ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

INTEGRATION OF OPC UA AT ELBE

K. Zenker*, R. Steinbrück, M. Kuntzsch,
Institute of Radiation Physics, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany

Abstract

The Electron Linac for beams with high Brilliance and low Emittance (ELBE) at Helmholtz-Zentrum Dresden-Rossendorf (HZDR) is operated using the SCADA system WinCC by Siemens. The majority of ELBE systems is connected to WinCC via industrial Ethernet and proprietary S7 communication. The integration of new subsystems based on MicroTCA.4 hardware, which do not provide S7 communication interfaces, into the existing infrastructure is based on OPC UA using the open source C++ software toolkit ChimeraTK.

This paper discusses OPC UA related contributions to ChimeraTK, that cover an OPC UA backend implementation, development of logging, data acquisition and history modules and improvements of the control system adapter for OPC UA in ChimeraTK. Furthermore, a user data interface based on OPC UA for ELBE is introduced and one ELBE real-life example is described, that makes use of all the afore-mentioned features.

INTRODUCTION

The Electron Linac for beams with high Brilliance and low Emittance (ELBE) at Helmholtz-Zentrum Dresden-Rossendorf (HZDR) is in operation since 2001. It is operated using the SCADA system WinCC by Siemens. The majority of ELBE systems is connected to WinCC [1] via industrial Ethernet and proprietary S7 communication. However, in recent years new subsystems had to be integrated into the existing infrastructure, which do not provide S7 communication interfaces. The Open Platform Communication (OPC) interoperability standard for the secure and reliable exchange of data in industrial automation environments is developed and maintained by the OPC foundation [2]. Its Extension called OPC Unified Architecture (OPC UA) [3] is an open standard for information modeling and machine-to-machine communication, that features e.g. build-in security/authentication and is the foundation of the so-called 4th industrial revolution in industry. It is platform independent and different open source implementations exists that provide support for various development environments like e.g. Python, C/C++, Java and Labview. Because of those features OPC UA has been chosen at ELBE to establish a link between the new subsystems and S7 devices.

As shown in [4] two types of communication were established in the past:

- OPC UA communication between an application and MicroTCA.4 [5] based hardware, i.e. FPGA via direct memory access (DMA) over PICe
- OPC UA communication between an application and Siemens S7-300/400 PLCs

The former communication is established based on the ChimeraTK framework introduced in the following section. The latter communication is based on commercially available OPC UA gateways introduced after. In the following sections OPC UA related components of ChimeraTK and an OPC UA based ELBE machine data interface are presented. Thereafter, the use of the new features is demonstrated by means of an ELBE subsystem as a real-life example.

OPC UA IN CHIMERATK

ChimeraTK [6, 7] is a C++ based open source toolkit for modular control application developments. ChimeraTK provides dedicated libraries for access to devices/data (DeviceAccess library [8]), the application itself (ApplicationCore [9]) and a control system server (ControlsystemAdapter [10]). Both, the Device Access Library and the Control System Adapter library, can be used to implement additional so called device backends and control system adapter implementations. The general structure is shown in Fig. 1. Highlighted in green are parts of ChimeraTK that were contributed as part of the OPC UA developments at ELBE.

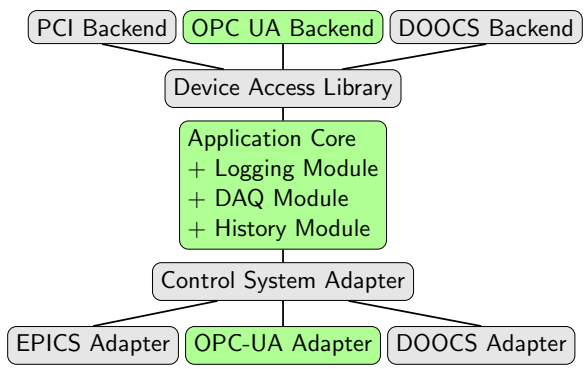

Figure 1: Overview of ChimeraTK. Only selected control system adapters and backends are shown. Contributions discussed in this paper are shown in green.

