# Unilac-BLM Programmer's Manual

GSI ACO HEL

December 5, 2024
ver. 0.1

# Contents


**1 Introduction** **1**

1.1 About this document . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Conventions . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2.1 Typesetting . . . . . . . . . . . . . . . . . . . . . . . 1

1.2.2 Reserved terms . . . . . . . . . . . . . . . . . . . . . 2

1.2.3 Numbering of entities . . . . . . . . . . . . . . . . . . 3

1.2.4 Version numbering . . . . . . . . . . . . . . . . . . . 3

**2 Hardware description** **5**

2.1 System layout . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 The Beam Loss Monitor (BLM) crate . . . . . . . . . . . . . . 6

2.3 Functional overview . . . . . . . . . . . . . . . . . . . . . . . 7

2.3.1 An up/down counter module . . . . . . . . . . . . . . . 7

2.3.2 Device architecture . . . . . . . . . . . . . . . . . . . 8

2.4 Event-driven operation . . . . . . . . . . . . . . . . . . . . . . 12

**3 Programmers interface** **13**

3.1 Register model . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.1.1 Notation . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.1.2 Data format . . . . . . . . . . . . . . . . . . . . . . . 13

3.2 Registers reference . . . . . . . . . . . . . . . . . . . . . . . . 14

3.2.1 Register superblocks . . . . . . . . . . . . . . . . . . . 14

3.2.2 DIOB configuration and status superblock (0x0000) . . 14

3.2.3 IO backplane mask superblock (0x0630) . . . . . . . . 14

3.2.4 IO backplane ID superblock (0x0638) . . . . . . . . . . 15

3.2.5 Status superblock (0x0700) . . . . . . . . . . . . . . . 16

3.2.6 Control superblock (0x0800) . . . . . . . . . . . . . . . 20

3.2.7 Event status superblock (0x0900) . . . . . . . . . . . . 22

3.2.8 Event control superblock (0x0A00) . . . . . . . . . . . 24

3.2.9 Input matrix superblock (0x1000) . . . . . . . . . . . . 26

3.2.10 Output matrix superblock (0x1100) . . . . . . . . . . . 27

3.2.11 Counter readout superblock (0x1200) . . . . . . . . . . . 29
3.2.12 Counter group assignment superblock (0x1600) . . . . 30
3.2.13 Threshold readout superblock (0x1800) . . . . . . . . . . 30
3.2.14 Threshold memory (0x8000) . . . . . . . . . . . . . . . . . 31
3.3 Event decoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.3.1 Event tag . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.3.2 Event commands . . . . . . . . . . . . . . . . . . . . . . . . 33

# Chapter 1

# Introduction

## 1.1 About this document

This document (called further the Manual) is about to describe the hardware and firmware for Unilac Beam Loss Monitor (BLM), especially the programmer's interface. It is the official reference for implementing related software as well as installation and commissioning of the system.

## 1.2 Conventions

### 1.2.1 Typesetting

This Manual uses special text decoration to emphasize words or fragments of text with special meaning. A summary of used text styles is shown in Table 1.1.

| | |
|---|---|
| Constants | are names which denote well defined numbers like register addresses, bit masks etc. |
| Reserved Terms | are important, well defined names, see Section 1.2.2. |
| *Constraints* | are such words as *shall* or *may* . |
| *Indexed terms* | are terms introduced for the first time. They can be found in the index. |
| *Citations* | are directly cited sentences or citation-like examples. |
| `Code snippets` | are pieces of programming code or pseudo-code. |
| **Important:** Important | are things which should be read twice. |
| **Comment:** Comments | are put mostly for clarification of statements with questionable meaning. |

Table 1.1: Summary of styles

### 1.2.2 Reserved terms

Some words in the Manual (like this one to the left) are treated as *reserved terms*: it means that they mean exactly what they should mean and *shall not* be used in a different context.

*Constraints* are a special form of reserved terms which define the area of Developer's freedom. For clarity, they are all always printed with specific typesetting. They are all listed in Table 1.2.

| | |
|---|---|
| *must* | means that given condition must be fulfilled under any circumstance |
| *must not* | means that something is forbidden |
| *shall* | means that the developer should proceed according the given specification if it's possible. Commonly, this term will be used together with a condition, e.g. *The device* shall *reply immediately if it doesn't affect its real-time functions* |
| *shall not* | means that something should be avoided if it's possible |
| *may* | means that it is allowed to do something and the Developer is free to choose |
| Manual | means this book |
| Developer | means a person who develops software based on the Manual. |
| high | means bit state of 1 |
| low | means bit state of 0 |

Table 1.2: Constraints and reserved terms

### 1.2.3 Numbering of entities

Every time when multiple entities of the same kind appear (like addresses in address space, registers, slots in the backplane, connectors, cookies), their numbering is **zero-based**.

### 1.2.4 Version numbering

*Version numbers* **TODO: Version numbers**

# Chapter 2

# Hardware description

## 2.1 System layout

Figure 2.1 shows a general system layout of the BLM.

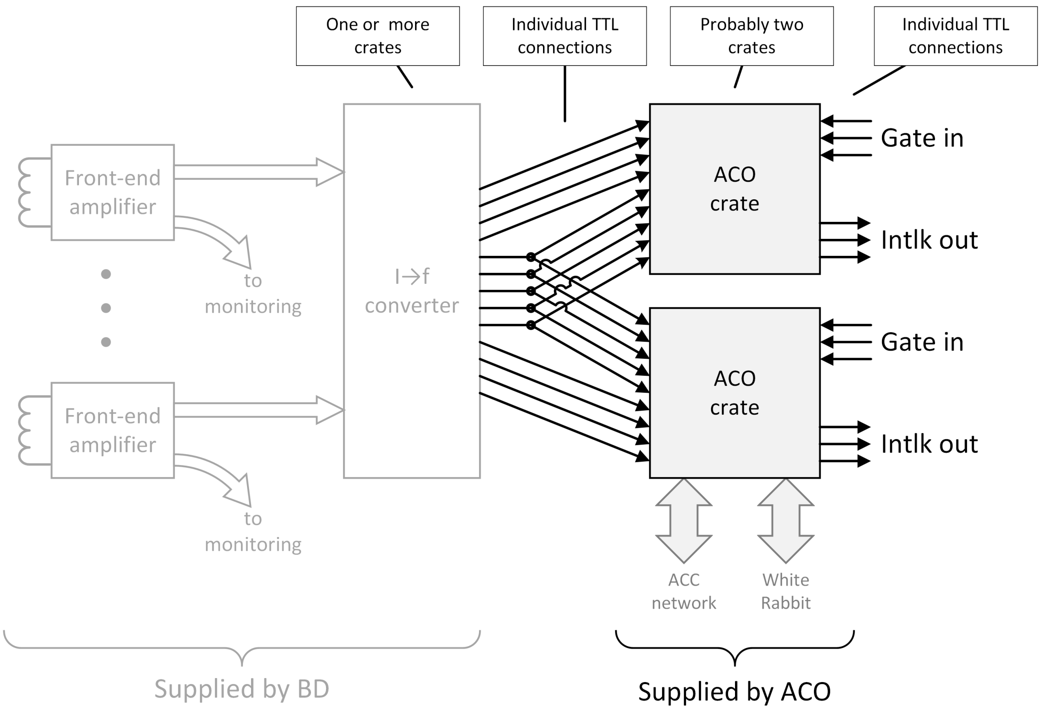


Figure 2.1: A possible system layout

The beam loss measurement at Unilac is based on comparing beam intensities measured at various places along the machine. Beam current is measured in multiple places by beam transformers, amplified by front-end amplifiers and sent to current-to-frequency converters. There, the analog signal is converted to a series of digital pulses, whose rate depends on the input current and may reach up to 25 MHz.

