|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| RF LPS Fast Interlock Control System- Fast Interlock Module (FIM) [VME Implementation] |
|  |
|  |

**RF LPS Fast Interlock Control System- Fast Interlock Module (FIM)**

**[VME Implementation]**

**System Requirements Specification**

|  | Name | **Role/Title** |
| --- | --- | --- |
| **Owner** | Niklas Claesson  Rafael Montano | Controls Engineer, ICS  RF Engineer, AD |
| **Reviewer** | Daniel Piso Fernandez  Annika Nordt  Hector Novella | HW&I Group Leader, ICS  Protection & Safety Group Leader, ICS  Deputy Project Manager, ICS |
| **Approver** | Timo Korhonen  Morten Jensen | Chief Engineer, ICS  RF WP Manager, AD |

|  |  |
| --- | --- |
| Table of content | Page |

1. Scope 4

2. Issuing organisation 4

3. Requirements 4

3.1. Functional Requirements 5

3.1.1. Electrical interface requirements 7

3.1.2. Control Requirements 8

3.1.3. Monitoring and Graphical User Interface Requirements 8

3.1.4. Data Acquisition and Archiving Requirements 8

3.2. FIM FPGA Module Requirements 9

3.2.1. Analog Input Pre-processing cell 9

3.2.1.1. Analog INPUT Pre-processing Level Discriminator 11

3.2.1.2. Analog INPUT Pre-processing Pulse with Monitor 12

3.2.1.3. Analog INPUT Signal Deviation Monitor 12

3.2.1.4. Analog INPUT Pre-processing Power Detector 12

3.2.2. Analog INPUT HISTORY Buffer 12

3.2.2.1. SMEM History Buffer 13

3.2.3. Digital Input pre-processing Cell 13

3.2.3.1. Digital Input pre-processing Functions 14

3.2.4. FIM State Machine 14

3.3. EPICS integration 15

3.4. Physical Interfaces 15

4. Glossary 16

5. references 16

Document Revision history 16

list of tables

**No table of figures entries found.**

list of Figures

Figure 1. Functional overview of the RF-LPS. 5

Figure 2. RF LPS VME Hardware Schematic -FIM 6

Figure 3: Fast Interlock Module Diagram 7

Figure 4: Analog Input Pre-processing Cell Diagram 9

Figure 5. A schematic view of a possible implementation of the analog processing function. 10

Figure 6. Digital Input pre-processing cell. 13

Figure 7.RF-LPS State machine 14

# Scope

This document contains the requirements for the RF LPS Fast Interlock Control System (VME Implementation)

# Issuing organisation

This document is issued by the Integrated Control System (ICS) division of ESS.

# Requirements

Requirements are identified with their device mnemonic from the ESS

| Id | Text | Trace up to |
| --- | --- | --- |
| xxx.nn1 | When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint],** so that signal x status can be assessed **<Rationale>.** | yyy.nn3 |
| xxx.nn2 | At sea state 1 **[Condition]**, the Radar System shall detect targets at ranges out to **[Action or Constraint]** 100 nautical miles **[Value],** so that **<Rationale>.** | yyy.239 |
| xxx.nn3 | The Invoice System **[Subject]** shall display pending customer invoices **[Action]** in ascending order **[Value]** in which invoices are to be paid**,** so that **<Rationale>.** | yyy.nn7 |

## Functional Requirements

The RF Local Protection System (RF-LPS) is in charge of preventing damage to the RF equipment within each RF cell and will also prevent any damage to other equipment related to the RF station where agreed, typically where a fast shutdown of the RF is required, e.g. arc detection, power detectors, etc. The RF LPS (Local Protection System) works independently from the other accelerator systems but can be configured and monitored through the main control system and via local human machine interface, e.g. touch screens etc.

A functional overview of the RF-LPS is presented in Figure 1. The system consists of a slow part (**Slow Interlock Module, SIM**) that takes care of signals that need to be monitored and reacted upon in the millisecond range, plus a fast part (**Fast Interlock Module, FIM)** (Figure 2) where signal monitoring, processing and reactions have to happen in the range of microseconds. The SIM is implemented using a PLC. The SIM and FIM communicate over discrete digital signals. The control system communicates with bot the SIM and FIM and handles the combination of them as a single unit.

This document is related to implementation of the fast part, or more precisely, a prototype implementation to verify the design principles using readily available components in VME form factor. This design will be migrated to the final platform based on MTCA.4 as soon as the prototype operation has been successfully demonstrated and the final MTCA components are available.

