

# LCLS Experiment Control Systems Platform Development

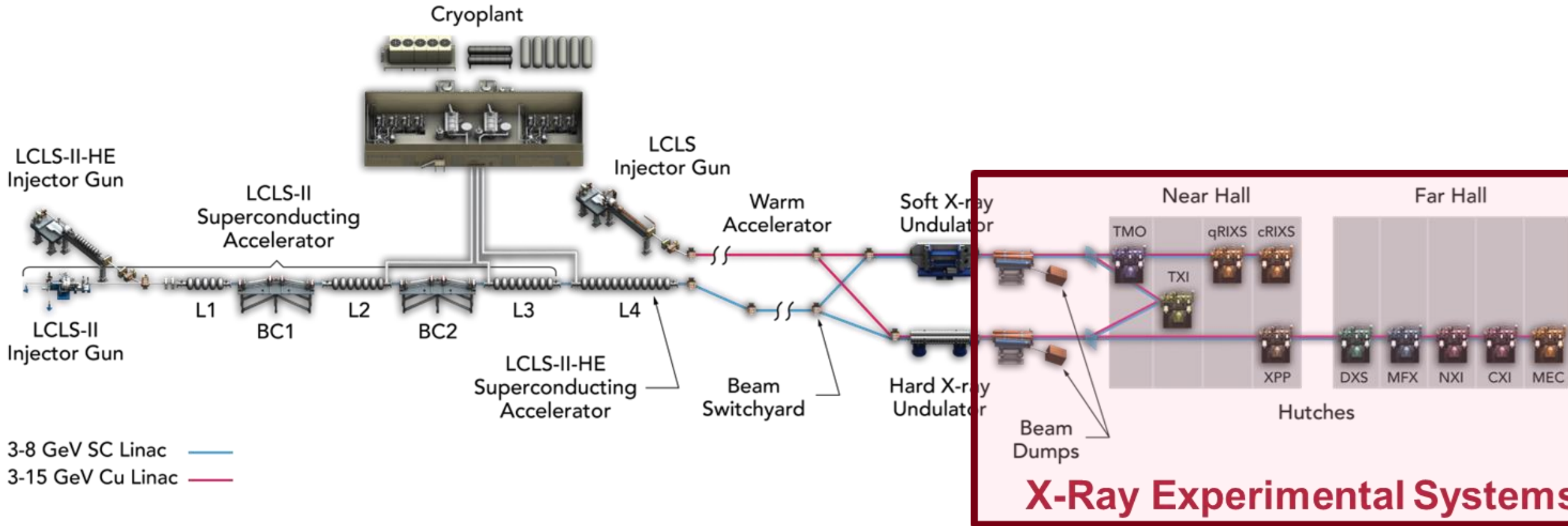
ICALEPCS2023 - PLC Workshop Presentation