In general, so far the OPC UA communication in ChimeraTK is implemented using client/server communication, based on TCP/IP. This means no real-time communication can be realized using this approach. Real-time communication would only be possible using the Publish/Subscribe mechanism of OPC UA in combination with Time-Sensitive Networking (TSN) and hardware that supports TSN.

* k.zenker@hzdr.de

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

Control System Adapter

The control system adapter that allows building OPC UA servers [11] was originally developed in cooperation with the Technical University of Dresden and DESY[12]. This work is presented in [3]. Now it is further developed together with the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB [13], who are also involved in the development of the OPC UA stack open62541 [14], which is used to implement the OPC UA control system adapter in ChimeraTK. Its main task is to expose all process variables and attached meta data like the variable description and engineering unit defined in the ChimeraTK application via an OPC UA server with corresponding nodes. The nodes are only updated with values of the corresponding ChimeraTK application variable upon request. If no client is connected to the OPC UA server updates of the process variables in the ChimeraTK application are not reflected on the OPC UA server side.

A major improvement compared to the version discussed in [3] is the extended mapping feature of the OPC UA control system adapter. In the past it was only possible to add additional static process variables to the variable tree via a mapping file. Now it is possible to extend the variable tree defined in the ChimeraTK application in an arbitrary manner, by copying or linking variables and complete folders including subfolders to different places in the variable tree. In addition, variable names and the meta data can be edited. If the variable name is changed the variable is copied in the variable tree, meaning a new OPC UA node is created. Else if not explicitly stated different in the mapping file the new location is simply linked to the source variable. This is done by adding a reference to the source OPC UA node keeping the number of nodes of the OPC UA server constant. Such a mapping feature is useful in case client applications expect a certain variable structure. For example, when the ChimeraTK application and its variable tree is changed, client applications are still operational if the previous variable tree is created via the mapping. This allows to decouple client developments from server developments.

Another major change is that numeric node identifiers of process variables, which were initially used for all variables of the ChimeraTK application, were replaced by string node identifiers. The string node identifier is created from the variable name and path as created in the ChimeraTK application. This solves the problem that node identifiers changed, when changing the variable tree of the ChimeraTK application. It happened, because node identifiers were assigned dynamically at server start by scanning the applications variable tree. Using a string node identifier guarantees a static node identifier as long as the variable name is not changed in the application.

Furthermore, changes in ChimeraTK, like the introduction of a data validity flag, were reflected in the OPC UA control system adapter by adding corresponding meta data to variables. Only recent changes in ChimeraTK, i.e. the introductions of the new data types *Bool* and *Void*, are not reflected yet.

Finally, the version of the open62541 stack was upgraded from version 0.2 to version 1.2. This new version includes full encryption support. An encryption endpoint for the OPC UA server will be added in the near future, thus two endpoints will be available following the security policies *None* and *Basic256Sha256* as defined in [15]. The *Basic256Sha256* endpoint will support only encrypted and signed messages and thus ensure a high security level.

Application Core

Different control systems supported by ChimeraTK include different control system specific features, e.g. for data archiving, that are not part of ChimeraTK. In case of OPC UA such features could be implemented in the OPC UA control system adapter utilizing the full functionality of OPC UA and e.g. its historizing mechanism. However, it was decided to add features that are needed at ELBE directly to the ApplicationCore library of ChimeraTK. Therefore, they can be used by all control system adapters of ChimeraTK applications.

First, a logging module was added. It covers four different severity levels and three different target streams – a message variable in the control system, a logging file and standard output to the console – that can be combined. The logging level, target stream and the number of messages included in the message variable can be configured at run time.

