ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

PROGRESSION TOWARDS ADAPTABILITY IN THE PLC LIBRARY AT THE EuXFEL

T. Freyermuth*, S. T. Huynh†, B. Baranasic, M. Bueno, N. Coppola, G. Giovanetti, S. Hauf, N. Jardón Bueno, N. Mashayekh, A. Silenzi, M. Stupar, M. Teichmann, J. Tolkiehn, L. Feltrin Zanellatto, P. Gessler, EuXFEL, Schenefeld, Germany

Abstract

In 2011, the European X-Ray Free Electron Laser (EuXFEL) commenced parallel developments of their control system (Karabo) and the Programmable Logic Controller (PLC) library. The PLC library was designed to control basic beamline components and under as set of initial assumptions, the automation component was deferred to the control system layer. After five years of operation, it can be seen that not all initial assumptions scaled well to the operational needs of the facility resulting in limitations hindering progress. Having identified the issues, the PLC development is now focused on providing a more cohesive and adaptable solution. In utilizing the IEC61131-3 (3rd edition) features, the PLC library has been restructured towards a layered architecture with loose coupling between function blocks. The ultimate goal is to achieve a PLC library which is not only test driven and capable of quickly integrating in new devices, but can achieve dynamic linking not only between hardware and software, but also across software devices, aiding the rapid development of more complex hardware integration and higher-level automation.

INTRODUCTION

Programmable Logic Controllers (PLCs) have been historically used to provide control and automation of larger systems, and this is no different at the European XFEL (EuXFEL)[1]. To further optimise the generation of PLC projects, a library has been developed to provide buildings blocks, which are pieced together to form the backbone of a PLC project. While initially developed to reduce development time for PLC projects, the PLC library also incorporates a bespoke communication protocol and provides functionality which over time became less efficient and harder to expand upon. As such, a new version of the library is underway, addressing some of the short-comings of the original:

- A strictly layered architecture that is testable
- Dynamic linking is provided by the TwinCAT Hardware Abstraction Layer (TcHAL)
- An enhanced communication protocol
- Simpler inter-device and inter-plc communication
- Additional tools

Paving the way for dynamic, adjustable and configurable complex automation processes, which can easily be developed and tested, whilst leaving room for future changes in a completely disentangled manner.

* Tobias.Freyermuth@xfel.eu
† Sylvia.Huynh@xfel.eu

CURRENT LIBRARY

The current PLC library is built up of a collection of software representations of hardware devices, which are known as softdevices. During early development, it was a priority to ensure that the rapid deployment milestones of PLC projects were met. The softdevice building blocks aided quick delivery through device instantiation, ensuring conformity amongst hardware devices performing a similar function set, and creating a simplified way with how these devices interact with the Supervisory Control and Data Acquisition (SCADA) system known as Karabo [2]. Using a TCP/IP communication protocol developed in-house, a connection header entailing basic device class information known as a self-description provided the SCADA system with a means to obtain information to aid User Interaction (UI). As the number of PLC projects grew in size, it became relevant to ensure a means for peer-to-peer communication between softdevices, in addition to basic equipment protection. This was achieved through shared variables and interlocks - a set of conditions which when met, would trigger a set of actions. Lastly, an additional tool called the PLC Management System (PLCMS)[3] was developed to aid the automatic generation of PLC projects with linking between the hardware Input and Output (I/O) and the previously defined softdevices. The structure of the current library can be seen in Fig. 1.

FUTURE LIBRARY

Building on from the lessons learned thus far in regards to the existing PLC library, this section will explore the changes planned for the next version of the EuXFEL PLC library, and how they work and will be integrated.

PLC Library Goals

Whilst the current library is functional, it is hoped that the many limitations that are currently imposed due to historical design decisions can be overcome in the next version. As many of the previous requirements remain valid, this section details on how they can be enhanced or evolved. Additionally, new requirements, which will facilitate to provide better building blocks for future PLC projects are introduced.