---

Alex Wallace / Dept. Head / ECS PD

07 October 2023

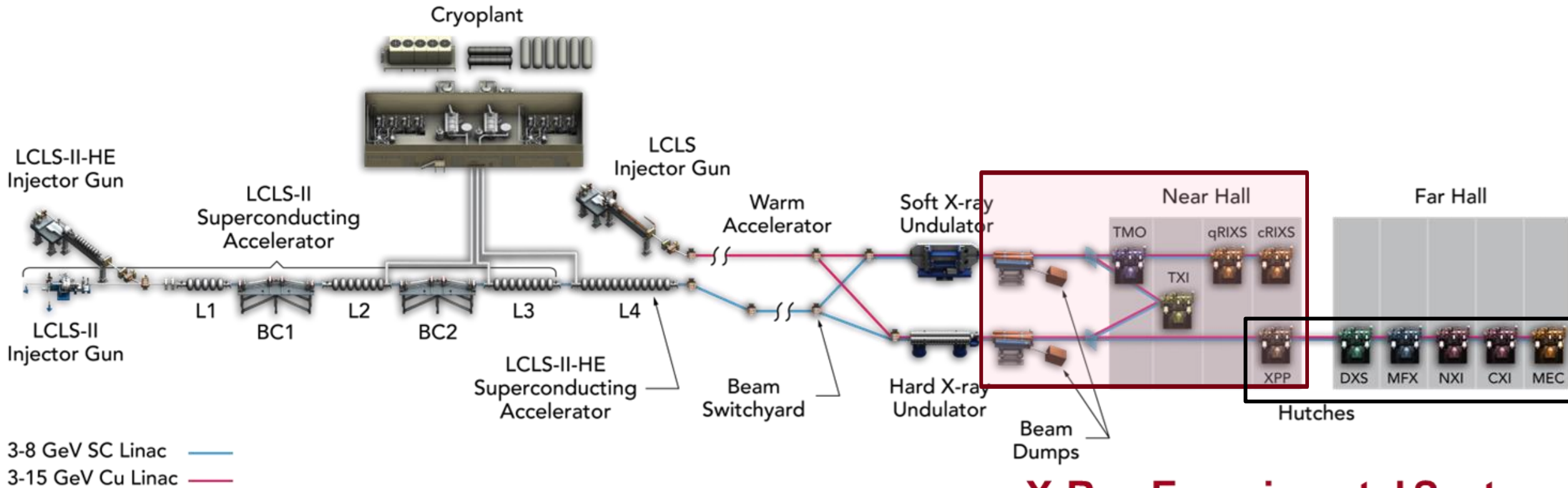
# LCLS Instrument Suite



# LCLS Instrument Suite

LCLS-I Controls

LCLS-II Controls



**X-Ray Experimental Systems**

# Experiment Control System Division Background

## Background

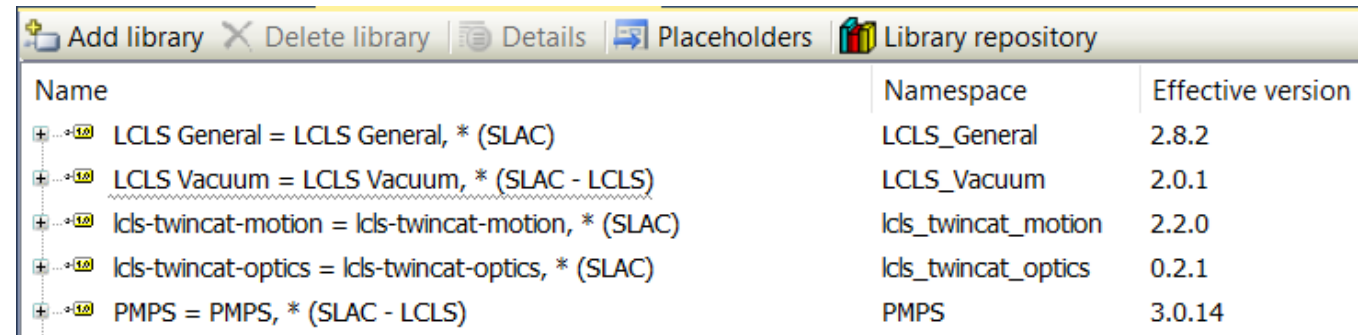
- ~25 control system engineers with a wide range of expertise
- Supporting day-to-day operations, experiment setup, XFEL beam delivery, and major projects
- In existence since roughly 2010, originally called the Photon Controls and Data System (PCDS) group



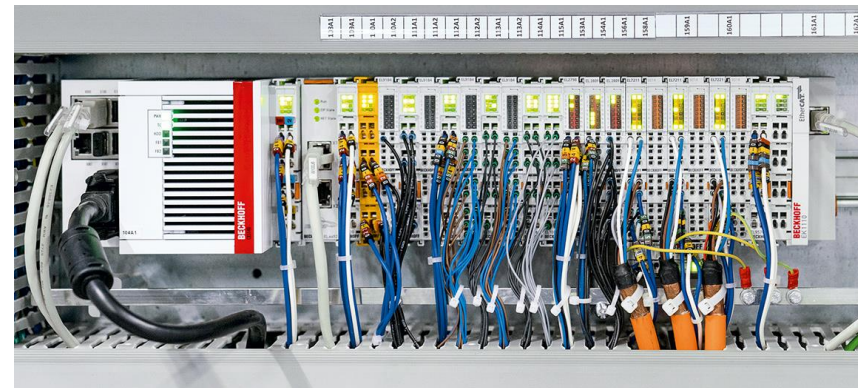
# Migrated towards a single PLC platform

30-50 Beckhoff PLCs deployed, 7 Automation Direction in legacy areas.

