Calibration Interface Board Status Update

Nuno Barros

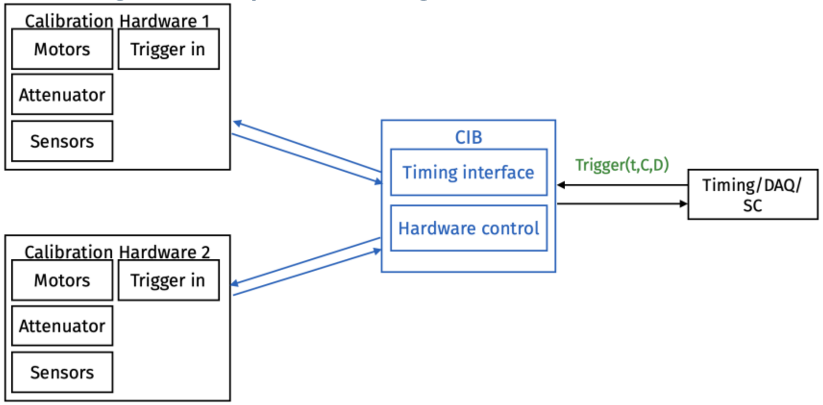
CALCI Meeting

April 2023

LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS

Calibration Interface Board (CIB)

- **Electronics board with FPGA/CPU to interface Control, DAQ and the IoLaser subcomponents**
  - Similar principle as ProtoDUNE-SP I trigger system (CTB)
  - SoC implementing FPGA fast logic and CPU to implement high level processing and ethernet communication with DAQ/SC



- **Implemented interfaces**
  - **Timing (FPGA)**
    - Receive clock, commands and send firing time
  - **Control (CPU/FPGA)**
    - Interface between Slow Control and different subcomponents
    - Communication with motors/encoders (through ethernet)
    - Communication with laser components
  - **DAQ**
    - Receive “fire allowed” signal through the timing system
    - Send firing time
    - Send laser position/direction

CIB Interfaces

- Laser interfaces:
  - External trigger for laser firing (DAT-Fire, DAT-Qswitch)
  - Photodiode input : Signal input + I^2C configurable discriminator
  - Shutter control output
  - Calibration laser output (5V)
  - Fire enable input (linear stage)
  - FSYNC, VSYNC (in principle won't be used)
  - Serial connection through USB/UART for laser configuration

- Motor Interfaces:
  - Motor phase inputs (2 phases per motor, max of 3 motors per periscope)
  - Configuration through software interface

- Timing interface
  - SFP/Fiber
  - Both new and old protocols (with CDR chip) in place (new protocol enabled by default)

- Spare I/Os:
  - 10 isolated (via optocoupler) inputs
  - 10 non-isolated, configurable input/outputs (3.3 or 5 V)

https://docs.google.com/spreadsheets/d/1IhIPEH3469wjSd4iTEx19AlFCEf5dIkC/edit#gid=404624908

IOLaser operation

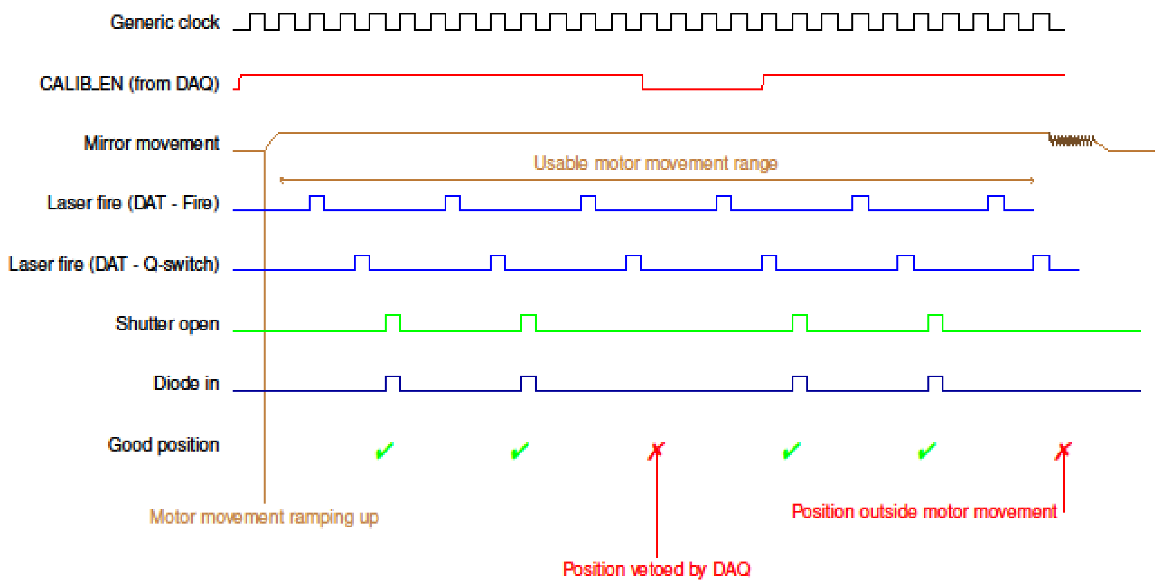
- Laser composed of a series of (mostly slow) components that must work in sync
  - Each has its own constraints that have to be handled in sync
  - For most the communication is done through serial (unclocked) connection

- **Laser system:**
  - Due to stability the laser must fire at 10 Hz
    - Can be controlled through serial connection (internal trigger), or **external trigger (DAT mode)**
- **Periscope motors**
  - 2 or 3 motors (depending on periscope model) control the direction of the mirrors
  - The movement profile is configurable (acceleration, deceleration, speed, target position)
  - There's backlash when the motors stop, so discrete movement is not desirable
    - instead do a raster of segments
- **Fast shutter**
  - Placed after the laser, blocking the light from reaching the cryostat
    - Shutter must be open when the laser is firing on an intended position, closed otherwise