A Complete Self-description of the PLC A highly important feature of the current library is the self descriptive nature of the PLC. When a connection is made to the PLC (via a TCP client), the PLC will send a description of all the devices currently on it. While incredibly useful, the self-description is still lacking some features for true com-

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

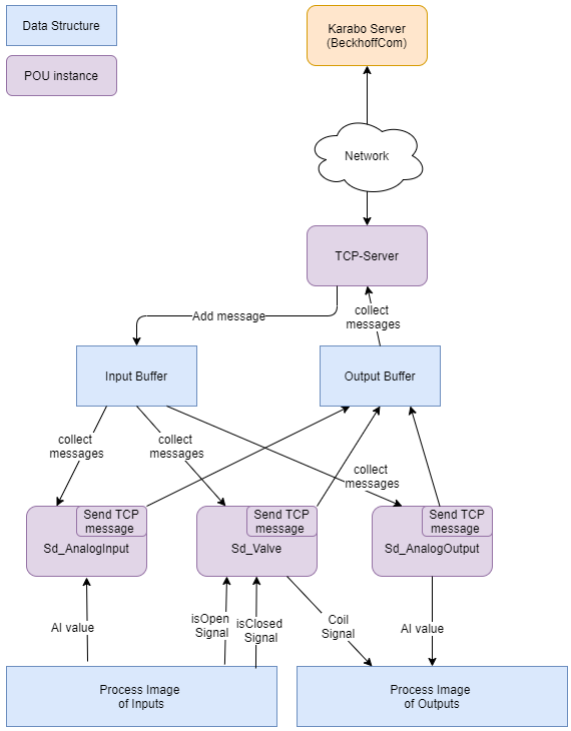

Figure 1: Structure of the Current Library.

pleteness. Firstly, it would be helpful for a textual description of each command and property to also be sent to the control system. This is currently only available in the online documentation of the library and not linked to the control system where it is used. Additionally, a description of the error states should be added provide context in comparison to the existing error code. These additional features could overcome the need for manual development of the corresponding devices on the control system, known as "Karabo devices".

Partial Regeneration and Handling of Custom Code in Projects The process of building a PLC project is automated to a high degree at EuXFEL, but the way it is currently implemented requires a complete re-build of the entire solution, overwriting any code that had been previously added. Despite the feature-richness of the tools available, this complete regeneration process has caused many issues when only a minor change is required. Due to these limitations, the future library and associated tools need to be more flexible, providing the ability to generate one or a set of sections of the PLC project, or a particular feature set.

Additional Abstraction of the Library The current version of the library does not utilize abstraction layers in the architecture, and so all softdevices have to deal with all the various functional layers that exist in the library. While developing a softdevice, the developer has to bear in mind very low level details of the terminals that might be connected to the softdevices while at the same time considering high level aspects, such as how to structure the interface for the end user, who interacts with the softdevice through the control system. To faciliate understanding and implementation of softdevices, it is planned to add a layered architecture to the future library. This should reduce the complexity of softdevices and thereby reducing the likelihood that bugs are introduced, while producing testable code. A useful feature when tackling new softdevices. In addition to this, it is foreseen that a higher layer to cover additional automation of systems and processes e.g. to handle beamline components may be desired. See Device Abstraction Layer.

Relinking on Runtime There have been instances when during operational periods, hardware (EtherCAT terminals/channels) had to be reallocated. Within the current library, hardware I/O are directly linked to the softdevice. This is a severe restriction due to the fact that most of the links can only be relinked with a rebuild and redeploy of the TwinCat configuration - something which is not possible during operation of the system. The future library will overcome this limitation through the additional TcHAL and an interface manager. All EtherCAT terminals will be abstracted by a dedicated call per terminal type, where all terminals and channels will also have a defined interface. These interfaces will on runtime be registered with the "interface manager" Programming Organisation Unit (POU), implemented as a singleton within the TcHAL, which, is responsible for handling both the interfaces to the TcHAL and the propagation of information to any compatible softdevice.

Generate Terminal I/O Linking Map Extending on from the ability to relink hardware during run-time, the information related to the current linking should be obtainable such that any previous wiring diagrams can be amended to reflect reality. Currently, there is no way to keep track or compare what has been configured, to what is expected. Not only does this hamper debugging, but it also adds confusion and produces unreliable documentation. Further utilisation of the configuration service tool will enable the exporting of this information.

Simplified Peer-to-Peer Communication Functionality for communication between softdevices already exists, however it does so in a rather complicated manner and involves a deep understanding of the device implementation. Softdevices are required to send "pairs" to other devices to issue commands or read and write requests, more or less utilising the same interface methods as the control system. This is not developer friendly or easily human-readable. An alternative method is to use a globally defined shared memory, which enables softdevices to read a predefined set of values for any softdevice on the system, again, requiring in depth knowledge as it is up to the developer of a softdevice to decide which properties are put into this shared memory. As this information is not self describing, this is a fragile way to exchange information, requiring type casting of each property, while considering that a data type might change over time.