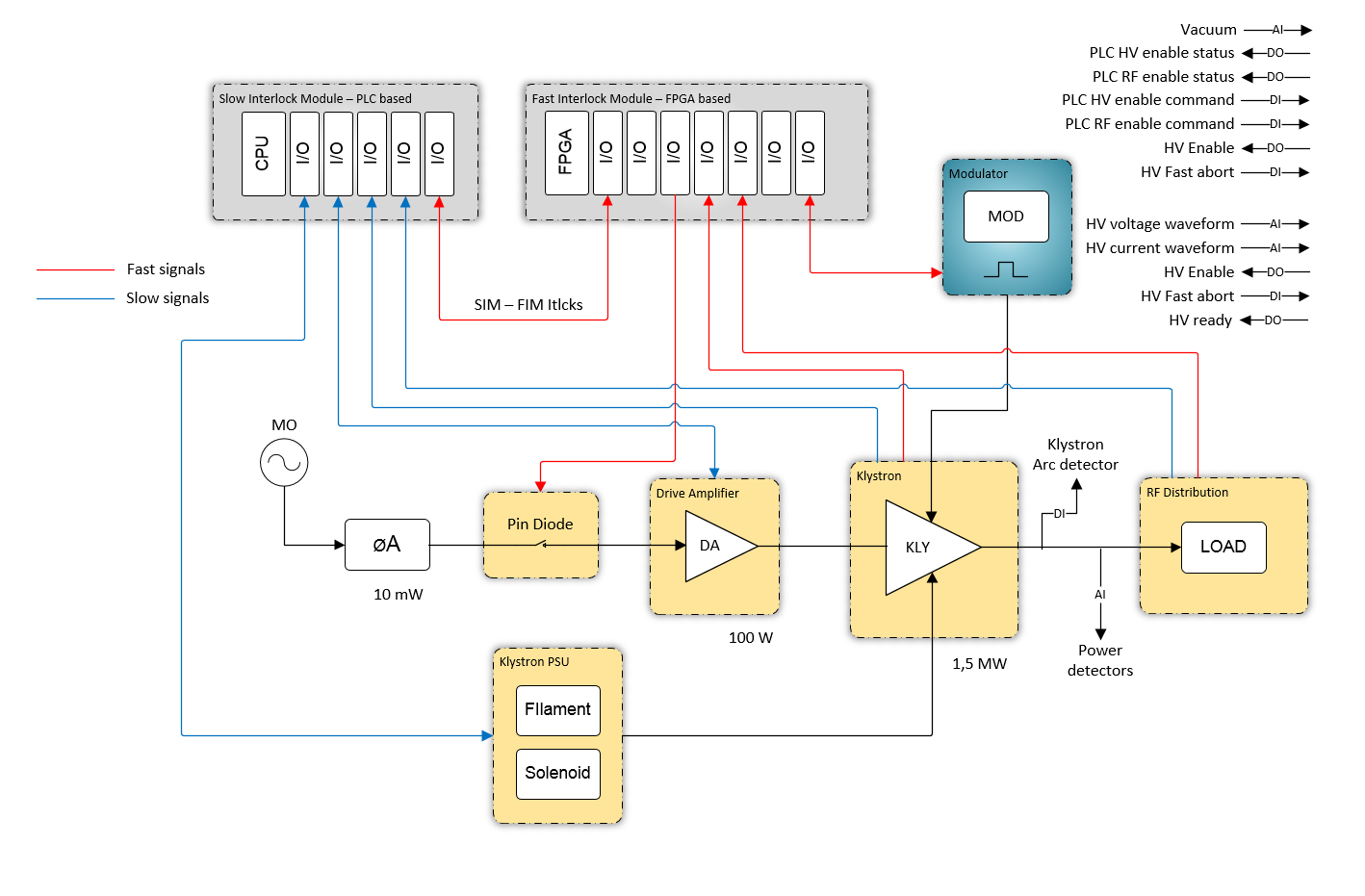


Figure 1. Functional overview of the RF-LPS.