IOLaser operation

- Since both laser and motor have operation constraints, all parts need to be controlled centrally
  - This is the job of the CIB
- The operation plan is as follows:
  1. In the control software choose the starting laser firing position and destination of the mirror
  2. The information is passed to the CIB that orders the mirrors to place themselves at the beginning of the segment
     - Note that the beginning of the segment should be before the start laser position, to account for the motor acceleration profile
  3. Initiate the motor movement (the CIB is counting the motor steps)
  4. Once the motor step count reaches the start laser firing position, initiate the laser firing sequence
     - DAT- fire signal (~180 μs before the lasing) - can tune through configuration
     - DAT-Qswitch (~170 ns before laser) - can tune through configuration
     - Open shutter (if DAQ allows calibration, and limit switches are not enabled)
  5. Receive the photodiode signal (confirming light injection in the cryostat) and timestamp it
  6. Send timing word to DAQ and record position (in step counts of all motors)
  7. Count 100 ms and repeat 4-6

IOLaser operation

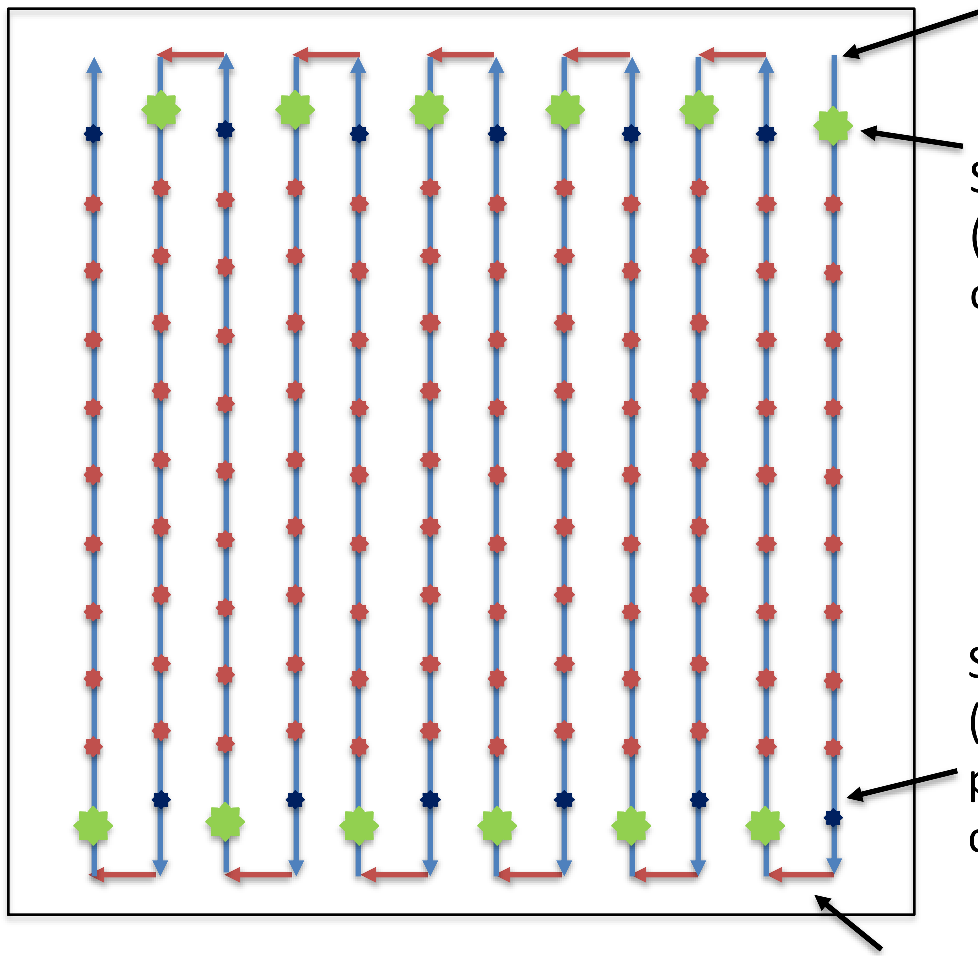


- There are situations where we may need to stop injecting light in the middle of a movement segment
  - DAQ revoke "authorization"
  - Mirror pointing to torlon
  - The laser is pointing to a sensitive (red) area
- The laser system keeps firing, but the shutter remains closed

IOLaser operation for a scan

Laser firing at 10 Hz (position determined from motor speed and starting firing position)

Start motor movement



Start laser firing (once motor is at constant speed)

Stop laser firing (computed from start position and motor configuration)

Mirror repositioning

Status of the CIB (HW/FW)

- Hardware:
  - All hardware components have been procured
  - PCB production and assembly ongoing
    - Should be ready within the month
- Firmware:
  - All major components have been implemented (Gil Madeira)
  - All modules have been tested individually
    - Currently assembling integrated design to test (simulated) full laser operation
- Software:
  - Currently in active development

Status of the CIB (SW)

- The CIB software is comprised of three loosely interconnected modules:
  - Slow Control Server
    - Responsible for configuration and control of all hardware
    - Is the core of the software infrastructure (everything else are subprocesses of this one)
    - **This is the only interface visible from outside**
  - DAQ interface + FPGA configuration (CPU/FPGA communication)
    - Most of the code in place (recycled from CTB)
    - Responsible for communicating back the fired directions back to the DAQ (to assemble the trigger primitives)
    - Responsible for loading the configuration parameters into the FPGA IPs
  - Hardware interfaces
    - Motors, laser, attenuator, power meter
      - These have already been developed by LIP and LANL
    - In direct communication with the Slow Control server (where the configurations are)

Why a SC server to operate the laser?