Testability of the Library The structure of the current library makes it very hard to test individual aspects, functions and components, which in turn makes it practically

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

impossible to unittest. This leads to a situation where a significant amount of manual testing is required, which is time-consuming and highly error-prone. With the proposed TcHAL, softdevices will become largely decoupled from the hardware and also from the SCADA system communication via the means of dedicated wrapper classes. As such, softdevices can be tested within each interface, and interaction between each level of abstraction can be completed with its own test case, opening up a better way to perform testing.

Resolve the Interlock Feature of the Current Library The current library implements a feature called "interlock", which allows a device to react in defined ways upon a set of predefined conditions. The interlock feature is intended to be used for equipment protection, however, over time has been extended to perform automation on the PLC as well, which is needed where a specific reaction time has to be achieved. The implementation of interfaces as well as layers within the future library will allow us to overcome the rigidity of the feature as currently implemented. The new structure will enable peer-to-peer interaction in a much more diverse and safer fashion. Combined with Simplified Peer-to-Peer Communication, it would be possible to write devices where the "interlock" feature becomes redundant.

Simplified Device Development Due to the implementation of layers and interfaces that are tailored to the needs of the PLC device developer, developing devices will become simpler and more efficient. This will open up the ability for "beamline component" experts to write their own devices, initially within the projects, then, eventually implemented within the "beamline component layer" of the library. Needless to say, the transformation of specific devices implemented by a "beamline component" expert within a project into a generic device becoming part of the future library, will be a joint effort of the "beamline component" expert and a PLC library developer.

Self-contained PLC System In order to make the PLC systems at EuXFEL self-contained, it is necessary to implement a solution to store and restore configuration data of PLCs on the Industrial PC (IPC) that is running the PLC. To be able to restore the previous configuration state of the PLC after a restart as well as after an update.

Structure of the Library

The structure of the future library will consist of at least two different layers; one for the hardware (field bus terminals) abstraction, provided by TcHAL, and one for the softdevice implementation, otherwise known as the Device Abstraction Layer (DAL). Communication between components across different layers will be carried out through interfaces, a concept within the IEC 61131-3 3rd edition which was not yet available when development of the initial library began within the TwinCAT 2 environment. The structure of the current library can be seen in Fig. 2.

Hardware Abstraction Layer The Hardware Abstraction Layer (HAL) provided by the library TcHAL is intended to abstract the EtherCAT fieldbus terminals. The TcHAL provides POUs for each fieldbus terminal used at EuXFEL. It will configure the terminal according to its configuration, provided through a configuration interface of the terminal-POU. Data will be periodically exchanged with the terminal via the EtherCAT master and translated into meaningful units, preferably in International System of Units (SI) where possible. Other components of the system will then obtain these values as required. These interfaces will be tailored to the needs of components of the DAL, see Device Abstraction Layer. As a result, the HAL aims to provide a clean and easy to interface into the hardware for the DAL device developers.

Device Abstraction Layer The DAL which sites above the HAL, will be the second layer of the future library as seen in Fig. 2. It is on this layer that softdevices are to be implemented, without the complexity of any hardware configuration.

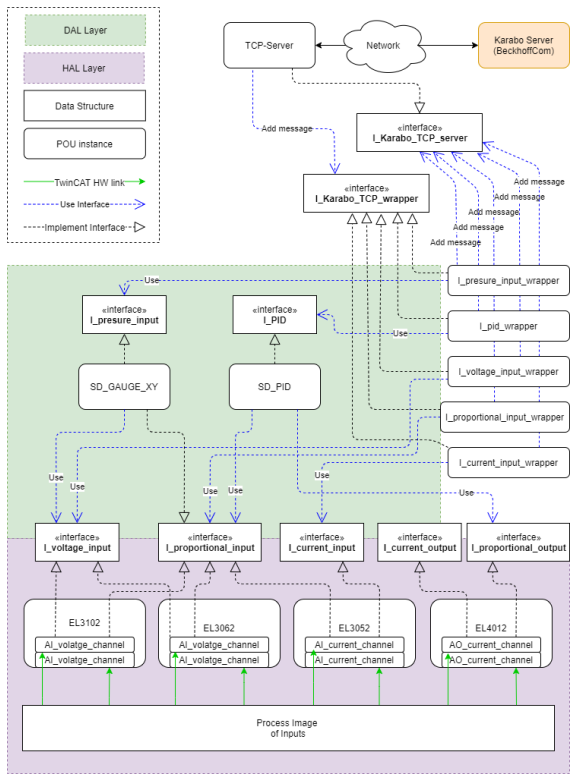

Figure 2: Structure of the Future Library.

