ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-THAPP01

AUTOMATIC GENERATION OF PLC PROJECTS USING STANDARDIZED COMPONENTS AND DATA MODELS

S. T. Huynh[†], H. Ali, B. Baranasic, N. Coppola, T. Freyermuth, P. Gessler, N. Jardón Bueno, M. Stupar, J. Tolkiehn, J. Zach, European X-Ray Free-Electron Laser, Schenefeld, Germany

Abstract

In an environment of rapidly expanding and changing control systems, a solution geared towards the automation of application dependent Programmable Logic Controller (PLC) projects becomes an increasing need at the European X-Ray Free Electron Laser (EuXFEL). Through the standardization of components in the PLC Framework, it becomes feasible to develop tools in order to automate the generation of over 100 Beckhoff PLC Projects. The focus will be on the PLC Management System (PLCMS) tool developed to achieve this. Provided with an electrical diagram markup (EPLAN XML export), the PLCMS queries the database model populated from the PLC Framework. It captures integration parameters and compatible EtherCAT fieldbus hardware. Additionally, inter-device communication and interlocking processes are integrated into the PLC from a defined user template by the PLCMS. The solution provides a flexible and scalable means for automatic and expedited deployment for the PLC control systems. The PLCMS can be further enhanced by interfacing into the Supervisory Control and Data Acquisition (SCADA) system for complete asset management of both PLC software and connected hard-ware across the facility.

INTRODUCTION

The European X-Ray Free-Electron Laser (EuXFEL) is a research facility that aspires to provide high brilliance X-ray beam for user experiments at six end stations with diverse experimental capabilities [1]. The facility is currently home to over a few thousand devices within the multiple photon beamlines, which are to be controlled remotely through the use of Programmable Logic Controllers (PLC).

As such, the PLC team was formed to bring about the design, implementation, deployment and commissioning of over a hundred PLCs with the various hardware components installed and/or proposed. Additionally, the PLCs are to be interfaced with the proprietary Supervisory Control and Data Acquisition (SCADA) system; Karabo [2], to provide a user interface for control.

In order to meet the demands relating to the rapid deployment of multiple PLC projects, it became apparent that an automated solution was required to compile and build the various PLC Projects. As such, the PLC Management System (PLCMS) tool was developed. Currently, the PLCMS performs several functions, all aimed to assist and expedite the existing manual processes related to PLC deployment. These can be largely divided into two categories: the PLC Framework and the PLC Projects, both of which are explored in more detail throughout the paper.

Firstly, the rationale behind the PLCMS will be introduced, followed by the functionality of the PLCMS, and how some of the shortcomings within the existing PLC generation process are addressed and resolved. Lastly, some of the ongoing developments to further enhance the PLCMS and its capabilities are explored.

THE PROBLEM

Since the inception of EuXFEL, there has been a rapid campaign to bring the facility online. In order to provide a facility where scientists can come to conduct research where the only heavy lifting comes down to a mouse click, many PLCs were required to be commissioned and deployed within a short period of time.

The PLC projects are designed around the infrastructure, whereby hardware components which belong to a similar sub-system, are grouped together and allocated a PLC. Each project is subsequently composed of standardised "soft" components called softdevices which are essentially a software representation of the hardware component within the PLC. As the PLC Projects often contain over a hundred instances of softdevices, with various initialisation and configuration parameters, this process would become tedious, time-consuming and error prone.

Utilising a library of the aforementioned softdevices would thereby enable device instantiation rather than continuous creation, and also provide a means for consistency. To be useful however, the library needs to maintain a standardised structure or template across all the softdevices. As the library evolves however, keeping track of changes within each softdevice across multiple versions, deployed across an array of PLCs becomes increasingly challenging. Adjustments to interlock parameters, initialisation values, configuration parameters, or hardware reconfiguration of softdevices; all changes that are applied on the PLC project layer, add to the complexity of maintaining these PLC projects by hand. It would be a fallacy to continue without another means for the design, development and generation of PLC projects that would achieve the results within a more effective and efficient manner.

THE PLCMS

The PLCMS has been developed using Python 3 in addition to Object-Relational Mapping (ORM) to interface with an SQL database backend. The PLCMS performs multiple functions, aimed to expedite the PLC framework and project generation processes in order to cope with the number of ongoing changes and vast quantity of PLCs used throughout the facility. Additionally, it provides the ability

† sylvia.huynh@xfel.eu

to analyse and capture a snapshot of the framework library across the different versions as they are developed. The ability to perform integrity checks through data mapping is taken advantage of by the Continuous Integration, Delivery and Deployment (CI/CD) functionality offered by GitLab [3].