The pulse signals are sent to *BLM crates* (called also ACO crates) via a number of individual LEMO cables, using Trasnsistor-Transistor Logic (TTL) standard levels. There, pulses are counted and the counts are used to generate signals on *interlock outputs* if certain thresholds are exceeded.

For enabling the counters only within the beam-on time window, *gate inputs* are used. Both gate inputs and interlock outputs use TTL standard levels.

The BLM is controlled over the ACC network. The needed real-time information is supplied via a White-Rabbit connection.

## 2.2 The BLM crate

A single crate contains required power supplies and the Scalable Control Unit (SCU). Further it is equipped with a Digital I/O Board (DIOB) module, which is the heart of the system and covers the whole BLM functionality described here. The DIOB is equipped with a semi-backplane and 12 input/output (I/O) modules for connecting all the needed signals, as shown in Figure 2.2.

Note that module layout is fixed and cannot be changed. However, exchanging modules in slots 9–11 to optical I/Os is possible.

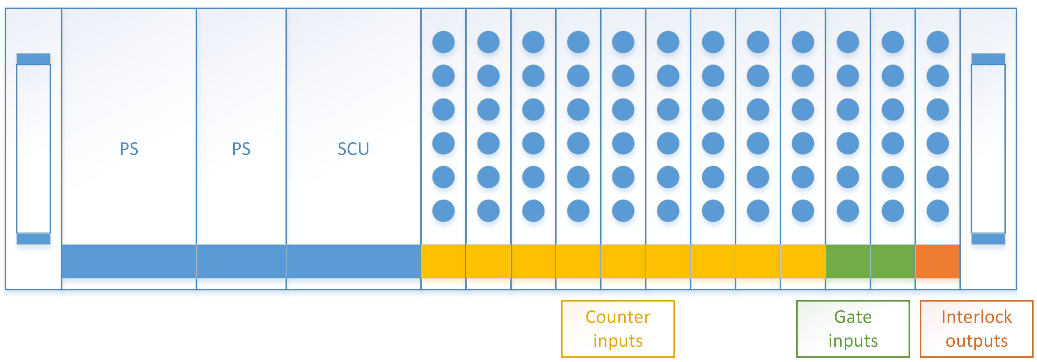


Figure 2.2: The BLM crate

| Function | Description | FG-Nr. | Slots | Count |
|---|---|---|---|---|
| Counter inputs | Fast TTL input with LEMO connectors | FG902.150 | 0–8 | 9 |
| Gate inputs | Isolated TTL input with LEMO connectors | FG902.130 | 9–10 | 2 |
| | Fibre optical input | FG902.110 | 9–10 | 2 |
| Interlock outputs | TTL output with LEMO connectors | FG902.140 | 11 | 1 |
| | Fibre optical output | FG902.120 | 11 | 1 |

Table 2.1: I/O module types used

## 2.3 Functional overview

### 2.3.1 An up/down counter module

The *up/down counter* is the most important elementary functional component of the BLM crate. It's shown in Figure 2.3.

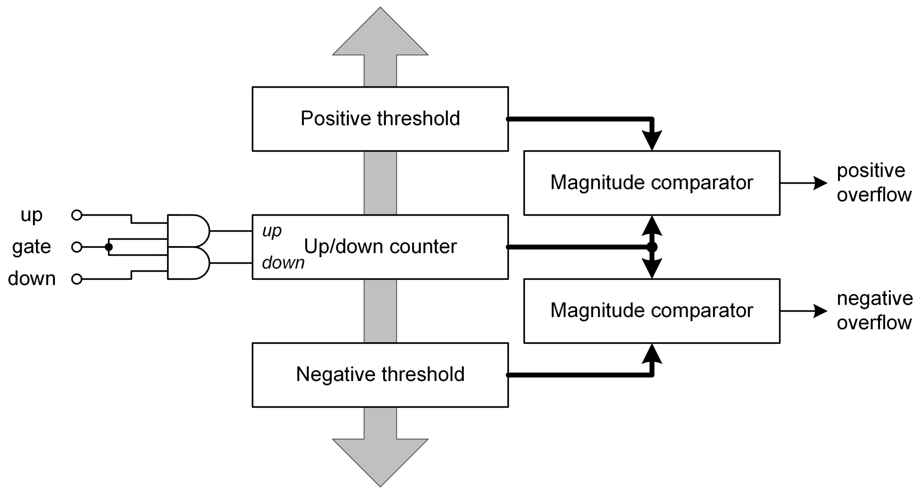


Figure 2.3: A single up/down counter

With the up- and downcounting capability, pulse rates from two detectors can be compared to calculate the beam loss. The gate input allows activating the counter only within the beam-on time window.

The count value is continuously compared against the *positive threshold* and *negative threshold*. In case of exceeding on of the thresholds, a corresponding *overflow signal* is asserted.

The count value and thresholds are represented with 32-bit two's complement code (U2)-encoded signed numbers. The current count value and thresholds can be read out via a common bus. Thresholds can be programmed as described later in the Manual.

There is no overflow protection, but an overflow is not probable in normal operation: one would need 85 s of open gate with an input frequency of 25 MHz.

### 2.3.2 Device architecture

A complete block diagram of the BLM crate is shown in Figure 2.4.

The central element is a pool of 128 up/down counter modules: the *counter pool*. The *threshold laoder* is responsible for loading values from the *threshold memory* into counter modules.

The counters can be connected to arbitrary inputs and outputs by means of patch matrices:

- *input patch matrix* for counter up/down signals,
- *gate patch matrix* for gate signals,
- *output patch matrix* for overflow outputs.

The *input watchdogs* are used for checking the input signal integrity.

To assure that measurement sequences are correct, *gate logic* is used.