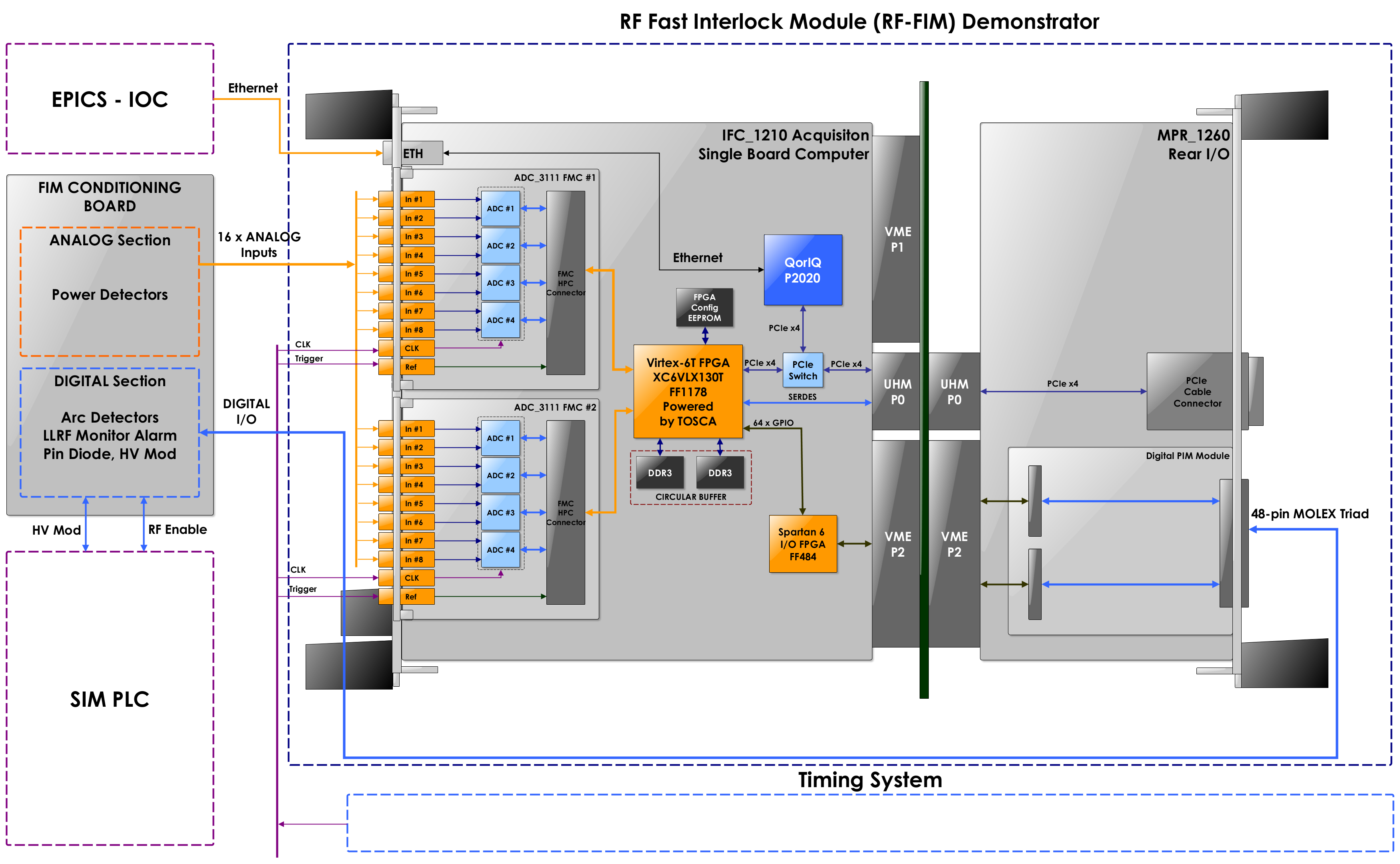


Figure 2. RF LPS VME Hardware Schematic -FIM

The fast interlock functions are critical and are implemented in a FPGA module in order to satisfy the reaction time requirement and to be deterministic and reliable. The RF equipment is expensive and any damage could cost thousands, even millions of euros to repair or replace. The control functions are not safety-critical and could be performed in the software configuration.

The components of the FIM are shown in Figure 3. The central parts of the FIM implementation are those in the FPGA, marked with red borders:

* Analogue signal pre-processing functions
* Digital signal pre-processing functions
* Interlock state machine
* Interlock state machine control, status and monitoring functions.



Figure 3: Fast Interlock Module Diagram

The RF LPS FIM continuously monitors a number of analog and digital signals.