The development and generation processes are illustrated in Fig. 1 and Fig. 2 respectively.

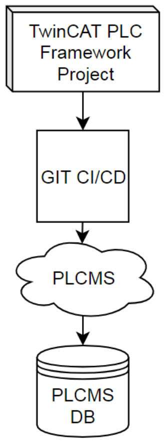

Figure 1: PLCMS Development Process.

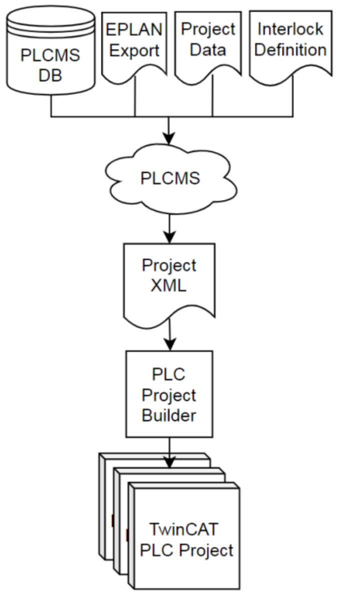

Figure 2: PLCMS Generation Process.

PLC Framework

The framework is developed in Beckhoff's TwinCAT 3 as a separate PLC project, and contains several classes or Program Organization Units (POUs) which represent the softdevices. The framework is compiled into a library file which is then used within all the PLC projects. Composed using Structured Text[1], the framework library is designed in such a way that the functionality of the softdevice is abstracted from the linking of hardware Inputs and Outputs (I/O). This accommodates flexibility in the physical implementation of the hardware. While the implementation of each softdevice can vary wildly; from procedural to finite state machines, the framework enforces a structural template. This is broken down into major aspects: global properties and softdevice specific templates. Combined, they aid both automation and data integrity of the subsequently compiled framework library leading into the PLC project itself.

Global Properties Facets of the framework library which are shared across all softdevices are incorporated into the global structure. This ensures that all softdevices developed will reference the same set of data structures, types or data sets. This includes:

- Access rights
- Units – prefixes and suffixes
- Error codes
- States
- Data types
- EtherCAT fieldbus terminals.

Softdevice Structure As the softdevices provide the interface to the SCADA system for remote control, they too adhere to a defined template. This ensures that they take on the same form and are compiled with the same structure. This includes:

- Softdevice functional commands
- Softdevice attributes
- Instantiation parameters as required
- Error code mapping
- Available set of hardware signals
- Compatible EtherCAT terminals for each defined I/O signal
- Device inheritances

Database Models

The PLCMS database comprises a SQL database backend, designed around the relational data model introduced by Ted Codd [4]. PostgreSQL was selected for being closer to the ANSI SQL standard compared to other open source databases [5]. Nonetheless, any transactional SQL database is both adequate and easily adaptable due to the use of ORM.

The database has been designed around the format of the framework library. After a new version is released, the PLCMS takes a reference to the TwinCAT framework project, parses the PLC code from the associated library into manageable relational datasets, and inserts the information into the PLCMS database. As such, the database is essentially a data capture of the framework library.

In utilising a relational data model, it is possible to ensure data integrity and structural integrity of the softdevice interfaces through data mapping. It also provides digestible bite sized relationships between all the aspects that make up a softdevice.

Through consistent naming conventions, it becomes clear which properties or components of a softdevice reference a global structure, data set or property.

1 Structured Text is the preferred IEC61131-3 language used at EuXFEL.

To aid the ongoing development of the framework library, the PLCMS database is made up of two schemas, one for production and another for development and testing.

Having all of this information stored within a database opens up the possibility to perform comprehensive data analysis on the framework library. This includes the automatic generation of change logs between release versions, generation of structural documentation, the ability to query aspects of the data model to determine inconsistencies or extract an array of data with a simple SQL query.

A simplified diagram of the PLCMS database can be seen in Figure 3.

