ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-WE1BC004

LCLS-II EXPERIMENT SYSTEMS VACUUM CONTROLS ARCHITECTURE*

M. Ghaly†, A. Wallace, SLAC National Accelerator Laboratory, Menlo Park CA, U.S.A

Abstract

The LCLS-II Experiment System Vacuum Controls Architecture is a collection of vacuum system design templates, interlock logic, supported components (eg. gauges, pumps, valves), interface I/O, and associated software libraries, which implement a baseline functionality and simulation. The architecture also includes a complement of engineering and deployment tools including cable test boxes or hardware simulators, as well as some automatic configuration tools. Vacuum controls at LCLS span from managing rough vacuum in complex pumping manifolds, protection of highly-sensitive x-ray optics using fast shutters, maintaining ultra-high vacuum in experimental sample delivery setups, and beyond. Often, the vacuum standards for LCLS systems exceed what most vendors are experienced with. The system must maintain high-availability while remaining flexible and adaptable to accommodate ongoing modifications. This paper provides an overview of the comprehensive architecture and the specific requirements of the LCLS systems. Additionally, it introduces how to utilize this architecture for new vacuum system designs. The architecture is intended to influence all phases of a vacuum system life-cycle, and ideally with the goal of becoming a shared project for installations beyond LCLS-II.

INTRODUCTION

The majority of LCLS experiments and measurements necessitate a vacuum environment as they rely on the absence of any background interference. Furthermore, various optic devices are highly sensitive to vacuum levels, with any level above Ultra High Vacuum (UHV) potentially accelerating the rate of contamination on the mirror surface.

The overarching Experiment Controls System (ECS) vacuum controls system is responsible for controlling vacuum devices and protecting both vacuum components and vacuum itself. Operating at a high availability, the system supports a diverse array of vacuum devices, including gauges, pumps, valves, and other related devices and controllers. Utilizing a single vacuum Programmable Logic Controller (PLC) for numerous devices, the system efficiently handles complex logic necessary to enforce protection interlock requirements. Vacuum interlocks established to prevent the system from entering an unsafe or undesired state. For instance, the system should inhibit the operator from opening a valve when the differential pressure across this valve exceeds a certain limit. Additionally, reactive interlocks are implemented for the closure and isolation of specific vacuum volumes to prevent the propagation of pressure events from adjacent sections.

Furthermore, the vacuum control system interfaces with other systems such as the Machine Protection System (MPS), enabling it to shut off the beam while a beamline valve is closing. This functionality protects the valve from potential damage caused by the beam.

ARCHITECTURE

The ECS vacuum controls architecture is a collection of software libraries, scripts, widgets and a complement of software and hardware tools that are fully integrated cross all layers of the controls stack.

The ECS vacuum controls system design is based on Beckhoff embedded controllers and associated industrial hardware and EtherCAT. The system's software stack includes the Experimental Physics and Industrial Control System (EPICS) layer, the Python layer, and the User Interface (UI) Layer, as illustrated in Fig. 1.

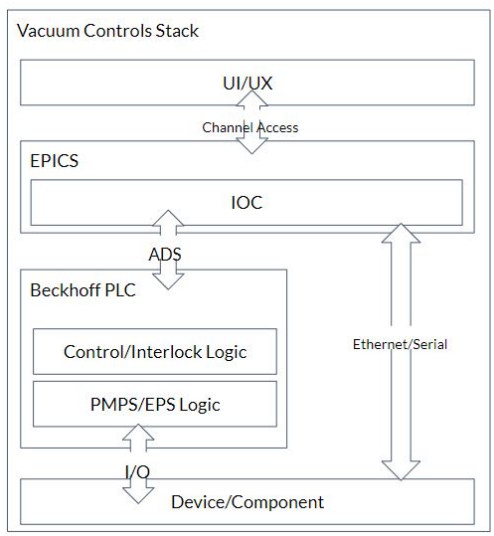

Figure 1: ECS Vacuum Controls Stack.

Device Hardware

The design process of every new vacuum device starts at the connector. Each vacuum device has a corresponding custom cable designed for it. These comprehensive cable design drawings include all the necessary information for cable fabrication, including specifications for the cable type, connector type, and fabrication instructions. Additionally, the cable designs contain pin-outs for the signals needed to control and monitor each device, as illustrated in Fig. 2.