### Electrical interface requirements

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM-001 | ***All signals related to the RF-FIM shall be connected through a conditioning board implementing  the following stages:***   * ***Amplification*** * ***Split*** * ***Isolation*** |  |
| FIM-002 | ***The FIM and the SIM shall communicate using a point-to-point connection.*** That is, there shall be a direct link between the FIM and SIM which shall not go over network switches, etc. |  |
| FIM-003 | ***The RF-LPS system shall be scalable to accept additional signals if required.*** |  |

### Control Requirements

|  |  |  |
| --- | --- | --- |
| FIM-004 | ***The RF interlock functions shall not be affected by external software related events such as: System reboot, Network malfunction,  Software update or accidental user access.***  If operations have to be done to the LPS that could compromise its function, the system has to enforce a safe state before such operations are allowed. |  |
| FIM-005 | ***The analog signal thresholds stored in the RF-FIM non-volatile memory shall be remotely  accessible by the EPICS IOC.*** |  |
| FIM-006 | ***The RF-LPS system shall keep record following information of the (TBD) last relevant interlock events:***   * ***Timestamp (to allow post-mortem analysis)*** * ***I/O status changes*** * ***Interval time with previous relevant event*** |  |

### Monitoring and Graphical User Interface Requirements

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM-007 | ***The waveform display shall be able to display coarser time resolution without changing the operating frequency of the interlock state machine.*** The full waveforms may not be possible to display if sampled at full speed (1 MHz). When displaying the waveform, the software should decimate to a lower frequency without changing the state machine operating speed. |  |

### Data Acquisition and Archiving Requirements

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM-008 | ***The analog and digital signal acquisition shall be active at all times, irrespective of the interlock status.*** However, after an interlock situation the readout buffers shall be “frozen” to store the status before and after the interlock event. |  |

## FIM FPGA Module Requirements

This section describes the requirements related to the firmware in the FPGA implementing the RF LPS interlock function: FIM.

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.009 | ***The configuration of the interlock shall be stored in a non-volatile memory on the device so that it can return to the correct values after power cycling.*** |  |
| FIM.010 | ***The LPS shall be capable of handling signals in less than 10 μs.*** |  |
| FIM.011 | ***Digital I/O shall support TTL signal levels.*** |  |
| FIM.012 | ***Interlock and alarm signals shall be managed through digital outputs*** |  |
| FIM.013 | ***Analog inputs shall support an input range of ±10V*** |  |
| FIM.014 | ***Analog inputs shall be capable of handling both differential and single-ended input signals*** |  |
| FIM.015 | ***Acquisition bandwidth of analog signals shall be 1 MSPS*** |  |
| FIM.016 | ***Resolution of the analog signal acquisition shall be 16 bits.*** |  |
| FIM.017 | ***Relevant interlock events shall be timestamped.*** |  |

