ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP088

LABVIEW-BASED TEMPLATE FOR ENHANCED ACCELERATOR SYSTEMS CONTROL: SOFTWARE SOLUTIONS FOR THE CERN-ISOLDE FACILITIES

C. Charrondiere, L. Le, R. Heinke, R. E. Rossel, S. Rothe, E. Galetti,
O. Ø. Andreassen, B. A. Marsh, S. Sudak, CERN, Geneva
A. Benoit, G. Boorman, ANGARA Technology, Geneva

Abstract

ISOLDE is part of the experimental infrastructure within the CERN accelerator complex that provides radioactive ion beams for studies of fundamental nuclear physics, astrophysics, condensed matter physics and medical applications. Complementing the available controls infrastructure, an easy-to-use set of applications was developed to allow operators to record and display signals from multiple sources, as well as to provide drivers for non-standard, custom-made instruments and specialized off-the-shelf components.

Aimed not only at software engineers but in general targeting developers without any specific background in software engineering, a generic and modular software template was developed in LabVIEW following a collaboration between CERN and ANGARA Technology. This unified template can be extended to support interaction with any instrument and incorporate any newly developed application to the existing control system and integrated into the CERN control and monitoring infrastructure. New modules and instrument drivers are easy to maintain as the structure and communication layers are all derived from the same template and based on the same components.

In this paper, we will explain the implementation, architecture, and structure of the template, as well as a wide variety of use cases - from motor control to image acquisition and laser-specific equipment control. We will also show use cases of applications developed and deployed within a few days in the ISOLDE facility.

MOTIVATION AND PURPOSE

The Isotope Separator On-Line (ISOLDE) [1] at CERN is a facility dedicated to the production of radioactive beams using a Resonance Ionization Laser Ion Source (RILIS) [2] to efficiently ionize specific isotopes of interest. In addition to ISOLDE, there is the MEDICIS [3] facility, including both RILIS and its counterpart MELISSA [4], which focuses on the production of medical radioisotopes. There are also several auxiliary offline test benches employed for quality control and research and development purposes [5], including two offline isotope mass separators known as OFFLINE1 and OFFLINE2 [6].

In this paper, we discuss the significance of avoiding large monolithic and complex applications, as such systems are typically challenging to maintain and sustain over time. We introduce the Module Template, which empowers developers to construct independent hardware drivers and software projects. These projects are designed to assist both machine operators and code maintainers when changes or updates are required.

The Module Template serves as a project template that already incorporates default functionality, offering a modular inter-process communication architecture. This framework enables developers to swiftly deploy an interface, promotes the upkeep of clean and organized code, and establishes a common communication approach for both software and hardware communication layers. Additionally, it streamlines the integration of existing software, such as that for the laser system [7], ensuring greater consistency and seamless integration into CERN's general network and data handling infrastructure.

CHALLENGES

An important variable in making design choices for the template is the consideration of the users and the code maintainers perspective: a balance must be struck between the amount of abstraction and hidden communication layers versus ease of use. For short-term affiliates such as students, for example, the aim is to reduce the initial effort required to make contributions. The template should also be comprehensible for the physicists and academics working at ISOLDE, allowing them to develop their own domain-specific applications.

The Module Template must be able to integrate a wide range of communication interfaces, such as Control Middleware (CMW) [8], a software communication protocol widely used at CERN, various hardware communication protocols, as well as a command line interface and the capability to handle a variety of configuration files.

TEMPLATE DESIGN

General Layout

Figure 1 shows the Module Template application architecture consisting of software modules that represent a distributed monitoring and control system. Each module can be comprised of one or more components (e.g. Manager and Worker). A module can be responsible for a specific task, such as interacting with a hardware device, performing stabilization tasks, or presenting a user interface. The Module Template is dependent on the LabVIEW RADE framework [9] and provides a coherent integration with the CERN control infrastructure [10].