For every device cable a number of I/O terminals are designated based on the device type enabling signals readouts

* Work supported by U.S. D.O.E Contract DE-AC02-76F00515.
† mghaly@slac.stanford.edu

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-WE1BC004

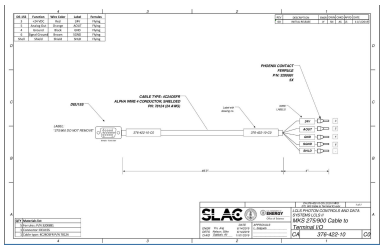

Figure 2: Gauge cable design drawing.

and control. Utilizing EPLAN as the CAD tool of choice for ECS, ECS designed an EPLAN Macro for every device. The macro consists of the aforementioned details and cable termination information as shown in Fig. 3.

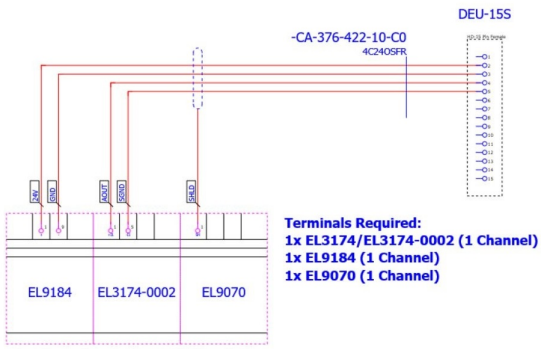

Figure 3: Gauge macro.

The ECS vacuum controls architecture was built with strong focus on comprehensive testing. Developing testing procedures, scripts and tools across all layers of the system stack was a fundamental objective, aiming to streamline the deployment process for beamlines and endstations. Consequently, for the majority of devices, a specially designed test box was designed to simulate the behavior of each respective device. Standardized test instructions are affixed to each test box, as illustrated in Fig. 4, facilitating the verification not only of cable fabrication but also of cable termination. It is important to note that these tests are currently conducted manually, as they have not yet been scripted or automated. Despite this, their implementation has significantly reduced the time required for cable pin-out and signal verification.

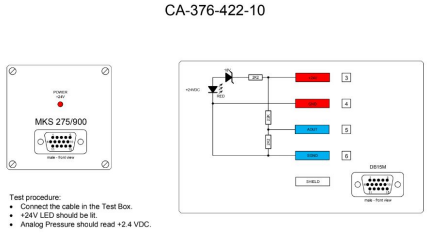

Figure 4: Gauge cable test box.

Custom Library Software

ECS developed its own custom TwinCAT vacuum library in IEC61131 Structured Text. The library is designed with a modular structure, incorporating fundamental operational functionalities and safety interlock logic for each device within a single function block. This approach ensures the encapsulation of logic alongside data variables, bundling them together as a cohesive unit. As a result, the logic, data variables, and I/Os are consolidated and safeguarded from external modifications in the production Programmable Logic Controller (PLC) [1], as shown in Fig. 5.

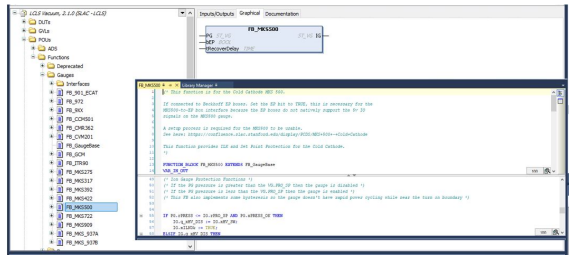

Figure 5: Gauge library function block.

Logging

In addition to the basic operational functionality and interlock logic, the library code integrates a comprehensive logging function. This feature enables the logging of messages at every PLC cycle, capturing essential information regarding device state changes, user actions, and interlock events within the logged messages, as shown in Fig. 6. These log messages are sent through a Logstash and ElasticSearch-based system [2] and can be viewed on Grafana.

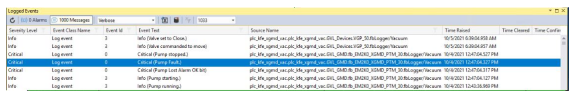

Figure 6: Example of logged messages.

Unit Tests

The library also incorporates a suite of tests based on tcUnit, a unit test type of framework designed for Beckhoff's TwinCAT 3 development environment. It comprises a testing library that can be seamlessly integrated into any existing TwinCAT 3 project [3]. Multiple tests are developed for each function block, guaranteeing the proper functionality of any released library code prior to production. These tests additionally ensure that subsequent modifications to any code components or function block do not break the logic or introduce any bugs. Illustrated in Fig. 7 are test outputs.

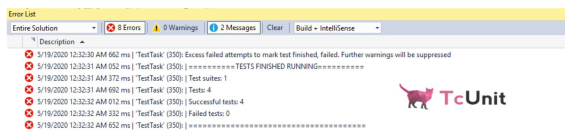

Figure 7: Test suite.

