RF Interlock Event Losses

Marek Penno

2014-01-20

Problem

- Interlock event data is lost with a certain probability
- When an RF event happens, it might happen that the real cause is unknown due to the event data loss
- Thats bad… physic‘s unable to determine real reason of RF event

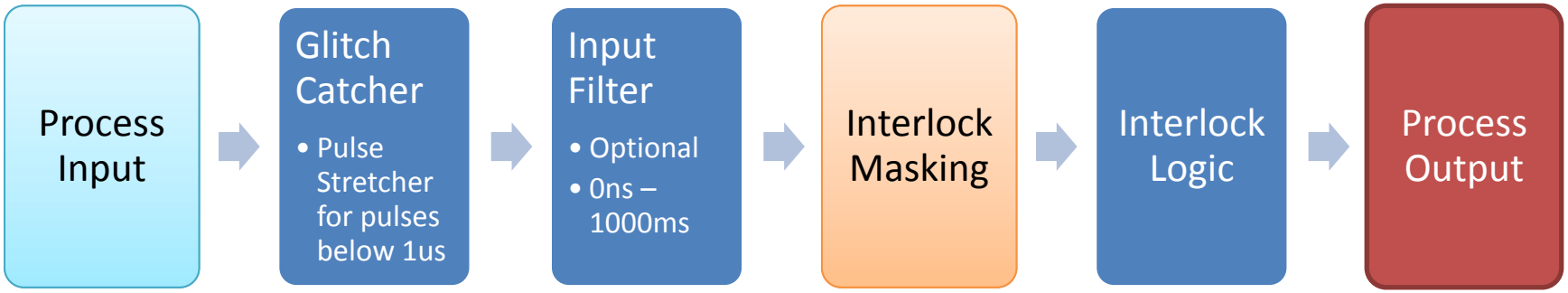
Possible causes

- Hardware problem?
- Firmware problem?
- Software problem?