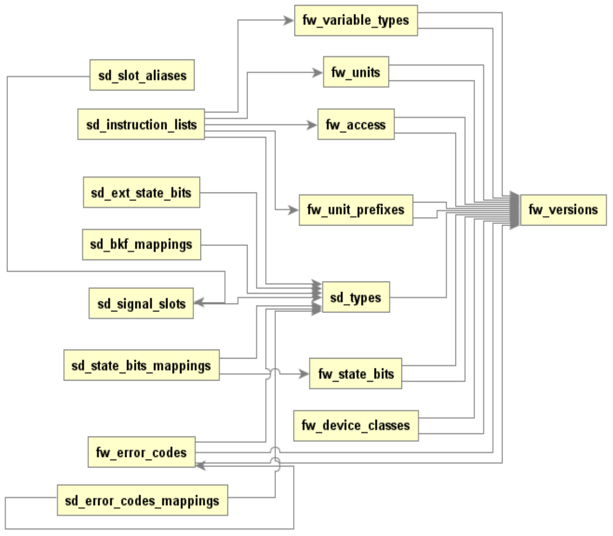

Figure 3: Simplified PLCMS database.

Continuous Integration, Delivery and Deployment

In having a framework library that is constantly evolving, it is necessary to maintain some level of version control. To address these needs, GitLab is used, along with the entire GitLab toolset.

As the framework is developed in Structured Text, GitLab is able to treat the TwinCAT files as standard source code. From here, with an appropriate ".yml" file, pipelines are set up and configured. The pipelines are built to act between the development branch and pre-release and from pre-release to release.

When a new version of the PLC framework project is ready for release, the pipeline is triggered automatically to take the PLC framework project and insert it into the PLCMS development database.

Regression tests in the form of integration tests are also completed within the pipeline and any failures at this stage are addressed manually. Regression testing is conducted as follows:

1. The PLC framework project is taken and compiled into a TwinCAT ".library" file.
2. The library file is installed within a PLC project designated for regression testing.
3. The regression test system is made up of two PLCs: One behaving as a standalone test PLC project utilising the framework library, and a mirror copy, acting as a hardware simulator for feedback.
4. Tests are run on all the softdevices to check access rights and that the defined attributes match up to the expected responses
5. Integration test are run on specific softdevices to ensure functionality and softdevice behaviour is consistent with the design requirements.

Independent of the regression testing within the pipeline, it is also possible to develop tests outside of this for individualised use during the developmental stages of a softdevice.

Upon the success of the regression tests, the pipeline between the pre-release and release comes into play.

Here the framework library:

- Is added to the PLCMS production database
- Documentation within the PLC framework library project is extracted and used to generate the documentation for the framework library version
- Generated documentation is deployed onto an internal ReadtheDocs [6].
- Is compiled as a TwinCAT ".library" file
- Is exported onto a server which is accessible to all the PLC Projects used at the multiple beamlines and endstations.
- Release notes are generated for the deployed library version

While further automation is certainly possible, the current implementation is nonetheless a convenient way to go from a TwinCAT project which contains the framework, to a compiled library file, ready for integration into other PLC projects.

Translating Wiring Diagrams

During the construction and design phase, the PLC allocation and the wiring design for the hardware up to the PLC has been defined using EPLAN.

Following collaboration between EPLAN and Beckhoff, it is possible to store terminal details of both Beckhoff and other EtherCAT based vendors within EPLAN. It is also possible to export wiring diagrams into markup languages – in this particular case, XML, - which can be better translated and handled by other programming languages. Similar to the file format required by Beckhoff's XCAD Interface.

This naturally leads the utilisation of EPLAN to map terminals to a collection of signals belonging to softdevices akin to that defined within the PLC Library. As a result, an EPLAN export not only contains the hierarchy of the PLC, its terminal configuration and EtherCAT addressing, but also the signal names and instance names which make up the collection of softdevices within the PLC.

In the same manner that the PLCMS is fed a TwinCAT Library PLC Project and breaks it down into relational data sets, it can inversely read an XML export defined as a hierarchal tree, and build the same relational data set which, when married together with the framework library data, is able to compile the building blocks of a PLC Project.

