The LCLS-II Precision Timing Control System

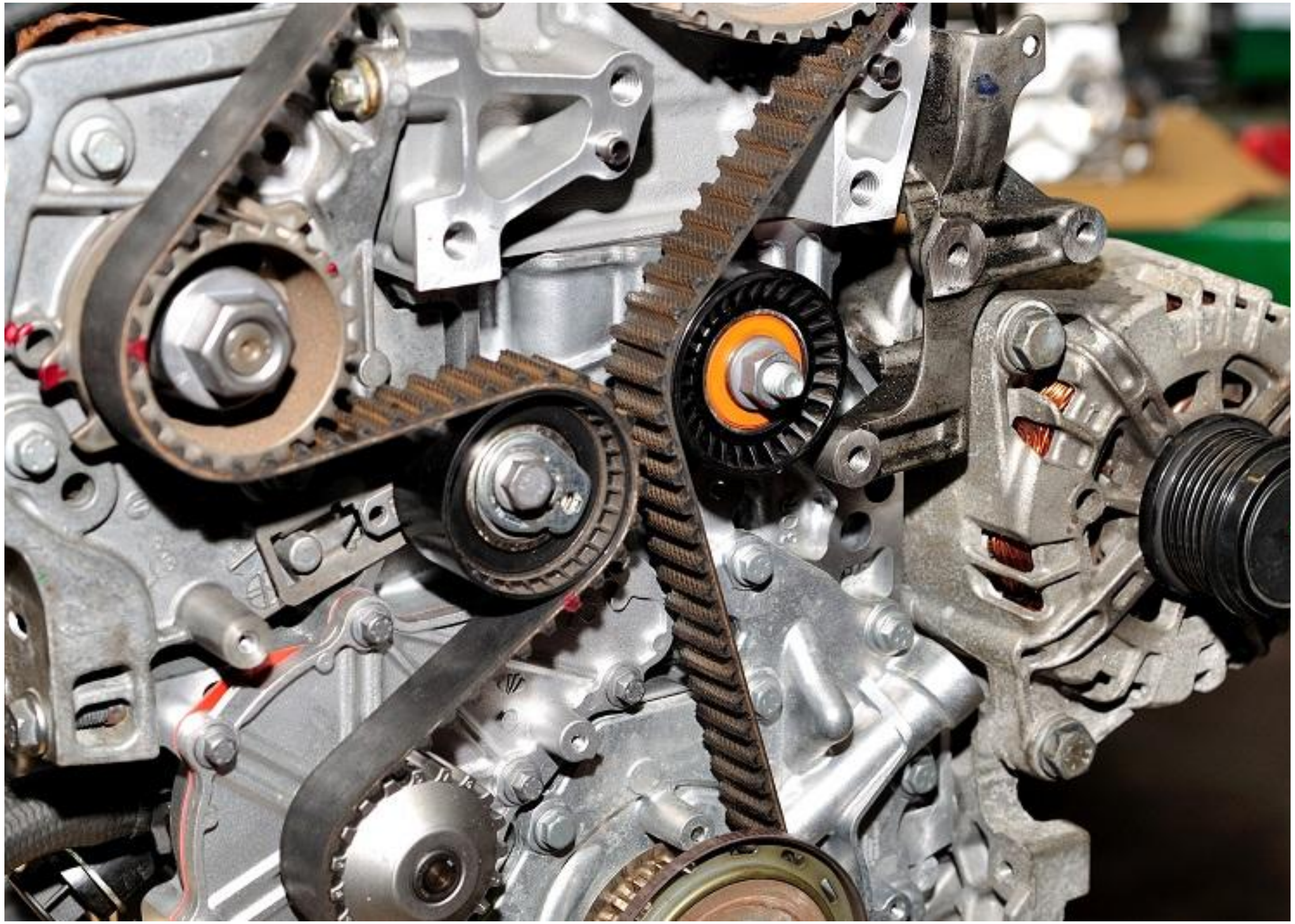
ICALEPCS 2023

Tyler Johnson
Control System Engineer
LCLS Experiment Control Systems

October 11, 2023

SLAC NATIONAL ACCELERATOR LABORATORY
Stanford University
U.S. DEPARTMENT OF ENERGY

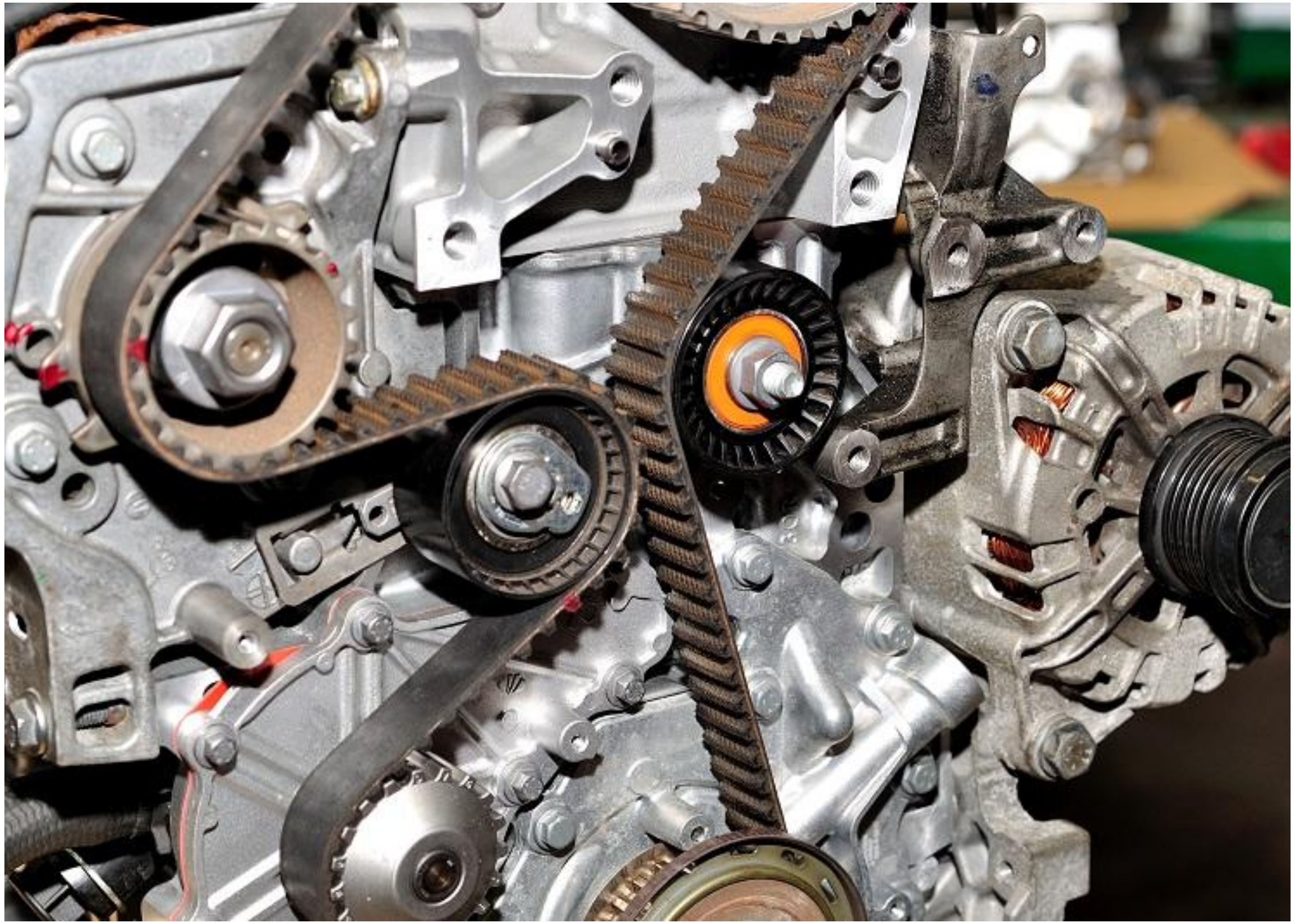
Why do we care about timing?