- Mix of answering concerns by the slow control group and optimisation of manpower
  - SC are not very keen on ad-hoc systems that require “a specific computer and specific software to something if there is trouble”
    - In a way it makes sense: there were bad experiences in the past

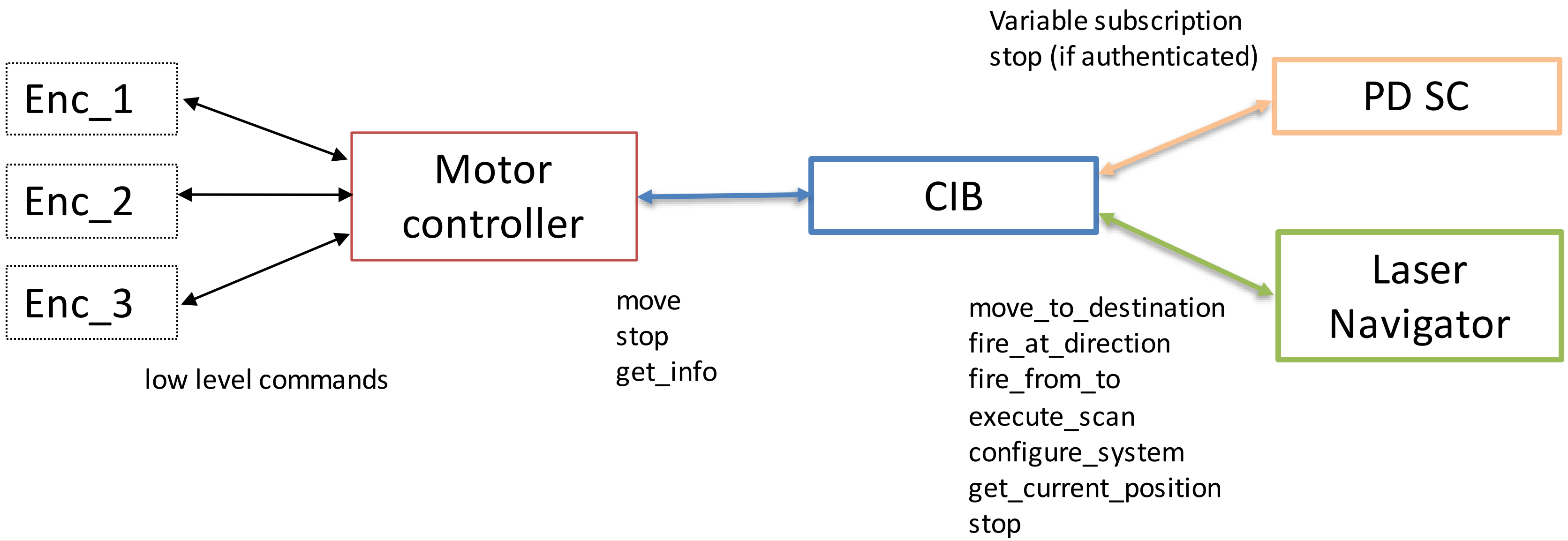
- Instead of implementing one interface for slow control (slow control server) and another to operate the laser system as a whole, implement a single OPC-UA server
  - Basically PD slow control central system (PD-SC) and the navigator are just two slow control clients talking to the CIB
  - By default the PD-SC only monitors (reads) the parameters served by the CIB
  - In order to operate the IoLaser you need to authenticate

CIB Software Status

- Discussions in Coimbra about interface and operation of different “modules”
  - Several small details that need to be considered
  - Overall operation is relatively straightforward, but the devil hides in the details:
    - Moving more than one motor (while lasing) at the same time is possible, but it is a lot of trouble to be done right
      - Both motor speed, acceleration and synchronized start of movement have to be calculated and set precisely to avoid missing target 3D direction
    - Operation conditions, who does what, overriding logic
  - Integration tests

IoLaser Control

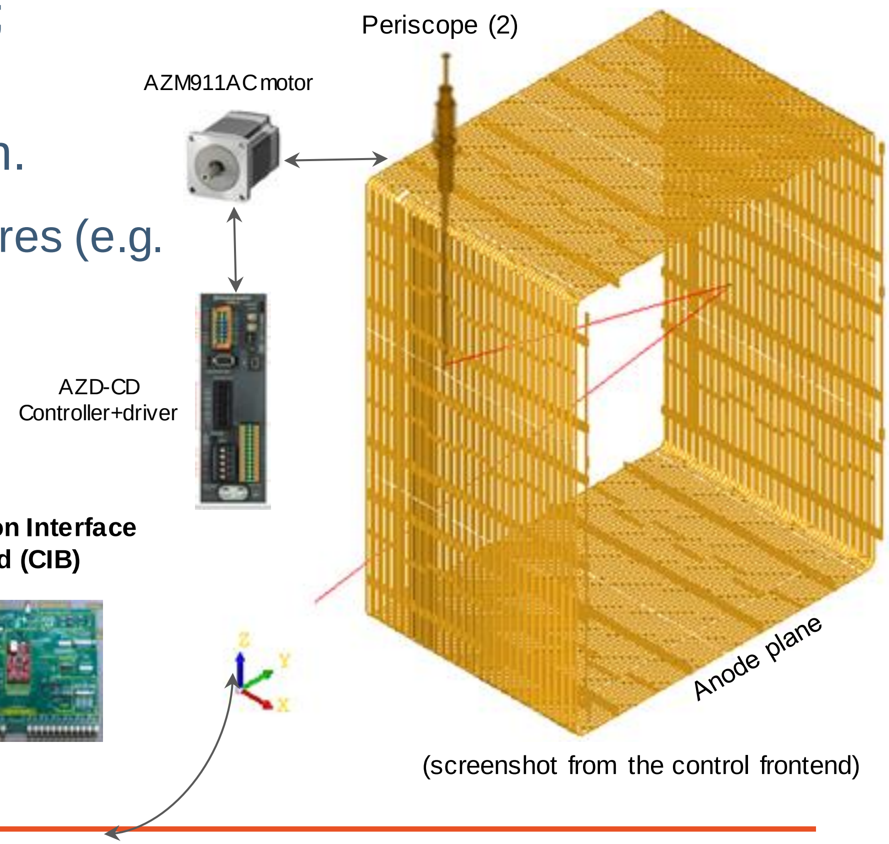
- Three modules discussed:
  - Laser navigator (FN) - GUI with operation functionality and laser “simulation”
  - CIB quasar server (NB) - Slow control server that interfaces all the hardware parts
  - Encoder controller (VS) - Motor control and monitoring software running on a RPi and implementing the low level communication with all the periscope motors



