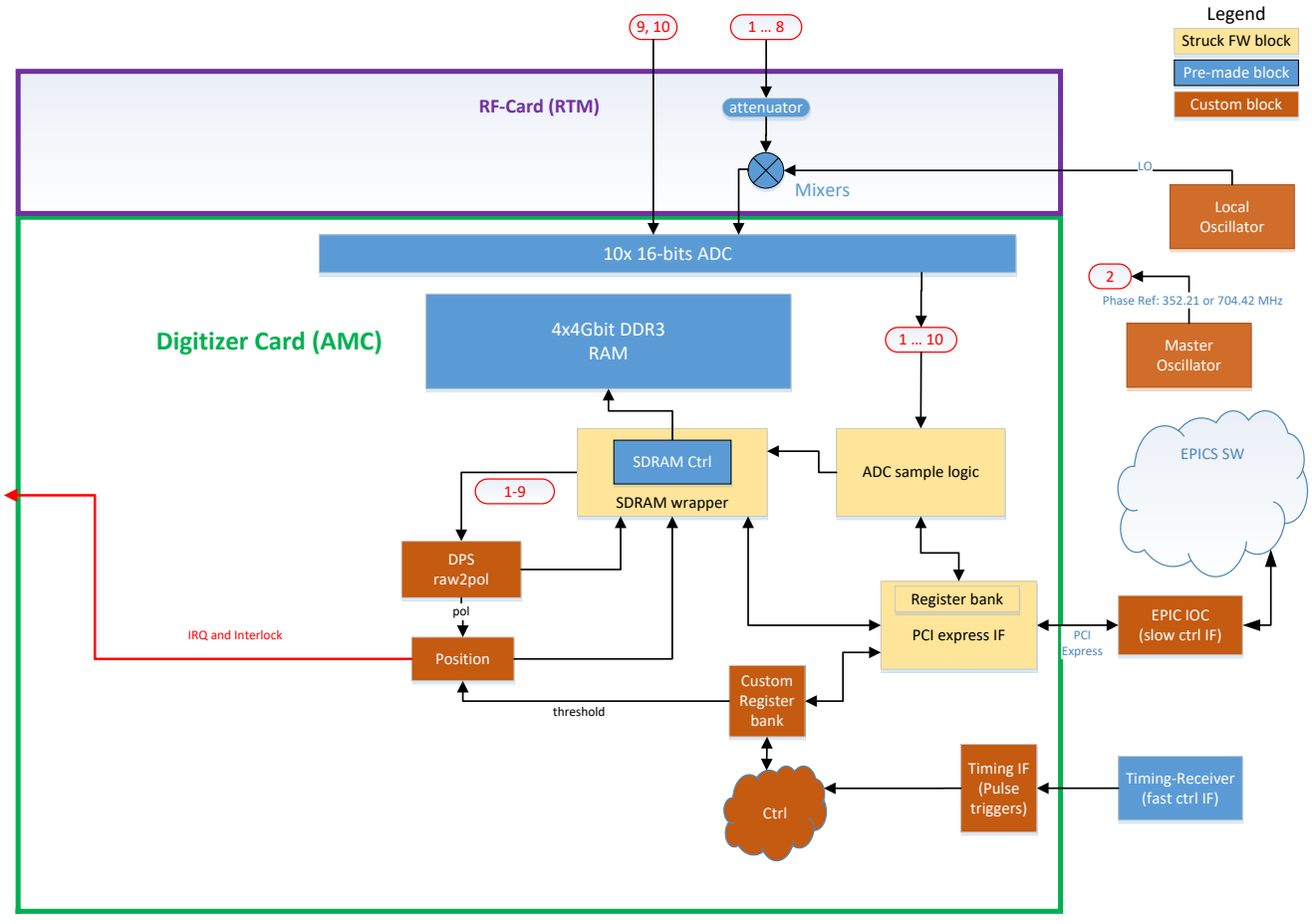

Figure 5. Digitizer and RTM board.

6.1 Board resources

The key properties of the SIS8300-L card are listed below and are described in [3].

- AMC .4 µTCA for Physics Board
- 4 Lane PCI Express Interface

- Dual SFP Card Cage for optional Multi Gigabit Link
- Xilinx Virtex 6 FPGA
- DDR3 Memory Interface
- 4 x 4Gbit default DDR3 memory
- Atmega128 IPMI
- External Clock and Trigger Inputs
- Frontpanel digital I/O (4in/4 out) on Harlink Connectors
- RTM ADC Analog Inputs, I2C-Bus, DAC Analog Outputs
- 10 ADC Channels 125MS/s, 16-Bit
- 2 DAC Channels 250MS/s, 16-Bit
- Clock distribution with phase shifting
- 8 M-LVDS µTCA Ports
- 2 µTCA Clocks

Where the main parts are the ADCs, DACs, FPGA and the DDR3 memory.

6.1.1 Input and output

The input to the ten ADCs are connect to the RTM connector together with output from the two DACs. The 8 M-LVDS and the PCI Express are connected to the MTCA backplane. The front panel has 4 input and 4 output harlink connecters, a clock input and the output from the two DACs.

6.2 Struck Firmware

The main parts of the provided Firmware (FW), is a PCI Express interface, an ADC control and a DDR3 memory interface. Some minor functionality for testing are also provided. A description of the FW is found in [3] and the source code is available through [4].

The PCI Express interface provides read and write (R/W) direct memory access (DMA) to the DDR3 memory and a register interface to the basic parameters of the FW. The register interface also has a section reserved for custom registers.

The ADC control includes internal trigger control, ring buffers, sample logic and DDR3 memory storage. Internal trigger control and ring buffers can be omitted through generics in the VHDL source code. There is also a histogram function that stores a histogram of the samples from channel 1 and 2. Sample logic can either be one control unit per ADC pair or one per two ADC pairs.

The memory interface provides access to the off-chip DDR3 memories. The FW by default allows the PCI DMA to read from memory and ADC samples to be written to memory. In order for the PCI DMA to write to memory the ADC samples to memory path has to be disabled through a register setting. The FW does not provide any read or write access to memory for custom FPGA functions.

6.2.1 FW changes

Internal trigger control and ring buffers are removed as well as the histogram function. External triggers are controlled by custom registers instead of the Struck registers. In **Table 2**, changes and usage of the Struck FW registers are shown. Registers that are available but not used in normal

operation are omitted from the list. Registers marked with "Not used" are still accessible but will not affect anything, i.e. they are disconnected after the register interface.

Table 2, Struck FW registers. Changes and usage.

Offset	Function	Notes
0x10	Arm/disable ADC.	Arm ADC before each pulse.
0x11	Enable ADC channels.	External trigger is always enabled.
0x12	LVDS Trigger ctrl.	Not used. Use Custom register BPM_BOARD_SETUP.
0x13	Harlink Trigger ctrl.	Used to ctrl Interlock input on Harlink1-4
0x40-0x43	Clock distribution/ctrl.	Route external clock to FPGA and ADCs.
0x45	DAC ctrl	Power up and set 2's compliment.
0x47	RTM I2C interface	Set attenuation of RTM input channels.
0x49	ADC input tap delay	Adjust data strobe timing.
0x100-0x119	Internal trigger logic.	Not used.
0x120-0x129	Sample memory address.	See DDR3 memory map.
0x12A	Sample length.	Set to ceil(ADC_clock*pulse_length)/16).
0x12B	Ring buffer.	Not used and removed.
0x12C-0x12E	Histogram.	Not used and removed.
0x12F	RTM LVDS IO-control	Used to control RTM board parameters.
0x200-0x214	DMA read and write.	Use provided SW.
0x220-0x223	Interrupt Ctrl.	Enable User_IRQ.
0x400-0x4FF	Custom registers.	See **Table 8**, custom register map.