Possible causes

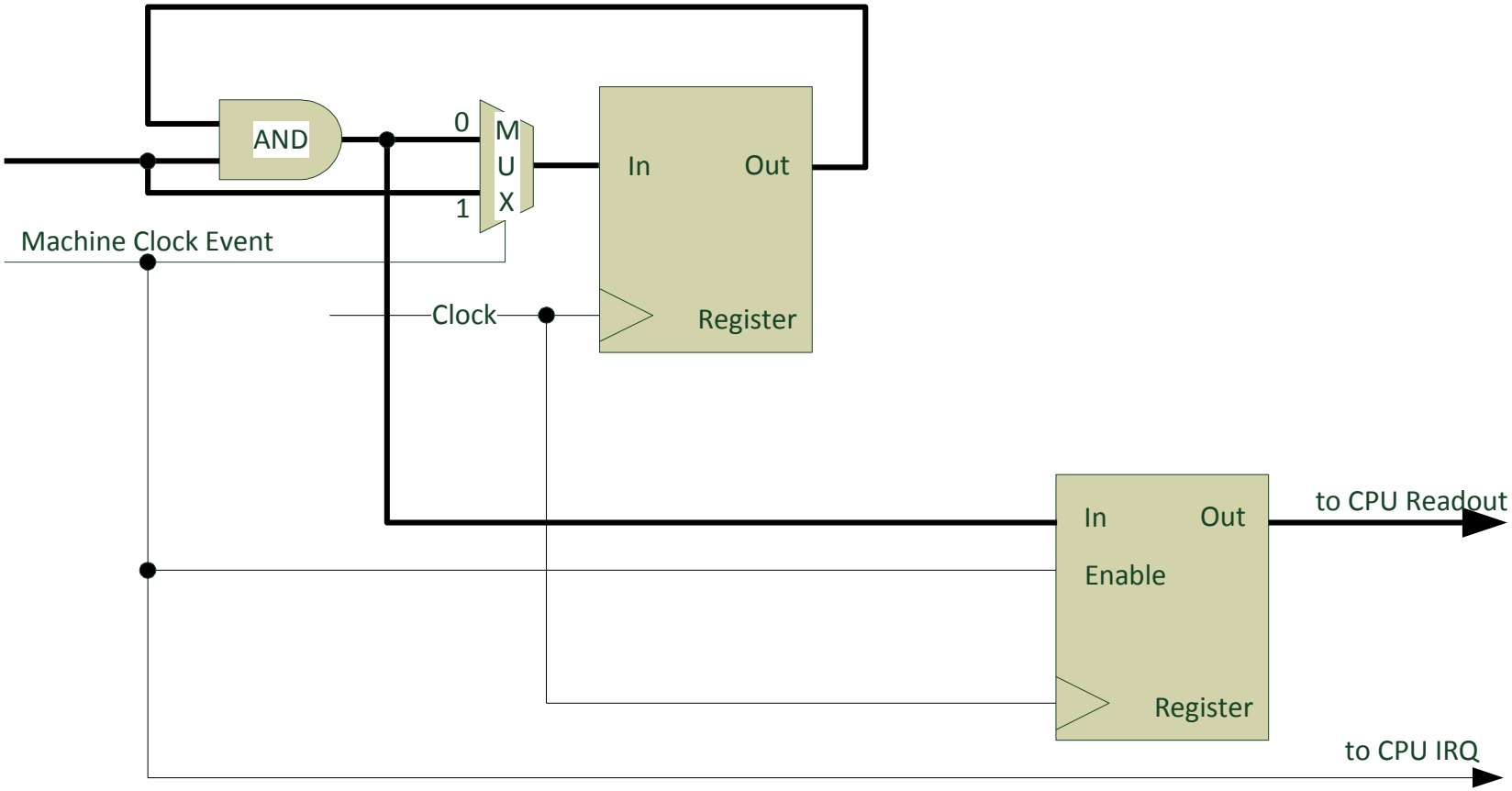
- ~~Hardware problem~~
  - Event whouldn‘t have been detected
- **Firmware problem**
  - Events lost because input pulses to short for readout but long enought for logic?
  - Capture logic for events not working correctly?
- Software problem?

Firmware Signal Processing



- Chain reaction time
  - for fast Signals (Light IO): 0.25us
  - for slow Signals (2us)

Firmware Readout Principle



- Tested and verified design

Possible causes

- ~~Hardware problem~~
- ~~Firmware problem~~
- **Software problem**
  - Sender problem?
  - Receiver problem?

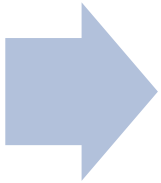
Network Protocol Versions

- **Protocol Vers.1 (Karen)**
  - UDP based, fixed format
  - Event data summed up to 1 sec.
  - Used at RF1/RF2, Interlock Rev. 3
- Protocol Vers.2 (Stefan Weisse)
  - Namely „Network Queue“
  - UDP based, loss detection, flexible extensible format
  - Event data summed up to 1 sec.
  - Used at GUN interlock, Interlock Rev. 3

Software Readout Principle

IRQ function

- Called at every machine clock
- Reads out hardware data
- Tags data with event number
- Writes data into event queue
- Optimized for speed (readout time <20T cpu clock cycles = 0.04% deadtime)