Timing is everything

- >75% of LCLS experiments require optical lasers
  - Pump-probe
  - “Molecular Movies”, etc.
- LCLS X-ray pulses are 100’s to 10’s of fs long
- LCLS optical laser pulses are of the same order
- Need to **establish**, **maintain**, and ***control*** the temporal overlap of these two very short pulses

Why do we care about timing?



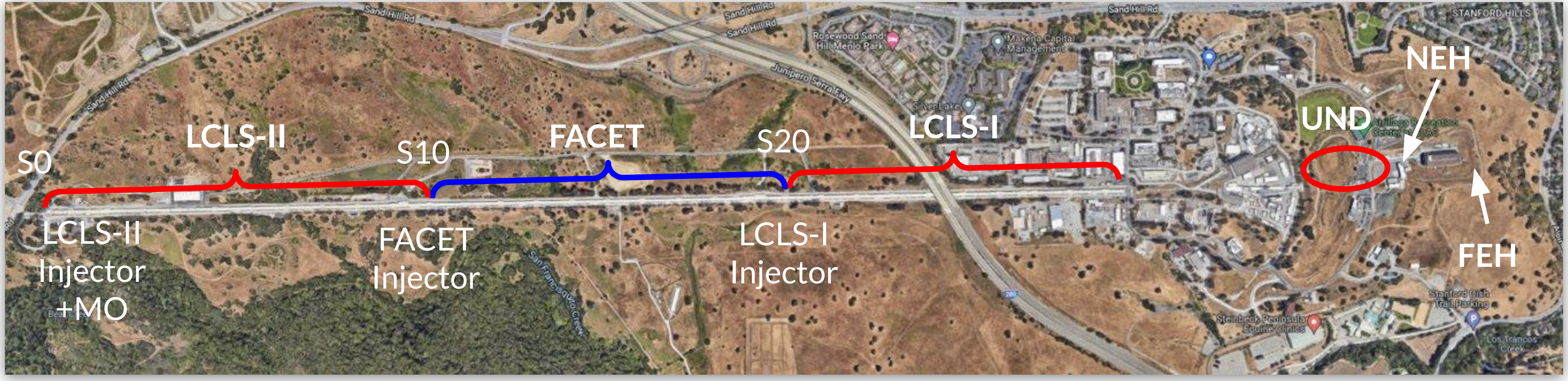
Timing is everything

- >75% of LCLS experiments require optical lasers
  - Pump-probe
  - “Molecular Movies”, etc.
- LCLS X-ray pulses are 100’s to 10’s of fs long
- LCLS optical laser pulses are of the same order
- Need to **establish**, **maintain**, and ***control*** the temporal overlap of these two very short pulses

Requires precise phase locking and phase control!

A 10,000 ft (3 km) view of LCLS

Multiple facilities, different needs



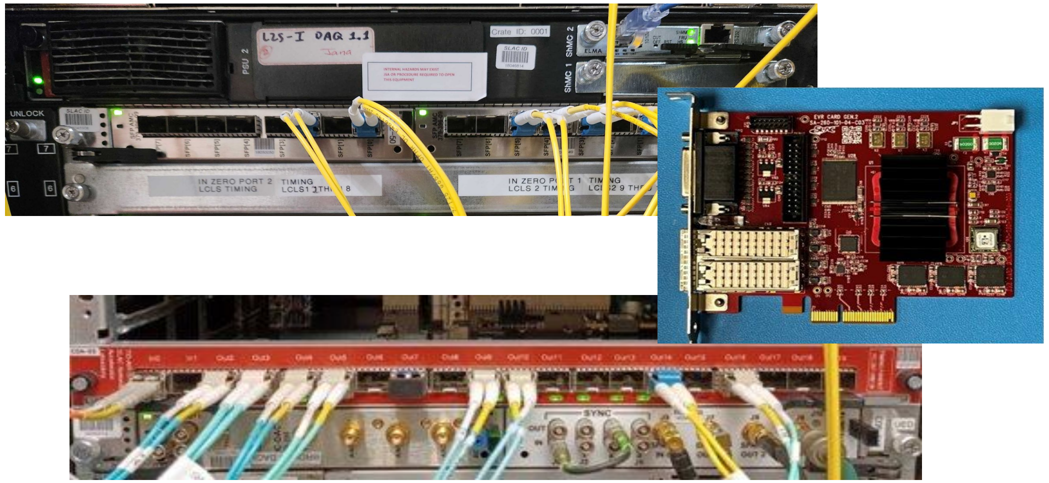
Event and RF timing is distributed to multiple facilities (AD/LCLS)

- LCLS-I S20 through S30 (approx. 100m/sector)
- FACET S10 through S20
- LCLS-II S0 through S10