Automatically Generating PLC Projects in XML

Provided that the required PLCs are defined in EPLAN, exports are generated and placed into a version control system. From this location, the PLCMS is provided with the EPLAN export and performs the following:

- Extracts the required framework version
- Builds the PLC I/O tree from the provided hardware hierarchy
- Checks that the EtherCAT fieldbus hardware information defined in EPLAN is consistent with the approved and supported list with the selected PLC library version from the project data
- Gathers a list of softdevices which are to be instantiated within the project

For every softdevice instance which is to be included in the PLC project, the PLCMS will query the library database to:

- Ensure the softdevice is supported
- Check that all signals in the softdevice have also been defined within the EPLAN export with an exception for optional signals
- Ensure that each signal is of the correct data type
- Ensure that each signal is connected to a compatible and approved EtherCAT terminal
- Connect each defined softdevice signal to a corresponding hardware I/O signal.

There are a few scenarios where the PLCMS will perform automatic overrides to the provided data when the requirements have not been met. When this occurs, a warning is added to the output log, likewise for any errors that are encountered. Some of these scenarios are described below:

- For softdevices which are not approved or available within the selected framework library version: the PLCMS will break the collection of I/O of the softdevice into a set of simplified components such that the project can still be generated.
- The PLCMS will highlight any discrepancies with signal addressing.
- Missing signals within a softdevice will be highlighted. E.g. If a valve requires two signals but only one is declared.
- Broken links within the terminal hierarchy are highlighted

With the above steps completed, the PLCMS creates another XML file which can be utilised to generate the Beckhoff TwinCAT project.

As all the aspects of the softdevice are available within the PLCMS, several integrity checks are applied during the creation of the PLC project XML. This catches a large number of errors which may have arisen during the planning or wiring phases. It also ensures consistency through the use of standardised softdevice templates across all of the PLCs at the facility. The main benefit of the PLCMS comes from all of the inherent checking and instantiation of softdevices which is no longer a manual task.

While PostgreSQL may not necessarily be optimised for connection and transactional times, creating a PLC project XML of approximately 1300 signals, originating from 250 instances of softdevices would take around 30 s to complete.

Beckhoff Automation Interface

Beckhoff provides an eXtended Automation Engineering interface (XAE) [7], which enables the creation of TwinCAT PLC projects generated from several scripting languages. In our case, a PLC Project Builder was developed in C# to interface with the XAE in order to create the final TwinCAT PLC Project.

The PLC Project Builder consumes the PLC Project XML file generated by the PLCMS, and uses this file to build the TwinCAT project with the defined I/O, linking and softdevice instances.

There are several major benefits to this, which are all capitalised heavily on within the creation of PLC projects at EuXFEL. Whether a PLC contains a single instance of a softdevice or 500, the total duration required to generate the PLC project falls to under a minute. Not only does this save a large amount of time, but with all the additional checks to ensure compatibility, completeness and data integrity, the likelihood of errors is also significantly reduced.

Extended Features

Further to the instantiation, mapping and linking of softdevices on a PLC, there are several other fundamental aspects which are required in order to provide a functional system. These include: interlocks, configuration and initialisation of parameters, and peer-to-peer communication.

During the generation of the PLC Project XML file, the PLCMS will also locate and read in auxiliary files pertaining to the extended features and incorporate them into the final PLC Projects. Data integrity checks are also applied to ensure correct data type usage, naming conventions and existing referencing.

Interlocks Combined with the equipment protection system, each of the beamlines and instruments have additional rules, relating to the photon beam properties and vacuum system integrity. These rules or conditions are defined by a team of experts within a Microsoft Excel spreadsheet, describing which properties from various softdevices or a particular status will trigger a series of protective actions once a defined threshold has been exceeded or met.

Configuration and Initialisation Parameters As a single softdevice template can be used to accommodate several different hardware models or vendors; configuration and initialisation parameters will differ from instance to instance.

These configurational, operational or initialisation properties are defined on an as-need basis for each softdevice instance. As some of these properties are only determined after commissioning, they are saved into a separate XML file referred to as the Project Data input file, containing the softdevice instance name, the property name and required value.

Peer-to-Peer Communication Occasionally, a softdevice from one PLC will require information from another PLC in order to function correctly. Whilst the infrastructure design attempts to reduce or prevent this situation from occurring, the PLCMS requires a method to handle the occasional peer-to-peer communication request. Note: Peer-to-Peer communication requires a minimum of two PLC cycles, resulting in a higher time cost which is undesirable.

Similarly to the configuration and initialisation parameters, softdevices which require data from an external PLC or vice versa are created as virtual softdevices and their details are defined within the project data. The PLCMS will parse the virtual device and incorporate it into the PLC project as required.