Simulator Library

With a strong emphasis on testing during development and the reduction of deployment time in production, ECS developed it's own TwinCAT vacuum simulator library. Each

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-WE1BCO04

device type in the vacuum library has corresponding simulation function block in the simulation library that emulates the device's functionality with the I/O definition inverted [4]. By leveraging EtherCAT simulation, the project's EtherCAT features are replicated on the simulation computer or PLC without the need for additional configuration.

An entire beamline vacuum project code, encompassing control logic and I/O mapping, can be tested, extending all the way up to the User Interface (UI), even before the procurement of any vacuum devices or I/O terminals.

EPICS Interface

EPICS interface is established through two ways. EPICS Input/Output Controllers (IOC) communicating with the PLCs directly; and EPICS device drivers connected directly to the device controller.

The EPICS-PLC communication is established via Automation Device Specification (ADS) communication protocol achieved via Ethernet connection. The custom vacuum library implements a standard EPICS interface, achieved though TwinCAT pragmas. TwinCAT natively supports a number of pragma attributes. The vacuum library uses a SLAC-defined Pytmc attribute to tag variables and data structures in the PLC code. The Pytmc tool automatically generates the Database (DB) records [5]. Additionally, a special SLAC built ADS-Deploy tool generates a templated IOC.

Additionally, a number of Templated IOCs have been developed for other device controllers, providing access to additional configuration variables and device meta-information through EPICS. Most of these IOCs communicate via StreamDevice or Modbus.

User Interface

The User Interface (UI) generally falls into two categories: custom overview screens and detailed engineering screens.

The custom overview screens are created using the PyDM Widgets library, which is based on PyQt and comprises a collection of symbol widgets. Each device widget is composed of an icon representing the device, a readback and control panel [6]. A unique widget, that matches the conventions of the vacuum schematics drawing, is created for every vacuum device group. A collection of these widgets and their arrangement constitutes a custom overview screen. Using the PyDm Designer, these vacuum widgets can be conveniently dragged and dropped to make the a user interface screen. Furthermore, A global style-sheet is used to customize the appear ace of these widgets based on the values of the Process Variables (PVs) attached to that specific widget. Figure 8 shows an example of a custom vacuum screen.

Detailed screens, also referred to as engineering screens, shown in Fig. 9, are automatically generated. Each vacuum device group has a corresponding ophyd object. A library of LCLS-specific ophyd object device classes, also known as pcdsdevices, has been developed for the various vacuum devices [7]. These Ophyd objects are utilized by Typhos to automatically generate these screens [8].

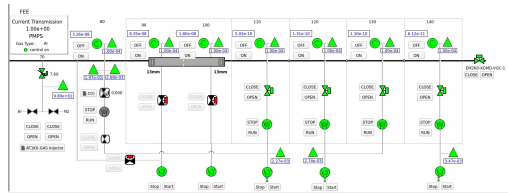

Figure 8: Custom Vacuum Screen.

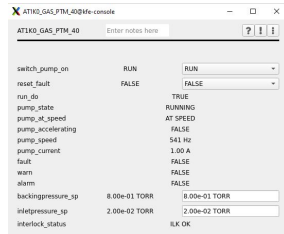

Figure 9: Engineering Screen.

Configuration Scripts

Custom scripts were developed for certain specific devices to simplify the initial configuration setup process. These scripts enable the team to promptly apply the necessary compatible configurations to each device and/or controller, ensuring timely deployment. Some of these scripts generate report files with the device serial numbers for documentation and record keeping purposes.

INTERFACE WITH OTHER SUBSYSTEMS

Within the vacuum library, a crucial feature is the interface and communication between the vacuum control systems and other subsystems, such as motion systems that controls all diagnostics components, as well as the Preemptive Machine Protection System (PMPS). The PMPS protects various beamline components and experiment devices from damage caused by XFEL photon x-ray beam [9].

In certain scenarios, a motion device may need to pass through a vacuum valve. To enable this safely, specific interlock logic is implemented to communicate with the motion system, granting permission to actuate the valve based on the position of the stage. This communication is established through EtherCAT between both subsystems.