Two flavors of timing at LCLS

Event Timing

- LVTTL triggers and CPU interrupts for data acquisition and/or PV processing
- Generator: synchronous timestamps & control on fiber optic distribution system
- Receiver: decodes messages, generates triggers for device control & acquisition



Precision Timing

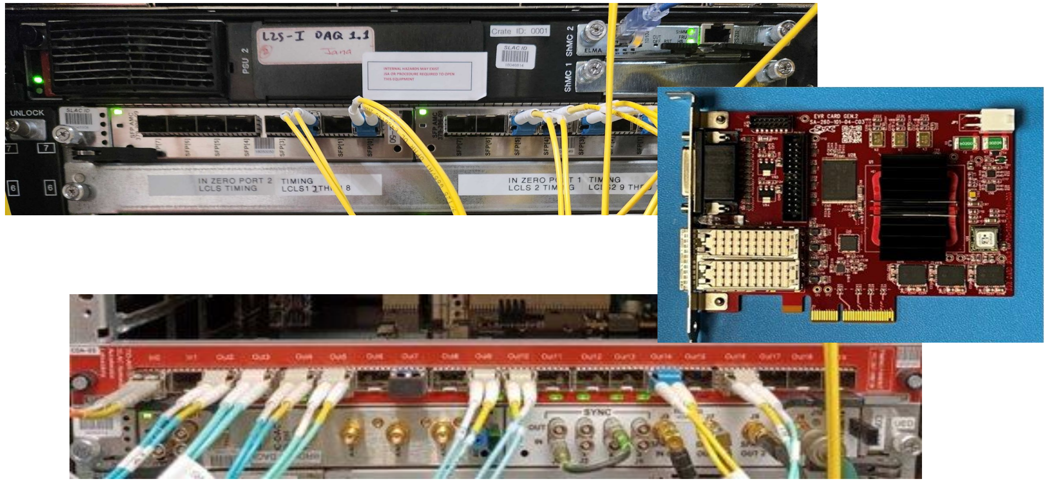
- Precise (~10’s fs) synchronization with XFEL
- RF synchronization of devices via hardware: PLL, FPGA, etc.
- Used for optical drive laser synchronization and delay control w.r.t. the XFEL

Two flavors of timing at LCLS

Focus for today

Event Timing

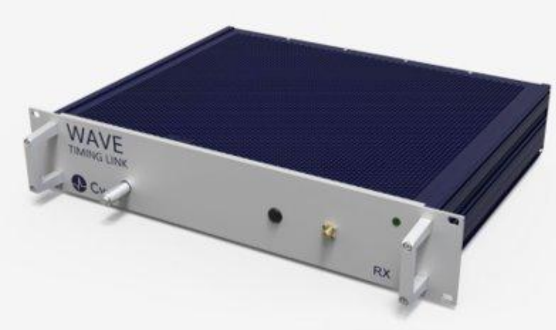
- LVTTL triggers and CPU interrupts for data acquisition and/or PV processing
- Generator: synchronous timestamps & control on fiber optic distribution system
- Receiver: decodes messages, generates triggers for device control & acquisition

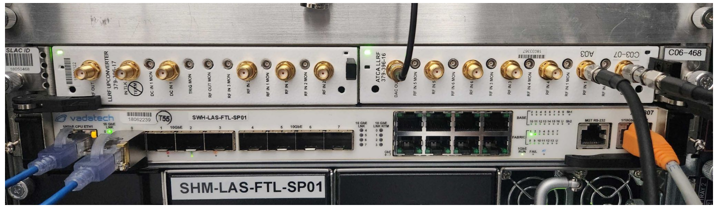


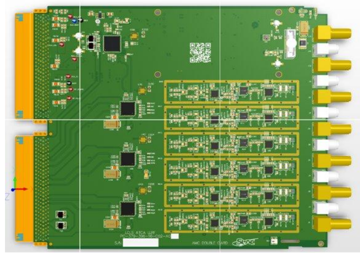
Precision Timing

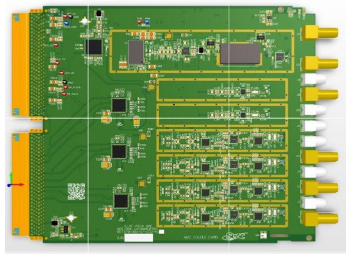
- Precise (~10’s fs) synchronization with XFEL
- RF synchronization of devices via hardware: PLL, FPGA, etc.
- Used for optical drive laser synchronization and delay control w.r.t. the XFEL

How do we do precision timing?









RF Distribution

- RF-Over-Fiber technology
- Commercial systems
- Transmit LCLS RF reference across >5km optical fiber
- Low jitter, low drift