Control software: user interface

- Frontend to operate/control the periscope and navigate the laser beam through the FC:
  - Uses directly the existing CAD drawings;
    - Based on OpenCascade (CE) libraries;
  - Able to identify the part being hit and determine the reflected beam direction.
    - Critical to avoid hitting sensitive structures (e.g. PDS)

Francisco Neves



(screenshot from the control frontend)

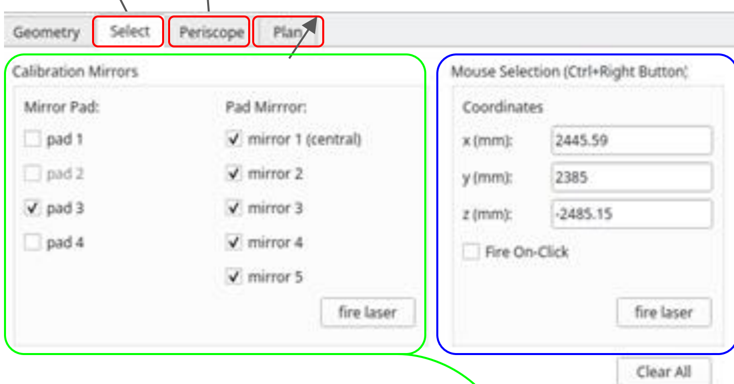
Control software: user interface

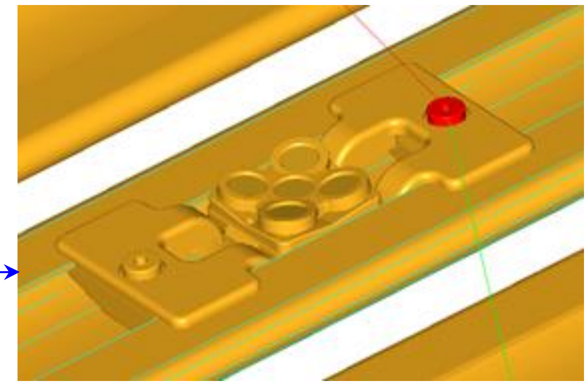
Francisco Neves

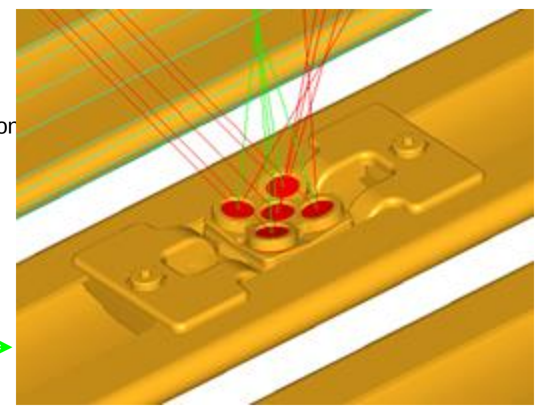
Select mirrors or target points using the mouse

Actuate directly the **periscope** motors (angles/motor steps) - *ongoing!*

Plan a calibration sequence by moving the periscope and firing the laser at predefined positions - *ongoing!*







Note: firing the laser will be interlocked/managed with the CIBs for security!

(screenshot from the control frontend)

Caveats raised in the discussions

- **Motors:**
  - We do not want to move more than one motor at the same time
    - Although multiple motor operation is possible, it is substantially harder to implement
- **Shutter:**
  - Need to check whether the shutter should be open and closed for each position, or remain open for a full segment (and closed in case the enable is lost)
    - From manual both ways are viable.
    - First option provides more security (no light entering the cryostat except when we explicitly open the shutter
- **Attenuator:**
  - Can accommodate changing settings **occasionally**
    - Example: at the end of a segment, after pausing operation
  - Changing settings on a per-position basis is not recommended
    - Logic complicates substantially
    - Timing constraints even more strict for a three-way synchronized operation (motor, laser and attenuator)

CIB Software Status

- **Collection of SC variables and methods available:**
  - https://docs.google.com/spreadsheets/d/1IhIPEH3469wjSd4iTEx19AlFCEf5dIkC/edit?usp=sharing&ouid=114399830641573474222&rtpof=true&sd=true
  - As of now:
    - ~40 variables
    - ~10 methods
    - More will certainly be added as integration of the different subsystems proceeds
- Motor control and monitoring interface fully implemented
  - Tested successfully last week at Coimbra
  - Moved, stopped, resumed, monitored status through a SC client talking to CIB software
- Example client code also implemented
  - Set up a dummy server running in simulation mode
  - Pass the code to Francisco to integrate into the laser navigator
- Currently implementing high-level laser operation functions
  - Fire at a single point (simple)
  - Fire at a segment (also simple)
  - Full scan (trickier)
    - This requires back and forth communication with the FPGA side to know when to load the next segment to execute

High level operation functions

- **configure_system**
  - submit a json object that configures the scan-wide parameters on the whole laser system (laser, attenuator, motor settings like acceleration, etc)
- **fire_at_point**
  - Move the motors so that the mirror points in a set direction and fire a single (or N) pulses
- **fire_segment**
  - start firing at a point $\mathbf{p_i}$, and keep firing (at 10 Hz) until you pass another point $\mathbf{p_f}$
  - A couple of caveats:
    - $\mathbf{p_i}$ and must $\mathbf{p_f}$ lie in the same segment <u>with only the mirror moving</u>
    - Motor must initially point to some point prior to be able to ramp up speed so that it is at cruising speed (i.e., speed set point) when it reaches $\mathbf{p_i}$
      - This may not be needed, if the only moving motor is the mirror (speed can be ramped up pretty much instantly, since the mirror has low inertia)
    - We may not fire at $\mathbf{p_f}$ (it depends on the speed of the motor)
      - But certainly we won't fire *past* $\mathbf{p_f}$