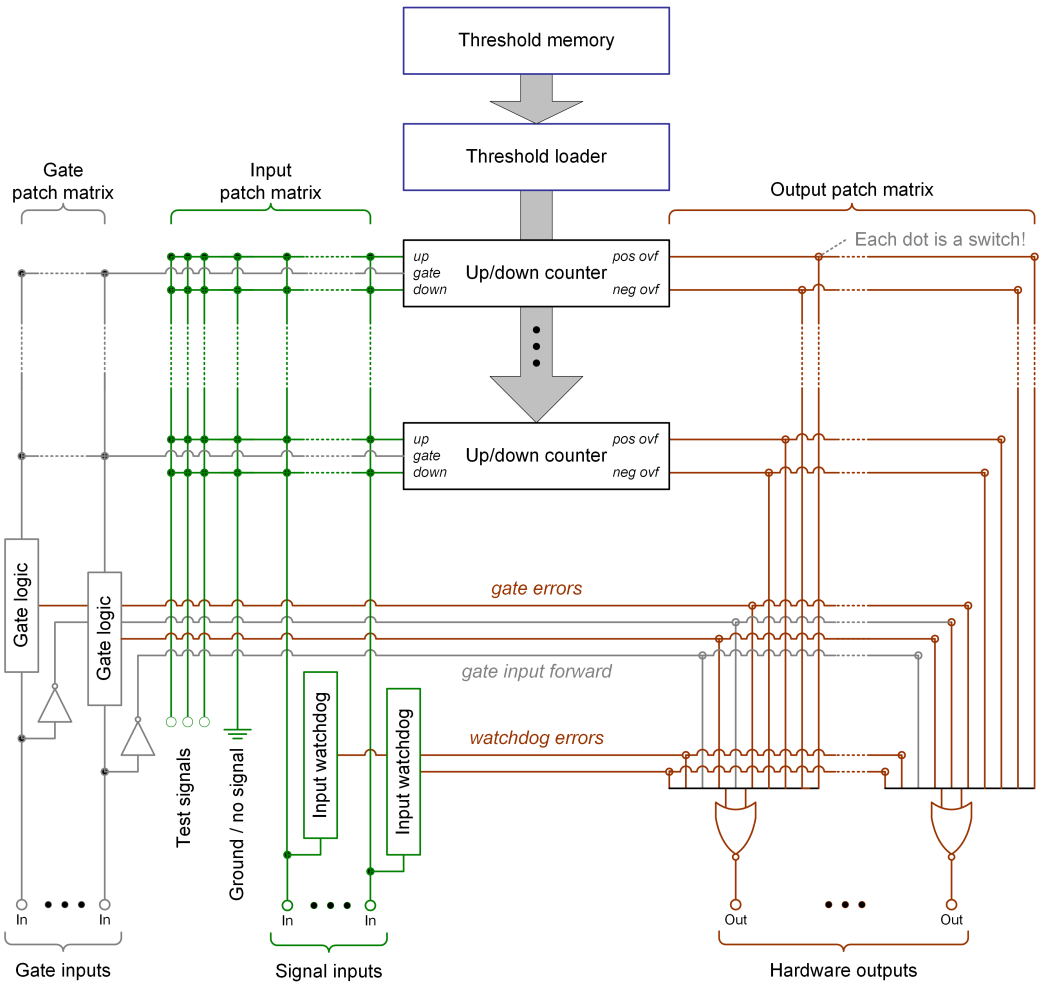


Figure 2.4: Block diagram of the device

### Threshold loader and memory

A full set of threshold values contains 256 numbers: such a bundle is called a *dataset*.

The threshold memory can store 32 datasets. The threshold loader allows almost instantaneous reloading the counter modules with a new dataset. To support the extraordinary Unilac's architecture with timing sections and virtual accelerators, the whole counter pool is divided into *groups*: each individual counter can be freely assigned to one of 16 groups.

Groups and datasets have direct relation to Unilac's timing sections and virtual accelerators:

- group → timing section

- dataset → virtual accelerator

A single action of the threshold loader (*threshold reload*) is no more no less than: ***reloading given group with given dataset***. In order to do it, the threshold loader cycles over all counters, checks their group numbers and applies new threshold values from memory if the group number matches the requested value. The whole procedure needs less than 5 µs. This time is independent of the size of the group to be reloaded.

The threshold loader is equipped with 16-stage First-In-First-Out (FIFO) buffer for enqueuing reload requests. This allows for accepting batches of reload requests via the event framework (see Section 2.4) in a very short time.

### Gate logic

Gate logic is used to check the integrity of gate signal connections. More precisely, it's not a pure logic, but rather a state machine, which works according to the following algorithm:

1. Wait for a *prepare signal*. The prepare signal is provided either by software or by the event system (see Section 2.4).
2. Wait for high signal on the input for a specified (programmable) time.
3. Keep the gate output asserted as long as the input remains asserted.
4. After deassertion of the input and prepare signal, wait for the next prepare signal.

Alternatively, the gate logic can be switched into *direct gate mode*, where the input is fed directly to the output. This allows operation without prepare signals for a cost of lacking integrity check.

The full state diagram is shown in Figure 2.5.

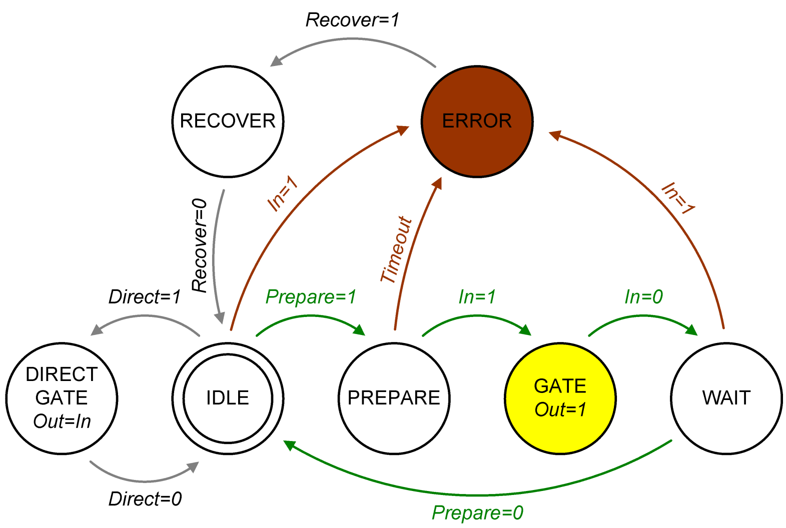


Figure 2.5: State diagram of the gate logic. Green arrows show the normal operation sequence

### Input watchdogs

Input watchdogs are used for continuous checking the signal path integrity between current-to-frequency converters and the BLM crate. The check procedure relies on a fact that the pulse rate never falls to zero: even with zero current signals, the converters output pulses at some base rate.

An input watchdog generates an error if it doesn't recognize a positive signal slope for a certain (programmable) time. Watchdog errors are latched and need an error reset procedure (see 3.2.6).

The input signals are forwarded to the input patch matrix independently of the watchdog error states. Watchdogs cannot be disabled, but their outputs may remain unconnected in the output patch matrix.

### Input and gate patch matrices

The input patch matrix allows connecting the up- and downcounting inputs of each counter:

- either to one of 54 signal inputs...
- or to one of internal test signals (see Table 3.25),...
- or to ground input (no signal).

The gate patch matrix allows connecting the gate input of each counter to one of the gate inputs via gate logic.

### Output patch matrix

The output patch matrix allows combining various signals into the 6 interlock output signals. The signals to be combined are:

- positive and negative overflow signals from counter modules,
- error signals from input watchdogs,
- error signals from gate logic,
- forwarded gate inputs.