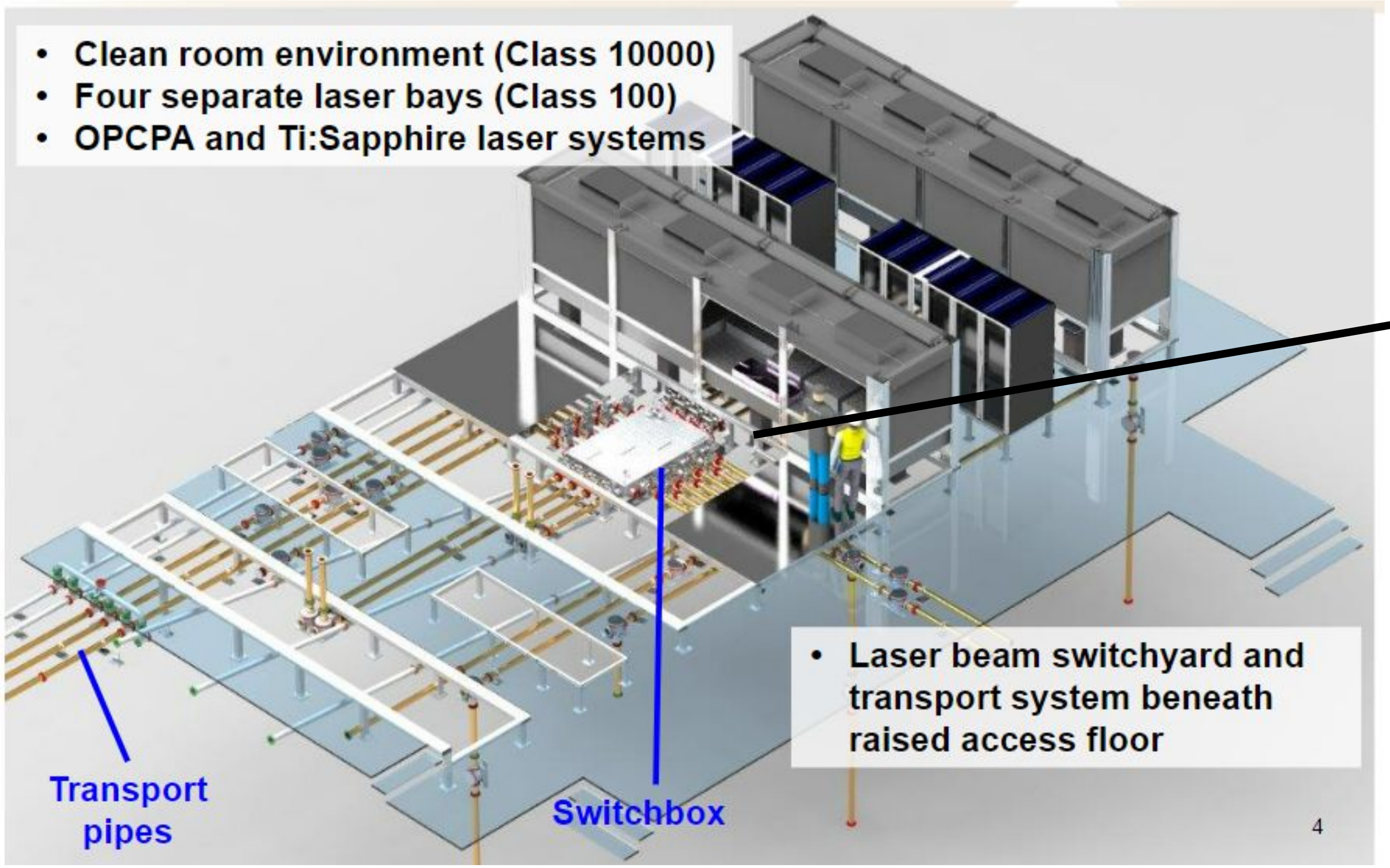
RF Locking

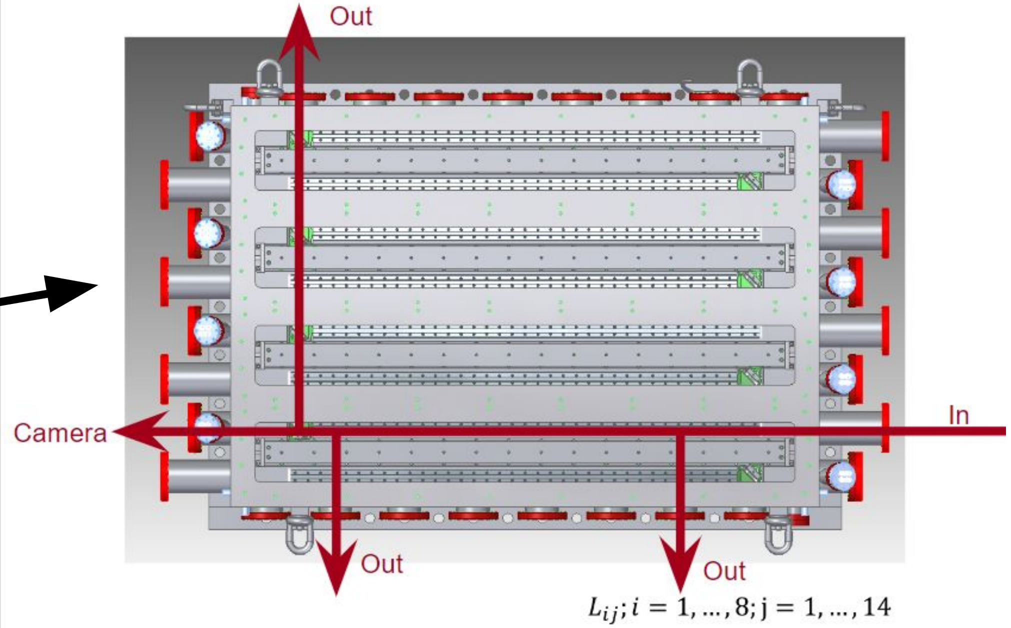
- Phase lock to RF reference
- Based on ATCA Common Platform developed for LCLS-II LINAC
- Delay control for pump-probe experiments

Optical Locking

- Based on commercial optical timing system
- Uses optical, rather than RF, phase comparison
- Provides sub-fs delay control and measurement