6.3 Custom Firmware implementation

The custom implementation of the LLRF controller is implemented in the FPGA on the Struck Digitizer board. It's implemented as an add-on to the provided FW from Struck. An overview of the Custom Logic (CL) is shown in **Figure 6**.

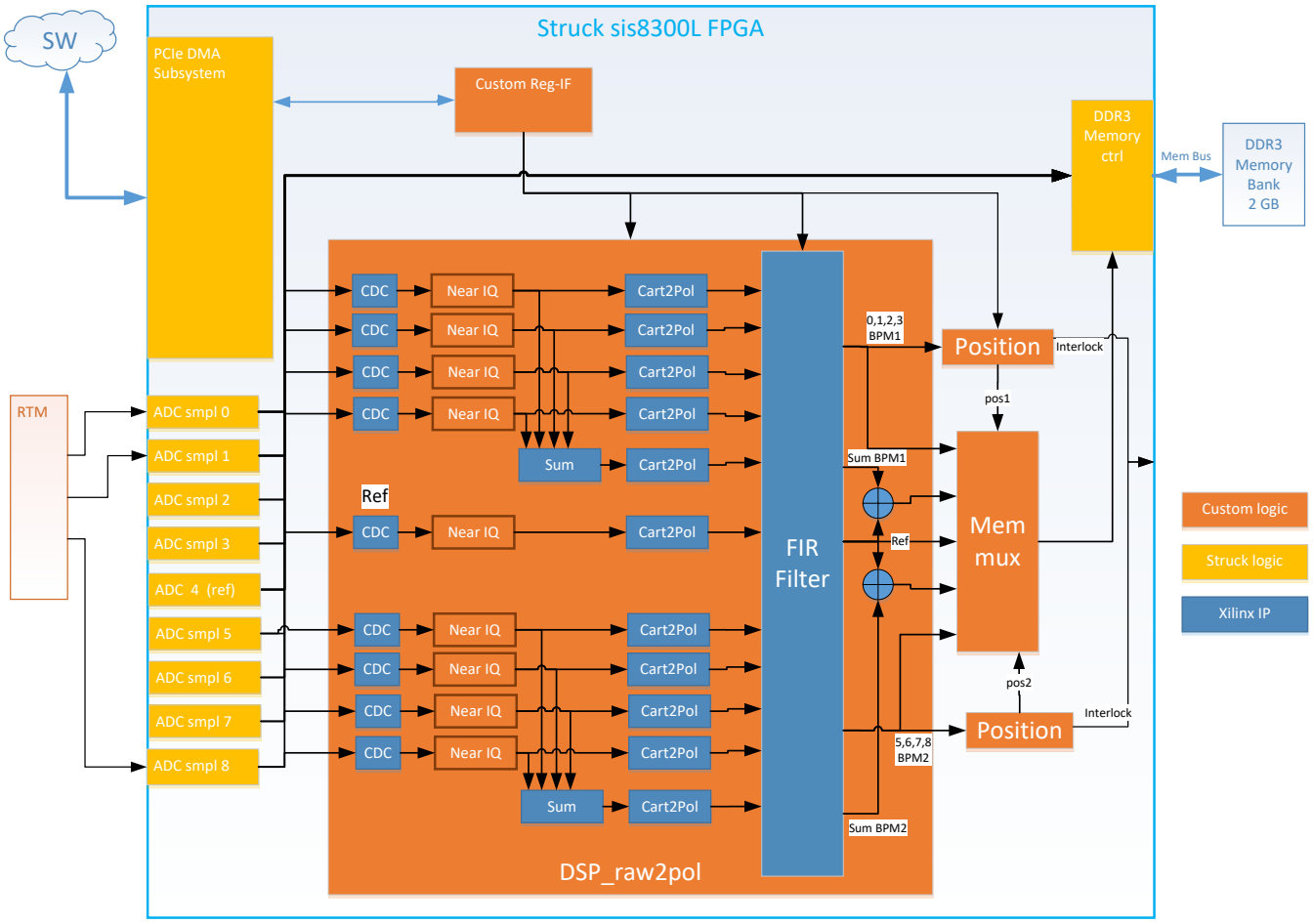

Figure 6, BPM custom logic overview.

The CL consists of nine major parts:

1. **IQ sampling:** Takes the sampled sinus signal from the cavity and reference line and converts them to IQ samples. The reference IQ samples are converted to magnitude and angle (MA) samples. The cavity line IQ samples are then rotated with the angle value of the reference. The output is the cavity IQ samples compensated with the reference angle, i.e. the reference angle is considered zero degrees.
2. **Filter:** All magnitude and angle values are FIR filtered.
3. **DSP_raw2pol:** Takes raw input samples and convert them to IQ-points using the near-IQ algorithm. After filtering, the IQ-points are converted to polar-points, i.e. amplitude and phase. In addition all four BPM antenna inputs are summed in the IQ-plane, converted to polar coordinates and phase compensated by the phase of the reference signal.

4. **Register interface:** Is connected to the PCIE bus and provides access to all parameters in the design. Parameter buffering and clock-domain-crossing (CDC) is also handled here.
5. **FSM controller:** Handles timing and control of the different parts based on the pulse trigger inputs and the register interface parameters.
6. **Position monitor:** Compares current position value to register controlled thresholds and raise an alarm if the values exceed the thresholds. One position monitor for each BPM, i.e. two.

6.3.1 Custom Logic physical connections

The custom logic uses a number of the physical inputs and outputs from the board. In **Table 3** the physical connects to the custom logic are listed.

Table 3, Physical connections.

Name	Physical connection	Comment
Pulse_start trigger	m-lvds1 or m-lvds5	Setup in register BPM_BOARD_SETUP.
Pulse_end trigger	m-lvds2 or m-lvds6	Setup in register BPM_BOARD_SETUP.
NOT USED	ADC_1	NOT USED
Antenna_A1	ADC_2	BPM1
Antenna_B1	ADC_3	BPM1
Antenna_C1	ADC_4	BPM1
Antenna_D1	ADC_5	BPM1
Antenna_A2	ADC_6	BPM2
Antenna_B2	ADC_7	BPM2
Antenna_C2	ADC_8	BPM2
Antenna_D2	ADC_9	BPM2
Phase_reference	ADC_10	Common for BPM1 and BPM2
PULSE_DONE, POS_IRQ_1 or POS_IRQ_2	PCIE USER_IRQ line	Interrupt generated when CL is done with a pulse or when a position alarm goes off.
POSITION_ALARM	Harlink output 1	Will be active if the position alarm goes off.

6.3.2 Bit width and number representation

All signals in the custom logic implementation is digital binary signals. To represent other values than binary, two or more binary signals are combined into one value. The number of binary signals used to represent a value is referred to as bits.
Value representation is by default unsigned integer numbers. Values that contain signed or fractional values will be marked with a **Signed(INT_bits, Frac_bits)** or **Unsigned(INT_bits, Frac_bits)**. All signed numbers are using two's complement number representation.
Examples:
0x7FFF0000 Signed(16,16) => 32767.0 decimal
0x80000000 Signed(16,16) => -32768.0 decimal
0xFFFF0000 Signed(16,16) => -1.0 decimal
0xFFFF8000 Signed(16,16) => -0.5 decimal

0xFFFF0000 Unsigned(32,0) => 4294901760 decimal
0x0000C000 Signed(16,16) => 0.75 decimal

The following Matlab function can be used to convert decimal values into signed.

```
function [v,v_int,error] = conv2hex(values,bits_int,bits_frac)
v_int = floor(values.*2^bits_frac);
v = v_int/2^bits_frac;
neg = 2^(bits_int+bits_frac);
v_fix = v_int;
v_fix(find(v_int<0)) = v_int(find(v_int<0))+neg;
[r,c] = size(v);
for i = 1:(r*c)
    disp(sprintf('v_%d:\t 0x%08X',i,v_fix(i)))
end
error = values - v;
```

6.3.3 Input, output and internal resolution and number representation

The ADCs and DACs on the Struck boards have 16-bits resolution, hence input and output resolution is restricted to a maximum of 16 bits. Please note that even though the DAC has 16-bits input the ADC performance corresponds to 12.6 true bits, i.e. the SNR of the ADC corresponds to a perfect ADC with 12.6 bits.