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP088

The Manager Module is at the centre of the architecture and acts as a waypoint and data dispatcher for communication between the other modules. It ensures that all the application-specific sequences and timing restrictions are met. The Manager module has a higher level of awareness: it implements the management logic of the whole template while keeping track of the states of all the other modules by buffering and relaying messages as needed.

Commonly Used Modules These modules include those dealing with CMW clients and servers, performing hardware interaction, configuration and logging, and are already provided within the Module Template. Unique or application-specific functionality must be designed and implemented by the developers themselves through the creation of custom modules, making use of the provided components that serve as template building blocks. This allows for easy testing, maintenance, and consistency within an application.

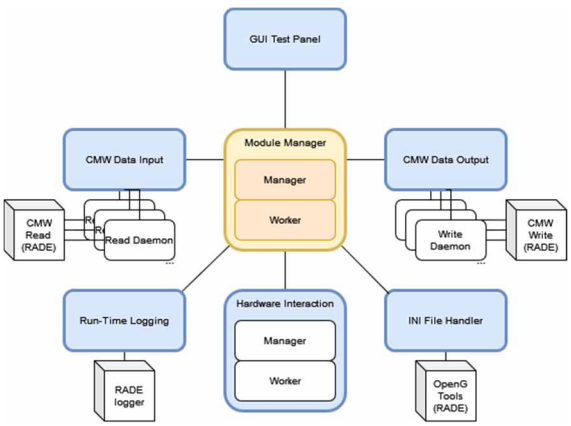

Figure 1: Illustration of a Queued-Message-Handler (QMH) template consisting of an Event Handler (Manager) and a Message Handler (Worker).

Components of a Module The Module Template provides components that can be used as a starting point for creating new modules. These components allow a module to send commands to itself (events or queues), provide a case structure to handle incoming commands (both externally and internally generated) and allow basic error-to-message conversion.

A module usually consists of two components: an Event Handler and a Message Handler. It may also consist of only one of these components when implementing a single loop that handles incoming messages intended for non-blocking tasks.

In many cases, the developer will combine the components to form a Queued-Message-Handler (QMH) template to deal with potentially blocking tasks, as illustrated within the Manager Module in Figure 1. The QMH template combines the Event Handler and the Message Handler components into a Manager-Worker pattern. The Manager component receives incoming messages from the components of other modules via an Application Programming Interface (API), handles the logic of the module, and can dispatch tasks to worker component(s). This allows for the delegation of blocking tasks to the Worker loop while the Manager loop remains responsive to incoming messages. At the same time, it allows for message buffering or sending accept/reject messages to other modules.

The Event Handler and the Message Handler are very similar in structure. The main difference is that the former consists of an Event Structure, while the latter is a process that receives queued messages. External modules may communicate with each other via API messages wherein modules explicitly specify commands that can be issued to this component. Each component contains its own execution logic that is executed when a command is received.

Communication Between Components

An API has been designed to facilitate inter-module communication. Every module is expected to feature an Event Handler component that includes a set of publicly accessible API messages. External modules can then employ this API to schedule parallel work, resulting in asynchronous messaging. Proper planning and coordination are paramount to ensure that the components are in the appropriate state when executing incoming tasks. Developers must carefully strategize which messages are invoked, when they are triggered, and from which sources. Equally important is devising contingency plans for scenarios where the component is occupied or, more critically, fails to execute a task successfully.

Each API message is furnished with a specific string command, aligning with an execution task contained within a component. If necessary, a relevant data payload can be specified. However, by default, a flexible variant data type is implemented to accommodate various data requirements.

USING THE TEMPLATE

Starting Point

The Module Template is a convenient starting point for developing LabVIEW-based applications that require parallel operations and integration with CERN data and infrastructure. However, depending on the development goal, many modules of the template may either need to be removed or heavily adapted. Since the template design is kept generic, and care has been taken to avoid obscuring its structure through advanced frameworks, it can be easier to either edit and modify existing applications or re-use modules and components rather than starting completely from scratch.