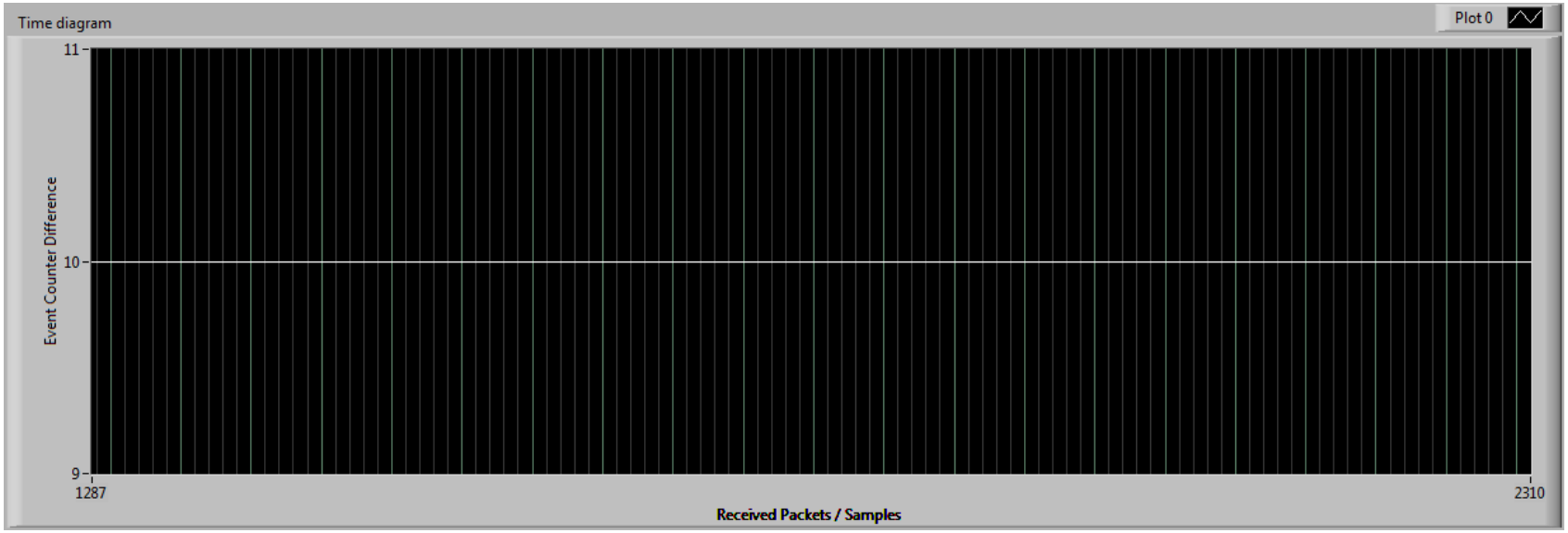
DAQ Listener Protocol Ver.1

- Reads data from queue
- Collects N events and sums them up into 1sec. Data
- Sends UDP packet to control system

Sender problem?

- Data is not send or is lost on the network
- Using a test receiver, that records the difference between the event numbers
- Event number should count up with a constant delta (= reprate)

Sender problem?

