ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

ADAPTATION OF CERN POWER CONVERTER CONTROLS FOR INTEGRATION INTO OTHER LABORATORIES USING EPICS AND TANGO

S. Page, J. Afonso, C. Ghabrous Larrea, J. Herttuainen, Q. King, B. Todd
CERN, Geneva, Switzerland

Abstract

Modern power converters (power supplies) at CERN use proprietary controls hardware, which is integrated into the wider control system by software device servers developed specifically for the CERN environment, built using CERN libraries and communication protocols. There is a growing need to allow other laboratories to make use of power converters that were originally developed for CERN and, consequently, a desire to allow for their efficient integration into control systems used at those laboratories, which are generally based upon either of the EPICS and TANGO frameworks.

This paper gives an overview of power converter equipment and software currently being provided to other laboratories through CERN's Knowledge and Technology Transfer programme and describes differences identified between CERN's control system model and that of EPICS, which needed to be accounted for. A reference EPICS implementation provided by CERN to other laboratories to facilitate integration of the CERN power converter controls is detailed and the prospects for the development of a TANGO equivalent in the future are also covered.

THE CERN KNOWLEDGE TRANSFER PROGRAMME

The Knowledge Transfer (KT) group at CERN aims to engage with experts in science, technology and industry in order to create opportunities for the transfer of CERN's technology and know-how. The ultimate goal is to accelerate innovation and maximise the global positive impact of CERN on society. This is done by promoting and transferring the technological and human capital developed at CERN.

CERN POWER CONVERTERS

Over the years, CERN has designed various families of power converters. The recent designs all work with the third-generation CERN-designed Function Generator/ Controller (FGC3) and five switched-mode 4-quadrant converter families and five fast-pulsed converter designs are now available under license [1]. The switched-mode converters range from the CUTE (±12.5 A ±15 V) to the SIRIUS (±450 A ±450 V peak, ±200 A ±80 V RMS), which can be combined in series or parallel combinations. The fast-pulsed converters range from 50 A to 3000 A with pulse durations of at least 5 ms and typical repetition rates of 1-2 Hz. The 320 A MaxiDisCap design can operate faster with support for pulse rates up to 10 Hz.

FGC3.1 AND FGC3.2

The current third-generation Function Generator/ Controller is FGC3.1 [2]. This small control computer was de-signed between 2007 and 2011 and went into operation in 2012. More than 2700 have been produced. It runs at 10 kHz and supports a current or field regulation period of 100 µs or multiples of 100 µs. This limits the regulator bandwidth for the rejection of perturbations to around 1 kHz.

Component obsolescence makes further production of the FGC3.1 difficult and some applications need more bandwidth than can be provided with 10 kHz regulation, so the FGC3.2 development was started in 2018 for operation from 2022 [3]. The FGC3.2 will be plug-compatible with the FGC3.1, except for increased power consumption on the +5 V. This may require an upgrade of the PSU, depending on how many other cards are sharing the supply.

If a lab or company wishes to license a CERN power converter design or use the FGC3 controls with an existing power converter, then FGC3.1s are available for small quantities that do not need more than 10 kHz bandwidth. For large quantities or high-performance applications, it will be necessary to wait for the FGC3.2 design to be ready.

Firmware and FPGA Programming

In both cases, CERN will provide the firmware, FPGA programming and support during integration and commissioning as well as long-term updates as part of the license agreement. All the firmware and FPGA programming source code will be available for review, but CERN will not give permission to compile them locally nor to create derivative versions or distribute this code. CERN invites bug reports, bug fixes and feature requests from licensees.

Despite the hardware reaching the end of its production life, the FGC3.1 programming will continue to be supported for at least 20 years. The FGC3.2 should be producible until at least 2027 and it will also be supported at CERN for at least 20 years. We anticipate future refreshes of the design after 2027 to replace obsolete components.

The step from FGC3.1 to FGC3.2 includes a significant change in architecture. The older design is based on the combination of a Microcontroller Unit (MCU) + Digital Signal Processor (DSP) + Field Programmable Gate Array (FPGA), while the newer design will use a multicore ARM-based System-on-Chip (SoC) with an FPGA. The FGC3.1 was a "bare-metal" design with a tiny real-time micro-kernel on the MCU and no operating system on the DSP. With FGC3.2, the System-on-Chip (SoC) will run Linux with the PREEMPT_RT patch.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

CONTROLS ELECTRONICS

The second-generation FGC2 is used with analogue voltage sources in the LHC. The FGC supports the digital current regulator in software with an analogue voltage reference emitted via a DAC. The FGC3 supports this mode of operation, however, it also has digital serial interfaces to work with digital voltage sources.