Internal resolution is 32-bits, in order to not lose any of the input precision during the algorithmic calculations. Number representation internally is different for the I, Q, magnitude and angle part of the implementation, since angle is restricted to +-pi while magnitude is an unsigned number from 0 to 1 and IQ is signed numbers between -1 and +1. However, to include over and underflow detection IQ and magnitude values are represented internally as signed numbers between -2 and +2.

Set-point and Feed Forward tables that are loaded from memory and the PI-error that is written to memory are restricted to 16-bits due to a limited amount of FPGA block RAM. FF and PI-error are delta values that could theoretically have values between -2 and +2, but to increase the resolution they have been limited to the range of -1 to +1.

Table 4 summarizes the bit resolution and number representation in the custom logic implementation. In the range field all values with a .999… ending, represent that all binary fraction bits are set to '1', e.g. maximum Input signal is 0x7FFF = b'0.111111111111111 ~ 0.999.

Table 4, Bit resolution and number representation.

Source	Resolution	Nbr representation	Range	Limiting factor
Input	16-bits	Signed(1,15)	0.999 to -1	ADC HW
Output	16-bits	Signed(1,15)	0.999 to -1	DAC HW
Internal: Magnitude path Angle path IQ path	 32-bits 32-bits 32-bits	 Signed(2,30) Signed(3,29) Signed(2,30)	 1.9999 to -2 3.9999 to -4 1.9999 to -2	Timing
DDR3 Memory	256-bits	-	-	DDR3

6.3.4 Memory map

In **Table 5** the memory map of the DDR3 memory is shown. It's a 32-bit byte-address with base address 0x00000000. Parameters set by Struck FW registers are shown as **SFW** 0xXX, where 0xXX is the register address. The memory can be accessed by three masters: PCIe-bus, AD-Converters and custom logic, depending on the memory area they have different read write access.

Note that the DDR3 memory is not directly accessible from SW. The DMA service provided in the Struck FW has to be used. A DMA is setup through the Struck FW registers, see **Table 2** in Chapter 6.2.1.

Table 5, DDR3 memory map

Base = 0x0	**Size**	**Allowed access**			**Storage**
Offset		**PCI**	**ADC**	**CL**	
SFW 0x120	**SFW** 0x12A	R/W	W	-	ADC channel 1 (ant A1 or A2)
SFW 0x121	**SFW** 0x12A	R/W	W	-	ADC channel 2 (ant B1 or B2)
SFW 0x122	**SFW** 0x12A	R/W	W	-	ADC channel 3 (ant C1 or C2)
SFW 0x123	**SFW** 0x12A	R/W	W	-	ADC channel 4 (ant D1 or D2)
SFW 0x124	**SFW** 0x12A	R/W	W	-	ADC channel 5 (Ref)
SFW 0x125	**SFW** 0x12A	R/W	W	W	ADC channel 6 (inter or A2)
SFW 0x126	**SFW** 0x12A	R/W	W	W	ADC channel 7 (inter or B2)
SFW 0x127	**SFW** 0x12A	R/W	W	W	ADC channel 8 (inter or C2)
SFW 0x128	**SFW** 0x12A	R/W	W	W	ADC channel 9 (inter or D2)
SFW 0x129	**SFW** 0x12A	R/W	-	W	ADC channel 10 (inter)

6.3.4.1 Memory content and organization

Memory area ADC channel 1-10 contains either ADC input in consecutive order, 16-bits per sample, or internal data signals according to the BPM_BOARD_SETUP register. Internal signals are all down sampled by the Near-IQ parameter N, i.e. for every N input ADC samples one internal value is generated. This results in empty spaces in memory, which is filled with 0xDEAD for SW identification. **Table 6** shows the different memory area contents and examples of the content. The actual content in memory depends the bpm_mux and mem_mux settings in register BPM_BOARD_SETUP.

Table 6, memory content.

Memory content	**Bits/sample**	**Example**
ADC ch 1-9	16	0x12345678 => Sample(i+1) = 0x1234, Sample(i) = 0x5678
Internal signal POS	16	…, 0xDEAD, POS_X_1, POS_Y_1, POS_X_2, POS_Y_2, 0xDEAD, … Valid X and Y values every N 16-bit sample. Not used 16-bit samples have the value 0xDEAD, i.e. X1Y1X2Y2 is followed by N-4 16-bit samples with value 0xDEAD.
Internal signal SUM	16	…, 0xDEAD, SUM_M_1,SUM_A_1, SUM_M_2, SUM_A_2, 0xDEAD, … Valid magnitude and angle values are found every N 16-bit sample. Not used 16-bit samples have the value 0xDEAD, i.e. sum_m1,sum_a1,sum_m2,sum_a2 is followed by N-4 16-bit samples with value 0xDEAD.

Internal signal antenna magnitude	16	…, 0xDEAD, ANT_A1_M, ANT_A2_M, ANT_B1_M, ANT_B2_M, ANT_C1_M, ANT_C2_M, ANT_D1_M, ANT_D2_M, 0xDEAD, … Valid magnitude values are found every N 16-bit sample. Not used 16-bit samples have the value 0xDEAD, i.e. a1,a2,b1,b2,c1,c2,d1,d2,ref is followed by N-9 16-bit samples with value 0xDEAD.
Internal signal antenna angle	16	…, 0xDEAD, ANT_A1_A, ANT_A2_A, ANT_B1_A, ANT_B2_A, ANT_C1_A, ANT_C2_A, ANT_D1_A, ANT_D2_A, 0xDEAD, … Valid angle values are found every N 16-bit sample. Not used 16-bit samples have the value 0xDEAD, i.e. a1,a2,b1,b2,c1,c2,d1,d2,ref is followed by N-9 16-bit samples with value 0xDEAD.

6.3.5 Register map

Registers can be accessed in four different ways, as shown in **Table 7**.

Table 7, Register access methods.

Access method	Abbreviation	Function
Read	R	Read 32-bits, where bits exceeding the size of the register are filled with zeroes.
Write	W	Write 32-bits, where bits exceeding the size of the register are ignored.
Set	S	Set 32-bits, where bits exceeding the size of the register are ignored. Writing to the set interface of a register will change the value of the register to one for all bits that are one in the write and leave the rest of the bits as they were, i.e. New_Reg_val = Old_Reg_val or Write_val.
Clear	C	Clear 32-bits, where bits exceeding the size of the register are ignored. Writing to the clear interface of a register will change the value of the register to zero for all bits that are one in the write and leave the rest of the bits as they were, i.e. New_Reg_val = Old_Reg_val and not(Write_val).

In **Table 8** custom registers are defined. It's a 32-bit address with base address 0x400.

Table 8, custom register map.