Second, a module for data acquisition (DAQ) and archiving was developed [16]. It includes two different backends – a ROOT [17] based and HDF5 [18] based backend. The backend type can be defined in the server configuration that is loaded at server start. The HDF5 backend does not yet cover compression and stores any ChimeraTK data type to a float data type in the HDF5 file. In case of the ROOT backend loss-less compression based on the ZStd algorithm [19] is supported and ChimeraTK data types are mapped to native ROOT data types, which improves the data compression rate.

In addition to this module for data archiving a software package with dedicated tools was developed [20]. The main component of this packages is a Graphical User Interface (GUI), that allows DAQ data inspections. It is a python GUI based on the widget toolkit Qt [21] and on the scientific graphics and GUI library pyqtgraph [22]. It can construct time lines out of the DAQ data and allows to visualize specific data periods. Also, it is possible to jump to events that fulfill a specific threshold criteria, e.g. that a certain temperature is above a certain threshold level. A table allows to inspect a statistical summary (mean and standard deviation) for time line data.

Finally, a history module was added, that allows to create a RAM based history for selected process variables. The history is available as a pair of two process variables each holding a data array. One variables holds the variable values and a second variable holds corresponding time stamps. Data of the history module is used in Human Machine In-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

terfaces (HMI), that show actual data trends for ChimeraTK applications. It complements the data available by the DAQ module described above with data that is not yet written by the DAQ module.

Device Access

The device access library allows access to devices that are not necessarily part of the machine, that is running the ChimeraTK application. E.g. as shown in Fig. 1, there is a PCIe backend that allows to access FPGA data via PCIe communication. At ELBE typically the FPGA is part of a MicroTCA.4 platform and it is added by the MicroTCA Carrier Hub (MCH) to the computers PCIe root complex.

In addition to such a classical device access often a communication between different applications is needed. Such a cross application communication can also cover communication between different control systems, e.g. if a ChimeraTK application using the OPC UA Control System Adapter communicates with a DOOCS [23] server via the DOOCS backend shown in Fig. 1. In case of OPC UA, this requires an OPC UA backend [24], which was developed at ELBE. In fact the backend is an OPC UA client and features synchronous and asynchronous data access. The latter is based on subscriptions in combination with items to be monitored that are attached to the subscription. Two methods to select process variables from the ChimeraTK application, i.e. the OPC UA server, to be monitored are available:

1. Map file based process variable selection.
2. Automatic process variable selection.

If using the map file, one needs to specify the OPC UA node identifier, the corresponding OPC UA name space and a name that is used to map the variable to ChimeraTK. The automatic mapping only works if one connects to an OPC UA server that is based on the ChimeraTK OPC UA control system adapter. It maps all process variables available on the server one to one to the ChimeraTK application. In some cases this might be not sufficient to create a process variable tree as needed by the application. In that case, one can use an additional mapping layer provided by ChimeraTK [7], that allows to restructure the variable tree of devices. In addition to the server address and port, the sampling interval can be specified in the backend configuration. It defines the update rate of monitored process variables independent of the update rate of the source process variable of the OPC UA server. The publishing interval is set equal to the sampling interval, which means each sampled value is sent without buffering directly to the backend by the OPC UA server.

OPC UA GATEWAY INFRASTRUCTURE

As discussed in [4] at ELBE commercially available OPC UA gateways [25] (UA Link) by IBHsoftec are used as a data bridge between Siemens S7-300/400 PLCs and OPC UA based applications. Initially only used for the LLRF system introduced in the following section, the gateway infrastructure was developed further to provide an OPC UA based ELBE machine data interface in general. A hierarchy of different gateways as shown in Fig. 2 was set up at ELBE.