Any laser, any hutch



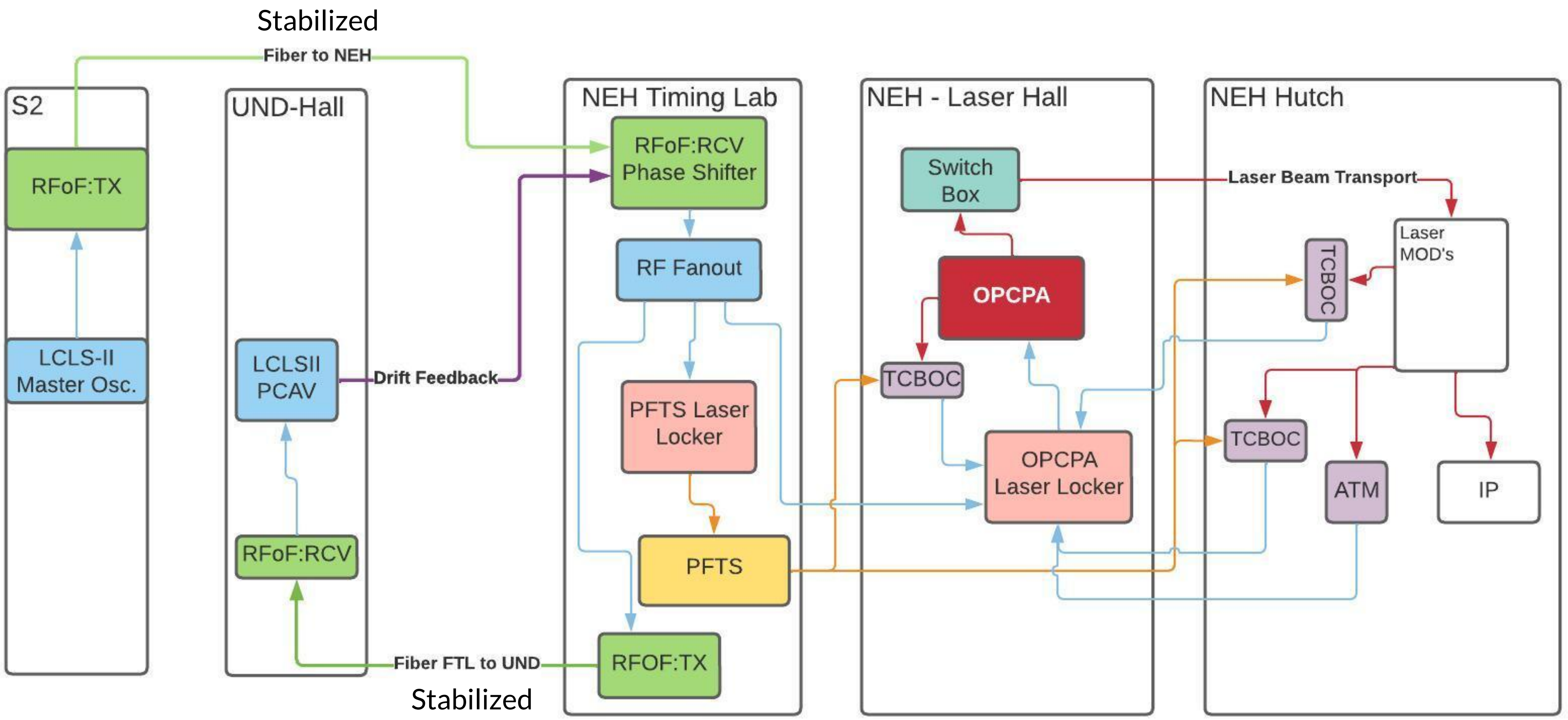


NEH Beam Transport System

- Optical laser switch box (8 in, 14 out)
- Evacuated transport lines to every IP
- Supplies optical laser for entire Near Expt. Hall

N:M relationship between hutches and the laser and timing systems!

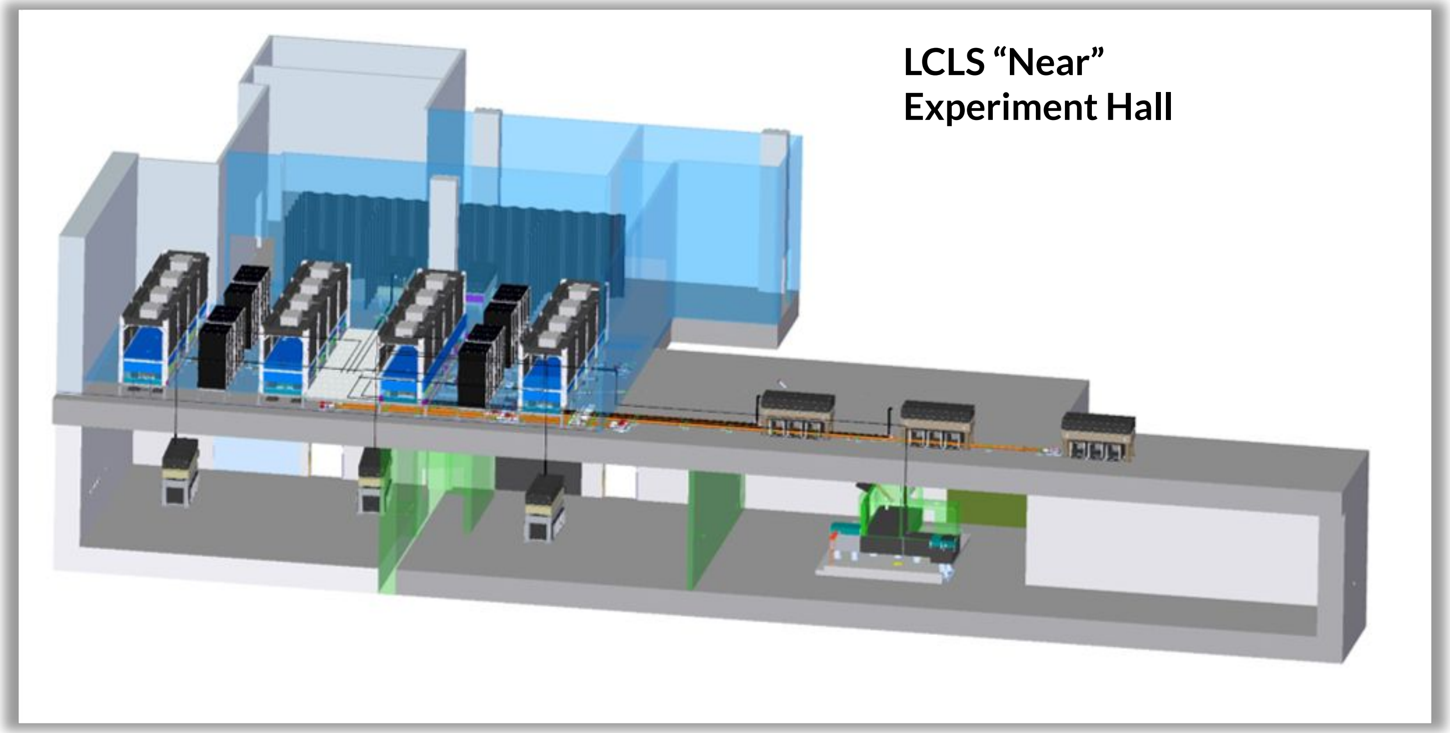
Putting it all together



What challenges do we face?