Offset (base = 0x400)	**Register**	**Access**	**Shadow register**	**Function**
0x00	BPM_ID	R	No	Module ID and FW version
0x01	BPM_INST_ID	R/W	No	SW instance ID

0x02	BPM_GOP	R/W[1]	No	General outputs
0x03	BPM_GIP	R/W	No	General inputs
0x04	BPM_SAMPLE_CNT	R	No	Nbr of input samples(Read only)
0x05	BPM_IQ_SAMPLE_CNT	R	No	Nbr of IQ samples (Read only)
0x06	BPM_BOARD_SETUP	R/W	No	DAC and Trigger setup
0x07	BPM_NEAR_IQ_1_PARAM	R/W	Yes	Near IQ sampling parameters
0x08	BPM_NEAR_IQ_2_PARAM	R/W	Yes	Near IQ sampling parameters
0x09	BPM_NEAR_IQ_DATA	R/W	No	Near IQ constants data access
0x0A	BPM_NEAR_IQ_ADDR	R/W	No	Near IQ constants data address
0x0B	BPM_REF_MA	R	No	Mag and Angle of reference signal
0x0C	BPM_SUM_1_MA	R	No	BPM1: Mag and Angle of Antenna Sum
0x0D	BPM_SUM_2_MA	R	No	BPM2: Mag and Angle of Antenna Sum
0x0E	BPM_POS_1_XY	R	No	BPM1: X and Y position
0x0F	BPM_POS_2_XY	R	No	BPM2: X and Y position
0x10	BPM_POS_PARAM_X_1	R/W	Yes	BPM1: Position X thresholds
0x11	BPM_POS_PARAM_Y_1	R/W	Yes	BPM1: Position Y thresholds
0x12	BPM_POS_MAG_CTRL_1	R/W	Yes	BPM1: Position Mag threshold and ctrl
0x13	BPM_POS_PARAM_X_2	R/W	Yes	BPM2: Position X thresholds
0x14	BPM_POS_PARAM_Y_2	R/W	Yes	BPM2: Position Y thresholds
0x15	BPM_POS_MAG_CTRL_2	R/W	Yes	BPM2: Position Mag threshold and ctrl
0x16	BPM_DSP_PARAM	R/W	Yes	DSP parameters (Placeholder)
0x17	BPM_FILTER	R/W	No	Filter parameter
0x18	BPM_FILTER_CTRL	R/W	No	Filter Ctrl
0x19	BPM_SELF_TRIG_PARAM	R/W	No	Self-triggering ctrl and threshold
0x1A	BPM_SELF_TRIG_CNT	R/W	No	Self-triggering sample cnt

Table 9, BPM_ID

Bits	Value	Function
31 - 16	0xCA5E	HW id of BPM ctrl digitizer
15 – 8	0x00	Custom FW Major revision number
7 – 0	0x0C	Custom FW Minor revision number

Back to custom register map.

Table 10, BPM_INST_ID

Bits	Default Value	Function
31 - 0	0x00000000	SW id of this instance of BPM ctrl digitizer

Back to custom register map.

Register BPM_GOP is a status register, i.e. only meant to be read, but it has the special function that writing any value to the register will clear bit 3 to 10, that is position out of bounds, read/write error, divide by zero and DAQ done signal from Struck FW. These bits are latched, meaning that they will remain high until cleared.

Table 11, BPM_GOP

Bits	Default Value	Function
31-16	0x0000	Pulse_done counter. Clear with GIP(7)

[1] Write only used to clear latched values in GOP.

15-12	0x00	Not used.
11	0	DAQ done signalled from Struck FW. Clear by writing to GOP.
10	0	BPM1: X-position divider, divisor was zero. Clear by writing to GOP.
9	0	BPM1: Y-position divider, divisor was zero. Clear by writing to GOP.
8	0	BPM2: X-position divider, divisor was zero. Clear by writing to GOP.
7	0	BPM2: Y-position divider, divisor was zero. Clear by writing to GOP.
6	0	Read error while accessing register. Clear by writing to GOP.
5	0	Write error while accessing register. Clear by writing to GOP.
4	0	Position 1 out of bounds status. Case 0: No position alarm. Case 1: Position alarm. Clear by writing to GOP. Position IRQ is sent if enabled (register BPM_BOARD_SETUP). Only the first position in a pulse that is out of bounds will generate an interrupt and set this status bit.
3	0	Position 2 out of bounds status. Case 0: No position alarm. Case 1: Position alarm. Clear by writing to GOP. Position IRQ is sent if enabled (register BPM_BOARD_SETUP). Only the first position in a pulse that is out of bounds will generate an interrupt and set this status bit.
2 -0	0x0	State of main ctrl FSM (Only for debug use)

Back to custom register map.

Table 12, BPM_GIP

Bits	Default Value	Function
7	0	Clear Pulse_done counter in GOP. Will always read as 0.
6	0	SW reset. Set or write this bit to SW reset the custom logic. Will always read as 0.
5	0	Force get_param. Debug. Will force the custom logic to immediately use new parameters and not wait until a new pulse is coming. (DEBUG) Will always read as 0.
4	0	Not used.
3	0	Force pulse_end trigger. (DEBUG) Will always read as 0.
2	0	Force pulse_start trigger. (DEBUG) Will always read as 0.
1	0	Update parameters. Set or write this bit to get the ctrl to use new parameters. Will always read as 0.
0	0	Init done. Set or write this bit when all initialization is done. Will always read as 0.

Back to custom register map.

Table 13, BPM_SAMPLE_CNT

Bits	Default Value	Function
31-0	0x00000000	Number of input samples between PULSE_START trigger and PULSE_END trigger. **Read Only, Unsigned(32,0)**

Back to custom register map.

Table 14, BPM_IQ_SAMPLE_CNT

Bits	Default Value	Function
31-0	0x00000000	Number of IQ-samples between PULSE_START trigger and PULSE_END trigger. **Read Only, Unsigned(32,0)**

Back to custom register map.

Table 15, BPM_BOARD_SETUP

Bits	Default Value	Function
22	0	Invert DAC output. Case 0: normal DAC output. Case 1: inverted DAC output.
21	0	Position 1 IRQ mask. Case 0: Position 1 IRQ disabled. Case 1: Position 1 IRQ enabled.
20	0	Position 2 IRQ mask. Case 0: Position 2 IRQ disabled. Case 1: Position 2 IRQ enabled.
19	0	Position 1 Interlock mask. Case 0: Position 1 Interlock disabled. Case 1: Position 1 Interlock enabled.
18	0	Position 2 Interlock mask. Case 0: Position 2 Interlock disabled. Case 1: Position 2 Interlock enabled.
17-14	0x0	Controls front panel DAC output source Case 0x0 or 0xC-0xF: zero output. Case 0x1: antenna A1 IQ Case 0x2: antenna B1 IQ Case 0x3: antenna C1 IQ Case 0x4: antenna D1 IQ Case 0x5: antenna sum1 IQ Case 0x6: Reference IQ Case 0x7: antenna A2 IQ Case 0x8: antenna B2 IQ Case 0x9: antenna C2 IQ Case 0xA: antenna D2 IQ Case 0xB: antenna sum2 IQ
13-10	0x0	Not used.
9-8	0x0	BPM_MEM_MUX: **Case 0x0 or 0x3:** BPM1 ADC signals are found on mem channel 1-5, internal signals on mem channel 6-10. Ch-1: Ant A1. Ch-2: Ant B1. Ch-3: Ant C1. Ch-4: Ant D1. Ch-5: Reference. Ch-6: SUM.

		Ch-7: Antenna magnitude. Ch-8: Antenna angle. Ch-9: Position. Ch-10: Internal see **CH10_MEM_MUX** below. **Case 0x1:** BPM2 ADC signals are found on mem channel 1-5, internal signals on mem channel 6-10. Ch-1: Ant A2. Ch-2: Ant B2. Ch-3: Ant C2. Ch-4: Ant D2. Ch-5: Reference. Ch-6: SUM. Ch-7: Antenna magnitude. Ch-8: Antenna angle. Ch-9: Position. Ch-10: Internal see **CH10_MEM_MUX** below. **Case 0x2:** BPM1 and BPM2 ADC signals are found on mem channel 1-9, internal signals on mem channel 10. Ch-1: Ant A1. Ch-2: Ant B1. Ch-3: Ant C1. Ch-4: Ant D1. Ch-5: Reference. Ch-6: Ant A2. Ch-7: Ant B2. Ch-8: Ant C2. Ch-9: Ant D2. Ch-10: Internal see **CH10_MEM_MUX** below.
7-6	0x0	**CH10_MEM_MUX:** **Case 0x0:** POS_X_1, POS_Y_1,POS_X_2,POS_Y2,0xDEAD… **Case 0x1:** SUM_1_m, SUM_1_a, SUM_2_m, SUM_2_a, 0xDEAD… **Case 0x2:** Ant_A1_m, Ant_A2_m, Ant_B1_m, Ant_B2_m, Ant_C1_m, Ant_C2_m, Ant_D1_m, Ant_D2_m, REF_m, 0xDEAD… **Case 0x3:** Ant_A1_a, Ant_A2_a, Ant_B1_a, Ant_B2_a, Ant_C1_a, Ant_C2_a, Ant_D1_a, Ant_D2_a, REF_a, 0xDEAD…
5-2	0x0	Force harlink out. (Will be force interlock out)
1-0	0x0	Trigger setup: Case 0,2 or 3 : Pulse_start_trigger on mlvds(1) Pulse_end_trigger on mlvds(2) Case 1 : Pulse_start_trigger on mlvds(5) Pulse_end_trigger on mlvds(6)

Back to custom register map.