Creating Modules

Considerations and Planning Following good software programming practice when creating a module, the developer must decide which functionality it will contain. If a non-blocking event-based action such as a GUI is needed, the Event Handler component can be re-used. If uninterrupted data streaming is needed, the Message Handler provides the corresponding actions. Lastly if there are potentially blocking or time-consuming stand-alone actions required, the components can be combined to form a Queued Message Handler (QMH) module.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP088

The Public APIs After the initial planning phase, it is natural to specify the actions and public API for the new module. A few default API messages such as "Stop" already exist inside the module, and the developer may simply copy and adapt the new VI with a new message and/or payload LabVIEW data type.

APPLICATION EXAMPLES

All the ISOLDE facilities have similar beam observation and beam manipulation systems, some applications are therefore intended for use at multiple locations. While the application deployed is more or less identical at every facility, it must be designed to launch with different parameters following a configuration-based approach.

XY Plotter

The XY Plotter is a simple graph and value indicator that displays and logs two data values either in relation to each other or versus time, as shown in Figure 2. The data displayed can be generated from an application developed using the template presented, or from any other CERN equipment publishing via CMW. A common concern with network subscriptions is the timing accuracy of the incoming data, which is discussed in more detail later.

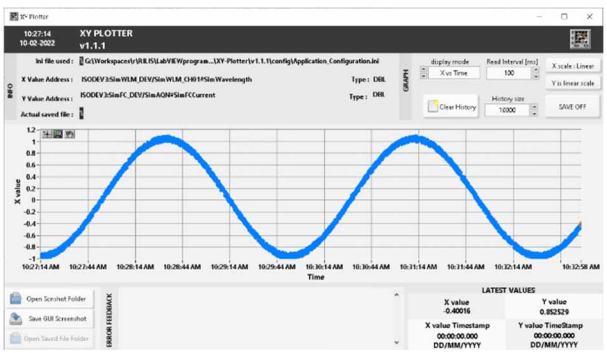

Figure 2: Graphical user interface of the XY Plotter application based on the Module Template.

Game Controller Interface

Dealing with lasers can be a tedious task especially when it comes to optimization of the laser beam position. Using a wireless game controller interface application, illustrated in Figure 3, along with an X-Y motor stage controller application, the vertical and horizontal positions of the mirror mount of an optomechanical setup can be manipulated using two independently communicating and device-specific applications. This gives the operator the freedom to move around the installation during operations while still viewing the screens on the walls where they can visualise the manipulations using, for example, the XY Plotter application. The Game Controller application also provides a local user interface with a simulated controller that can be used in parallel with the physical controller, in case the battery is drained, or only remote access to the control PC is possible.

Figure 3: Game controller interface.

Using the CMW input and output modules present in the template, the game controller interface communicates to the motorized mirror mount device driver application, where it can send commands and read status. Using adaptable configuration files, this application can easily be reused without any change in the code to control other devices for similar applications.

COMMUNICATION BETWEEN APPLICATIONS

The developed template allows the combination of a set of small and structurally similar applications that are each adapted to performing a specific task and thus form a flex-ible and distributed control application. An application can publish data (output) or receive data (input) via CMW. For each CMW device used by the application, a unique paral-lel process (daemon) is launched when the application starts up that listens for new data that to be received or to be published. Multiple applications can be interconnected, thus creating more complex control and acquisition solu-tions in a modular and extendible way.

Multiple Applications

An example of using multiple applications together is shown in Figure 4. Application 1 has a CMW client that can subscribe to data published by existing CMW devices, in this example a pair of PLCs. In addition, a CMW server running on Application 1 can communicate with the CMW client of Application 2. The module template allows the creation of one CMW server hosting multiple devices per application and permits any number of clients to connect to CMW devices hosted on other machines.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP088

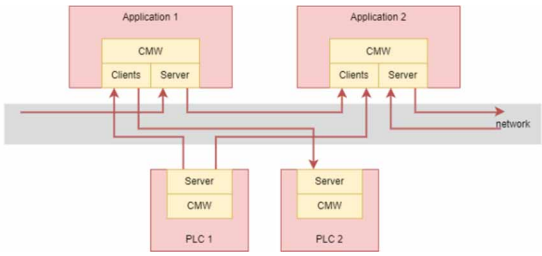

Figure 4: Communication between applications and other CMW devices on a network.