Multi-user systems

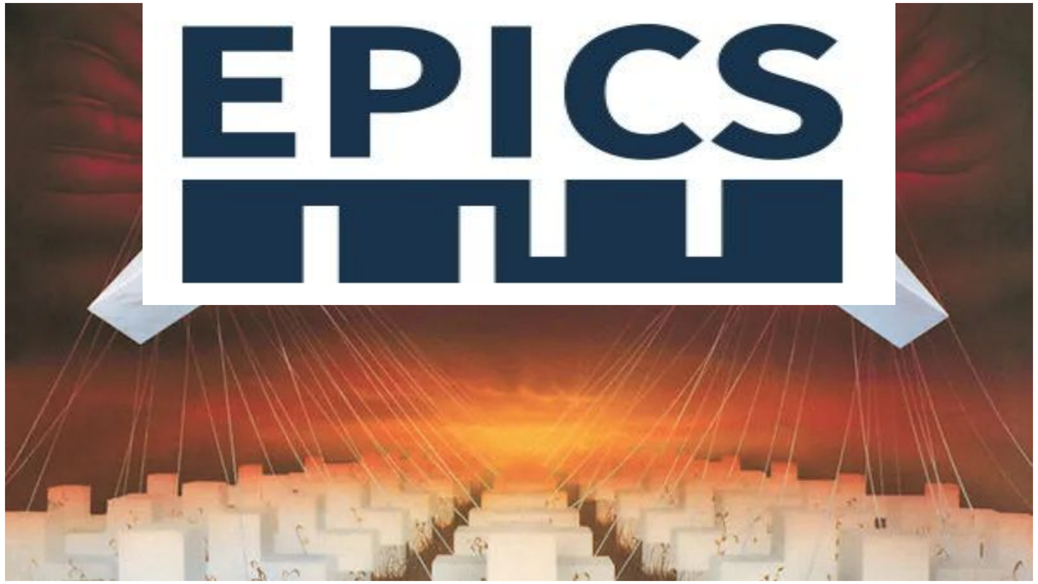
- LCLS-I lasers and laser lockers are 1:1
  - No "sharing"
- LCLS-II OPCPA laser can be delivered to up to 7 different interaction points
- Up to 4 experiment changes per day
- **Goal:** allow beamline operators to changeover laser and timing w/out expert assistance
- Need to deconflict shared resources
  - Drive laser
  - Laser locker
  - Stabilized timing links



From Python to EPICS

All EPICS, all the time

- Most beamline automation at LCLS is done via Python
    - ophyd, bluesky, etc
- LCLS-I: EPICS IOC + Python script
    - IOC: basic functions, frequency locking, EPID record
    - Python: high level interface PVs, calibration routine, delay calculations and delay control
- LCLS-II: Why not *100%* EPICS?
    - Most of the LCLS-I locker could be handled by EPICS
    - Can have an all in one system application
- <u>Missing capabilities</u>:
    - Mutual exclusion
    - Easy automation procedures



Resource arbitration at the record level

The arbiter record

- 30 separate request lines (potential owners)
- Reports current owner and owned status
- Requests can be cleared and arbiter released
- Used in global and local contexts throughout the PFTS automation system

FIELD	Summary
REQ0 to REQ29	30 long inputs. Write a non-zero value to request the arbiter, and zero to release it (or cancel the request).
OWN0 to OWN29	30 string input links. These describe the thirty input requests.
OWNER	A string value giving the name of the current owner using the OWNx field. If OWNx is not defined, this will be “REQ*n*” if the owner is *n*, or “None” if the arbiter is not owned.
VAL	A long value indicating the current owner, or -1 if it is not currently owned.
CLEAR	Writing a non-zero value will clear all requests and release the arbiter. This field is cleared after handling the request.

Example: Global PFTS arbiter record

```
record(arbiter, $(P):GLOBAL:ARB) {
    field(OWN0, "$(P):OWN0 NPP NMS")
    field(OWN1, "$(P):OWN1 NPP NMS")
    field(OWN2, "$(P):OWN2 NPP NMS")
    field(OWN3, "$(P):OWN3 NPP NMS")
    field(OWN4, "$(P):OWN4 NPP NMS")
    field(OWN5, "$(P):OWN5 NPP NMS")
    field(OWN6, "$(P):OWN6 NPP NMS")
    field(OWN7, "$(P):OWN7 NPP NMS")
    field(OWN8, "$(P):OWN8 NPP NMS")
    field(OWN9, "$(P):OWN9 NPP NMS")
```

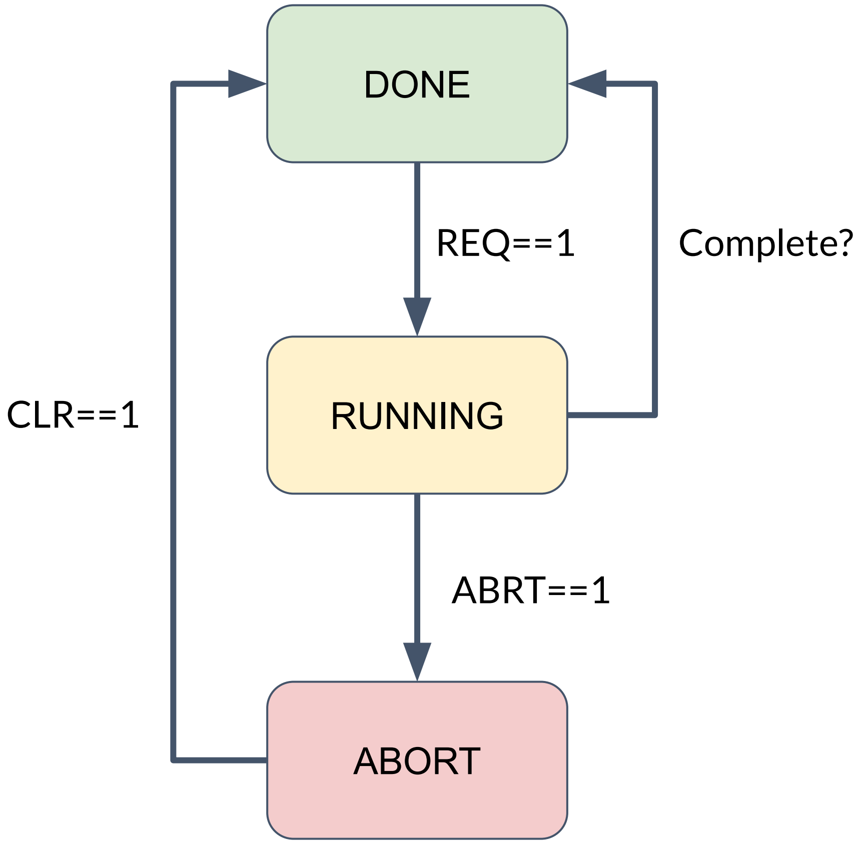
Example: Hutch interface PVs accessing global lock

```
# Global action lock.

# Write 1/0 to lock/unlock the global lock.
record(longout, $(P):QRIX:ACTREQ) {
    field(VAL,  "0")
    field(PINI, "YES")
    field(FLNK, "$(P):QRIX:ACTREQ_ASUB")
}

# Write $(P):QRIX:ACTREQ to $(P):GLOBAL:ARB.REQ0.
record(longout, $(P):QRIX:ACTREQ_ASUB) {
    field(OMSL, "closed_loop")
    field(DOL,  "$(P):QRIX:ACTREQ NMS NPP")
    field(OUT,  "$(P):GLOBAL:ARB.REQ0 NMS PP")
}
```

