CONTROL USING BECKHOFF DISTRIBUTED RAIL SYSTEMS AT THE EUROPEAN XFEL

N.Coppola*, J.Tolkiehn, C.Youngman, European X-Ray Free Electron Laser Facility GmbH, Albert-Einstein-Ring 19, 22761 Hamburg, Germany

Abstract

The European XFEL project is a 4th generation light source producing spatially coherent 80 fs short photon x-ray pulses with a peak brilliance of 10^{32}-10^{34} photons/s/mm^2/mrad^2/0.1% BW in the energy range from 0.26 to 24 keV at an electron beam energy 14 GeV. Six experiment stations will start data taking in 2016.

In order to provide a simple, homogeneous solution, the DAQ and control systems group at the European XFEL are standardizing on COTS control hardware for use in experiment and photon beam line tunnels. A common factor within this standardization requirement is the integration with the Karabo software framework of Beckhoff TwinCAT 2.11 or TwinCAT3 PLCs and EtherCAT. The latter provides the high degree of reliability required and the desirable characteristics of real time capability, fast I/O channels, distributed flexible terminal topologies, and low cost per channel.

In this contribution we describe how Beckhoff PLC and EtherCAT terminals will be used to control experiment and beam line systems. This allows a high degree of standardization for control and monitoring of systems.

SYSTEM OVERVIEW

At project [1] start-up stage three undulator systems SASE1, SASE2 and SASE3 (see Fig. 1) will be used to produce photon beams. At the end of each undulator system a photon beam transport system is planned, which comprises: mirrors, slits, diagnostic pop-in cameras, monochromators, attenuators, Compound Refractive Lenses (CRL), vacuum pumps and gauges. These systems extend up to the experimental hall.

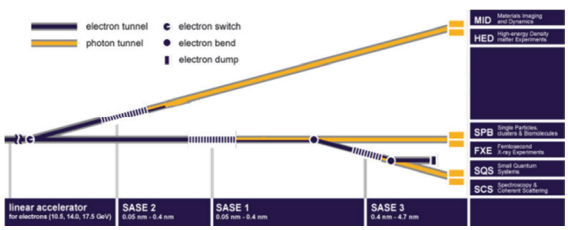

Figure 1: Schematic layout of the electron and photon beam distribution.

In order to control all these devices a set of Programmable Logic Computer (PLC) will be used. Each system will be partitioned into three sub-systems per SASE, with ring topology, enabling cable redundancy within each ring. The first ring will control mostly vacuum devices. The second ring will be in charge of controlling movements of mirrors, slits and pop-in cameras. The third is there to control special devices, in case it is needed, or to be used as spare to add and/or remove devices that are not installed all the time, or that needs testing, in order to minimize down time or maintenance interventions on the other 2 rings.

Two experimental areas are directly connected at the end of each SASE branch, each experimental area comprises an experimental hutch, one optics hutch and one optical pump-probe laser coupling room. These areas are planned to be build and instrumented to make maximal use of the high intensities and rates given by the accelerator and undulators. Environmental sensors, sample injectors, beam diagnostics, temperature sensors, vacuum devices and, finally, up to 250-300 axes need to be controlled also to guarantee safety of all detectors within each of these areas. In the same way as for the devices in the SASEs tunnels, also here a set of PLCs will be used. It is foreseen to have in the order of five to ten

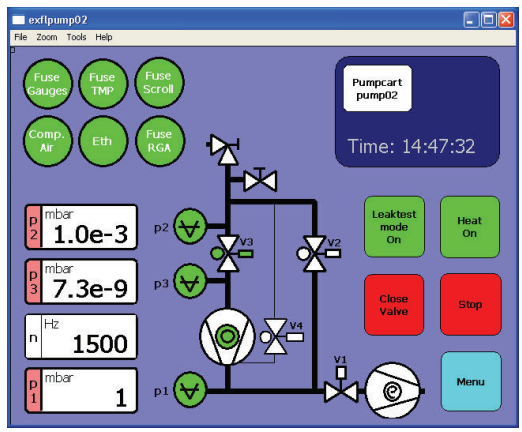

Figure 2: Example of GUI running locally on a WindowsCE touch screen Beckhoff CPU, implemented using the TwinCAT visualisation library. The PLC is used to start the pumping down procedures of the photon beam line system, to perform bake out operations of vacuum elements and to check for leaks.

cabinet CPUs per experiment, with WindowsCE as operating system (see Fig. 2 as an example of a PLC with local HMI).