Applications 1 and 2 can be run on the same or different PCs, making use of the abstracted CMW data communication layer. Template applications can also run on a National Instruments real-time hardware such as a compactRIO (cRIO) or PXI system.

Non-CMW Communication

In the future, the data transfer mechanism could be changed from CMW to low-level communication protocols such as TCP or MQTT, or other middleware solutions such as EPICS, TANGO or OPC UA. The modules "CMW Data Input" and "CMW Data Output", shown previously in Figure 1, would then be replaced by new modules containing the appropriate communication mechanism for receiving and sending data to other applications. The Module Template allows for multiple communication protocols, simply by adding the corresponding components, thus acting as a bridge between protocols.

Multi-Application Example: Laser Timing Feedback Loop

Stand-alone applications like the XY plotter and the Game Controller are simple in design, to perform small, application-specific tasks. However, by combining multiple applications, they can work together and perform more complex operations.

An example is an automated pulse timing feedback loop of the RILIS laser system. It is crucial that the laser pulses are perfectly synchronized in time, while the internal processing time from the trigger signal to the pulse output of the laser system can vary depending on operational parameters. Monitoring of the laser output, data processing and trigger timing adjustments can be handled by a combination of three applications as schematically displayed in Figure 5:

- The first application acts as a driver to communicate with a digital oscilloscope (pico technology PicoScope 6402c). It reads out the traces of the photodiodes that monitor the actual laser system outputs. The time delay of the pulse peak with respect to a master clock trigger is published in CMW.
- A generic feedback loop application compares this read-out delay value (or potentially a time-average of it) with a set value provided by the operator. The difference is applied as a correction to the individual laser trigger timings. These corrected timing set-points are then again shared via CMW.
- The last application is a driver to directly control the pulse timing hardware (Quantum Composers 9538). Possible longer internal processing times in the lasers are compensated by an earlier trigger command.

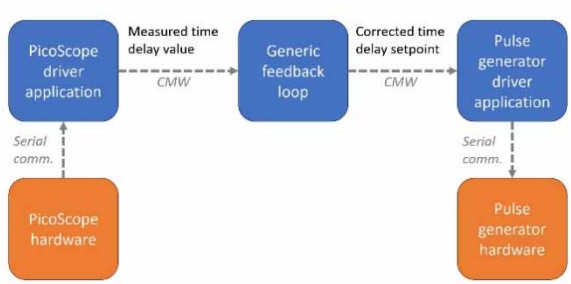

Figure 5: Schematic overview of the communication data flow for a laser pulse timing feedback loop, constructed from three basic applications (blue) and the respective hardware (orange).

Note that this modularization ensures full flexibility for different operation modes, instead of limiting the usage as in a monolithic program. For example, the generic feedback loop can be used in any stabilisation applications, or the pulse timing can be controlled by an operator directly, or linked to the duty cycle of detector units.

Considerations of the Communication Time

Combined applications naturally raise concerns about timing, delay, and synchronization of data. The applications developed are very diverse, ranging from feedback systems, control of optomechanical actuators, or data display and visualization. Data transfer mechanisms such as CMW are limited by the network speed and pose a limitation to timing and delays. Thus, distributed applications developed using the template are certainly not intended to perform fast feedback loop operations, but can be useful tools for exchanging data and information from different systems when working beyond the millisecond timeframe.

CONCLUSION AND OUTLOOK

A LabVIEW-based template was developed, tested, and successfully deployed for the control of accelerator systems within the ISOLDE facilities. This template serves as the foundational framework for a variety of stand-alone and combined expert applications.

The outcome is an entry-level application template that is modular, adaptable, and easy to maintain. It allows rapid development and commissioning of new applications without necessitating a background in software engineering. The template approach and the chosen design patterns ensure straightforward and unified code maintenance, all while providing flexibility and accessibility for novice developers.