- Legacy Automation Direct, phasing out
- Beckhoff PLC and TwinCAT 3 in all new areas and any new projects
- Structured Text everywhere (Ladder in the Automation Direct legacy systems)
- Common libraries, Vacuum, Motion, Machine Protection, Optics, General



Name	Namespace	Effective version
LCLS General = LCLS General, * (SLAC)	LCLS_General	2.8.2
LCLS Vacuum = LCLS Vacuum, * (SLAC - LCLS)	LCLS_Vacuum	2.0.1
lcls-twinCAT-motion = lcls-twinCAT-motion, * (SLAC)	lcls_twinCAT_motion	2.2.0
lcls-twinCAT-optics = lcls-twinCAT-optics, * (SLAC)	lcls_twinCAT_optics	0.2.1
PMPS = PMPS, * (SLAC - LCLS)	PMPS	3.0.14



**BECKHOFF**

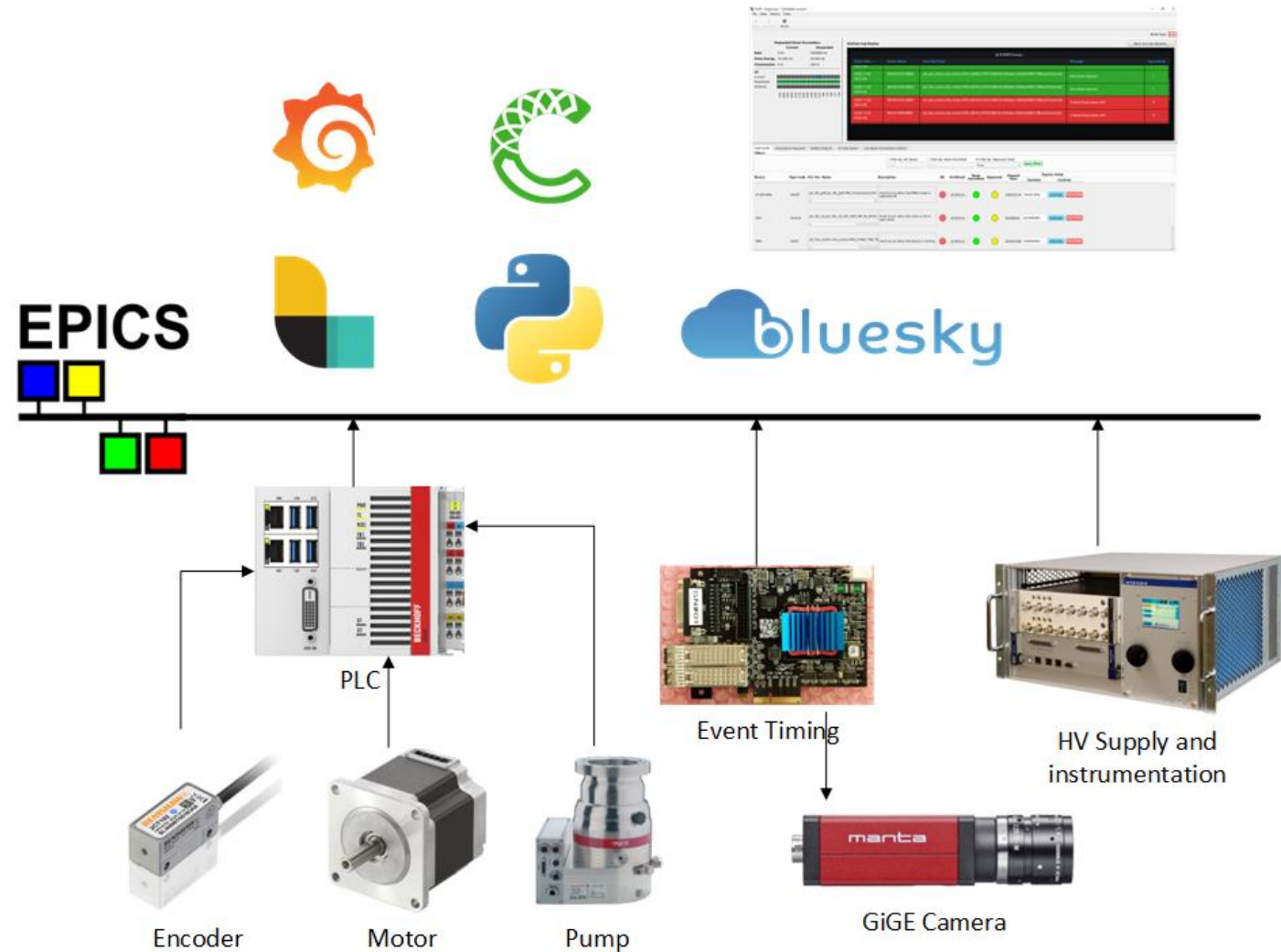
# Experiment Control Systems is a SCADA

## Subsystems using PLCs

- Vacuum
- Stepper Motors
- Diagnostics
- Optics
- Machine protection
- Sample Delivery
- Optical laser delivery
- etc.