Table 16, BPM_NEAR_IQ_1_PARAM

Bits	Default Value	Function
23-16	0x0	Near IQ parameter N.
7-0	0x0	Near IQ parameter M. Not used by the FPGA, only kept as user info.

Back to custom register map.

Table 17, BPM_NEAR_IQ_2_PARAM

Bits	Default Value	Function
31-0	0x0	Near IQ parameter 2/N. **Signed(2,30)**

Back to custom register map.

Table 18, BPM_NEAR_IQ_DATA

Bits	Default Value	Function
31-0	0x0	Data access to near-iq sampling constants memory. Will write to or read from address specified in BPM_NEAR_IQ_ADDR. Up on write to this register, register BPM_NEAR_IQ_ADDR will auto increase with 1. **Signed(2,30)**

Back to custom register map.

Table 19, BPM_NEAR_IQ_ADDR

Bits	Default Value	Function
7-0	0x0	Address register for data access to near-iq sampling constants memory. Address will auto increase with 1 if register BPM_NEAR_IQ_DATA is written to. **Unsigned(8,0)**

Back to custom register map.

Table 20, BPM_REF_MA

Bits	Default Value	Function
31-16	0x0000	Magnitude of Reference signal (0-1.99). **Unsigned(1,15)**
15-0	0x0000	Angle of Reference signal (+-pi). **Signed(3,13)**

Back to custom register map.

Table 21, BPM_SUM_1_MA

Bits	Default Value	Function
31-16	0x0000	BPM1: Magnitude of antenna sum signal (0-1.99). **Unsigned(1,15)**
15-0	0x0000	BPM1: Angle of antenna sum signal (+-pi). **Signed(3,13)**

Back to custom register map.

Table 22, BPM_SUM_2_MA

Bits	Default Value	Function
31-16	0x0000	BPM2: Magnitude of antenna sum signal (0-1.99). **Unsigned(1,15)**
15-0	0x0000	BPM2: Angle of antenna sum signal (+-pi). **Signed(3,13)**

Back to custom register map.

Table 23, BPM_POS_1_XY

Bits	Default Value	Function
31-16	0x0000	BPM1: X position (-1 - 0.99). **Signed(1,15)**
15-0	0x0000	BPM1: Y position (-1 - 0.99). **Signed(1,15)**

Back to custom register map.

Table 24, BPM_POS_2_XY

Bits	Default Value	Function
31-16	0x0000	BPM2: X position (-1 - 0.99). **Signed(1,15)**
15-0	0x0000	BPM2: Y position (-1 - 0.99). **Signed(1,15)**

Back to custom register map.

Table 25, BPM_POS_PARAM_X_1

Bits	Default Value	Function
31-16	0x0	BPM1: Position-X high threshold. **Signed(1,15)** An X-position larger than this will raise interlock and an IRQ.
15-0	0x0	BPM1: Position-X low threshold. **Signed(1,15)** An X-position lower than this will raise interlock and an IRQ.

Back to custom register map.

Table 26, BPM_POS_PARAM_Y_1

Bits	Default Value	Function
31-16	0x0	BPM1: Position-Y high threshold. **Signed(1,15)** A Y-position larger than this will raise interlock and an IRQ.
15-0	0x0	BPM1: Position-Y low threshold. **Signed(1,15)** A Y-position lower than this will raise interlock and an IRQ.

Back to custom register map.

Table 27, BPM_POS_MAG_CTRL_1

Bits	Default Value	Function
16	0x0	BPM1: Use magnitude or XY threshold. Case 0: Use XY threshold. Case 1: Use Magnitude threshold.
15-0	0x0	BPM1: Position magnitude threshold. **Unsigned(1,15)** A position magnitude larger than this will raise interlock and an IRQ.

Back to custom register map.

Table 28, BPM_POS_PARAM_X_2

Bits	Default Value	Function
31-16	0x0	BPM2: Position-X high threshold. **Signed(1,15)** An X-position larger than this will raise interlock and an IRQ.
15-0	0x0	BPM2: Position-X low threshold. **Signed(1,15)** An X-position lower than this will raise interlock and an IRQ.

Back to custom register map.

Table 29, BPM_POS_PARAM_Y_2

Bits	Default Value	Function
31-16	0x0	BPM2: Position-Y high threshold. **Signed(1,15)** A Y-position larger than this will raise interlock and an IRQ.
15-0	0x0	BPM2: Position-Y low threshold. **Signed(1,15)** A Y-position lower than this will raise interlock and an IRQ.

Back to custom register map.

Table 30, BPM_POS_MAG_CTRL_2

Bits	Default Value	Function
16	0x0	BPM2: Use magnitude or XY threshold. Case 0: Use XY threshold. Case 1: Use Magnitude threshold.
15-0	0x0	BPM2: Position magnitude threshold. **Unsigned(1,15)** A position magnitude larger than this will raise interlock and an IRQ.

Back to custom register map.

Table 31, BPM_DSP_PARAM

Bits	Default Value	Function
31-0	0x0	Not used.

Back to custom register map.

Table 32, BPM_FILTER

Bits	Default Value	Function
15-0	0x0	New coefficient data input. **Signed(16,0)**

Back to custom register map.

Table 33, BPM_FILTER_CTRL

Bits	Default Value	Function
31-2	0x0	Not used.
1	0	Load new coefficients to the FIR. Will always read 0.
0	0	Case 1: FIR filter enabled. Case 0: FIR filter bypassed.

Back to custom register map.

Table 34, BPM_SELF_TRIG_PARAM

Bits	Default Value	Function
31-16	0x0000	Threshold. ADC values higher than this value will trigger a start-trigger, acting in the same way as an external start-trigger. Only positive valued adc samples are checked against the threshold. **Unsigned(0,16)**
15-6	0x000	ADCs that are checked against the threshold: (A '1' => Checked against threshold, a '0' => not checked) Bit 15 : Not used. Bit 14 : bpm_ant_D2. Bit 13 : bpm_ant_C2. Bit 12 : bpm_ant_B2. Bit 11 : bpm_ant_A2. Bit 10 : bpm_reference_line. Bit 9 : bpm_ant_D1. Bit 8 : bpm_ant_C1. Bit 7 : bpm_ant_B1. Bit 6 : bpm_ant_A1.
5-1	0	Not used.
0	0	Case 1: Self-triggering enabled. Case 0: Self-triggering disabled.

Back to custom register map.

Table 35, BPM_SELF_TRIG_CNT

Bits	Default Value	Function
31-0	0x0000	Sample count. When a self-triggered start-trigger is generated a stop-trigger is generated this many raw samples later. Recommendation is to use a smaller amount than the Struck sample length (SFW 0x12A). **Unsigned(16,0)**

Back to custom register map.

6.3.6 Interlock

The BPM support **External output interlock**, which is triggered when beam position is outside the allowed area. **Figure 7** shows the interlock FW connections.