### Analog Input Pre-processing cell

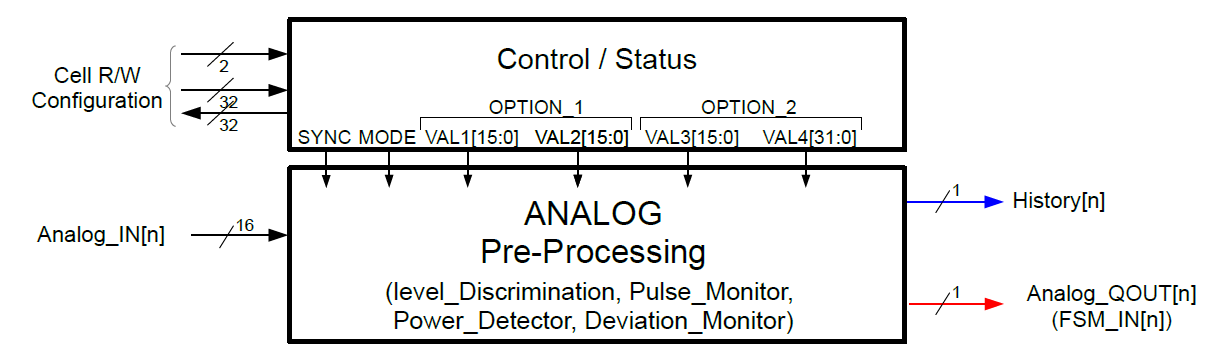


Figure 4: Analog Input Pre-processing Cell Diagram

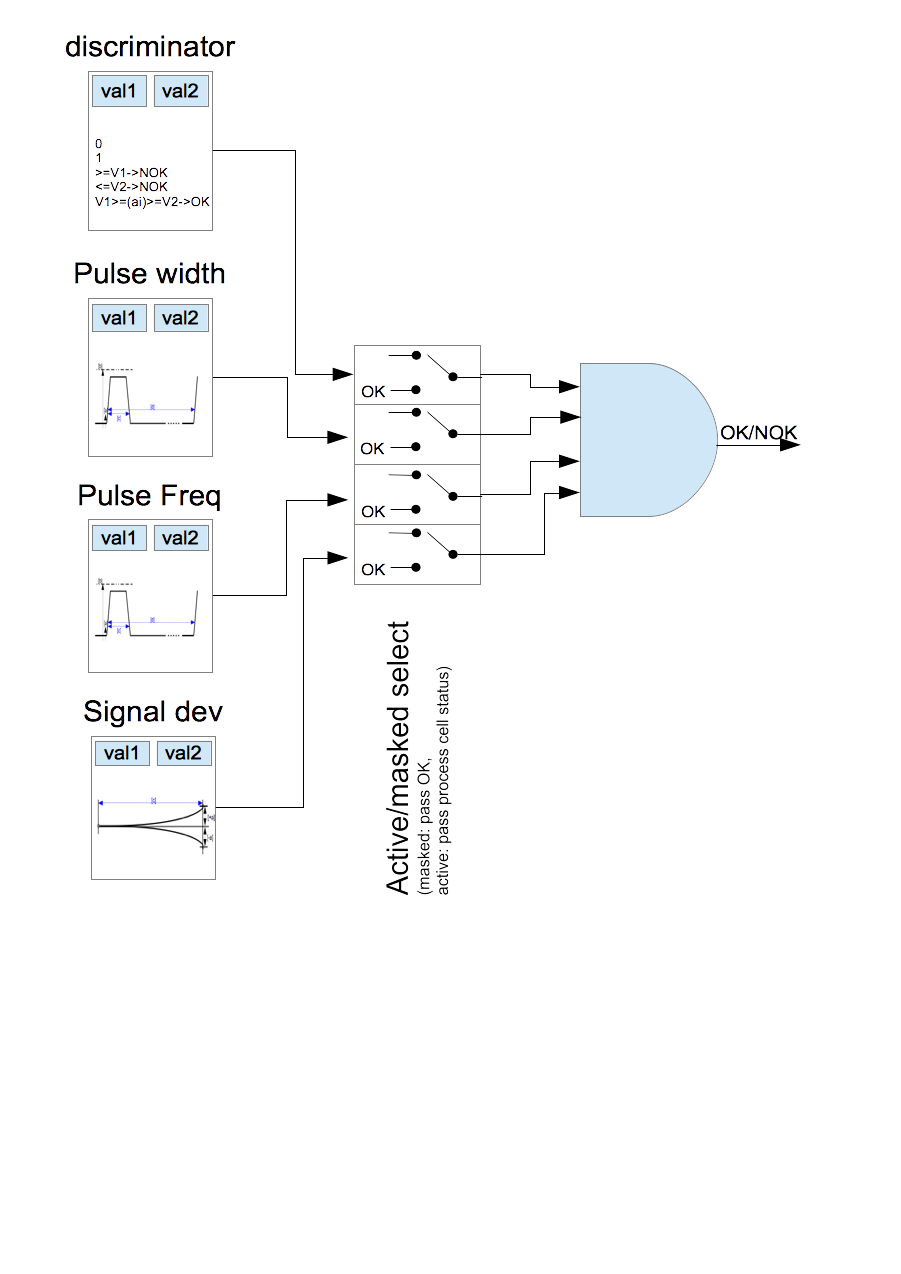


Figure 5. A schematic view of a possible implementation of the analog processing function.

Figure 5 presents schematically a possible implementation of one analog processing function cell. The idea is that an analog signal input feeds into multiple functions that are evaluated simultaneously. Outputs of these functions are states OK and NOK (“Not OK”), defined as logical 1 = OK, logical 0 = NOK. The outputs go to a select array, usually a register where the input would be activated by setting the corresponding register bit to one (1). Output of the entire function is NOK if any of the selected functions is in NOK state.

In addition to the function described in Figure 1 that takes single input channels as input, there shall also be functions that take two signals as input, apply a series of operations to the inputs and output a NOK or OK signal.

This function block, expressed graphically in Figure 6 , shall take as inputs any two analog input signals, multiply them, apply a breakpoint conversion (piecewise linear approximation) and apply a sliding window average function to the result value. The sliding average is compared to a threshold and a value that exceeds the set threshold shall trigger a NOK status.