## Subsystems not using PLCs

- High speed DAQ
- Camera systems
- Piezo Motors
- Timing systems (event, precision)
- Misc. instrumentation and diagnostics

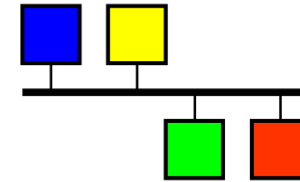


# ECS uses Beckhoff and EPICS and a Python-based framework

## Beckhoff PLC

- ECS maintains a network-based distributed controls system built on top of EPICS.
- Control system is primarily Beckhoff PLCs
  - Mostly 2 part numbers CX5010 and CX5020
  - Consolidating to CX5240 on Tc/BSD
- EPICS
  - Beckhoff communication interface over TCP/IP: ADS (Automation Device Specification)
  - Automatic IOC generation

**EPICS**

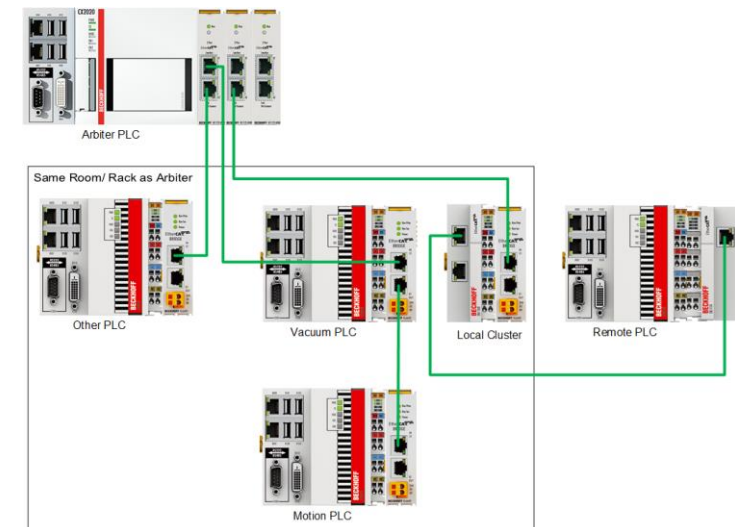


**BECKHOFF**

**ADS**

**TwinCAT®**

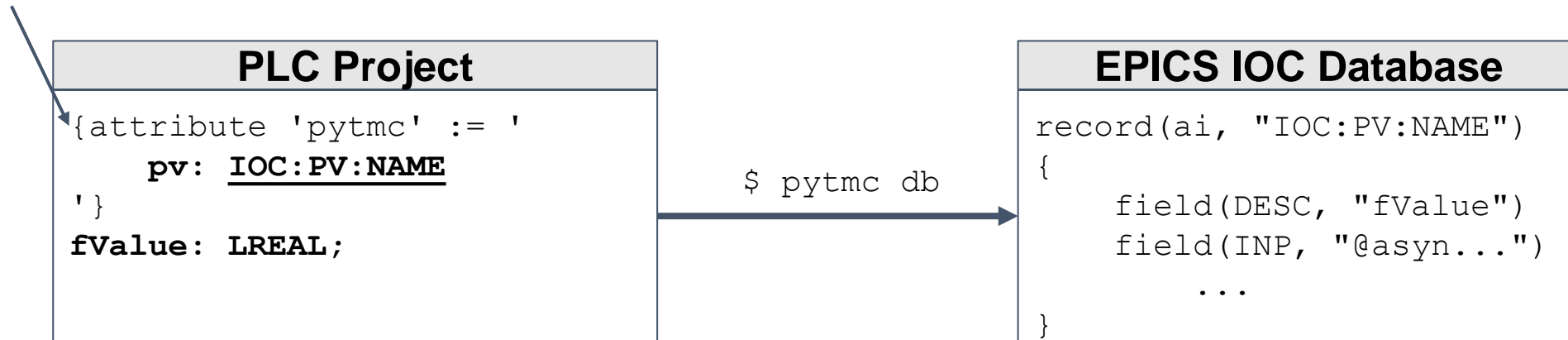
**EtherCAT® I/O**



# Automatic EPICS IOC Generation from PLC Source Code

IOCs made easy: [pytmc pragmas](#)