The feature of gate input forwarding is meant for daisy-chaining or networking multiple crates, where outputs from one or more crates can be merged in by another crate.

For each output, all selected signals are logically or-ed. For compatibility with inverse Fast Beam Abort System (FBAS)-type logic, the output signal is negated. Signals from gate inputs are pre-inverted to maintain correct logic.

## 2.4 Event-driven operation

Some operations of the BLM (like reloading thresholds or preparing gates) can be triggered by events on the SCU bus. An *event* is a special SCU bus cycle which is well-timed by means of Event-Condition-Action (ECA) controller of the SCU. It provides a 32-bit *event tag* to the device, which hold the information regarding the action to be performed.

For a full description of event tags, see Section 3.3.

# Chapter 3

# Programmers interface

## 3.1 Register model

### 3.1.1 Notation

The device is accessed via registers. All registers are 16-bit wide and are word-addressed. The 16-bit address space allows addressing up to 65536 registers.

The Manual uses a strictly structurized register model with following levels:

- *superblocks*
- *blocks* (optional)
- *subblocks* (optional)
- *registers*
- *bits* or *bit groups* (optional)

Correct notation includes all these levels separated by dots. Bits and bit groups are separated by a colon:

SUPERBLOCK[.BLOCK[.SUBBLOCK]].REGISTER[:BIT]

Every existing register *must* have a name! Bits or bit groups *may* have a name. If not defined, default names for bits are bit0 for least significant bit (lsb) to bit15 for most significant bit (msb).

### 3.1.2 Data format

Unless otherwise noted, registers represent 16-bit unsigned integers.

Unless otherwise noted, bits are used in positive logic:

- if a particular bit is used as switch, high state means 'switch on'.

- if a particular bit is used as trigger, then a transition from low to high will be used.

## 3.2 Registers reference

### 3.2.1 Register superblocks

Table 3.1 shows the general register layout of the DIOB code (housekeeping registers are not included).

| Word address | Symbol | Description | See |
|---|---|---|---|
| 0x0000 | DIOB_CS | DIOB configuration and status superblock | 3.2.2 |
| 0x0630 | IOBP_MASK | IO backplane mask superblock | 3.2.3 |
| 0x0638 | IOBP_ID | IO backplane ID superblock | 3.2.4 |
| 0x0700 | STATUS | Status superblock | 3.2.5 |
| 0x0800 | CTRL | Control superblock | 3.2.6 |
| 0x0900 | EVT_STATUS | Event status superblock | 3.2.7 |
| 0x0A00 | EVT_CTRL | Event control registers | 3.2.8 |
| 0x1000 | IN_SEL | Input matrix superblock | 3.2.9 |
| 0x1100 | OUT_SEL | Output matrix superblock | 3.2.10 |
| 0x1200 | CTR_READOUT | Counter readout superblock | 3.2.11 |
| 0x1600 | CTR_GROUPS | Counter group assignment superblock | 3.2.12 |
| 0x1800 | THR | Threshold readout superblock | 3.2.13 |
| 0x8000 | THR_RAM | Threshold memory | 3.2.14 |

Table 3.1: General layout of register blocks

### 3.2.2 DIOB configuration and status superblock (0x0000)

Currently, this superblock doesn't contain any registers.

### 3.2.3 IO backplane mask superblock (0x0630)

The IO backplane mask registers allow controlling the red Light-Emitting Diode (LED)s on I/O modules.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | BP_MASK0-1 | LED states for module 0 and 1 |
| ... | | | |
| 0x0005 | R/W | BP_MASK10-11 | LED states for module 10 and 11 |

Table 3.2: DIOB configuration and status superblock layout. Base: 0x0630

| Bits | Symbol | Description |
|---|---|---|
| 5–0 | LED0_5 downto LED0_0 | LED states for the first I/O module |
| 11–6 | LED1_5 downto LED1_0 | LED states for the second I/O module |
| 15–12 | | (not used) |

Table 3.3: Bit definition of IO backplane mask registers

### 3.2.4 IO backplane ID superblock (0x0638)

The IO backplane ID registers allow checking which types of I/O modules are attached to the DIOB inter-backplane. There are 6 registers in total - each register handles two I/O modules - the first module occupies 8 lsbs and the second one 8 msbs as shown in Table 3.5.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | BP_ID0-1 | ID read out from I/O module 0 and 1 |
| ... | | | |
| 0x0005 | R/o | BP_ID10-11 | ID read out from I/O module 10 and 11 |

Table 3.4: DIOB configuration and status superblock layout. Base: 0x0638

Table 3.6 shows IDs for used module types.

| Bits | Symbol | Description |
|---|---|---|
| 7–0 | ID0 | ID of the first I/O module |
| 15–8 | ID1 | ID of the second I/O module |

Table 3.5: Bit definition of IO backplane ID registers

| Value | Symbol | Description | FG-Nr. |
|---|---|---|---|
| 0x03 | IO_LEMOIN | Isolated TTL input with LEMO connectors | FG902.130 |
| 0x04 | IO_FIBREIN | Fibre optical input | FG902.110 |
| 0x05 | IO_FIBREOUT | Fibre optical output | FG902.120 |
| 0x06 | IO_LEMOOUT | TTL output with LEMO connectors | FG902.140 |
| 0x07 | IO_DIGIN | Fast TTL input with LEMO connectors | FG902.150 |

Table 3.6: IDs for used I/O module types

### 3.2.5 Status superblock (0x0700)

The status superblock is used to read the overall device state, including a number of signals, errors and state machine states.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | NEG_OVERFLOW | Start of the negative overflow block |
| 0x0008 | R/o | POS_OVERFLOW | Start of the positive overflow block |
| 0x0010 | R/o | GATE_ERROR | Gate error register |
| 0x0011 | R/o | WD_ERROR | Start of the watchdog error block |
| 0x0017 | R/o | GATE_IN | Gate input monitoring register |
| 0x0018 | R/o | GATE_OUT | Gate output monitoring register |
| 0x0019 | R/o | OUT_SIGNAL | Output monitoring register |
| 0x001A | R/o | GATE_STATE | Gate state monitoring block |
| 0x001D | R/o | IO_CTRL_STATE | I/O slow control state monitoring register |

Table 3.7: Status superblock layout. base: 0x0700

### Negative overflow block (0x0700+0x0000)

This block contains 8 registers for reporting negative overflow signals from the 128 counters. Within each register, lsb corresponds to the counter with the lowest number.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | CTR0-15 | Overflow signals for counters 0 to 15 |
| ... | | | |
| 0x0007 | R/o | CTR112-127 | Overflow signals for counters 112 to 127 |

Table 3.8: Negative overflow block layout. Base: 0x0700+0x0000

### Positive overflow block (0x0700+0x0008)

This block contains 8 registers for reporting positive overflow signals from the 128 counters. Within each register, lsb corresponds to the counter with the lowest number.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | CTR0-15 | Overflow signals for counters 0 to 15 |
| ... | | | |
| 0x0007 | R/o | CTR112-127 | Overflow signals for counters 112 to 127 |

Table 3.9: Positive overflow block layout. Base: 0x0700+0x0008

### Gate error register (0x0700+0x0010)

The gate error register contains error information about the 12 gates; lsb corresponds to gate 0. Bits 12 to 15 are not used.