Communication Protocol

The future library will implement an extended version of the custom TCP-protocol of the current library. This is predominately required to overcome the limitations seen during operation, specially with large PLC-projects which encompass hundreds of devices.

Extended Self-Description The current implementation of the self-description sends a list of all softdevices on connect. It also sends a description of each softdevice-class including all properties and commands, with the name,

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

data type, access level, unit and unit prefix of the associated properties and commands on a device-instance or object level, and also for any type variations which deviate from the standard device-instance. The future library will extend this self-description with a description of each command and property alongside the name, aimed at providing additional information to the end user. Additionally, a mapping between available commands and properties for a given Karabo state is planned. Beyond the existing message types, an additional message type to define error codes with their associated description is a future feature.

Separation Between Self-Description and Operation The current version of the library implements the self-description of the system on the same server (port) as operationally related communications. This is not ideal, since the self-description is sent to the client on connect, not on request. When this occurs, it creates a notably large load on the client using a protocol which is not optimized for large amounts of data, but rather for small amounts of data at high data rates. It is planned to differentiate the data required for operation: device state updates, reads and writes, and command requests; and the self-description: description of all devices on the system and associated details such as which commands and properties are available. The new self-description data will be provided on a dedicated port, with JavaScript Object Notation (JSON) being the intended format. This fits in nicely with the other planned changes given its amenability strings.

Additional Tooling

The current PLC library uses a set of tools which where predominantly implemented in Python, and one in C#. There one commonality is that they are not part of the development environment of the PLC developers. Some tools are part of continuous integration (CI)-pipelines, and others are to be used from the command line on dedicated development machines. Due to the fact that these tools were not envisioned at the beginning of the current library development, they rely on unconventional ways to retrieve the necessary information from the library. Much of this information is gathered through several iterations of source code parsing, which makes the "interface" quite fragile. Given that the code and/or comments written are not checked by the PLC compiler (IDE) or any subsequent build tool to ensure adherence and compatibility, this process becomes highly susceptible to error. This is exacerbated by the fact that errors may not be caught until after the various CI pipelines have run, or only during PLC project builds, making the entire process tedious. In order to provide a more convenient and comprehensive solution for PLC developers, it is aimed to have these tools integrated into the PLC IDE via plugins to Visual Studio. In conjunction with the other changes planned for the library, the parsing of source files will become redundant. Ultimately, this means the tools will be re-written in a .Net language as an overarching "Configuration Manager" as well as a "Development Toolbox".

Check and Configure Connected Hardware The "Configuration Manager" will be implemented as a tool to check the configured hardware of a TwinCAT project. It will retrieve all the necessary information with how to set up each terminal of the project, based on the requirements of the TcHAL POUs in the selected EuXFEL PLC library. The settings are to be applied to the terminals, skipping over and flagging those that are unsupported or incompatible.

Generate and Link TcHAL POUs In order to generate and link all TcHAL POUs according to the hardware configured in a TwinCAT project, the "Configuration Manager" will contain a tool that provides the user with an easy-to-use interface.

Apply EuXFEL Terminal Naming Convention Currently, there are two different naming conventions at EuXFEL defining how to name automation components such as fieldbus terminals and connected equipment. These naming conventions are reflected within the hardware configuration of TwinCAT projects and in the hardware abstraction. The "Configuration Manager" plugin will contain a tool, that will allow PLC experts to apply one of these naming conventions to hardware configurations of a TwinCAT project comfortably. This will reduce the work while using scanned hardware, in this case TwinCAT applies generic names to all scanned components, or switching from one to the other naming convention.