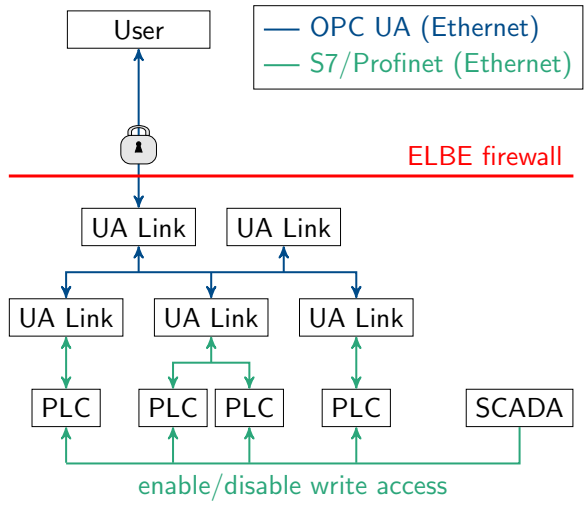

Figure 2: UA Link hierarchy used at ELBE.

In the lowest hierarchy level, UA Link devices are connected to individual PLCs. Already at this level data of different PLCs is aggregated at ELBE. At ELBE typically about 20000 PLC process variables per UA Link are made available via OPC UA. In the hierarchy level above two aggregating UA Links are connected to all the UA Links in the hierarchy level below. They serve as access points to the OPC UA user interface and include an identical process variable tree, that is a a subset of the process variables available in the hierarchy level below. If needed UA Links at this level can also be replaced by workstations running a software[26] that directly uses the configuration of the UA Links.

One of the aggregating UA Link acts as the access point for external users via an encryption based secured connection. For this UA Link machine data is accessible read-only. The second UA Link in this hierarchy level is used for ELBE internal applications. Read and write access is granted to this UA link. However, all PLC registers published via the OPC UA user interface are duplicated on the PLC, which allows to disable write access on the PLC level. Thus, only the duplicates are accessed by the UA Link and operators can control external write access via OPC UA for dedicated process variables or at all from the SCADA system. The duplication also allows data type adjustment and scaling. Restructuring and renaming of process variables is done in the upper UA Link hierarchy level.

In addition to PLC data published via the UA Link, special process variables dedicated to the users can be added to the variable tree on the UA Link. Users can have read and write access to those variables. One example for such variable is a count rate measured by the users and exposed via OPC UA to the SCADA system. Like that such information is available to operators for machine setting optimization.

User authentication for external users is done via user certificates following the X.509 standard in combination with a Certification Authority (CA). A Public-Key-Infrastructure

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

(PKI) including a ROOT CA and dedicated user group CAs was set up at ELBE. Thus, only the CAs certificates and the corresponding revocation lists need to be placed on the UA Link, which simplifies the configuration. User certificates are handed out by the ELBE group and are signed with the corresponding user group CA certificate.

EXAMPLE: THE LLRF SUBSYSTEM AT ELBE

The digital low level radio frequency (LLRF) system of ELBE was the first sub system building on the OPC UA control system adapter discussed here. It is used to control the amplitude and phase of cavities used for particle acceleration and bunching at ELBE [27]. Operators interact with the LLRF system via the SCADA system WinCC. It runs OPC UA clients that are connected to the ChimeraTK LLRF applications (LLRF server). Currently at ELBE seven LLRF servers are running. A simplified overview of the system including only a single LLRF server is shown in Fig. 3.

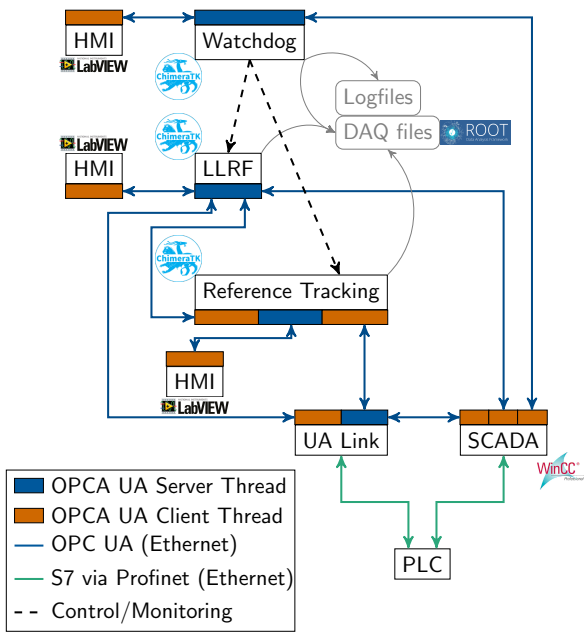

Figure 3: Overview of the control system infrastructure related to the LLRF system at ELBE. Here only one out of seven LLRF applications is shown.