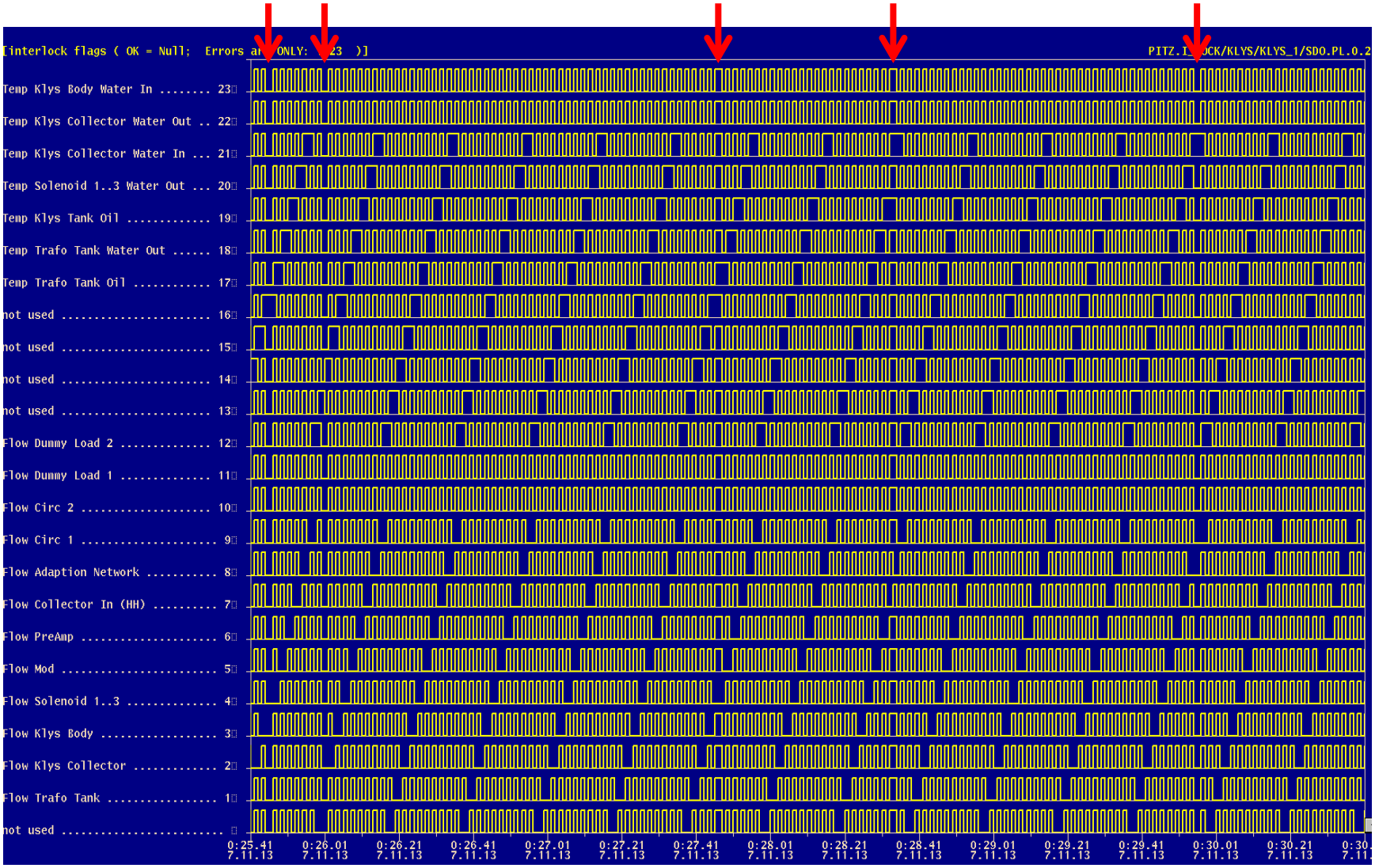


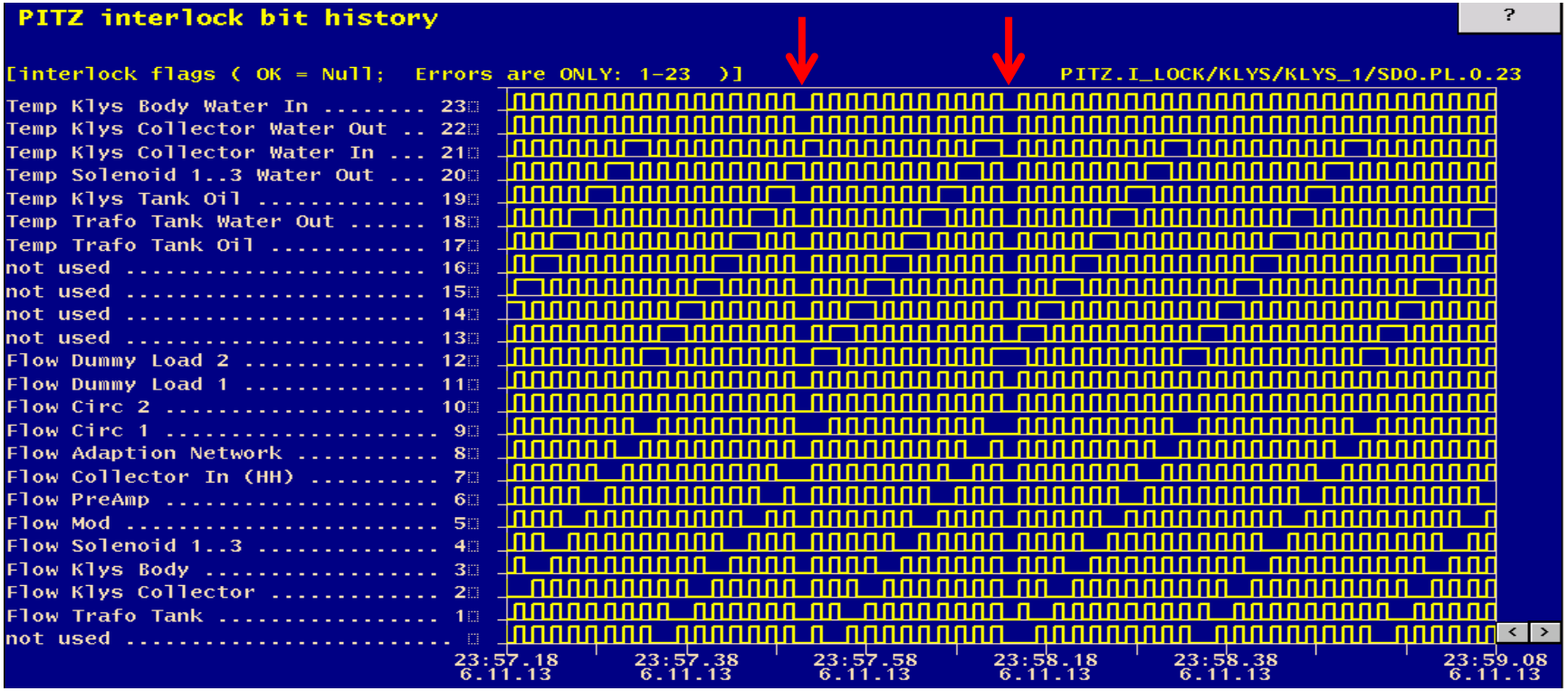
- Test over 1000 seconds
- constant delta = 10

Receiver problem?

- Testing receiver with test pattern generator
- Sending test pattern, alternating bits with periodic ‘holes’
- Visual check for “jumps” in pattern

- Discovering „gaps“ in the pattern
- Probability: ~ 1.8% (14 – 15 out of ~840 events)





- When a „Jump“ happen the last event data is repeated
  - Conclusion: Data was not updated in time
  - Idea: caused by internal periodic update method in the DOOCS server , which is not called perfectly periodically but with some jitter

Receiver problem?

- Looking at the source code…
- DOOCS Server receives data in thread A and writes data into buffer at 1 sec period
- DOOCS update function is periodically called by thread B at 1 sec. period and reads data from buffer
- Could work if updates periods are perfectly constant, but update periods do have jitter because of:
  - Network delays (few ms)
  - Operation System scheduler adds jitter to sleep function calls (>10ms)

Solutions?

- Fix server?
  - Using server-locks for updating inside of udp thread
  - A bit change of concept of the server… coding style… I whould like to reduce types of servers
- Change to Protocol Ver.2?

- Change to Interlock 4 Protocol?
  - Needs some software work on Interlock 3
  - Long term solution

Interlock 4 protocol

- Basic DOOCS server exists already
- Features:
  - TCP based communication, ZMQ extension in mind
  - Fully generic, adapts to any interlock configuration
  - Configurable signal arrangement at control system view
  - Update data by using server locks
  - Processes data on machine clock level
  - Transfers analog data and plots (if available)
  - Transfers more metadata (masks, filters, min/max thresholds, signal names)
  - Support for full event history (advanced archiver)