Generate Device Skeleton To provide consistency across the future library and projects with custom devices, a tool to support device development is planned as a part of the "Development Toolbox" plugin. It will allow the user to generate a device skeleton on any layer (see Additional Abstraction of the Library) of the future library (or project). It will give the developer the opportunity to define which interface(s) should be implemented and which name the device should have. It will then generate skeleton code for the POU adhering to the EuXFEL PLC library naming convention, as well as boilerplate code to implement methods and commands.

Generate Karabo Wrapper Class This tool which is part of the "Development Toolbox" plugin, is to provide the user with a POU template generator, which can be pointed to an interface and is made available to the SCADA system in order to ease implementation of the interface wrapper class.

EuXFEL PLC Linter Expanding upon the naming convention tool, an additional feature will be a check function which can be performed across the entire code base. It is intended to be added to the "Development Toolbox" plugin.

Architecture Violation Checker In the long run, a feature to define architectural rules will also be added to the "Development Toolbox" plugin. Additionally, a checking feature will be provided to ensure the code adheres to the defined rules.

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

Configuration Management

The current library does not have a proper concept to handle persistent configuration data of softdevices. It is envisioned to implement a dedicated configuration interface on every POU that is required to handle persistent data. On the IPC running the PLC, a service will be implemented using Beckhoff's ADS-protocol to retrieve and restore data as requested by the POU. The data will be stored on the IPC with the possibility of an additional backup in the event that the IPC crashes. This approach will not only allow the system to retrieve configurations after a restart or a power cut of a PLC, but also after an update or partial reconfiguration of a PLC.

MIGRATION STRATEGY

It is intended to perform the refactoring of the current library in four major steps while ensuring as much backwards compatibility with the currently running projects, build tools and finally, control system, as is feasible. If a new feature breaks compatibility, as is the case with the evolved communication protocol (between the PLC and the control-system), a migration strategy has to be formulated, to ensure adequate uptime across all involved systems.

Hardware Abstraction Layer

The hardware abstraction layer will be added as soon as all the terminals used at EuXFEL have been implemented in the new library. As the existing softdevices are linked directly to the hardware, all softdevices will have to be migrated one by one to adapt to the new structure and interfaces provided by terminal POUs of the TcHAL. To aid this process, a compatibility adapter will be made available for the existing devices, linking them to the new interfaces on the other side. These adapters will be removed once all softdevices have been migrated to incorporate in TcHAL interfaces.

Layers and Interfaces

In order to establish a layered architecture in the future library, it is necessary to refactor all softdevices to implement commands and properties via methods organized in interfaces. The refactoring can be completed softdevice by softdevice, ensuring no unintended effects on the overall system during the migration period. This will make it possible to add the next layer onto the DAL. The TcHAL Hardware Abstraction Layer will make up the second layer.

Updated Communication Protocol

Lastly, migrating to a new communication protocol is planned once the design has been completed and a client is available. As soon as the self-description service, which is serving the JSON self-description is done, the existing implementation of all softdevices will be changed to not send a self-description on connect anymore. The structure of the future library enforces a decoupling between the softdevice and communication component to the control system. All new softdevices will be implemented following this approach, whereas the old devices will get adapters until they are migrated internally.

Add Configuration Persistence

Softdevices that need to store/restore configuration parameters will make use of configuration interface they define, being implemented by configuration-devices. These interfaces will be passed into the softdevice via a reference, which will allow the softdevice to figure out if the interface is implemented by a configuration-devices on runtime. If this is not the case, the softdevices can make use of the old hardcoded configuration values, which are passed via inputs. This approach will make a piecemeal migration possible, softdevice by softdevice.

SUMMARY

Through several years of refinement, it can be seen that the existing PLC Library requires a significant restructuring. It is hoped that though refined abstraction layers and advanced tooling, we can achieve an easily (re)configurable PLC library which can provide further automation and efficient development and deployment.

REFERENCES

[1] T. Tschentscher *et al.*, "Photon beam transport and scientific instruments at the European XFEL," *Appl. Sci.*, vol. 7, no. 6, p. 592, 2017. doi:10.3390/app7060592

[2] S. Hauf *et al.*, "The Karabo distributed control system," *J. Synchrotron Radiat.*, vol. 26, no. 5, pp. 1448–1461, 2019. doi:10.1107/S1600577519006696

[3] S. Huynh *et al.*, "Automatic Generation of PLC Projects Using Standardized Components and Data Models," in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 1532–1537. doi:10.18429/JACoW-ICALEPCS2019-THAPP01