### Watchdog error block (0x0700+0x0011)

This block contains 4 registers for reporting watchdog error signals from the 54 input watchdogs. Within each register, lsb corresponds to the channel with the lowest number. In case of IN48-53, 10 msbs are not used.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | IN0-15 | Watchdog error signals for inputs 0 to 15 |
| 0x0001 | R/o | IN16-31 | Watchdog error signals for inputs 16 to 31 |
| 0x0002 | R/o | IN32-47 | Watchdog error signals for inputs 32 to 47 |
| 0x0003 | R/o | IN48-53 | Watchdog error signals for inputs 48 to 53 |

Table 3.10: Watchdog error block layout. Base: 0x0700+0x0011

### Gate input monitoring register (0x0700+0x0017)

The gate input monitoring register contains the current input state of the 12 gates; lsb corresponds to the gate 0. Bits 12 to 15 are not used.

### Gate output monitoring register (0x0700+0x0018)

The gate output monitoring register contains the current output state of the gate logic for all 12 gates; lsb corresponds to gate 0. Bits 12 to 15 are not used.

### Output monitoring register (0x0700+0x0019)

The output monitoring register contains the current state of the 6 interlock outputs; lsb corresponds to output 0. Bits 6 to 15 are not used.

### Gate state monitoring block (0x0700+0x001A)

This block contains 3 registers for reporting current state of the gate logic state machines. Each register contains a 4-bit state ID for four gates.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | GATE0-3 | Current state for gates 0 to 3 |
| 0x0001 | R/o | GATE4-7 | Current state for gates 4 to 7 |
| 0x0002 | R/o | GATE8-11 | Current state for gates 8 to 11 |

Table 3.11: Gate state monitoring block layout. Base: 0x0700+0x001A

| Bits | Symbol | Description |
|---|---|---|
| 3–0 | GATE0 | State of gate N+0 |
| 7–4 | GATE1 | State of gate N+1 |
| 11–8 | GATE2 | State of gate N+2 |
| 15–12 | GATE3 | State of gate N+3 |

Table 3.12: Bit definition of gate state monitoring registers

| Value | Symbol | Description |
|---|---|---|
| 0x0 | IDLE | Waiting for prepare |
| 0x1 | PREPARE | Prepare received, waiting for gate signal |
| 0x2 | GATE | Gate signal asserted, waiting for deassertion, output high |
| 0x3 | WAITING | |
| 0x4 | ERROR | Timout or forbidden sequence detected |
| 0x5 | RECOVER | Intermediate state in the error recovery procedure |
| 0x6 | DIRECT_GATE | Working in direct gate mode, input is forwarded to output |

Table 3.13: State IDs for gate logic. See Figure 2.5 for state explanation.

#### I/O slow control state monitoring register (0x0700+0x001D)

This register is used for monitoring the operation of the I/O slow-control subsystem, which is responsible for enumerating I/O modules and controlling front-panel LEDs. It is used solely for debugging purposes.

### 3.2.6 Control superblock (0x0800)

The control superblock contains registers basic device control and configuration. For setting up patch matrices, see Sections 3.2.9 and 3.2.10. For configuring the event framework, see Section 3.2.8.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | WDOG_TIMEOUT | Watchdog timeout register |
| 0x0001 | R/W | COUNTERS | Counters control register |
| 0x0002 | R/W | GATE_MODE | Gate mode register |
| 0x0004 | R/W | GATE_TIMEOUT | Start of the gate timeout block |
| 0x0010 | R/W | WD_RESET | Start of the watchdog reset block |

Table 3.14: Control superblock layout. Base: 0x0800

#### Watchdog timeout register (0x0800+0x0000)

Configures the timeout for all watchdogs in 1024 ns steps. Watchdogs will generate error if the input signal frequency is lower than 1/(WDOG_TIMEOUT·

1024ns). With 16-bit registers, the smallest programmable frequency is about 14.9 Hz.

### Counters control register (0x0800+0x0001)

The counters control register is used for:

1. Resetting all counters. This is an alternative for event-based control of counter reset (see Section 3.3.2).

2. Enabling counter auto-reset. If counter auto-reset is enabled, the counters will clear their counts on the rising slope of incoming gate signal.

| Bits | Symbol | Description |
|---|---|---|
| 0 | RESET | Reset all counters |
| 1 | AUTORESET | Enable counter auto-reset |
| 15–2 | | (not used) |

Table 3.15: Bit definition of the counters control register

> **Important:** Counters remain in reset state as long as the RESET bit is asserted!

**TODO: Check if it's true! By the way, this might be dangerous!**

### Gate mode register (0x0800+0x0002)

The gate mode register is used for configuring gates to direct mode; lsb corresponds to the gate 0. Bits 12 to 15 are not used. Note that switching a gate into direct mode is only possible when it's in IDLE state (see Table 3.13).

### Gate timeout block (0x0800+0x0004)

This block contains 12 registers for configuring individual timeouts for all gates. Gate timeouts are configured in steps of 131.072 µs. With 16-bit registers, the longest possible timeout is about 8.5 s.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | GATE0 | Timeout value for gate 0 |
| ... | | | |
| 0x000B | R/o | GATE11 | Timeout value for gate 11 |

Table 3.16: Gate timeout block layout

**Watchdog reset block (0x0800+0x0010)**

This block contains 4 registers for resetting watchdog errors for the 54 input watchdogs. Within each register, lsb corresponds to the channel with the lowest number. In case of IN48-53, 10 msbs are not used.

A watchdog error reset procedure requires first asserting and then de-asserting the relevant bit.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | IN0-15 | Watchdog reset signals for inputs 0 to 15 |
| 0x0000 | R/o | IN16-31 | Watchdog reset signals for inputs 16 to 31 |
| 0x0000 | R/o | IN32-47 | Watchdog reset signals for inputs 32 to 47 |
| 0x0000 | R/o | IN48-53 | Watchdog reset signals for inputs 48 to 53 |

Table 3.17: Watchdog reset block layout

### 3.2.7 Event status superblock (0x0900)

The event status superblock allows monitoring the state of the event receiver and the threshold loader.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | TAG_LOW | Last event tag low register |
| 0x0001 | R/o | TAG_HIGH | Last event tag high register |
| 0x0002 | R/o | CODE | Last accepted tag register |
| 0x0003 | R/o | THR_LOADER_STATE | Threshold loader state register |
| 0x0004 | R/o | THR_LOADER_ITER | Threshold loader iterator monitoring register |
| 0x0005 | R/o | THR_LOADER_ENQ_STATE | Threshold loader FIFO enqueue framework state register |

Table 3.18: Event status superblock layout. Base: 0x0900

#### Last event tag registers (0x0900+0x0000, 0x0900+0x0001)

The last event tag low and high registers show a full 32-bit value of the last received tag. All tags are reported if events are enabled, even these with bad key.