High level operation functions

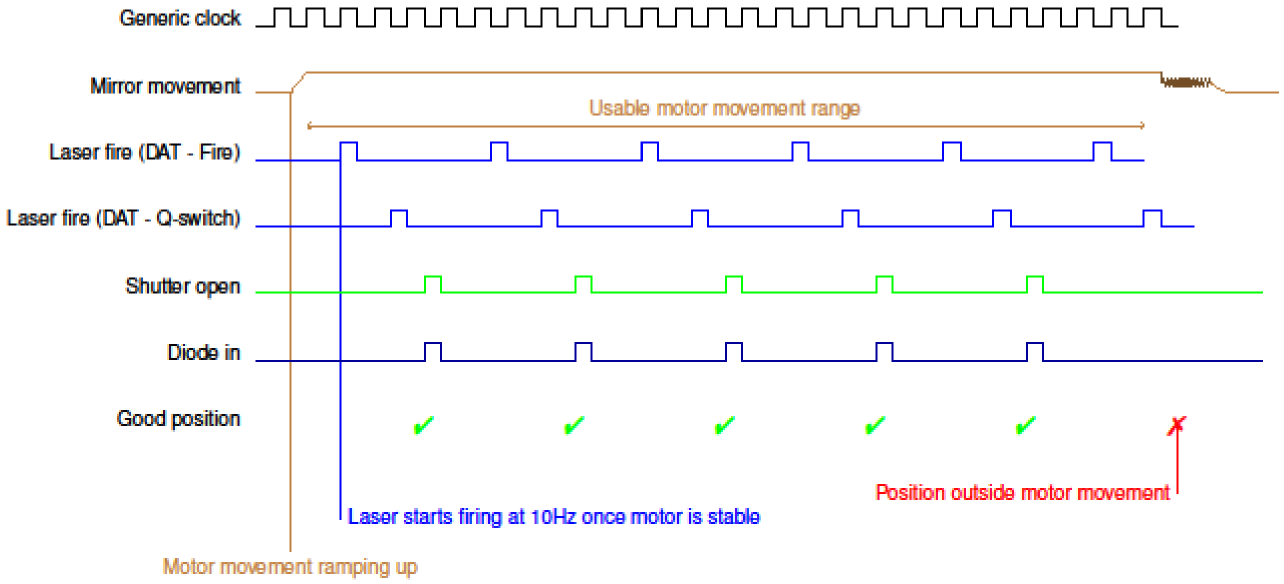
- **Execute_scan**
  - submit a json object that specifies sequences of segments to fire
  - Internally will sequentialize calls to fire_segment with relocation movements in between with the shutter closed

Next steps

- Prepare for QA/QC CIB once it is ready (NB)
- Test fully assembled CIB firmware (GM)
- Integrate the other IoLaser control software components (laser, piezo-actuator, attenuator) into the CIB server (NB, LANL)
- Implement the configuration IP in the FPGA (NB, GM)
- Integrate the slow control client example into the navigator (FN)
- Add some extra monitoring quantities to the motor communication interface (VS)
    - RPi load, memory usage, ?

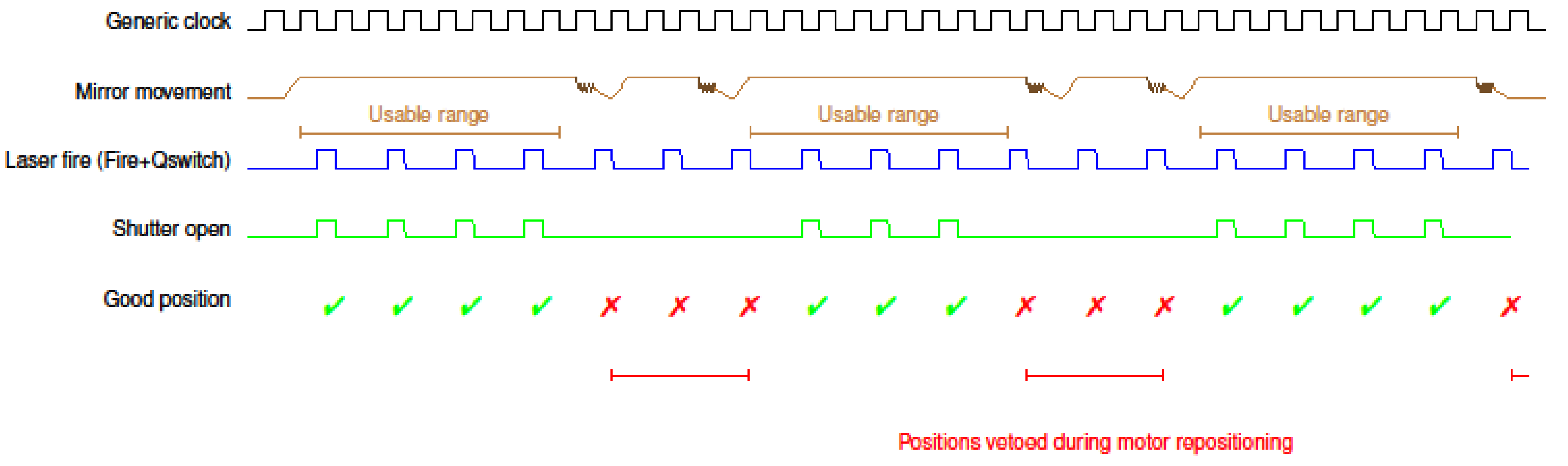
Backup Slides

IOLaser (normal) operation



- Note that we need to stop firing before the motor stops, as there seems to be some backlash in the motor movement