BACKGROUND

In automation, PLC programs are used to control devices with actuators based on feedback from sensors. The PLC runs in a loop: first the sensor values are copied into the program inputs, after which the PLC program is

*Contact author: nicola.coppola@xfel.eu

ISBN 978-3-95450-139-7

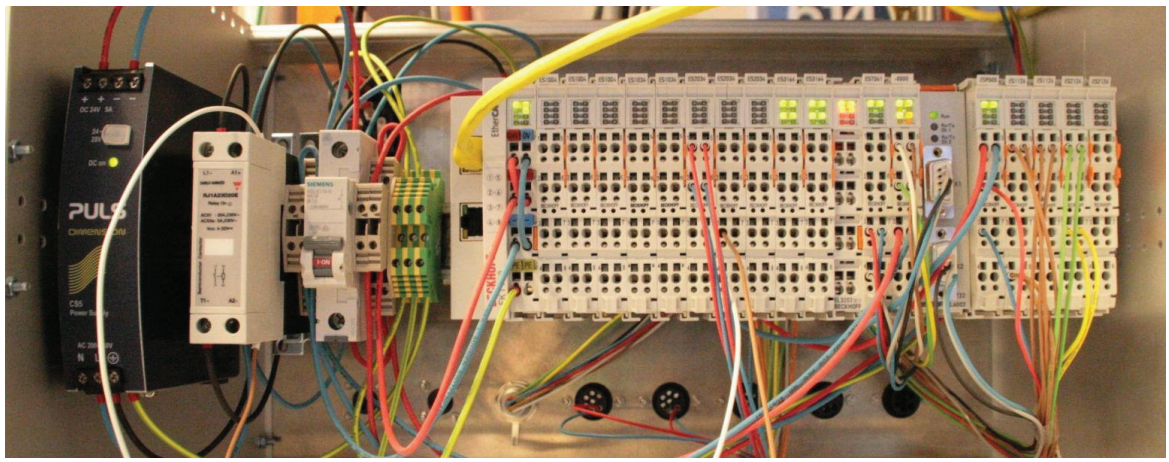

Figure 3: Example of a Beckhoff EtherCAT DIN-rail, used in a test stand, to simultaneously control: scroll pump, turbo pump, vacuum valves, vacuum gauges, motorised manipulators and their interlock. This CPU on which the PLC is running is not shown, but resides in the same rack.

executed to calculate the output based on the values from the input, and eventually the program outputs are applied to the actuators. PLCs also provide network access to internal memory and variables in order to modify the behaviour of the program and to read out and export status information. The PLC is supported by a Numeric Control (NC for example the TcMC2 library where applicable) to control positioning of movable axes.

The control of the photon beam line system of the European XFEL is realized with Beckhoff [2] TwinCAT [3] (software framework implementing all five IEC 61131-3 defined programming languages), using Structured Text (ST) in-house written programs, which offers real-time PLC/NC on Windows based PCs. The PC running Beckhoff TwinCAT PLC acts like a virtual machine and up to four TwinCAT PLCs can run in a PC (such limitation has been lifted with TwinCAT3.x, with this new version it is possible also to distribute the computing load to more than one CPU or to different cores within one host, we plan in future to upgrade to this version). TwinCAT PLC NC is compliant with the PLCopen Motion Control standard [4]. The PLC/NC is connected to the peripheral devices via the EtherCAT fieldbus [5].

The choice of Beckhoff PLC and of EtherCAT as fieldbus is based on the fact that this hardware is Commodity Out The Shelf (COTS), that it gives a nice standard to build on, both at the programming level with TwinCAT and hardware with the fast EtherCAT fieldbus and protocol, and that allows to easily integrate equipment, even if this is not equipped with an EtherCAT interface, which, in this case, can be controlled and monitored via digital and/or analogue input/output (I/O) signals or via serial interfaces (for example: RS232, RS422 or RS485). Beckhoff offers a multitude of different I/O DIN terminals with high density of channels per unit and at reasonable prices. A system comprises at least one PLC and one or many rails (a rail is a set of one coupler and one or more terminals, see Fig. 3), which can be geographically distributed over large distances, in such a way to have the controlling terminal as near to the device to control as needed (rails can be placed as far as 100 m apart from each other using CAT5 cables or up to 20 km apart using optical fibres). The system can easily be expanded even at run time profiting from the use of HotConnect EtherCAT couplers [6]. The fact, that products of this firm were already used at DESY and, in special mode, by the undulator group, has eventually strengthen this decision.