Furthermore, in many cases, a beamline gate valve must fault the beam before it enters the beam path to prevent any damage to the valve. To achieve this, an interface with the PMPS system is established, incorporating PMPS logic within the valve function block. This logic preemptively awaits the appropriate beam conditions before the valve is closed. Notably, in this specific scenario, this logic sequence is only executed when the valve is instructed to close by the operator and not in the case of an interlock triggered by a vacuum event. In the event of a vacuum-related interlock, the vacuum controls system faults the beam through direct interface with the Machine Protection System (MPS). That sequence is as illustrated in Fig. 10.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-WE1BC004

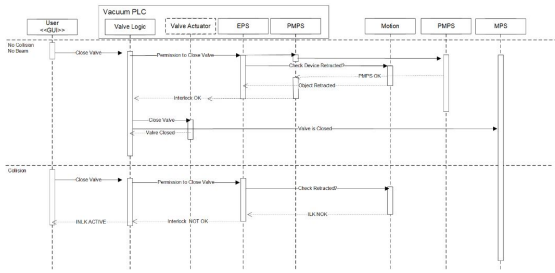

Figure 10: Sequence diagram of subystem-interface.

STANDARDIZATION

Another key goal of the new vacuum controls system was to establish standardization. To achieve this, a Supported Device List (SDL) was created, outlining all the vacuum gauges, pumps, controllers, and valves that are fully supported by the vacuum controls architecture. Additionally, vacuum design templates were developed, providing architecture diagrams for various systems that detail all devices, their expected arrangements, interlock logic, and the current LCLS facility-wide setpoints.

The SDL currently includes over 60 devices, including 24 gauges and gauge controllers, 26 pumps and their controller such as turbo pumps, ion pumps and roughing pumps, as well as over 10 different types of valves including pneumatic, electromagnetic, motorized valves, among others.

The following tables [Table 1] and [Table 2] present some of the specifications of the vacuum controls system:

Table 1: Specifications

Specification	Value
Typical min. measurable pressure	5×10^{-10} Torr
Specialized min. measurable pressure	1.5×10^{-12} Torr
Reaction time of fast shutter system	500 µs-100 ms
Typical pressure measurement delay	120 ms

Table 2: Digital-to-Analog Conversion

DAC Bits	Nominal Range	Unit/Bit	Conversion Time
12	0 V-10V	2.44 mV	0.625 ms
16	−10 V-10V	327 µV	50 µs
16	4 mA-20mA	1.2 µA	50 µs

CONCLUSION

This architecture has been successfully deployed for the LCLS-II project, particularly in the Electron Beam Dump (EBD) and Front End Enclosure (FEE), and has been reused for the Soft-Xray hutches: the Time resolved atomic, Molecular and Optical instrument (TMO), the Resonant Inelastic X-ray Scattering instrument (RIXS) and the Tender X-Ray Instrument (TXI). It was designed to influence all phases of the vacuum system's lifecycle, from design to installation and checkouts, and was intended to be reusable by subsequent projects and third-party integrators. The entire architecture of SDL, templates, software and hardware designs, and tools is meant to enable the straightforward delivery of vacuum controls systems where no additional design work is necessary, only integration and deployment efforts.

ACKNOWLEDGEMENTS

The Experiment Controls Architecture as it stands now, is culmination of years team efforts of the entire ECS team, achieved through close coordination with our stakeholders. Jing Yin made numerous contributions providing essential reviews of the entire architecture and library code, and took over the vacuum library development beyond the LCLS-II project. Ken Lauer developed Pytmc and the ADS-deploy tool that enabled the automation of EPICS IOC deployment, and developed the logging system. Zachary Lentz providing essential guidance and reviews to the ophyd vacuum device library. Additionally, Hugo Slepicka was responsible for the development the PyDM vacuum widget library.

REFERENCES

[1] Experiment controls vacuum system twincat library for LCLS-II, https://github.com/pcdshub/lcls-twincat-vacuum/releases/tag/v2.3.0

[2] K. Lauer, "Centralized logging and alerts for EPICS-based control systems with logstash and grafana", presented at ICALEPCS'23, Cape Town, South Africa, 2023, paper TH-PDP089, this conference.

[3] TcUnit, https://github.com/pcdshub/lcls-twincat-pmps/releases/tag/v2.2.1

[4] Experiment controls vacuum system simulator twincat library for LCLS-II, https://github.com/pcdshub/lcls-twincat-vacuum-system-simulator

[5] pytmc, https://github.com/pcdshub/pytmc

[6] Vacuum Widgets, https://pcdshub.github.io/pcdswidgets/master/vacuum.html

[7] pcdsdevices, https://pcdshub.github.io/pcdsdevices/v8.1.0/

[8] Typhos, https://pcdshub.github.io/typhos/v3.0.0/

[9] A. Wallace, "The LCLS-II experiment controls preemptive machine protection system", presented at ICALEPCS'23, Cape Town, South Africa, 2023, paper TUPDP129, this conference.