#### Last accepted tag register (0x0900+0x0002)

The last accepted tag register shows lower 16 bits (the code) of the last accepted tag. Tags with bad key are not reported.

#### Threshold loader state register (0x0900+0x0003)

This register shows the current state of the threshold loader state machine. It is used solely for debugging purposes.

| Value | Symbol | Description |
|---|---|---|
| 0x0 | IDLE | Idle, waiting for a trigger |
| 0x1 | WAIT1 | Wait cycle to adapt for memory latency |
| 0x2 | WAIT2 | Wait cycle to adapt for memory latency |
| 0x3 | READ | Writing a value to a register |
| 0x4 | FINISH | Intermediate state to finish the procedure |

Table 3.19: State IDs for threshold loader

Note that all states but IDLE are temporary.

#### Threshold loader iterator monitoring register (0x0900+0x0004)

The threshold loader cycles over all registers when reloading thresholds. This register shows the number of the last iterated counter. It is used solely for debugging purposes.

#### Threshold loader FIFO enqueue framework state register (0x0900+0x0005)

This register shows the current state of the FIFO enqueue framework state machine for the threshold loader. It is used solely for debugging purposes.

**TODO: Which value is for which source?**

### 3.2.8 Event control superblock (0x0A00)

The event control superblock allows configuring the event receiver. Further, it provides registers for manual triggering of the event-driven actions.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | KEY | Event key register |
| 0x0001 | R/W | CTRL | Event control register |
| 0x0002 | R/W | THR_RELOAD | Threshold reload register |
| 0x0003 | R/W | GATE_PREP | Gate prepare register |
| 0x0004 | R/W | GATE_RECOVER | Gate recovery register |

Table 3.20: Event control superblock layout. Base: 0x0A00

#### Event key register (0x0A00+0x0000)

This register defines the key for incoming events. For all incoming events, their 16 msbs are compared with the key and only events witch matching key are accepted.

#### Event control register (0x0A00+0x0001)

This register is used for enabling event-driven operation.

| Bits | Symbol | Description |
|---|---|---|
| 0 | EVT_ENABLE | Enable event-driven operation |
| 15–1 | | (not used) |

Table 3.21: Bit definition of the event control register

### Threshold reload register (0x0A00+0x0002)

This register allows manual operation of the threshold loader. It is an alternative for event-driven operation (see Section 3.3.2 for more details).

In order to reload given group of counters with given dataset, one needs to set up DATASET and GROUP and pulse TRIGGER. A transition of TRIGGER from 0 to 1 will trigger the threshold reloading procedure. Setting DATASET, GROUP and asserting TRIGGER can be done with a single write operation. All operations are queued in a 16-place FIFO buffer and executed as quickly as possible. Parallel register-driven and event-driven operation is possible and safe.

| Bits | Symbol | Description |
|---|---|---|
| 7–0 | DATASET | Dataset to be loaded |
| 11–8 | GROUP | Group of counters to be loaded to |
| 12 | TRIGGER | Threshold reload trigger |
| 15–13 | | (not used) |

Table 3.22: Bit definition of the threshold loader control register

### Gate prepare register (0x0A00+0x0003)

The gate prepare register is used for manual control of the prepare signal for all the gates; lsb corresponds to the gate 0. Bits 12 to 15 are not used.

This is an alternative for event-based control of prepare signals (see Section 3.3.2).

> **Important:** Gates will react on the rising slope of the prepare bit state. However, as long as certain bits in the register are set, the state machine will stop in WAIT state (see Figure 2.5) and prepare events for corresponding gates will be disregarded. Therefore, concurrent event-based and register-based operation is not recommended.

**TODO: Check if it's true!**

**Gate recovery register (0x0A00+0x0004)**

The gate recovery register is used to manually control error recovery for all the gates; lsb corresponds to gate 0. Bits 12 to 15 are not used. A gate recovery procedure requires first asserting and then deasserting the relevant bit.

This is an alternative for event-based control of gate error recovery (see Section 3.3.2).

### 3.2.9 Input matrix superblock (0x1000)

The input matrix register superblock is used to configure the input and gate patch matrices (see Section 2.3.2). It contains one R/W register per counter (a total of 128 registers).

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | CTR0 | Counter 0 input configuration register |
| ... | | | |
| 0x007F | R/W | CTR127 | Counter 127 input configuration register |

Table 3.23: Intput matrix configuration superblock layout. Base: 0x1000

| Bits | Symbol | Description |
|---|---|---|
| 5-0 | UP_SEL | Channel selection for upcounting input (see Table 3.25 for values) |
| 11-6 | DOWN_SEL | Channel selection for downcounting input (see Table 3.25 for values) |
| 15-12 | GATE_SEL | Channel selection for gate input (0–11) |

Table 3.24: Bit definition of input matrix superblock registers

| Value | Symbol | Description |
|---|---|---|
| 0–53 | INPUTnn | Signal inputs 0–53 |
| 54 | GND | No signal |
| 55 | TEST_10K | 10 kHz test signal |
| 56 | TEST_99K | 99 kHz test signal |
| 57 | TEST_100K | 100 kHz test signal |
| 58 | TEST_990K | 990 kHz test signal |
| 59 | TEST_1M | 1 MHz test signal |
| 60 | TEST_9M9 | 9.9 MHz test signal |
| 61 | TEST_10M | 10 MHz test signal |
| 62 | TEST_24M9 | 24.9 MHz test signal |
| 63 | TEST_25M | 25 MHz test signal |

Table 3.25: Up/downcounting input mapping for counters

### 3.2.10 Output matrix superblock (0x1100)

The output matrix superblock is used to configure the output match matrix (see Section 2.3.2).

This superblock contains a set of 6 sub-blocks, each 22 registers long. The blocks are placed with a raster of 32 addresses, as stated in Table 3.26. Each sub-block defines the configuration of one output, as shown in Table 3.27.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | OUT0 | Start of output 0 block |
| 0x0020 | R/W | OUT1 | Start of output 1 block |
| ... | | | |
| 0x00A0 | R/W | OUT5 | Start of output 5 block |

Table 3.26: Output matrix configuration superblock layout. Base: 0x1100

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | NEG_OVF | Start of the negative counter overflow selection subblock |
| 0x0008 | R/W | POS_OVF | Start of the positive counter overflow selection subblock |
| 0x0010 | R/W | GATE_ERR | Gate error selection register |
| 0x0011 | R/W | WD_ERR | Start of the watchdog error selection subblock |
| 0x0015 | R/W | GATE_IN_ FORWARD | Gate input forward selection register |

Table 3.27: Output matrix configuration block for a single output.
Base: 0x1100+$n$·0x0020

### Counter overflow selection subblocks (0x1100+$n$·0x0020+0x0000, 0x1100+$n$·0x0020+0x0008)

These two identical subblocks allow selecting which counters will forward their negative and positive counter overflow signal to the given output. There is one bit per counter; for each register lsb always corresponds to the counter with lowest number.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | CTR0-15 | Counters 0–15 |
| ... | | | |
| 0x0007 | R/W | CTR112-127 | Counters 112–127 |

Table 3.28: Counter overflow selection subblocks layout.
Base: 0x1100+$n$·0x0020+0x0000, 0x1100+$n$·0x0020+0x0008

### Gate error selection register (0x1100+$n$·0x0020+0x0010)

This register allows selecting which gates will forward their error signal to the given output. There is one bit per gate; lsb corresponds to gate 0. Bits 12 to 15 are not used.