The LLRF servers are controlled and monitored by a second ChimeraTK application called Watchdog server[28], which also monitors the computer where the applications are running. WinCC runs also an OPC UA client that is connected to the Watchdog server, which allows to forward the LLRF Server status to WinCC and also to restart LLRF severs via WinCC. Logging output of all applications controlled by the watchdog are available via OPC UA and it is written in addition to log files stored on disc via the logging module discussed above.

A simple feedback algorithm [27] correcting long term drifts of amplitude and phase introduced by temperature variations is implemented in a separate ChimeraTK application called Reference Tracking server (see also [27]). This server needs data from the LLRF servers as well as machine data, which is read from ELBE PLCs via the UA Link by the OPC UA backend. Calculated corrections are written back to the LLRF server via the OPC UA backend. The Reference Tracking server calculates corrections for all LLRF servers meaning it creates seven OPC UA client threads for the LLRF servers and one OPC UA client thread to the UA Link.

All ChimeraTK applications include the DAQ module introduced before and produce ROOT data files that are written to disc as indicated in Fig. 3. In addition to the WinCC interface to the ChimeraTK servers, application specific expert HMI GUIs have been set up at ELBE. They are implemented using LabVIEW, which features an OPC UA client. Details about LabVIEW client applications are presented in [4].

CONCLUSION

The SCADA system of ELBE is the S7 communication based software WinCC. New subsystems that are integrated to ELBE, like the LLRF system that is based on the MicroTCA.4 platform, do not support S7 communication. In this paper we have shown how such systems are integrated using OPC UA. The integration is based on the C++ toolkit ChimeraTK. The already existing OPC UA control system adapter was extended by additional mapping features that allow to decouple the HMI and application development. Also it was updated to the latest version of the open62541 OPC UA stack it is based on. That allows to make full usage of the encryption provided by OPC UA. In addition ChimeraTK was extended by an OPC UA backend, common modules for logging, data acquisition and online data history, to fulfill the needs at ELBE. Those developments are not only relevant if using the OPC UA control system adapter of ChimeraTK. All other supported control systems benefit, either by the common modules or by the OPC UA backend that allows to integrate data from any OPC UA server to any ChimeraTK application.

The OPC UA based machine data interface presented in this paper, allows external users secure access to ELBE data, which was not available before.

In future, we plan to support more features of OPC UA in the OPC UA control system adapter, like support for OPC UA events or registration to local or global discovery servers according to the OPC UA standard.

ACKNOWLEDGEMENTS

The authors would like to thank our IOSB collaborators and the ChimeraTK team, especially Martin Killenberg and Martin Hierholzer, for their continuous support and feedback.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

REFERENCES

[1] *SIMATIC WinCC SCADA*, Siemens. https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada.html

[2] *OPC Unified Architecture*, The OPC Foundation. http://opcfoundation.org/opc-ua/

[3] "OPC Unified Architecture - Part 1: Overview and Concepts", Tech. Rep. IEC TR 62541-1:2020, 2020. https://www.vde-verlag.de/iec-normen/249339/iec-tr-62541-1-2020.html

[4] R. Steinbrück *et al.*, "Control System Integration of a MicroTCA.4 Based Digital LLRF Using the ChimeraTK OPC UA Adapter", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017*, (Barcelona, Spain), Geneva, Switzerland: JACoW Publishing, Jan. 2018, pp. 1811–1814, ISBN: 978-3-95450-193-9.