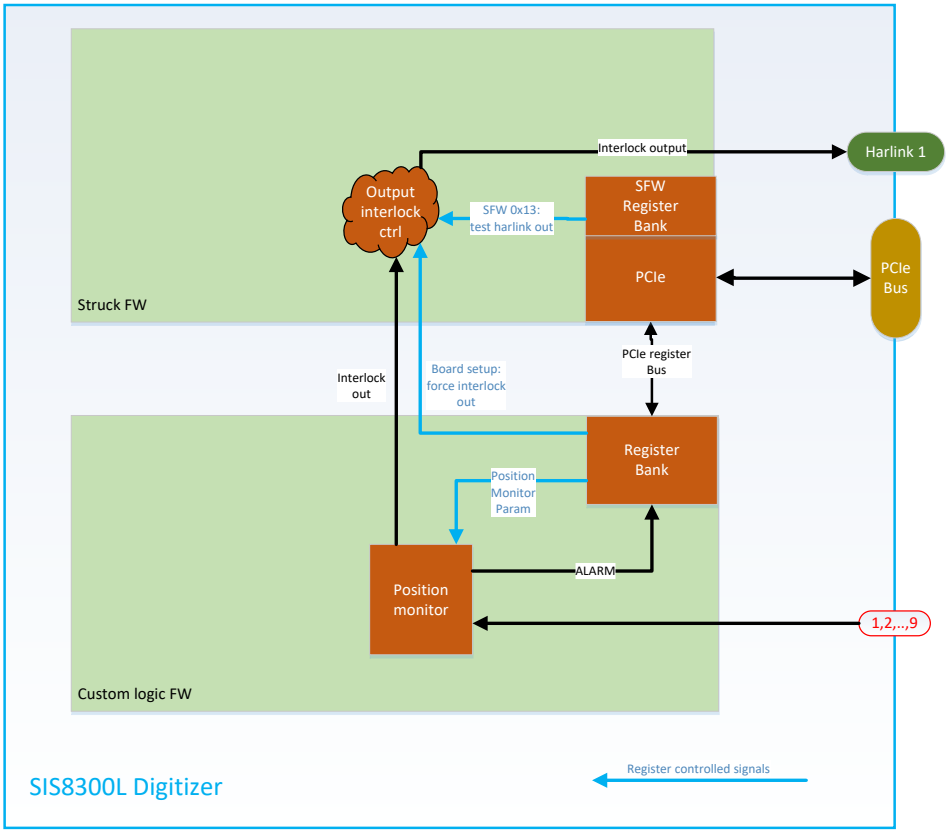

Figure 7, Output interlock connection.

Table 36, BPM Interlock connectors.

Interlock	Connector	Note
Output external interlock	Front panel Harlink out 1	Will be connected to LPS.

6.3.7 Position monitor

In **Figure 8**, it's shown how the beam position is calculated from the magnitudes of antenna A, B, C and D. The position is compared to either a square or a circular threshold, if exciding the threshold an interlock and IRQ is raised.

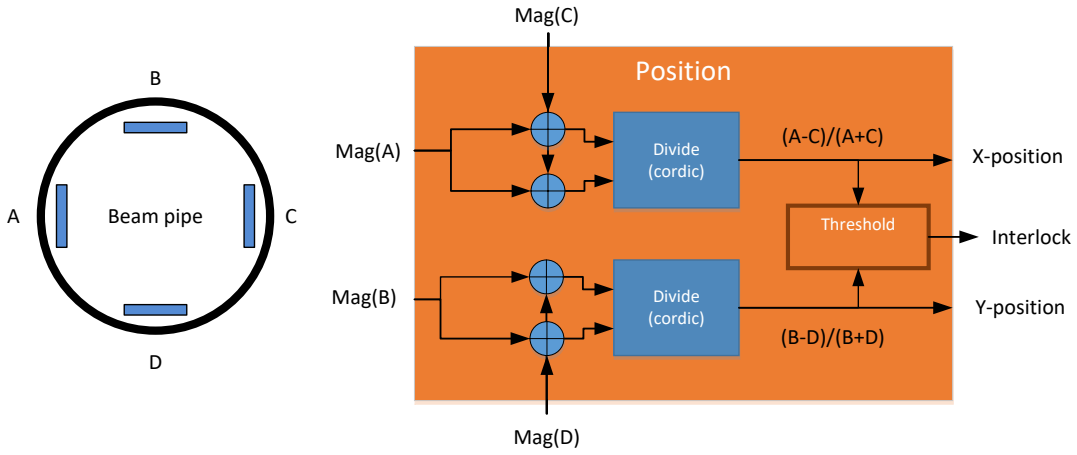

Figure 8, Beam position calculation unit.

6.3.8 FIR filter

A Xilinx core-generator FIR filter is used. It has four parallel data paths and is time-multiplexed by six, i.e. each data path takes six different input data streams and generates six output data streams. This is required since there are eleven parallel magnitude-angle data streams, a total of twenty two streams that needs to be filtered. As a consequence of the time multiplexed input, the FIR filter needs twelve clock-cycles to accept all input data. The number of available clock-cycles is depending on the Near-IQ parameter N, since it determines how many input samples is required to create an IQ-sample. With the used FIR implementation N must be twelve or larger.

The FIR filter is order ten and requires symmetric filter coefficients and that the filter gain is one. Details on how the coefficients are updated is found in Section 6.7.8.

6.3.9 SW Interrupt

The custom logic has three SW interrupts that are or:ed together on the user interrupt line provided by the Struck FW. Enabling and clearing of the user interrupt is handled by the Struck FW registers, see **Table 2**.

Table 37, BPM SW interrupt sources

Interrupt	Default Value	Function
PULSE_DONE	0	Goes high when the controller is done with the current pulse and goes to IDLE state. At this point SW is free to read and write to register and memory.
POSITION_1	0	Goes high when XY position of BPM1 is out of bounds. Status is shown in register BPM_GOP(4).
POSITION_2	0	Goes high when XY position of BPM2 is out of bounds. Status is shown in register BPM_GOP(3).

6.4 Verification

Performance measurement and verification is documented in TBD, and a regression test plan is found in TBD.

6.5 FW usage

6.5.1 ADC/FPGA Clock Setup

The FPGA and the ADCs on the sis8300L card use the same clock source that can be provided through the front panel, the backplane or be generated on the card. **Table 2** shows which registers are used to control clock routing.

The clock frequency that can be used depends on both physical limits, e.g. FPGA and ADC speed, and on algorithm settings, e.g. Near-IQ setup. Furthermore, there are RF limits on the input signal to the ADC, for more details see [7] found in [8].

The clock frequency used needs to be selected so that the following criteria are met:

1. Clock frequency must be below the ADCs max operational speed 125 MHz.
2. Clock frequency must be below the FPGA max operational speed 100 MHz.
3. The RF frequency of the input to the ADCs, ADC_RF_INPUT_FREQUENCY, should be between 10-20% of the RF frequency of the system (352.21 or 704.42 MHz). Near-IQ parameters N and M must be chosen so that the clock frequency is equal to ADC_RF_INPUT_FREQUENCY * (N/M).
4. Near-IQ parameter N must be larger than 12 to allow the FIR filter to handle all signals.

As an example, this is the setup used at the ESS testbench:

- ADC_RF_INPUT_FREQUENCY = 23.6 MHz
- Near-IQ: M=4, N=15 => clock frequency = 23.6*15/4 = 88.5 MHz.
- IQ-sample speed = 88.5/15 = 5.9 MHz.

where all criteria are met.

6.6 FW capabilities

Parameter	Value	Note
Max operating frequency (f_{op})	MHz	See Section 8.
Latency	97 clock cycles	Time until an input sample affects the position interlock signal.