How do we do procedures in EPICS?

stepSequence Record

- What do we *want* in a procedure?
    - Start, Stop, Status
    - Clear, consistent interface
- “stepSequence” record: simple FSM interface
- stepSequence provides 5 main interface fields:
    - **REQ**: Set to 1 to ask the stepSequence to start.
    - **ABRT**: Set to 1 to ask the stepSequence to abort.
    - **STATE**: Indicates whether the stepSequence is DONE (0), RUNNING (1), or ABORTED (2).
    - **CLR:** Inlink. Set to 1 to move from ABORT to DONE
    - **STEPNAME**: Description of current step. Concatenation of step prefixes and step names.



We automate the use of the stepSequence record!

Database Preprocessor

- Provides a simple, consistent method for generating stepSequence records
- Preprocessor is applied to any .dbs files found during compilation
- Recognizes sequence { … } macros
- Decodes simple macros into stepSequence records and supporting EPICS records
- “Normal” records are left untouched
- StepSequence records can be chained together, or used as steps *within* other stepSequences!

Example

```
sequence($(P):$(H):R$(N):REQ_SRC) {
    SET_INT("Start Operation", $(P):$(H):R$(N):LAST_REQ, 0)
    SET_INT("Src ownership", $(P):$(H):R$(N):SRC_REQ, 1)
    WAIT("Waiting for src ownership", $(P):$(H):R$(N):SRC_OWNED, 5)
}
```

.dbs file

stepSequence Preprocessor

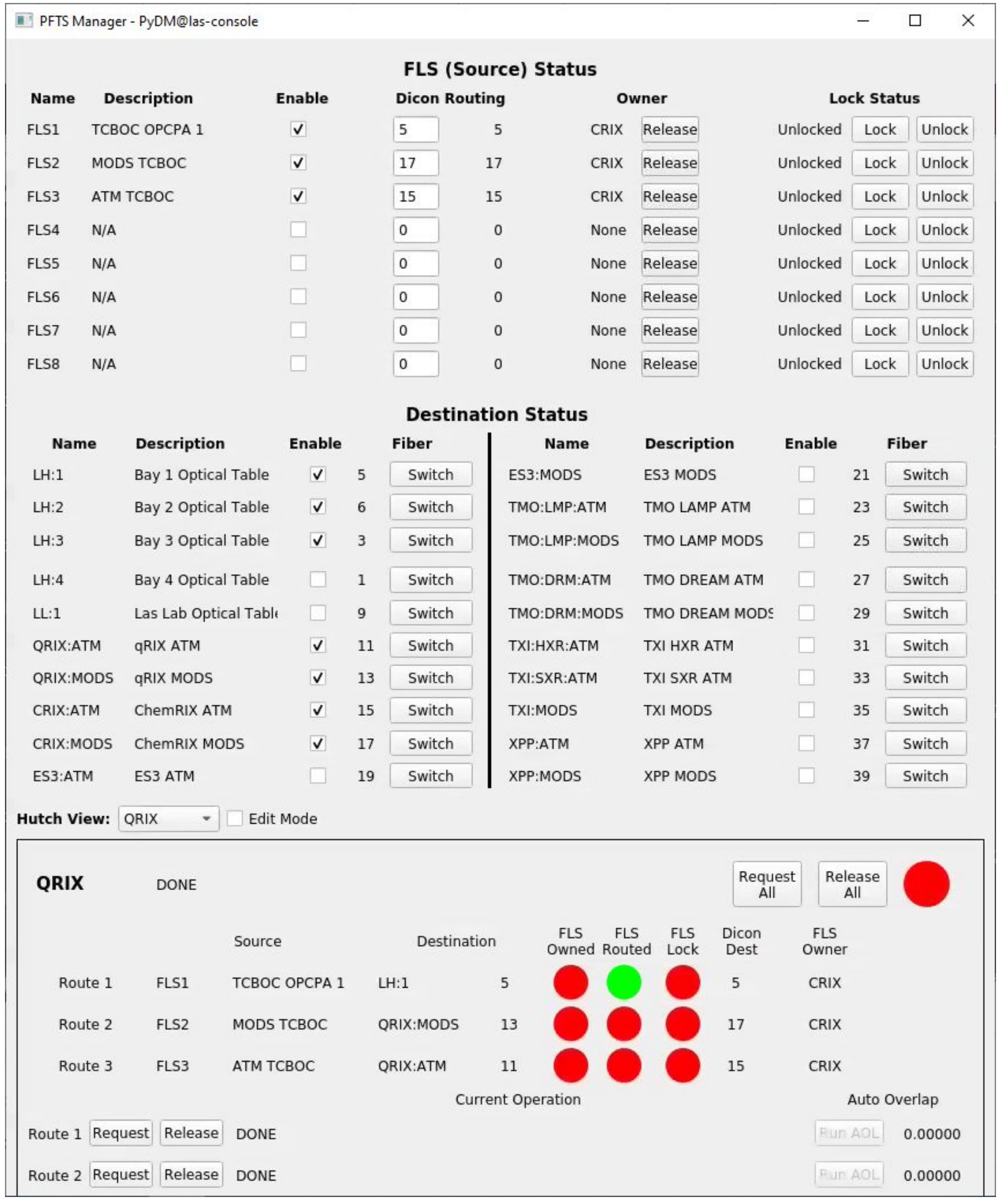
.db file

```
record(stepSequence, $(P):$(H):R$(N):REQ_SRC) {
    field(PRE0,       "Start Operation")
    field(REQ0,       "$(P):$(H):R$(N):REQ_SRC:_V0.PROC")
    field(PRE1,       "Src ownership")
    field(REQ1,       "$(P):$(H):R$(N):REQ_SRC:_V1.PROC")
    field(PRE2,       "Waiting for src ownership")
    field(REQ2,       "$(P):$(H):R$(N):REQ_SRC:ST2.PROC")
    field(ABRT2,      "$(P):$(H):R$(N):REQ_SRC:AB2.PROC")
    field(STATE2,     "$(P):$(H):R$(N):REQ_SRC:_S2 CPP")
}
```