CERN has developed a family of low-level power converter control cards and crates that can cover a wide range of different topologies. Collectively, these are known as RegFGC3 cards and crates. At the heart of switched-mode designs is the VSREG_DSP board, which combines the TMS320C28346 DSP with a Spartan 3 FPGA. This DSP has nine high-resolution PWM generators, sufficient to drive four IGBT H-Bridges. It supports the voltage regulator, filter damping and firing control in software.

An analogue thyristor firing card is included in the catalogue of RegFGC3 cards. When this is used, the FGC3 implements the voltage regulator and filter damping loop in software and the firing reference is emitted via the DAC.

Other cards support state control, analogue and digital interlocks, fast-accelerator-interlocks, White Rabbit networking, external high-performance ADCs and other facilities.

RegFGC3 card and crate designs are available under license. The first KT agreement was signed with a European manufacturer in 2016. They successfully used the switched-mode controls crate and cards with their 20 kA 80 V converter design for the TRIUMF main cyclotron magnet [4]. This was commissioned in March 2018.

SOFTWARE SERVICES AND EXPERT TOOLS

CERN has more than five thousand power converters across the accelerators, including 1750 in the LHC. Around four thousand are now controlled by FGCs, physical or virtual, and this number continues to rise. Managing such a large park of equipment requires effective expert tools that scale to such large deployments. These are part of CERN's FGC offer [5].

PowerSpy and FortLogs

PowerSpy is an advanced web application that was developed at CERN since 2015. It allows power converter experts to acquire and analyse analogue and digital signals as well as tabular log data from FGCs and other sources. Common analysis features such as fast Fourier transforms and first-order time constant analysis are provided. A public standalone version of the application is available for testing online [6].

FortLogs is a log acquisition database that is in development for operation in 2020. Based on Postgres, it presents a RESTful API to allow applications to store, stream, tag, search and retrieve up to a million multi-megabyte acquisitions. A Python library makes it easy for Python programs to use the FortLogs API. PowerSpy uses a Python back-end under Apache and it will use the library to save all acquisitions in FortLogs as well as provide an elegant UI to search the contents. Automatic nightly clean-up will delete old acquisitions that have not been pinned. PowerSpy, FortLogs and the Python library can be licensed independently and used as the acquisition and analysis tool for any data source.

FortLogs provides the foundation for the FGC logging server, which automatically reads out log data from FGCs if their power converter trips.

Terminal

FGC3s have a USB port to connect a PC running a terminal emulator. It is possible to connect to the same terminal via the fieldbus using PowerSpy or a command-line tool.

THE CERN CONTROLS MODEL

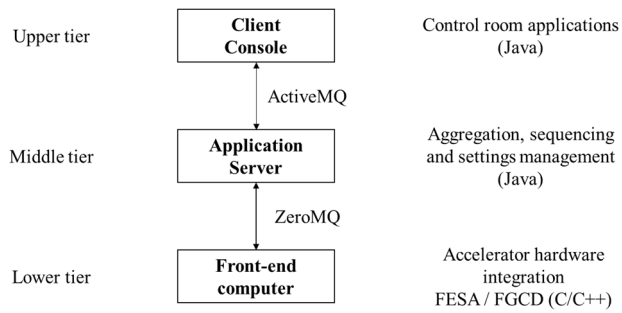

Figure 1: The CERN controls model.

Three-Tier Control System

The CERN control system is based around what is known as the three-tier model, as shown in Fig. 1. The upper tier is typically an application running on a console in the control room. The lowest of the three tiers is a diskless front-end computer, which is physically connected to one or more pieces of hardware to be controlled, either directly or via a fieldbus. The lower tier is responsible for the integration of the hardware into the wider control system, offering a controls interface, reporting alarms and driving the hardware in real time. Between the upper and lower tiers, there is a middle tier, which aggregates and manages the devices presented by the lower-tier front-ends.

Communication between the three tiers is handled by standardised middleware.

The Device / Property Model

Physical or logical equipment in the CERN control system is represented by what is known as the Device / Property Model. Each piece of equipment is represented by a named device. Attributes of the device which either represent settings affecting the operation of the device or report data acquired by the device are known as properties. A client may set, get or subscribe to those properties in order to modify or retrieve their values.

Implementation

Control-room applications (the upper tier) at CERN are mostly written in Java, although Python is rapidly growing in popularity. The middle tier, which provides business

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

logic, settings management and sequencing, is also implemented in Java, with ActiveMQ middleware used between those two tiers for compatibility with the Java Message Service (JMS).