Access from the higher level control program is provided via a supplement TCP/IPserver library from Beckhoff, which allows the use of an open protocol, developed within European XFEL, to be used with the Karabo software framework[7]. This allows platform independency as far as it is permitted by the use of Karabo (Linux ScientificLinux and Ubuntu, Mac OS X and in the future, possibly, Windows).

IMPLEMENTATION

Each PC will be configured to run 2 PLCs in parallel: the first is in charge of administrative tasks: checking that the version of the program running on the second one is up to date, stopping this in case it is not, uploading an appropriate version, CRC checking that the file of the uploaded one was properly copied and that this corresponds to a valid version of the PLC and restarting of the same. The second PLC is in charge of I/O and controlling of all connected devices. This firmware is the entity of the control system that is in charge of hardware safety.

General Processes and Devices Layout

In order to control multiple devices and still keep the number of used TCP/IP ports to a minimum on the PLC a purely software device is run, which has the only task of reading in data sent via the standard switched network from Karabo and sending update information to Karabo. This device forwards then instructions to other encapsulated uniquely identified, in-house developed, devices running inside the PLC, which are controlling the

ISBN 978-3-95450-139-7

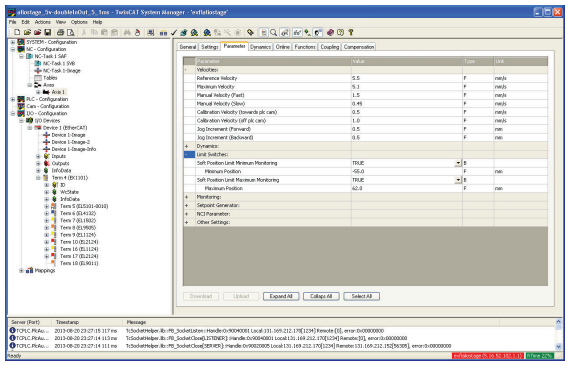

Figure 4: The SystemManager of TwinCAT: a graphical interface to configure EtherCAT buses and terminals.

real hardware peripherals, in a simple or composite fashion. These firmware devices implement all needed algorithms to control and monitor the associated hardware together with Finite State Machine (FSM) to insure that only safe actions, for the same hardware and the nearby or possibly concerned devices (interlock), are allowed to be carried on, and only when the status of the hardware is proper. In order to improve on flexibility, a forceToValue mechanism is introduced, to allow the same interlock algorithm to work also if some sensors are misbehaving or are not installed yet (this enables to postpone hardware exchange or minimize the versions of firmware to be deployed).

All facets of the configuration of each firmware block or of each I/O terminal are, as far as possible, performed programmatically, trying to minimize, or remove altogether, the use of the standard graphical interface SystemManager (the configuration graphical user interface provided within TwinCAT 2.xx, see Fig. 4) to minimize the chance of human errors and the need of a large number of highly specialized programmers. At bootstrap phase of the PLC (or when needed due to experts' request), the list of terminals in the fieldbus is compared to the expected one, according to the configuration known by the equipment's database, as is the configurations of the terminals themselves. To this aim also the configuration of the EtherCAT bus, the phase of assigning which signal-variable corresponds to which channel of which terminal, will be automatized using either the EPLAN Electric P8 PPE software plugin [8] or the use of XML-files generated from electrical schemas (useful for TwinCAT3.x). In the same way information on how many and which devices need to be controlled is used to generate as far as possible the firmware that is then run on the PLCs [9].

Each device running at the firmware level is mirrored on the Karabo side by one process running an instance of a software device (in a similar way as done within Tango [10]), a Device class specific to the type of hardware to be controlled. Each Device class, even the simplest, implements a FSM, using Boost Meta State Machine (MSM) [11], mirroring the one existing at the PLC level.

Also at Karabo level there is a communication device, the BeckhoffComDevice, which acts as man-in-the-middle, receives, translates among protocols, if needed, and distributes all needed messages to the appropriate partners, via jms-broker-based message communication or directly via TCP/IP using the open protocol above mentioned. The latter is a purely binary protocol (using containers with: deviceId, key, value), which implements all needed entities: commands, properties, configuration parameters and attributes.