Over 30 applications have been created so far using the Module Template, resulting in a standardized structure and significantly reducing the annual maintenance time for small applications from 7 to 2 days.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP088

Given the substantial number of deployed applications and the frequent changes requested by scientific users, this represents a substantial long-term efficiency gain.

The template has been very successful in integrating instruments and communication protocols into the CERN infrastructure, seamlessly interfacing with independent and newly installed CERN equipment.

Moreover, the template's flexible design facilitates smooth integration and interaction with temporary ISOLDE experimental setups. These systems are often operated by visiting teams who bring their own detector equipment and corresponding data acquisition software. While these teams typically use non-CERN-standard communication protocols and interfaces, they can still easily integrate these with the existing control and monitoring infrastructure using the template.

The primary goal of this module development was to provide accessible and flexible entry-level access to LabVIEW's potent parallelization capabilities for non-software engineers. Additionally, it offers a unified and configuration-based approach for code maintainers, thereby reducing implementation times when adapting to new requirements and deployment changes.

REFERENCES

[1] R. Catherall *et al.*, "The Isolde facility", *J of Phys G: Nucl. and Part. Phys.*, vol. 44, no. 9, p. 094002, Aug. 2017. doi:10.1088/1361-6471/aa7eba

[2] V. Fedosseev *et al.*, "Ion beam production and study of radioactive isotopes with the laser ion source at ISOLDE", *J. of Phys. G: Nucl. and Part. Phys.*, vol. 44, no. 8, p. 084006, Aug. 2017. doi:10.1088/1361-6471/aa78e0

[3] C. Duchemin *et al.*, "CERN-MEDICIS: A Review Since Commissioning in 2017", *Frontiers in Medicine*, vol. 8, p. 693682, Jul. 2021. doi:10.3389/fmed.2021.693682

[4] V.M. Gadelshin *et al.*, "MELISSA: Laser ion source setup at CERN-MEDICIS facility. Blueprint", *Nucl. Instrum. and Methods in Phys Res. Sect. B: Beam Interact. with Mater and At.*, vol. 463, pp. 460-463, Jan. 2020. doi:10.1016/J.NIMB.2019.04.024

[5] S. Rothe *et al.*, "Targets and ion sources at CERN-ISOLDE — Facilities and developments", *Nucl. Instrum. and Methods in Phys Res. Sect. B: Beam Interact. with Mater and At.*, vol. 542, pp. 38-44, Sep. 2023. doi:10.1016/j.nimb.2023.05.058

[6] S. Warren *et al.*, "Offline 2, ISOLDE's target, laser and beams development facility", *Nucl. Instrum. and Methods in Phys. Res., Sect B: Beam Interact. with Mater. and At.*, vol. 463, pp. 115-118, Jan. 2020. doi:10.1016/j.nimb.2019.07.016

[7] R. E. Rossel *et al.*, "Data acquisition, remote control and equipment monitoring for ISOLDE RILIS", *Nucl. Instrum. and Methods in Phys. Res., Sect. B: Beam Interact. with Mater. and At.*, vol. 317, pp. 557-560, Dec. 2013. doi:10.1016/j.nimb.2013.05.048

[8] K. Kostro, J. Andersson, F. Di Maio, S. Jensen, and N. Trofimov, "The Controls Middleware (CMW) at CERN - Status and Usage", in *Proc. ICALEPCS'03*, Gyeongju, Korea, Oct. 2003, paper WE201, pp. 318-321.

[9] O. O. Andreassen, D. Kudryavtsev, A. Raimondo, A. Rijllart, S. Shaipov, and R. Sorokoletov, "The LabVIEW RADE Framework Distributed Architecture", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper WEMAU003, pp. 658-661.

[10] S. Deghaye and E. Fortescue-Bec, "Introduction to the BE-CO Control System", CERN, Geneva, Switzerland, Rep. CERN-202-0069, Dec. 2020.