[5] PICMG, *PICMG specification MTCA.4 Rev. 1.0*, Aug. 2011. https://www.picmg.org/openstandards/microtca

[6] M. Killenberg *et al.*, "Abstracted Hardware and Middleware Access in Control Applications", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17)*, (Barcelona, Spain), Geneva, Switzerland: JACoW Publishing, Jan. 2018, pp. 840–845, ISBN: 978-3-95450-193-9. DOI: 10.18429/JACoW-ICALEPCS2017-TUPHA178.

[7] G. Varghese *et al.*, "ChimeraTK - A Software Tool Kit for Control Applications", in *Proc. of International Particle Accelerator Conference (IPAC'17)*, (Copenhagen, Denmark), ser. International Particle Accelerator Conference, Geneva, Switzerland: JACoW, May 2017, pp. 1798–1801, ISBN: 978-3-95450-182-3.

[8] *Device Access library*, ChimeraTK. https://github.com/ChimeraTK/DeviceAccess

[9] *ApplicationCore library*, ChimeraTK. https://github.com/ChimeraTK/ApplicationCore

[10] *ApplicationCore library*, ChimeraTK. https://github.com/ChimeraTK/ControlSystemAdapter

[11] *OPC UA Control System Adapter*, ChimeraTK. https://github.com/ChimeraTK/ControlSystemAdapter-OPC-UA-Adapter

[12] Deutsches Elektronen-Synchrotron. https://www.desy.de

[13] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB. https://www.iosb.fraunhofer.de

[14] *open62541 — An open source and free C (C99) OPC UA stack licensed under LGPL + static linking exception*. http://open62541.org

[15] "OPC Unified Architecture - Part 7: Profiles", Tech. Rep. IEC 62541-7:2020 RVL, 2020. https://www.vde-verlag.de/iec-normen/248879/iec-62541-7-2020-rlv.html

[16] K. Zenker, *DAQ Module*. https://github.com/ChimeraTK/ApplicationCore-MicroDAQ

[17] R. Brun and F. Rademakers, "Root — an object oriented data analysis framework", *Nucl. Inst. & Meth. in Phys. Res. A*, vol. 389, no. 1, pp. 81–86, 1997, See also "ROOT" [software], Release v6.20/04, 01/04/2020. DOI: 10.5281/zenodo.3895855.

[18] The HDF Group, *Hierarchical data format version 5*. http://www.hdfgroup.org/solutions/hdf5/

[19] Y. Collet and E. M. Kucherawy, "Zstandard compression and the application/zstd media type", *RFC 8478*, 2018. DOI: 10.17487/RFC8478.

[20] K. Zenker, *DAQ Tools*. https://github.com/ChimeraTK/ApplicationCore-MicroDAQ-Tools

[21] *GUI widget tool kit Qt*, The Qt Company. https://code.qt.io/cgit/qt/qtbase.git

[22] *PyQtGraph - Scientific Graphics and GUI Library for Python*. https://www.pyqtgraph.org

[23] K. Rehlich *et al.*, "DOOCS: an Object Oriented Control System as the Integrating Part for the TTF Linac", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'97)*, (Beijing, China), 1997. https://www3.aps.anl.gov/News/Conferences/1997/icalepcs/schedual.html

[24] *OPC UA Device Access*, ChimeraTK. https://github.com/ChimeraTK/DeviceAccess-OpcUaBackend

[25] *IBH Link UA*, IBHsoftec. https://www.ibhsoftec.com

[26] *IBH OPC UA Server/Client*, IBHsoftec. https://www.ibhsoftec.com

[27] K. Zenker *et al.*, "Microtca.4-based low-level rf for continuous wave mode operation at the elbe accelerator", *IEEE Transactions on Nuclear Science*, vol. 68, no. 9, pp. 2326–2333, 2021. DOI: 10.1109/TNS.2021.3096757.

[28] K. Zenker, *Watchdog application*. https://github.com/ChimeraTK/Watchdog