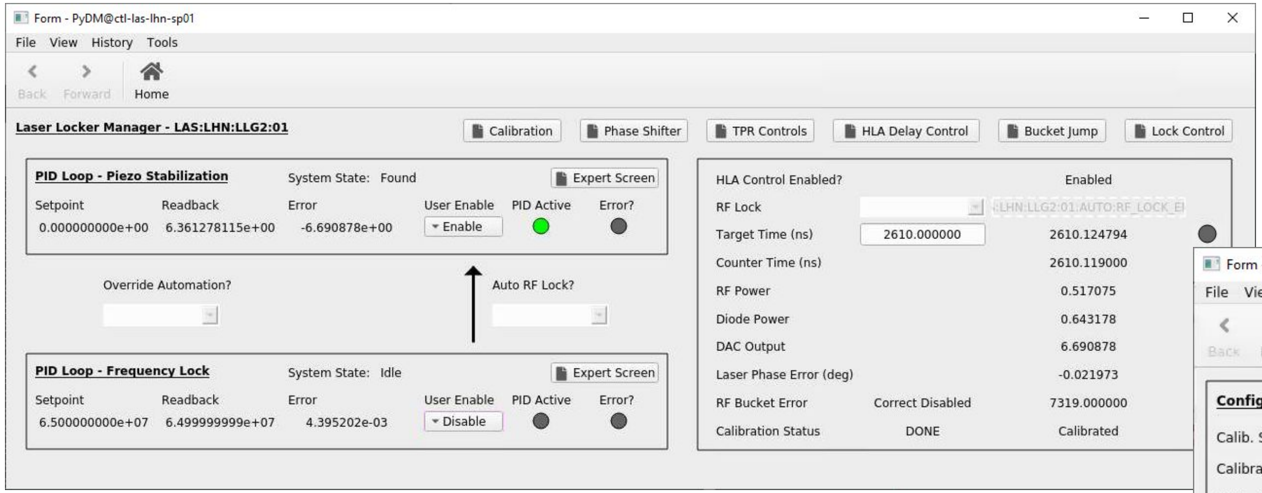
Automated optical reference distribution

PFTS Manager

- Provides arbitration of shared resources
- Makes heavy use of the stepSequence and Arbiter records
- Provides facility for requesting/releasing control of up to 3 PFTS stabilized links for TCBOCs
- Automatically performs signal routing via fiber matrix switch
- Performs stabilized link locking and unlocking as needed for fiber routing
- Displays current system status
- Provides both “user” and “expert” interfaces to subsystems

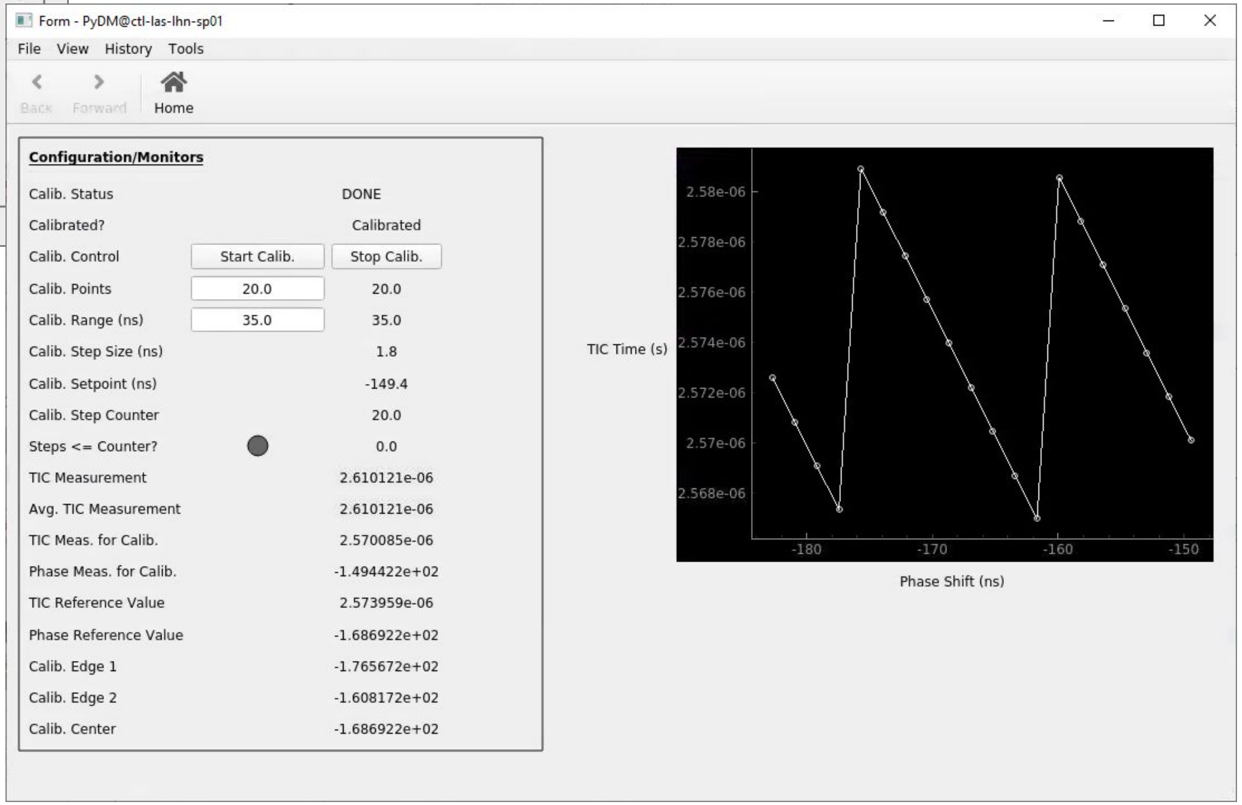


Automated pump-probe delay control



Laser Locker Manager

- Primarily EPICS record primitives
  - stepSequence for calibration
- Single “Target Time” delay control
- Automated system calibration, locking, and RF/laser bucket jump corrections
- TODO: PFTS-style arbitration



SLAC

Acknowledgments

LCLS Control Systems

- Mike Browne
- Christina Pino

LCLS Timing Systems

- Stefan Droste
- Dring Xiang
- Charlie Xu

Acknowledgments

LCLS Control Systems

- Mike Browne
- Christina Pino

LCLS Timing Systems

- Stefan Droste
- Dring Xiang
- Charlie Xu

…. And all of you!

Questions?