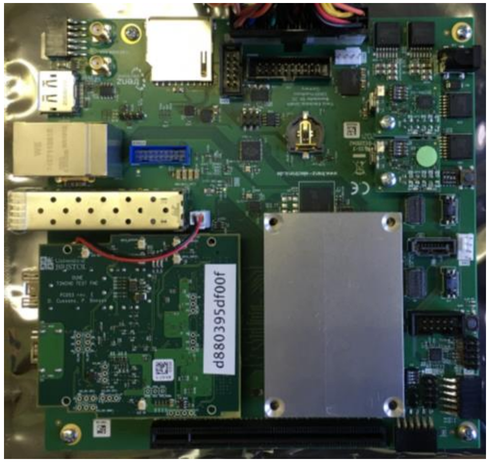
IOLaser operation for a scan

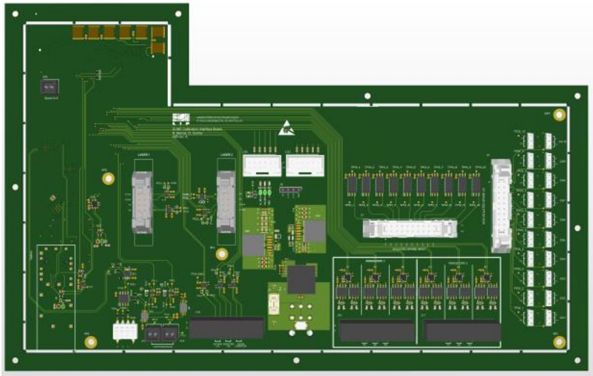


- The plan is to not stop the laser once a scan is started
- The shutter is used to control which positions actually have light injection into the cryostat

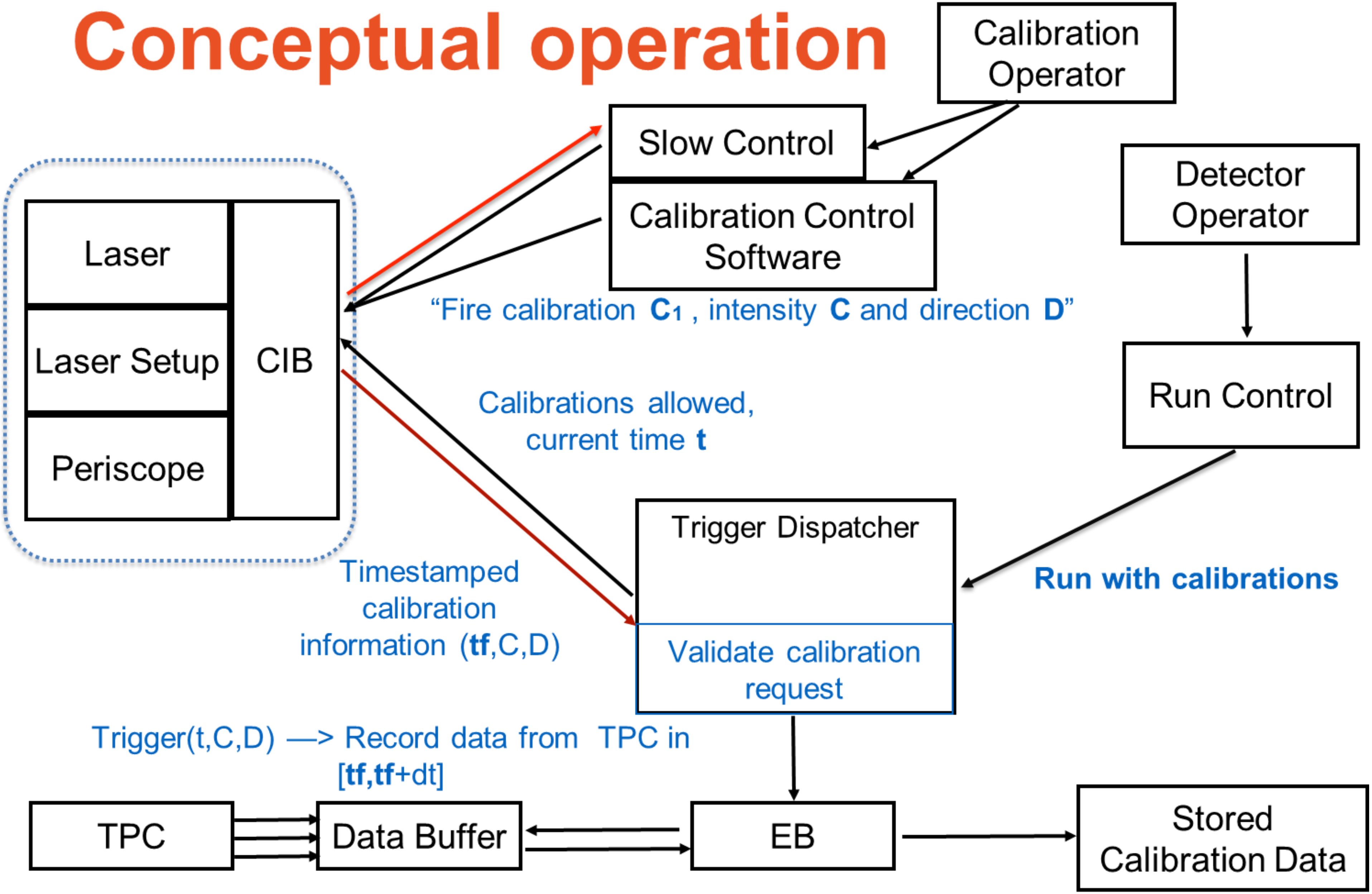
CIB Hardware

- CIB is composed of 2 parts:
  - Carrier + SoC (Trenz TEBF0808 + TE0803)
    - UltraScale+ ZU2EG
    - A quad-core Cortex-A53 CPU (64 bit, 1.5 GHz)
    - 4 GB of RAM
    - Running linux (lightweight Ubuntu server) with:
      - A Slow Control Server
      - Software interface to the DAQ
      - Software for laser configuration
  - Custom electronics condensed in single FMC daughterboard
    - Laser interfaces
    - Periscope interfaces
    - Timing interface
    - Spares