### Watchdog error selection subblock (0x1100+$n$·0x0020+0x0011)

This subblock allows selecting which input watchdogs will forward their error signal to the given output. There is one bit per input; for each register lsb always corresponds to the input with lowest number.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | IN0-15 | Inputs 0–15 |
| 0x0001 | R/W | IN16-31 | Inputs 16–31 |
| 0x0002 | R/W | IN32-47 | Inputs 32–47 |
| 0x0003 | R/W | IN48-53 | Inputs 48–53 (bits 6–15 unused) |

Table 3.29: Watchdog error selection subblocks layout.
Base: 0x1100+...+0x0011

### Gate input forward selection register (0x1100+$n$·0x0020+0x0015)

This register allows selecting which gate insputs will be forwarded to the given output. There is one bit per gate; lsb corresponds to gate 0. Bits 12 to 15 are not used.

## 3.2.11 Counter readout superblock (0x1200)

The counter readout register superblock contains 128 two-register blocks to read out current count value. Table 3.31 shows a layout of a single block.

The two registers form a 32-bit U2 signed integer which is the count value.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | CTR0 | Counter 0 readout block |
| 0x0002 | R/o | CTR1 | Counter 1 readout block |
| ... | | | |
| 0x00FE | R/o | CTR127 | Counter 127 readout block |

Table 3.30: Counter readout superblock layout. Base: 0x1200

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | LOW | lsbs of the count value |
| 0x0001 | R/o | HIGH | msbs of the count value |

Table 3.31: Counter readout block layout for a single counter.
Base: 0x1200+$n$·0x0002

### 3.2.12 Counter group assignment superblock (0x1600)

The counter group assignment superblock contains one R/W register per four counters (a total of 32 registers) for assigning counters to groups. Group numbers are four bits wide (0 to 15).

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | CTR0-3_GROUP | Group assignment for counter 0–3 |
| ... | | | |
| 0x001F | R/W | CTR124-127_GROUP | Group assignment for counters 124–127 |

Table 3.32: Counter group assignment superblock layout. Base: 0x1600

| Bits | Symbol | Description |
|---|---|---|
| 3–0 | CTR0 | Group selection for counter 4·N+0 |
| 7–4 | CTR1 | Group selection for counter 4·N+1 |
| 11–8 | CTR2 | Group selection for counter 4·N+2 |
| 15–12 | CTR3 | Group selection for counter 4·N+3 |

Table 3.33: Bit mapping of group assignment register N

### 3.2.13 Threshold readout superblock (0x1800)

The threshold readout register superblock contains 128 four-register blocks to read out currently programmed threshold values. Table 3.34 shows a layout of a single block.

Register pairs form 32-bit U2 signed integers which are the threshold values. Note that for correct operation, the positive threshold *must* be a positive number and the negative threshold *must* be a negative number.

Threshold programming is possible only via threshold memory (see Section 3.2.14).

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | CTR0 | Counter 0 threshold readout block |
| 0x0004 | R/o | CTR1 | Counter 1 threshold readout block |
| ... | | | |
| 0x01FC | R/o | CTR127 | Counter 127 threshold readout block |

Table 3.34: Counter readout superblock layout. Base: 0x1800

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/o | POS_THR_LOW | lsbs of the positive threshold |
| 0x0001 | R/o | POS_THR_HIGH | msbs of the positive threshold |
| 0x0002 | R/o | NEG_THR_LOW | lsbs of the negative threshold |
| 0x0003 | R/o | NEG_THR_HIGH | msbs of the negative threshold |

Table 3.35: Threshold readout block layout for a single counter.
Base: 0x1800+$n$·0x0004

### 3.2.14 Threshold memory (0x8000)

Threshold memory is used for providing threshold data. It's mapped into the device's register addres space. It contains space for 32 complete datasets. The datasets are stored as consecutive blocks. The data format for a single dataset is equal with the data format for threshold readout superblock (see Table 3.34).

The threshold memory is freely readable and writable by user. Two-port Random Access Memory (RAM) architecture allows user operations even while threshold loading with no dangerous side effects.

> **Comment:** An intermittent data inconsistency is possible if a dataset is written during threshold loading with the same dataset. In this case, partly new and partly old data can be loaded.

| Word offset | R/W | Symbol | Description |
|---|---|---|---|
| 0x0000 | R/W | RAM_DATASET0 | Thresholds for dataset 0 as described in Table 3.34 |
| ... | | | |
| 0x3E00 | R/W | RAM_DATASET31 | Thresholds for dataset 31 as described in Table 3.34 |

Table 3.36: Threshold memory layout. Base: 0x8000

## 3.3 Event decoding

### 3.3.1 Event tag

Every event tag contains three main parts, as stated in Table 3.37: *event key*, *event command* and *event parameter*.

| Bits | Symbol | Description |
|---|---|---|
| 11-0 | EV_PARAM | Event parameter |
| 15-12 | EV_CMD | Event command |
| 31-16 | EV_KEY | Event key |

Table 3.37: Bit mapping for an event tag

#### Event key

Event key is used to filter event tags. Only these tags are accepted, where event key matches the setting stored in EVT_CTRL.KEY register (see Section 3.2.8.)

#### Event command

Event command defines the action to be executed. A full list of commands is presented in Table 3.38. See Section 3.3.2 for more detailed description of each command.

| Value | Symbol | Description |
|---|---|---|
| 0x0 | EVT_CMD_NOP | Do nothing |
| 0x1 | EVT_CMD_RELOAD_THR | Reload thresholds |
| 0x2 | EVT_CMD_PREPARE | Prepare gates |
| 0x3 | EVT_CMD_RECOVER | Recover gates |
| 0x4 | EVT_CMD_RESET_CTR | Reset all counters |
| 0x5–0xD | | Reserved for future use |
| 0xE–0xF | | Reserved for development and debugging |

Table 3.38: Event command numbers

#### Event parameter

Event parameter contains supplementary information. Its interpretation depends on the command. See Section 3.3.2 for details.

### 3.3.2 Event commands

#### Do nothing

This command is guaranteed to do nothing in the current and future firmware releases.

| Bits | Symbol | Description |
|---|---|---|
| 11–0 | | (not used) |

Table 3.39: Parameter bits for 'Do nothing' command

#### Reload thresholds

Perform threshold reload for given group of counters and with given dataset. Group and dataset numbers are provided via event parameter as shown in Table 3.40.