FUTURE DEVELOPMENTS

Whilst the PLCMS is a convenient tool to aid PLC project generation through automation, there are still many aspects which can be improved or further enhanced, some of which are currently in development but not yet in production.

PLCMS Projects Database Expansion

Currently the PLCMS only stores static data related to the Library version. Ideally, the database will also store static information relating to each PLC in production – from the names of all the softdevice instances, the EtherCAT addressing, the number of PLCs and which instantiated softdevice can be found where.

After the PLCMS reads in the EPLAN export file, it can have the option to save the data directly into the database itself. This will readily enable the tracking of PLC Projects as they evolve and provide a means for data analysis regarding the hardware assets which are connected to the PLC without being dependant on any proprietary PLC software. Additionally, the ability to identify where a softdevice type has been implemented will aid future roll out of updated PLC Projects when the softdevice template has been updated or evolved.

Web Interface

To increase usability, an improved Human Machine Interface (HMI) web interface can be designed as a front end to the PLCMS. Currently, a series of scripts are run in order to complete the PLC project generation tasks.

Additionally, the defining of interlocks could be relocated to the web interface. The PLCMS should be able to display a list of the available softdevices and their associated properties and functional commands. This will aid the interlock definition design process greatly as it is not always clear which instantiated device originates from which PLC, or what properties or functions are available.

Difficulty relating to versioning is also a major drawback whenever Excel spreadsheets are concerned. By deprecating this step using a web interface, versioning can be handled easily, along with change logs relating to historical trigger conditions and actions.

Configuration Settings Integration

While there is currently a crude method to store configuration, initialisation and operational settings, it would be much more elegant and convenient if the PLCMS was able to store and read the settings directly from the database.

Having a set of default properties for each hardware model or vender located within the database will also enable the PLCMS to extract and configure the correct settings while the EPLAN export is being processed. Conversely, if a property on an instantiated softdevice has been changed after the PLC Project is active; the parameters will be lost after a restart or power loss. Having this information stored and accessible from the database would provide a means for it to be automatically updated and readily extracted.

SCADA Integration

Furthermore, by integrating the existing SCADA system with the PLCMS, changes to any important parameters, functional or otherwise made via the SCADA system can be written directly during runtime into the PLCMS, preventing any critical data loss on update or outage.

Database Independence

Currently the PLCMS takes information from the PLC Library project and stores it into the database. When generating the PLC Project XML, the PLCMS again queries the database, several times, extracting information as required. This dependency results in an expected uptime requirement for the PLCMS database and a reliance on a database connection. If the database aspects of the PLCMS were to be abstracted from the rest of the functionality, this would lead to a more versatile solution where the PLCMS holds the information of the entire library project in Random Access Memory (RAM). This way, by having a local copy of a PLC Library project and an EPLAN export, it would be possible to generate the PLC Project XML without the need for an active database connection.

The intent here is to provide an alternative way to generate a PLC project if required, rather than removing the requirement for a database altogether.

SUMMARY

The PLCMS is a tool used to aid the automatic generation of PLC projects in an expedited fashion while reducing the potential for error. It is designed to be scalable and in using ORMs and with Open Source tools in mind, maintains a level of flexibility and adaptability.

ACKNOWLEDGEMENTS

The PLC Team and authors of this paper worked closely with other European XFEL scientific support groups and acknowledge their continuous efforts, input and cooperation. We thank the rest of the Electronic and Electrical Engineering (EEE) group, the Information Technology and Data Management (ITDM) group and the Controls Software and Data Analysis groups.

REFERENCES

[1] T. Tschentscher *et al.* "Photon Beam Transport and Scientific Instruments at the European XFEL", *Applied Sciences*, vol. 7, no. 6, p. 592, Jun. 2017, doi:10.3390/app7060592

[2] S. Hauf *et al.*, "The Karabo distributed control system", *J. Synch. Radiat.*, vol. 26, no. 5, p. 1448-1461, Sep. 2019, doi.org/10.1107/S160057751900669

[3] GIT, https://docs.gitlab.com/ee/ci/introduction

[4] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems 6th Ed.* Boston, MA, USA: Addison-Wesley, 2011.

[5] PostgreSQL, https://www.postgresql.org/docs/current/features.html

[6] ReadTheDocs Open Source documentation tool, https://readthedocs.org

[7] Beckhoff, https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_automationinterface/index.html