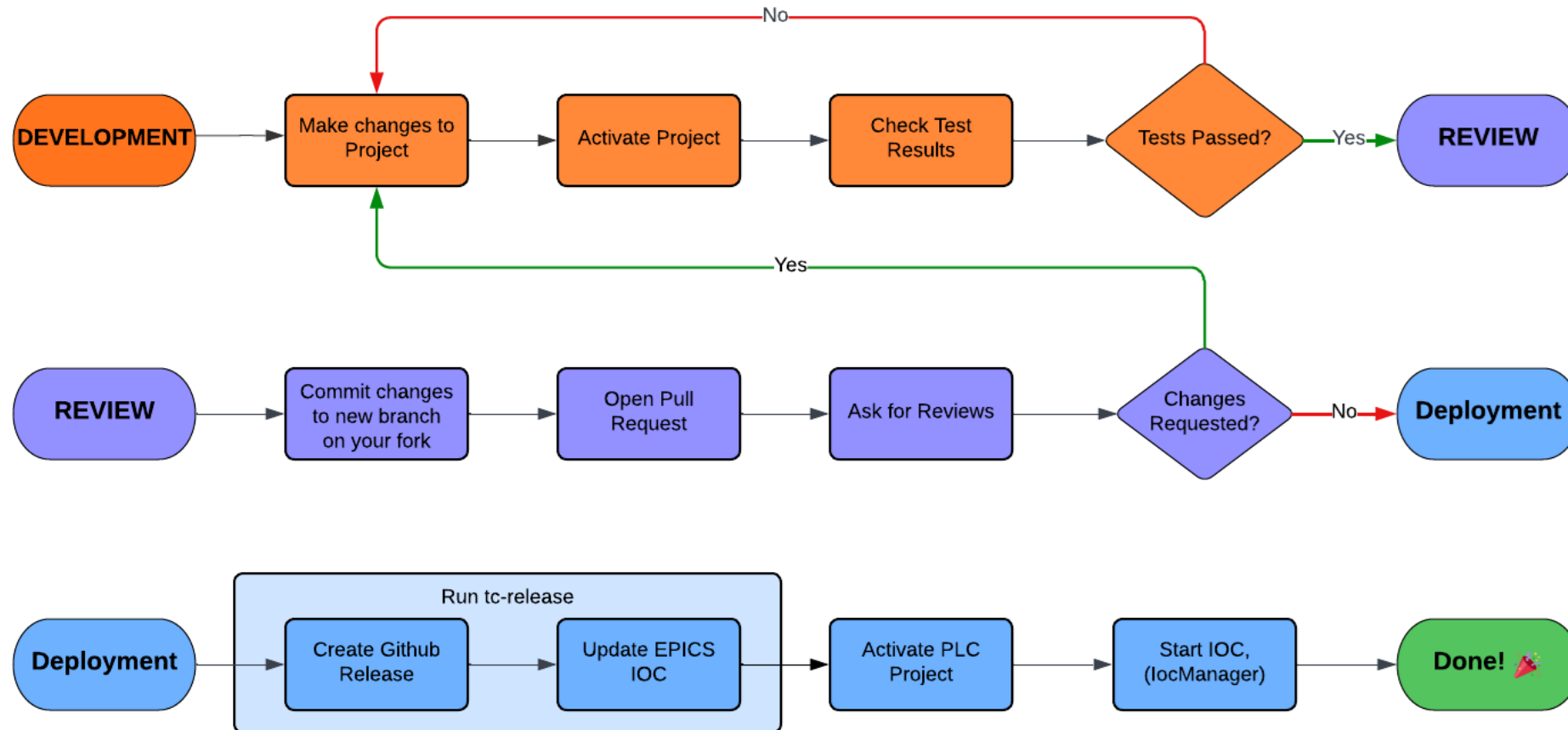
IEC 61131-3 pragma annotating the following line - the variable declaration of "fValue"