This command has the same effect as using EVT_CTRL.THR_RELOAD register (see Section 3.2.8).

| Bits | Symbol | Description |
|---|---|---|
| 7–0 | DATASET | Dataset to be loaded |
| 11–8 | GROUP | Group of counters to be loaded to |

Table 3.40: Parameter bits for 'Reload thresholds' command

#### Prepare gates

Send prepare signal to selected gates. Gates to be affected are selected by event parameter as shown in Table 3.41.

This command has the same effect as using EVT_CTRL.GATE_PREP register (see Section 3.2.8).

| Bits | Symbol | Description |
|---|---|---|
| 0 | GATE0 | Apply for gate 0 |
| ... | | |
| 11 | GATE11 | Apply for gate 11 |

Table 3.41: Parameter bits for 'Prepare gates' command

#### Recover gates

Recover selected gates from error state. Gates to be affected are selected by event parameter as shown in Table 3.42.

This command has the same effect as using EVT_CTRL.GATE_RECOVER register (see Section 3.2.8).

| Bits | Symbol | Description |
|---|---|---|
| 0 | GATE0 | Apply for gate 0 |
| ... | | |
| 11 | GATE11 | Apply for gate 11 |

Table 3.42: Parameter bits for 'Recover gates' command

#### Reset all counters

Set all counters to 0. This command has the same effect as using CTRL.COUNTERS:RESET bit (see Section 3.2.6).

| Bits | Symbol | Description |
|---|---|---|
| 11–0 | | (not used) |

Table 3.43: Parameter bits for 'Reset all counters' command

# Index

BLM crates, 6
Constraints, 2
Indexed terms, 2
Version numbers, 3
Bit groups, 13
Bits, 13
Blocks, 13
Counter pool, 8
Dataset, 9
Direct gate mode, 10
Event command, 32
Event key, 32
Event parameter, 32
Event tag, 12
Event, 12
Gate inputs, 6
Gate logic, 8
Gate patch matrix, 8
Groups, 9
Input patch matrix, 8
Input watchdogs, 8
Interlock outputs, 6
Negative threshold, 8
Output patch matrix, 8
Overflow signal, 8
Positive threshold, 7
Prepare signal, 10
Registers, 13
Reserved terms, 2
Subblocks, 13
Superblocks, 13
Threshold laoder, 8
Threshold memory, 8
Threshold reload, 10
Up/down counter, 7

# All musts

definition of must, 3
definition of must not, 3

negative threshold values, 30

positive threshold values, 30

register names, 13

# Todos

Check if it's true, 25

By the way, this might be dangerous, 21

Version numbers, 3

Which value is for which source?, 24

# Acronyms

**BLM** Beam Loss Monitor

**DIOB** Digital I/O Board

**ECA** Event-Condition-Action

**FBAS** Fast Beam Abort System

**FIFO** First-In-First-Out

**I/O** input/output

**LED** Light-Emitting Diode

**lsb** least significant bit

**msb** most significant bit

**RAM** Random Access Memory

**SCU** Scalable Control Unit

**TTL** Trasnsistor-Transistor Logic

**U2** two's complement code

# List of Figures


2.1 A possible system layout . . . . . . . . . . . . . . . . . . . . . . . 5
2.2 The BLM crate . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
2.3 A single up/down counter . . . . . . . . . . . . . . . . . . . . . . 7
2.4 Block diagram of the device . . . . . . . . . . . . . . . . . . . . . 9
2.5 State diagram of the gate logic. Green arrows show the normal operation sequence . . . . . . . . . . . . . . . . . . . . . . . . . 11

# List of Tables


1.1 Summary of styles . . . 2
1.2 Constraints and reserved terms . . . 3

2.1 I/O module types used . . . 7

3.1 General layout of register blocks . . . 14
3.2 DIOB configuration and status superblock layout. Base: 0x0630 15
3.3 Bit definition of IO backplane mask registers . . . 15
3.4 DIOB configuration and status superblock layout. Base: 0x0638 15
3.5 Bit definition of IO backplane ID registers . . . 16
3.6 IDs for used I/O module types . . . 16
3.7 Status superblock layout. base: 0x0700 . . . 17
3.8 Negative overflow block layout. Base: 0x0700+0x0000 . . . 17
3.9 Positive overflow block layout. Base: 0x0700+0x0008 . . . 18
3.10 Watchdog error block layout. Base: 0x0700+0x0011 . . . 18
3.11 Gate state monitoring block layout. Base: 0x0700+0x001A . . 19
3.12 Bit definition of gate state monitoring registers . . . 19
3.13 State IDs for gate logic. See Figure 2.5 for state explanation. . 20
3.14 Control superblock layout. Base: 0x0800 . . . 20
3.15 Bit definition of the counters control register . . . 21
3.16 Gate timeout block layout . . . 22
3.17 Watchdog reset block layout . . . 22
3.18 Event status superblock layout. Base: 0x0900 . . . 23
3.19 State IDs for threshold loader . . . 23
3.20 Event control superblock layout. Base: 0x0A00 . . . 24
3.21 Bit definition of the event control register . . . 25
3.22 Bit definition of the threshold loader control register . . . 25
3.23 Intput matrix configuration superblock layout. Base: 0x1000 . 26
3.24 Bit definition of input matrix superblock registers . . . 26
3.25 Up/downcounting input mapping for counters . . . 27
3.26 Output matrix configuration superblock layout. Base: 0x1100 27

3.27 Output matrix configuration block for a single output.
Base: 0x1100+$n$·0x0020 . . . . . . . . . . . . . . . . . . . . . . 28
3.28 Counter overflow selection subblocks layout.
Base: 0x1100+$n$·0x0020+0x0000, 0x1100+$n$·0x0020+0x0008 . 28
3.29 Watchdog error selection subblocks layout.
Base: 0x1100+...+0x0011 . . . . . . . . . . . . . . . . . . . . . 29
3.30 Counter readout superblock layout. Base: 0x1200 . . . . . . . 29
3.31 Counter readout block layout for a single counter.
Base: 0x1200+$n$·0x0002 . . . . . . . . . . . . . . . . . . . . . . 30
3.32 Counter group assignment superblock layout. Base: 0x1600 . . 30
3.33 Bit mapping of group assignment register N . . . . . . . . . . 30
3.34 Counter readout superblock layout. Base: 0x1800 . . . . . . . 31
3.35 Threshold readout block layout for a single counter.
Base: 0x1800+$n$·0x0004 . . . . . . . . . . . . . . . . . . . . . . 31
3.36 Threshold memory layout. Base: 0x8000 . . . . . . . . . . . . 32
3.37 Bit mapping for an event tag . . . . . . . . . . . . . . . . . . . 32
3.38 Event command numbers . . . . . . . . . . . . . . . . . . . . . . 33
3.39 Parameter bits for 'Do nothing' command . . . . . . . . . . . 33
3.40 Parameter bits for 'Reload thresholds' command . . . . . . . . 34
3.41 Parameter bits for 'Prepare gates' command . . . . . . . . . . 34
3.42 Parameter bits for 'Recover gates' command . . . . . . . . . . 34
3.43 Parameter bits for 'Reset all counters' command . . . . . . . . 35

# Bibliography