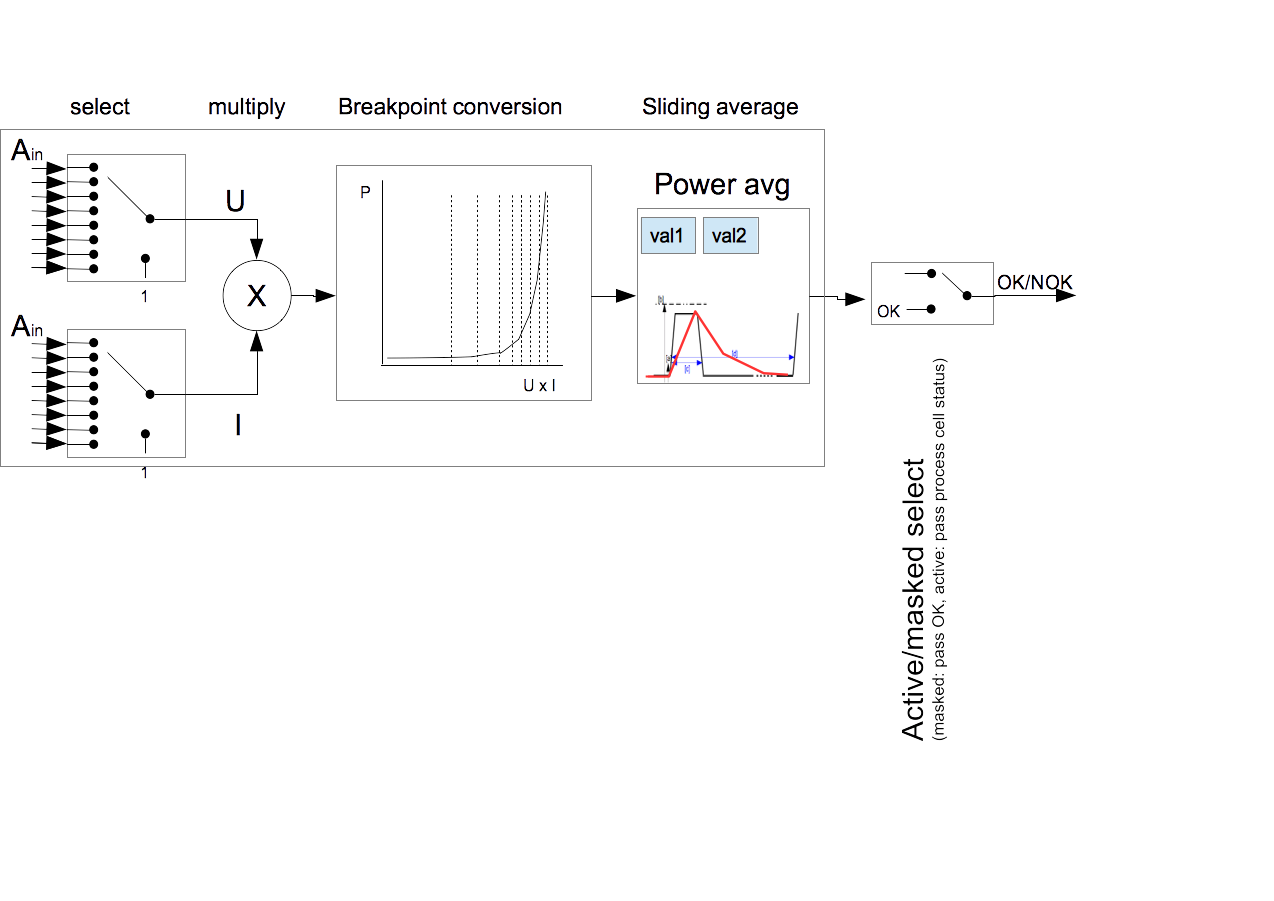


Figure 6. Function cell for average power monitoring

In the implementation, the number of single input cells shall be the same as the number of analogue input channels (16) and the number of power averaging functions shall be four (to be discussed).

Outputs of all these functions shall be feeding into the condition matching registers of the FIM State machine (3.2.4) and used to control the state transitions.