- Uses Structured Text to mark PLC Variables for exposing data to the EPICS IOC layer
- Our tooling looks at a build artifact from TwinCAT (.tmc) to generate an IOC database file (.db)
  - ADS communication allows for access by way of symbol (PLC variable) name
  - Creating an IOC becomes an automated process!
  - No more modbus address maps or keeping separate mappings of PLC variable/symbol names to PV names

# PLC code development uses a common software development workflow, fork-pull.

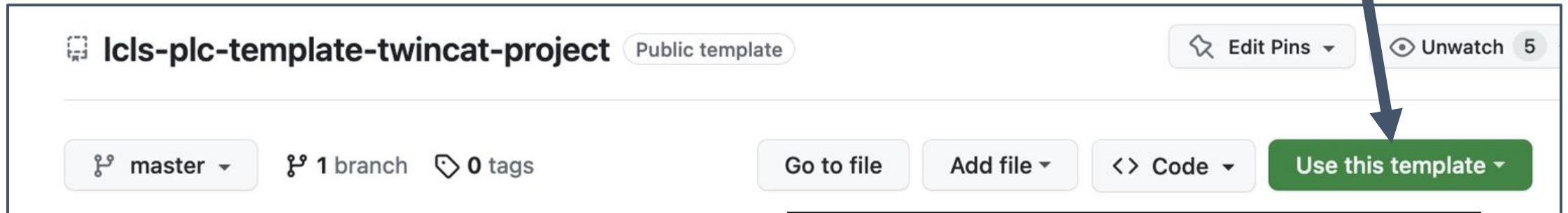
## TwinCAT PLC workflow



# We use the Github template system for new project templates

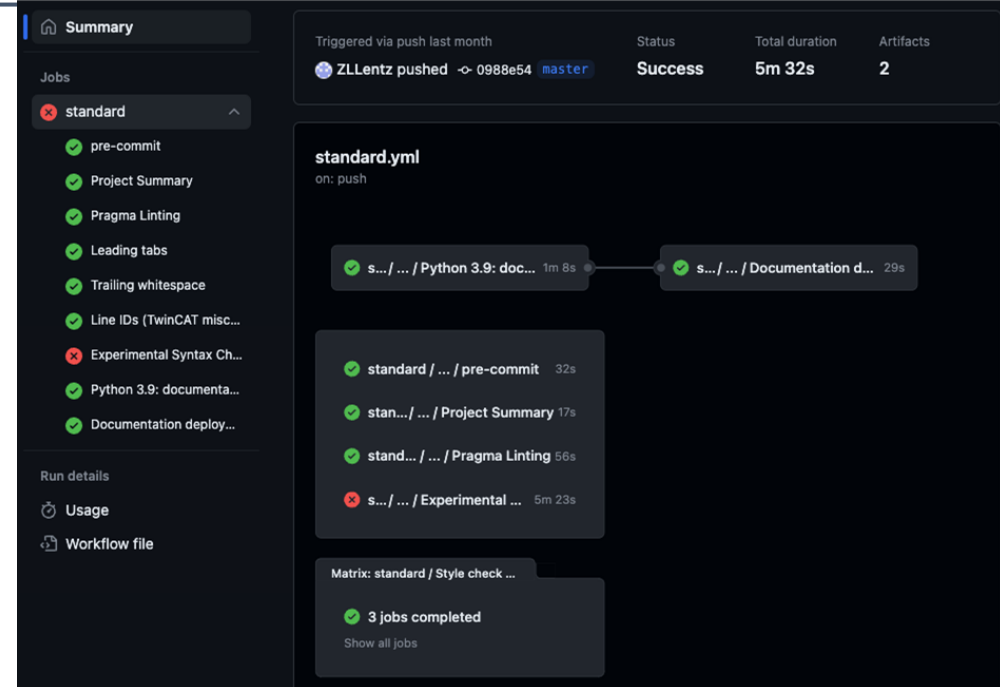
<https://github.com/pcdshub/lcls-plc-template-twincat-project>

Using the template is as easy as clicking a button:



## Includes

- Pre-commit configuration
- Standard GitHub actions
  - Pre-commit
  - pytmc summary (inspects .tsproj files)
  - pytmc pragma linting
  - Style, syntax checks
- Documentation building, template