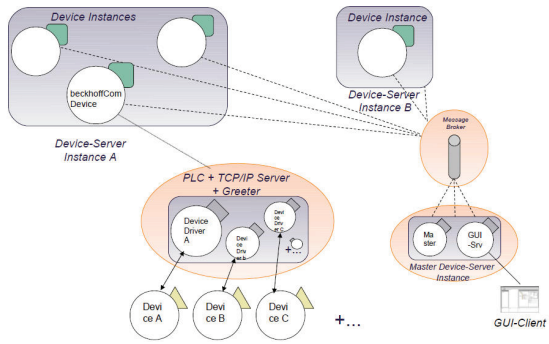

Figure 5: Graphical representation of the distributions of firmware (TwinCAT) and software (Karabo) devices and message paths thereof.

Karabo is a purely event-based control framework, therefore the BeckhoffComDevice instance acts as watchdog over the status of the TCP/IP connection, sending out an heartbeat message and checking that a similar heartbeat is send out from the PLC, in order to recognize silent disconnect errors.

The BeckhoffComDevice has still a very important task, as an instance, running inside a deviceServer, initiate the connection with a PLC, this PLC communicates which devices are running and, in a totally distributed and automatic way, processes are spawned, for each Karabo device instance, within one or more deviceServers running on one or more hosts. As each of these devices has completed its instantiation, it requests a status update to the PLC via the BeckhoffComDevice, the device on the PLC replies writing out all the values of its configuration parameters, attributes and properties, at any later time, values are communicated onChange only. Once the information arrives at the software device the status of the FSM is, by default, adjusted to the state of the device at the firmware level (at instantiation time or later while the device is running, this behaviour can be changed among: followSilently, followAndIssueWarning or goToErrorState according to users' wishes) after that the device is ready to be fully reconfigured, monitored and controlled via the many API realized within Karabo.

Synchronization with Linac Timing System

In order to correctly and uniquely identify to which photon train the status and the values of attributes and properties of the controlled devices within the PLC relate to, each PLC will be interfaced to the accelerator timing system, or Clock and Control (CC), via an External

ISBN 978-3-95450-139-7

Timing Adapter [12], which has already been designed, implemented and tested (see Fig. 6).

Figure 6: Interfacing tests of a Beckhoff PLC to the Clock and Control (CC) system which is delivering time information from the accelerator via an external timing adapter [5]. Data transmission starts at spike in the blue line, data is transmitted red curve. The PLC, running at 1 kHz, generates a digital signal as soon as it has received and interpreted all input data from CC, containing macro bunch number and beam-operation related status information, yellow curve. The jitter in time arrival is 1.3 ms, well below the inter-train time interval or before the arrival of photons.

CONCLUSION

In this paper the design of the photon beam line control system based on Beckhoff TwinCAT PLC using EtherCAT fieldbus to be realized at European XFEL has been described together with the explanation of the concepts and choices underlying it.

ACKNOWLEDGMENT

The authors would like to express their gratitude to J. Kuhn from Beckhoff GmbH Lübeck for his invaluable help and kindness, and all those not explicitly mentioned, but who contributed to the developments described.

REFERENCES

[1] M. Alterelli et al., "XFEL: the European X-ray Free-Electron Laser technical design", DESY XFEL Project Group (2006).
[2] Beckhoff GmbH; http://www.beckhoff.com
[3] TwinCAT; http://infosys.beckhoff.com/
[4] PLCopen; http://www.plcopen.org/
[5] D. Jamsen, H. Buttner "Real-Rime Ethernet: the EtherCAT solution". IET Computing Control Engineering, Feb 2004 vol. 15 issue 1; http://www.ieee.org/computingmagazine
[6] HotConnect: http://infosys.beckhoff.com/index.php?content=../content/1 031/ethercatsystem/html/bt_ethercat_master_fasthotconnect.htm
[7] B.C. Heisen et al., "Karabo: An integrated software-frame-work combining control, data management, and sci-entific computing tasks", these proceedings, TUPPC105, ICALEPCS'13, San Francisco, CA, USA.
[8] EPLAN Electric P8 software; http://www.eplan.de/en/
[9] D. Fernández-Carreira et al., "The design of the ALBA control system: a cost-effective distributed hardware and software architecture" ICALEPS2011, Grenoble, France, September 2011, FRBHMUST01, p. 1318 (2011); http://www.JACoW.org
[10] The Tango collaboration: http://www.tango-controls.org/
[11] Boost libray: http://www.boost.org
[12] C. Youngman, B. Fernandes, P. Gessler, "Electronics developments for high speed data throughout and processing", TUPPC086, and references therein, these proceedings. ICALEPCS'13, San Francisco, CA, USA.

ISBN 978-3-95450-139-7