6.7 SW usage

6.7.1 Timing dependencies

The BPM controller is designed so that SW should in general not need to take timing into account when accessing registers or memory. However, some dependencies are unavoidable. In **Figure 9**, the main FSM of the BPM controller is shown, in which two timing dependencies can be found. First, parameters affecting the next pulse must be written before the "puls_start_trigger" and secondly SW access to the DDR3 memory will be affected by which state the FSM is in. SW is not allowed to access memory when the FSM is in ACTIVE_PULSE or WAIT_ADC, from a SW perspective that is from SW arms the digitizer until it receives the PULSE_DONE interrupt it shall not access DDR memory.

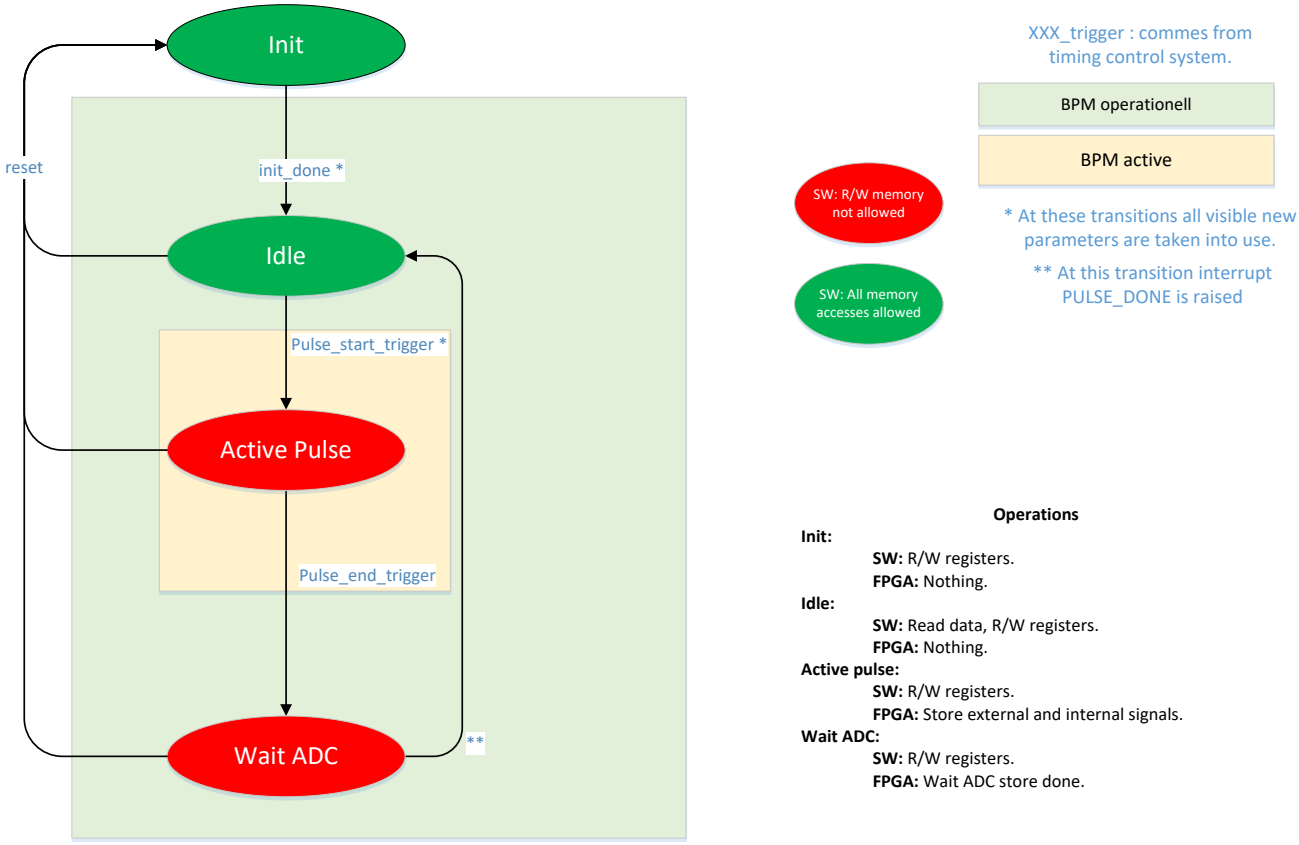

Figure 9, Main FSM of the custom logic.

6.7.2 Output interlock

An external output interlock can be generated by the position monitor or forced through register settings. The following register control the behaviour:

Register	Field/Bits	Notes
SFW 0x13	20	Test harlink out enabled
SFW 0x13	16	Harlink 1 output
BPM_BOARD_SETUP	Force harlink 1	Harlink 1 output

BPM_POS_PARAM_X_1	All	Position monitor setup BPM1
BPM_POS_PARAM_Y_1	All	Position monitor setup BPM1
BPM_POS_MAG_CTRL_1	All	Position monitor setup BPM1
BPM_POS_PARAM_X_2	All	Position monitor setup BPM1
BPM_POS_PARAM_Y_2	All	Position monitor setup BPM1
BPM_POS_MAG_CTRL_2	All	Position monitor setup BPM1

6.7.3 SW interrupt

There are two interrupt sources, see Section 6.3.7. These are or:ed together on one user interrupt line, hence SW needs to determine which source coursed the interrupt. This is done by reading register BPM_GOP, if POSITION is high this is the active interrupt source else it is a PULSE_DONE interrupt. In **Table 38**, the expected SW behaviour is described for the different interrupt sources.

Table 38, Expected SW interrupt behaviour.

Active interrupt source	**Expected SW behaviour**
PULSE_DONE	Clear user interrupt. Read all signals from memory for the done pulse. If required, setup new parameters for the next pulse and arm the struck board for the next pulse, see **Table 2**.
POSITION	Clear user interrupt. Write register BPM_GOP, to clear status. Only the first position in a pulse that is out of bounds will generate an interrupt.

6.7.4 Interlock

An external output interlock can be generated by the position monitor or forced through register settings. The following register control the behavior:

Register	**Field/Bits**	**Notes**
SFW 0x13	20	Test harlink out enabled
SFW 0x13	16	Harlink 1 output forced to ‘1’ if enabled.
BPM_BOARD_SETUP	Force harlink 1	Harlink 1 output forced to ‘1’
BPM_POS_PARAM_X_1	All	Position monitor setup
…	All	…
BPM_POS_MAG_CTRL_2	All	Position monitor setup
BPM_BOARD_SETUP	18-19	Enable position interlock out

6.7.5 Sampling

Samples from ADC1-ADC9 are stored to memory starting when trigger PULSE_START is received. The number of samples to store is set in **SFW 0x12A**. At the same time, starting with PULSE_START trigger and ending when PULSE_END trigger is received, internal signals are stored to DDR. An overview is shown in **Figure 10**.

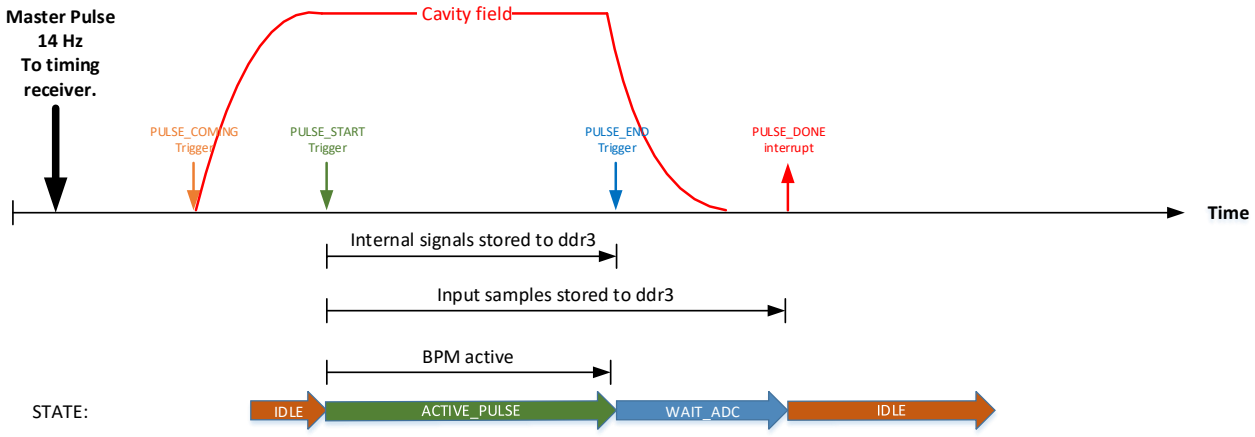

Figure 10, Sampling overview.