# Github works well for source control, could be even better

## ECS uses GitHub

Progression:

- TwinCAT 3
  - Started with Beckhoff integrated SVN
    - Worked OK
  - Git shows up, much better than SVN
  - Beckhoff integrates git into TwinCAT
    - Works OK
  - Went back to using standalone git, either git-bash, or GitHub's nice GUI system
    - Works best
- Moved *everything* to GitHub
  - We made a nice system for viewing TwinCAT projects in web pages hosted by GH
  - Actions automatically update these pages with repo commits; see testing slides!
  - Currently investigating ways to make GitHub identify and display .xml TwinCAT project files nicer/natively



# GitHub Pages PLC Code View

Per-PLC project web view “documentation” of sorts, plus facility-wide summary

pcdshub/lcls-plc-lfe-motion

Search docs

PLC-LFE-MOTION

- Pragmas
- NC Settings
- Box Hierarchy
- Boxes
- Links
- LFE\_MOTION
- Settings
- Pragmas
- Libraries
- Symbols
- Data Types
- Database Records

## RTDSL0

```
//{attribute 'qualified_only'}
VAR_GLOBAL
  {attribute 'pytmc' :=' pv: RTDSL0:MPA:01 '}
  RTDSL0_MPA_01: FB_MPA := (sName := 'RTDSL0:MPA:01');

  {attribute 'pytmc' :=' pv: RTDSL0:MPA:02 '}
  RTDSL0_MPA_02: FB_MPA := (sName := 'RTDSL0:MPA:02');

  {attribute 'pytmc' :=' pv: RTDSL0:MPA:03 '}
  RTDSL0_MPA_03: FB_MPA := (sName := 'RTDSL0:MPA:03');

  {attribute 'pytmc' :=' pv: RTDSL0:MPA:04 '}
  RTDSL0_MPA_04: FB_MPA := (sName := 'RTDSL0:MPA:04');

END_VAR
```

### Related:

- [FB\\_MPA](#)

## POUs

### FB\_AttenuatorElementDensity

```
FUNCTION_BLOCK FB_AttenuatorElementDensity
VAR_INPUT
  sName : STRING;
END_VAR
VAR_OUTPUT
  fDensity : LREAL;
  bFound : BOOL;
END_VAR
VAR
  fbElementDensity : FB_ElementDensity;
  fDensity_gm3 : LREAL;
END_VAR
IF sName = 'C' THEN
  (* Special-case diamond here *)
  fDensity := 3.51E6; (* C (Diamond) g/m^3 *)
ELSE
```

## plc-summary

### INFORMATION

#### ject

- cls-plc-bergmann-kohzu
- cls-plc-crixs-motion
- cls-plc-crixs-vac
- cls-plc-cvmi-motion
- cls-plc-cvmi-vac
- cls-plc-cxi-fms
- cls-plc-dream-motion
- cls-plc-dream-vac
- cls-plc-ftl-leak-det
- cls-plc-hxx-vonhamos
- cls-plc-kfe-arbiter
- cls-plc-kfe-gatt
- cls-plc-kfe-gmd-vac
- cls-plc-kfe-motion
- cls-plc-kfe-rix-motion
- cls-plc-kfe-rix-vac
- cls-plc-kfe-vac
- cls-plc-kfe-xgmd-vac
- cls-plc-lamp-motion
- cls-plc-lamp-vac
- cls-plc-lamp-vac-1
- cls-plc-las-bts
- cls-plc-las-lps-01
- cls-plc-lfe-arbiter
- cls-plc-lfe-gem
- cls-plc-lfe-motion
- cls-plc-lfe-motion-kmono
- cls-plc-lfe-optics

🏠 / Axes by Project

[View page source](#)

## Axes by Project

This summary was generated Oct 06 2023 at 04:20:40 (-0700).