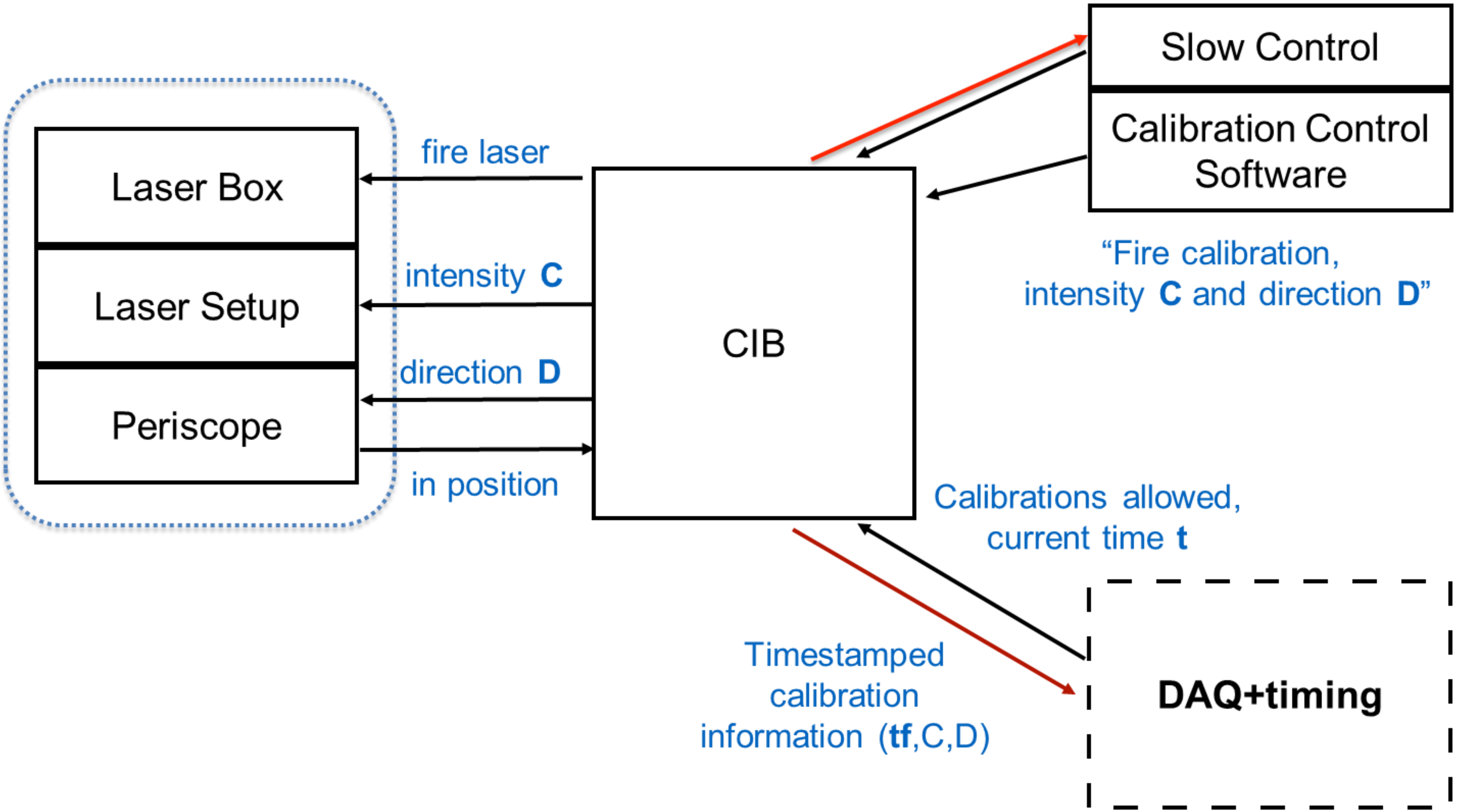


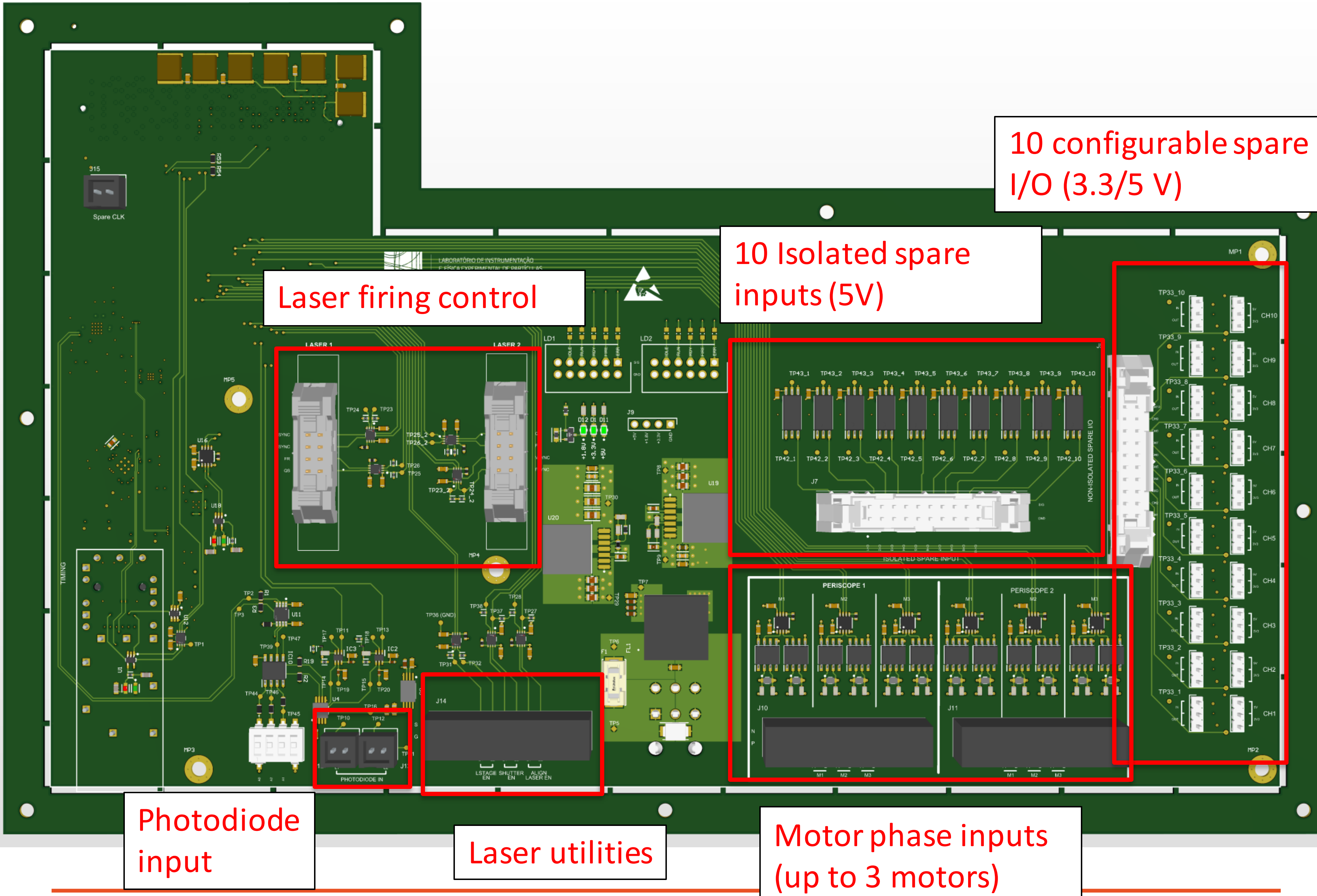


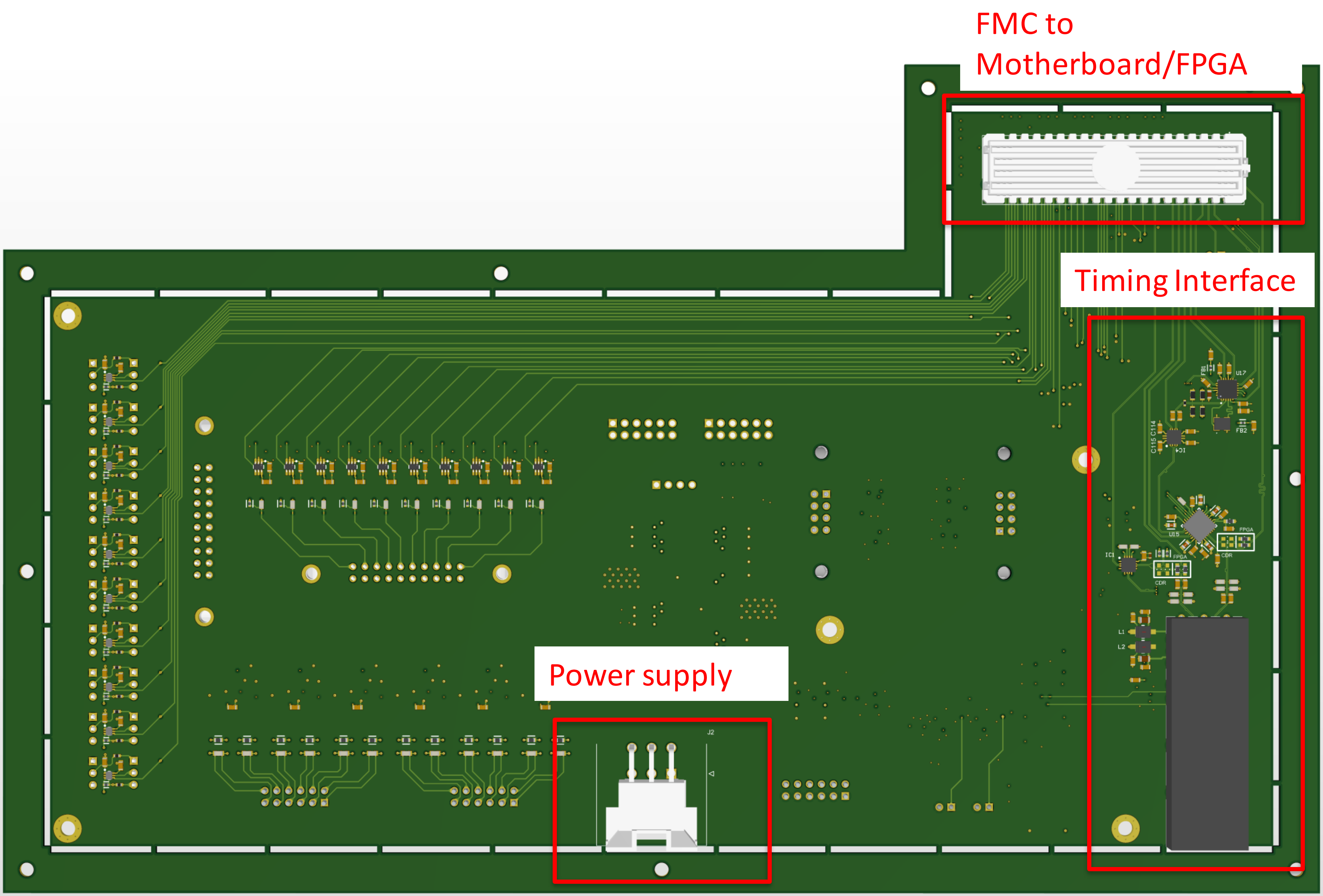
Conceptual operation



Conceptual operation (detail with CIB)




10 configurable spare I/O (3.3/5 V)
10 Isolated spare inputs (5V)
Laser firing control
Photodiode input
Laser utilities
Motor phase inputs (up to 3 motors)


FMC to
Motherboard/FPGA
Timing Interface
Power supply

Slow Control

- Slow Control System established to be based on the OPC UA libraries
- CIB will act as Slow Control server for the laser system, aggregating the information that the then served to the ProtoDUNE Slow Control Infrastructure
  - Almost everything in the laser is Slow Control data
  - List of parameters served:
    - https://docs.google.com/spreadsheets/d/1IhIPEH3469wjSd4iTEx19AlFCEf5dIkC/edit#gid=404624908

- Test server implemented
  - Used open62541 (open-source implementation of OPC UA libraries)
  - Tested locally
  - Need to reactivate discussions at CERN for integration tests