Table 39 lists the different status registers and when they are valid. For safe and good SW practice, it's recommended that these registers are only read once when PULSE_DONE interrupt is received.

Table 39, Sample info registers.

Register	Valid from:	Valid to:	Info
BPM_SAMPLE_CNT	PULSE_END	PULSE_START	Nbr of input samples received during PULSE_ACTIVE state.
BPM_IQ_CNT	PULSE_END	PULSE_START	Nbr of IQ-samples received during PULSE_ACTIVE state.

6.7.6 Register interface and control

All registers in **Table 8** marked as shadow registers will not be seen by the BPM controller until they have been triggered, i.e. just writing to a shadow register will not make the BPM controller use the new value. There are currently 2 different triggers all located in the BPM_GIP register. The different triggers are explained in **Table 40**. After a shadow register has been made visible by one of the triggers the BPM controller will first take the new values into use when the FSM, shown in **Figure 9**, is in the correct state. This is done in order to ensure that no new values will be taken into use while there is an active pulse.

Table 40, SW triggers.

Trigger	Function
Init_done	Will make shadow registers visible in the BPM controller. Will move the FSM to IDLE state, where registers will be taken into use.
Update_parameters	Will make shadow registers visible in the BPM controller.

All registers that are not marked as shadow registers will be taken into use as soon as they are written.

6.7.7 Near-IQ sampling constant memory

To calculate IQ-points from the input signals the BPM use near-IQ sampling, also known as non-IQ sampling. For an efficient HW implementation pre-calculated constants are used, as described in

Section 4.1, these are located in the near-IQ sampling memory. It's the responsibility of the SW to fill this memory at start up, using the following registers:

- BPM_NEAR_IQ_ADDR, points to the location in memory where the data will be written. Will auto increase every time data are written to BPM_NEAR_IQ_DATA.
- BPM_NEAR_IQ_DATA, data to write to memory.

The layout of the memory is shown in **Table 41**, where M and N is the near-IQ parameters specified in register BPM_NEAR_IQ_1_PARAM. Maximum N supported by the design is 255.

Table 41, Near-IQ sampling constant memory.

Address	Data signed(2,30)	i
0	Sin(2*pi*M/N*i)	0
1	Cos(2*pi*M/N*i)	0
2	Sin(2*pi*M/N*i)	1
3	Cos(2*pi*M/N*i)	1
…	…	…
2*N-2	Sin(2*pi*M/N*i)	N-1
2*N-1	Cos(2*pi*M/N*i)	N-1
…	Not used	-
511	Not used	-

The following procedure is used to fill the constant memory:

1. Set address to zero: write 0x0 to BPM_NEAR_IQ_ADDR.
2. Write data:
   for(i=0; i<N; i++){
   write Sin(2*pi*M/N*i) to BPM_NEAR_IQ_DATA
   write Cos(2*pi*M/N*i) to BPM_NEAR_IQ_DATA
   }

6.7.8 Filters

How to change filter coefficients:

1. Put the BPM in INIT or IDLE state.
2. Write '1' to the "new coefficient" bit in the BPM_FILTER_CTRL register.
3. for(i=0; i<6; i++){
   write coeff(i) to BPM_FILTER
   }

New coefficients are **NOT** written in straight order, they have to be written in the order shown in **Table 42**. The corresponding filter index and original decimal filter coefficients are shown in the last two columns of **Table 42**. The coefficients shown in **Table 42** are the default coefficients that are loaded when the FW is flashed, i.e. it's not required to load coefficients at start up.

Table 42, filter coefficients of the symmetric order ten FIR filter.

Index i:	Coeff(i)	Filter index	Original filter coefficients.
0	9691	4	0.295731148152960
1	16131	5	0.492273184395108
2	-1739	2	-0.053060037295737
3	185	3	0.005647590076694
4	236	0	0.007209976642777
5	-64	1	-0.001953094774248

6.7.9 Typical operation procedure

This is an example of the normal operational procedure:

1. Boot the digitizer.
2. Enable clks to the digitizer.
3. Write all registers.
    a. Use the register interface to fill the Near-IQ sampling constant memory as described in Section 6.7.7.
4. Write or set “init_done” trigger in BPM_GIP register.
5. Arm the struck board, see **Table 2**.
6. Wait for user interrupt:
    a. See Section 6.7.2, for expected SW behaviour.

Changing and updating BPM parameters examples:

I. If new parameters are required for pulse 10:
    a. Wait until pulse 9 is done, i.e. after PULSE_DONE interrupt for pulse 9.
    b. Write new values to all affected registers.
    c. Write or set “update _parameters” trigger in BPM_GIP register.

7 Common errors and debug guide

Common errors when using the BPM are listed in **Table 43**, together with possible solutions.

Table 43, Common errors and possible solutions.

ERROR	Solution
Not sampling correctly	1. Check input signals and clk signal, correct frequency and amplitude? 2. Check that clk frequency is below maximum allowed FW frequency, see Chapter 8. 3. Change ADC tap-delay value, in Struck FW register 0x49.
Not correct or stable IQ values on reference input.	1. Check that reference input is sampled correctly else see “Not sampling correctly”-error. 2. Check input signals clk and reference, is it correct frequency and amplitude, are they phased locked. 3. Check Near-IQ sampling parameters N and M. Is the IF (LO-RF) frequency*N/M equal to the clk frequency?
Output after filtering is wrong	Check that near-IQ parameter N is larger than 12.

8 FW versions

Table 44, FW versions**.**

FW	Changes	Timing closure	Notes
BPM_0v1	Initial FW	100 MHz	
BPM_0v2		100 MHz	
BPM_0v3		100 MHz	
BPM_0v4		100 MHz	
BPM_0v5		100 MHz	
BPM_0v6	2nd BPM added	100 MHz	

9 Bibliography

[1] Steve Peggs, Edditor, “Technical Design Report,” European Spallation Source, Lund, 2013.

[2] T. Schilcher, “The CERN Accelerator School,” 6 June 2007. [Online]. Available: https://cas.web.cern.ch/cas/Sweden-2007/Lectures/Web-versions/Schilcher-1.pdf.

[3] Struck Innovative Systeme, “SIS8300-L, µTCA FOR PHYSICS Digitizer, User Manual,” 2013.

[4] Struck Innovative Systeme, “http://www.struck.de/,” Struck Innovative Systeme. [Online].

[5] F. Kristensen, “doc/LLRF_performance_evaluation_vx_x,” 2015. [Online]. Available: https://bitbucket.org/europeanspallationsource/llrf_digital/.

[6] F. Kristensen, “doc/Test_specifications_LLRF_vx_x,” [Online]. Available: https://bitbucket.org/europeanspallationsource/llrf_digital/.

[7] A. Svensson, “LLRF_frequencies_and_clock_signals_0v1,” Lund, 2014.

[8] F. Kristensen, “LLRF_digital,” EIT, 2014. [Online]. Available: https://bitbucket.org/europeanspallationsource/llrf_digital/overview.

[9] US Government, “http://www.gps.gov/,” [Online].

[10] J. D. e. a. T. Allison, “A digital self excited loop for accelerating cavity field control,” in *Proceedings of PAC07*, Albuquerque, New Mexico, USA, 2007.

[11] O. Troeng and B. Bernhardsson, “Cavity Field Control in the Presence of Passband Modes,” Lund, 2016.

[12] B. B. Olof Troeng, “Modulator Ripple Control,” Lund, 2016.