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***Analog processing shall include a function for threshold comparison (‘discriminator’) for interlock generation .*** Note: the already implemented function is sufficient and shall be kept as is. |  |
| FIM.00x | ***Analog processing shall include a function for pulse width monitoring against user defined values. Pulse width that exceeds the set threshold shall trigger a NOK status.*** |  |
| FIM.00x | ***Analog processing shall include a function for pulse frequency monitoring against user defined values. Pulse frequency above the set threshold shall trigger a NOK status.*** |  |
| FIM.00x | ***Analog processing shall include a function for monitoring signal deviation. (Note: not needed but as it has already been implemented, it can be kept as is.)*** |  |
| FIM.00x | ***Analog processing shall include a function for calculating sliding power average, with a settable time window. Power is calculated as a product of two channels (U\*I), which shall be freely selectable. If the average power exceeds the set threshold, an NOK status shall be generated.*** |  |
| FIM.00x | ***Acquired analog signals shall be stored in a buffer for later (post-mortem, etc.) analysis.*** |  |
| FIM.003 | ***Different processing functions shall be simultaneously available and configurable for all signals.*** (Currently the measurements are implemented as 1 measurement per AI or a set of specific measurements per AI it is not possible to select multiple arbitrary measurements per AI.) |  |
| FIM.00x | ***Threshold values for processing functions shall be independent of each other and settable for each function independently.*** However, the “zero level” signal for pulse width and pulse frequency shall be common within the function cell. |  |
| FIM.004 | ***It shall be possible to store permanent configurations for different modes.*** Currently VAL 1/2/3/4 are shared between modes and it is not possible to store permanent configurations for different modes. The type of VAL1/2/3/4 also changes depending on mode, which enforces the software to wipe out the value when the mode is changed. The proposed implementation would have separate threshold values for each function, thus each configuration can be saved individually. |  |

#### Analog INPUT Pre-processing Level Discriminator

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***The firmware shall implement a function to compare an input value to a high threshold limit, a low threshold limit or an interval (OK between low and high limits, NOK outside of those limits.)*** |  |
|  |  |  |
|  |  |  |
|  |  |  |

#### Analog INPUT Pre-processing Pulse Width Monitor

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***The firmware shall implement a function to monitor width of a pulse. Pulse width that exceeds a set threshold shall cause a NOK state.*** |  |
| FIM.00x | ***The pulse width monitor function has to be configurable, with a “zero level” setting and pulse width (in time) setting.*** |  |
| FIM.0xx | ***The “zero level” setting shall be common with the Pulse Frequency Monitor function.*** When the input signal level is over the zero level, it indicates being within a pulse. |  |
|  |  |  |

#### Analog INPUT Pre-processing Pulse Frequency Monitor

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***The firmware shall implement a function to monitor the frequency of pulses. Occurrence of pulses more often than a specified frequency shall trigger a NOK state.*** |  |
| FIM.00x | ***The pulse frequency monitor function has to be configurable, with a “zero level” setting and pulse frequency setting.*** The function should cover frequencies down to 0.1 Hz. (does this make sense?) |  |
| FIM.0xx | ***The “zero level” setting shall be common with the Pulse Width Monitor function*** |  |
|  |  |  |

#### Analog INPUT Signal Deviation Monitor

(Note: this function is not needed at this point of time but can be kept as it has been implemented and it could be useful in the future.)

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***The firmware shall implement a function to monitor the deviation of a signal from a nominal value.*** |  |
| FIM.00x | ***The signal deviation monitor function has to be configurable.*** |  |
|  |  |  |
|  |  |  |

#### Analog INPUT Pre-processing Power Detector

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***The firmware shall implement a function to convert the ADC value to power units***. Power is a non-linear function of the ADC value. Possible implementations could be a lookup table or piecewise interpolation. |  |
| FIM.005 | ***The power conversion function has to be configurable.***  The conversion factors may change when cables are changed (connected and disconnected). For example, if the implementation uses a lookup table, the tables need to be writable and readable. |  |
|  | ***The power conversion function shall take two inputs that shall be selectable.***  It should also be possible to tie the input to a “zero” level. |  |
| FIM.0xx | ***The power conversion function shall also implement a function to apply a configurable conversion from the multiplied signal.*** One possible way is to have a breakpoint table that can be configured. The size of the breakpoint table shall be at least (how many??) points. |  |
| FIM.0xx | ***The power conversion function shall implement a sliding average. The width of the averaging window shall be configurable (in number of samples).*** |  |