### pcdshub/lcls-plc-bergmann-kohzu

Version [b4ba7a2](#) - [Documentation](#)

*pcdshub/lcls-plc-bergmann-kohzu*  
Axes

Axis ID	Axis Name
Axis 1	XTAL_SAMPLE_Y
Axis 2	XTAL1_X
Axis 3	XTAL1_CHI
Axis 4	XTAL1_TH
Axis 5	XTA2L_CHI
Axis 6	XTA2L_TH
Axis 7	XTAL3_CHI
Axis 8	XTAL3_TH

### pcdshub/lcls-plc-crixs-motion

Version [8a67cdc](#) - [Documentation](#)

*pcdshub/lcls-plc-crixs-motion* Axes

Axis ID	Axis Name
Axis 1	Axis 1 - Diagnostic Paddle X
Axis 2	Axis 2 - Diagnostic Paddle Y
Axis 3	Axis 3 - Diagnostic Paddle Z
Axis 4	Axis 4 - Objectives X

## DUTs

- DUT\_SATT\_Filter
- E\_PositionState
- ST\_MPA

## GVLs

## POUs

# Development and deployment systems

---

## TwinCAT development environment installs and updates for free

- Managing TwinCAT target versions vs. development environments can be tricky
- We use a centralized development environment and deploy to any PLC via the network
- Uses a hypervisor provided Windows Server instance which is utilized by the whole team *simultaneously*
  - We've seen up to ~10 simultaneous users
  - Originally used a “fleet” of 4 machines which we tried to keep synchronized
    - Very difficult
    - Environments quickly diverged
    - Load balancing doesn't quite work
  - The centralized environment maintains consistency between our engineers
  - Generally runs OK
    - Headcount has grown; issues are starting to occur more frequently
    - This is not a “supported” way of using TwinCAT
  - This machine sits inside a secure enclave
    - Accessed via rdesktop
    - Uses a proxy for GitHub access

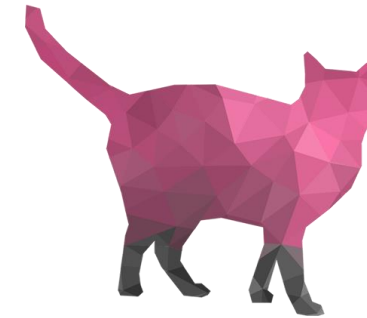
# Testing

Most testing is done via procedure during qualifications, but some is automatic

- Linting, or making sure the ST code is formatted consistently
- Some of our core libraries are covered by TcUnit tests which are embedded in runtime projects included in the library
  - So a developer can deploy a “library” to a PLC and run it and receive a list of the test results
  - PLC code unit tests look a lot like an ordinary software unit test

```
1 FUNCTION_BLOCK FB_Sum_Test EXTENDS TcUnit.FB_TestSuite
2 VAR
3 END_VAR
```

```
1 METHOD TwoPlusTwoEqualsFour
2 VAR
3     Sum : FB_Sum;
4     Result : UINT;
5     ExpectedSum : UINT := 4;
6 END_VAR
7
8 TEST('TwoPlusTwoEqualsFour');
9
10 Sum(one := 2, two := 2, result => Result);
11
12 AssertEquals(Expected := ExpectedSum,
13              Actual := Result,
14              Message := 'The calculation is not correct');
15
16 TEST_FINISHEDC);
```



**TcUnit**  
TwinCAT unit testing framework

```
=====TESTS FINISHED RUNNING=====
| Test suites: 5
| Tests: 17
| Successful tests: 11
| Failed tests : 6
|=====
```

# Automated testing with TcUnit

---

## TcBSD and GitHub Actions holds some promise for Continuous Integration for PLC code

- Currently collaborating with J. Sagatowski on a proof of concept for a continuous integration “pipeline” for automated execution of TcUnit test suites and other code quality checks.
- The concept:
  - An engineer commits a change to a library or implementation project git branch
  - The new code (including tests) is deployed to either a virtual PLC, or a physical testing PLC
  - The tests run, and results are returned to GitHub
  - The results are displayed inline with the branch
  - Can gate the integration of the code change into the main branch by the tests
- Physical PLCs for testing
  - Future may include virtualized PLC instances (TcBSD) for hardware-less testing

# Acknowledgements

---

## ECS Team and our collaborators

- Margaret Ghaly
  - Will give a talk about our vacuum controls systems!
- Tyler Johnson
  - Will give a talk about our laser timing control systems!
- Ken Lauer
  - Posters about TcBSD, our PLC logging system, and an EPICS parser.
- Jakob Sagatowski
  - Originator of the TcUnit framework, extremely knowledgeable of all things Beckhoff
- ESS
  - Original authors of the EPICS device support for our PLCs