### Analog INPUT HISTORY Buffer

| Id | Text | Trace up to | |
| --- | --- | --- | --- |
| FIM.00x | ***The measured values shall be made available to the software.*** Recording the ADC values continuously into a history buffer is important to see the characteristics of the measured signal. | FIM.00x | |
| FIM.002 | ***After an interlock the history of all the values shall be made available to the software.*** During regular operations it is enough to poll the measured values in an OPI friendly frequency. But after an interlock the history of all the values are needed as well. Updates to the buffer shall be frozen until read out by the software and the memory updates are again re-enabled. | |  | |
| FIM.00x | ***The number of values stored after an interlock shall be configurable.*** The current version of the firmware stores 500 values after post-interlock (on abort), not 1k as documented. |  | |

#### SMEM History Buffer

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.008 | ***The whole available RAM shall be possible to use for the history buffer.*** Having the size definable does not prevent from using the whole memory. The default setting should however be to use the whole available memory space. |  |
| FIM.010 | ***ADC values shall be stored per channel.*** Since the ADC values are interleaved the CPU has to spend a lot of time reshuffling them. If ADC values are stored per channel,software can decide which channels to read out. This improves real time performance. However, this should only be implemented if there are no other major implications to the performance of the application. Response times etc. shall not be compromised by the implementation. |  |
| FIM.011 | ***Reading out the data in the FPGA in the CPU order (big-endian, same as in the CPU) shall be possible.*** This would reduce the CPU power needed for data processing. However, this feature is probably best to keep configurable so that different process architectures can be accommodated. Also, PCI Express is little-endian by specification. |  |

Remarks concerning the current implementation:

* When the board goes into interlock all history will be read out. ***The read address pointer can be removed.***
* Depending on mode the first address in the history buffer is updated with the last value. ***This feature is not needed and can be removed.***

### Digital Input pre-processing Cell



Figure 7. Digital Input pre-processing cell.

#### Digital Input pre-processing Functions

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.00x | ***Digital processing shall include a configurable function for processing the digital input signal for interlock generation as follows:***  ***Function 1: always output NOK***  ***Function 2: always output OK***  ***Function 3: pass through the input signal as is***  ***Function 4: invert the input signal*** |  |
| FIM.003 | ***The processing function shall be simultaneously available and configurable for all signals.*** |  |

### FIM State Machine



Figure 8.RF-LPS State machine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FSM** | **IDLE** | **PRE** | **RUN** | **ABO** |
| **HV\_Enable**  (VME P2\_A03 / LVTTL) | '0' (NOK) | '1' (OK) | '1' (OK) | '0' (NOK) |
| **RF\_Enable**  (VME P2\_C05 / LVTTL) | '0' (NOK) | '0' (NOK) | '1' (OK) | '0' (NOK) |
| **LLRF\_Enable**  (VME P2\_C02 / LVDS) | '0' (NOK) | '0' (NOK) | '1' (OK) | '0' (NOK) |
| **Spare**  (VME P2\_C04 / LVTTL) | '0' (NOK) | '0' (NOK) | '0' (NOK) | '1' (OK) |

| Id | Text | Trace up to |
| --- | --- | --- |
| FIM.0xx | **The RF-LPS FIM shall implement the state machine as drawn in** Figure 7**.** |  |
| FIM.0xx | **The state machine shall implement the state transitions according to pre-definable, configurable conditions** |  |
| FIM.0xx | **All input signals after pre-processing shall be possible to include as conditions in the state transitions.** Not all signals are required to be used either, so a way to define what is included is required. Current implementation uses configurations for this, which is a good way. |  |
| FIM.0xx | **State transitions, except moving to an ABO state, shall need an active command to happen.** |  |
| FIM.0xx | **If the history buffer is enabled it shall always record data, until an interlock status (ABO) is entered.**The different *History\_Modes* are not necessary. |  |
| FIM.013 | ***The number of values stored after an interlock shall be configurable.*** The current version of the firmware stores 500 values after post-interlock (on abort), not 1k as documented. |  |
|  |  |  |

## EPICS integration

This section will contain the requirements related to the EPICS integration

| Id | Text | Trace up to |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

## Physical Interfaces

The requirements for physical interfaces (e.g connectors) are described in this section

| Id | Text | Trace up to |
| --- | --- | --- |
|  |  |  |

# Glossary

| Term | Definition |
| --- | --- |
| ICS | Integrated Control System |
|  |  |
| GUI | Graphical User Interface |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# references

1. -

Document Revision history

| Revision | Reason for and description of change | Author | Date |
| --- | --- | --- | --- |
| 1 | First issue | <<Name>> | <<YYYY-MM-DD>> |
|  | <<Keep only full number revisions when approving document>> |  |  |
|  |  |  |  |