Software on the lower tier consists of a combination of drivers for connected hardware and a real-time software device server, built using one of the FESA or FGCD frameworks common at CERN. This software is implemented in C/C++. Communication with the lower tier uses ZeroMQ middleware to present the device / property interface.

The FGC Lower Tier

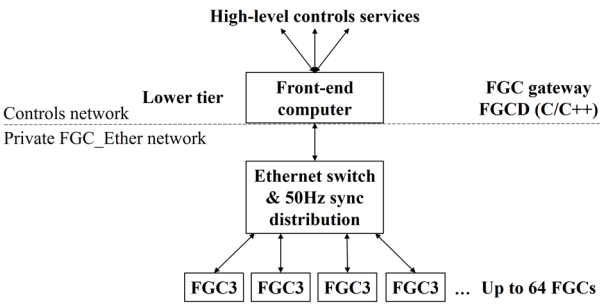

Figure 2: The FGC lower tier.

In the case of the FGC power converter controls, the lower tier front-end computer is connected to an Ethernet-based real-time fieldbus, known as FGC_Ether, to which up to 64 FGC power converter controllers may be connected, as shown in Fig. 2. The computer runs software that communicates with the FGCs in real time using raw Ethernet and acts as a gateway to the wider control system.

THE EPICS CONTROLS MODEL

The Experimental Physics and Industrial Control System (EPICS) is a set of tools and software components that can be adopted by developers to create and implement control systems. Its development began around 1991 and since then it has been adopted by many accelerator laboratories as well as other physics facilities. It provides a tool-based, distributed, event-driven architecture that can scale from small facilities to large laboratories.

Like the CERN controls model, it consists of a 3-tier control system. Both 1st and 3rd layers are similar in the sense that they refer to client applications and front-ends (FGC_Ether gateway / FESA device server or IOC), respectively. In contrast, the EPICS 2nd layer refers to the network, connected through the Channel Access (CA) protocol, while at CERN it refers to physical application servers.

The Channel Access Protocol

Channel Access (CA) offers clients the possibility to get, put or monitor Process Variables (PVs), which are identified by a unique name across the entire network. These commands are equivalent to CERN's get, set and subscribe commands.

In addition, CA also provides a search command, which is used to find out which CA servers contain a set of PVs, by broadcasting the request to all CA servers. The servers containing the PVs will respond back. At CERN, this functionality is instead provided by a central directory service which can be queried.

Input / Output Controller

An important module in the EPICS architecture is the Input / Output Controller (IOC). This module serves mainly as a CA server, although it can also be a CA client to other IOCs.

As mentioned earlier, the IOC fills the 3rd layer in the EPICS control model, and can be connected to hardware or software interfaces. It is comparable in its functions to CERNs FGC gateway or FESA device server.

In the next section we will describe our approach to exposing FGC properties via an EPICS IOC.

EPICS IMPLEMENTATION

The IOC contains several components, following a modular design approach.

One of the most important is the IOC database, where the records reside. The records themselves, depending upon their type, can contain different fields, and are able to perform operations on their data as well as access other record values. Since they fill a similar role to CERNs properties, they are the logical choice for exposing them within an EPICS system.

Three other important components are record support, device support and device drivers. Record support routines implement all record-specific behaviour, making it unnecessary for the database to have any record-type specific knowledge. In a similar way, device support modules implement the logic required for interacting with a device, making it unnecessary for records to have any knowledge about the devices they are handling. Finally, device drivers can be developed to handle more complex hardware access, if device support is not sufficient.

Many custom tools have been developed to fit these layers. They abstract complex EPICS and CA concepts from the developer, letting them focus on the actual interaction with the devices. To integrate FGC property handling with EPICS, we decided to use two well-known device support development tools: asynDriver and StreamDevice. Both are widely known by the EPICS community and maintained by experienced EPICS developers.

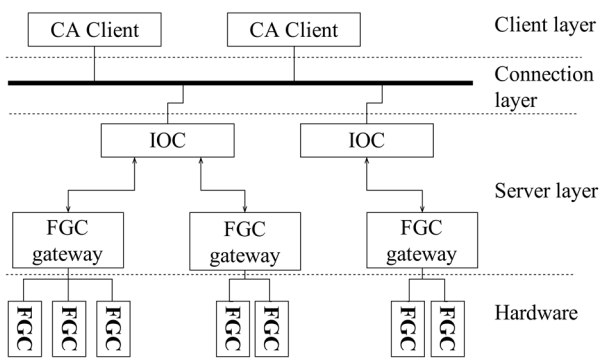

Figure 3: Merging of EPICS and FGC architectures.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

Figure 3 shows how EPICS and FGC systems can be combined together. However, since both FGCs and IOCs take the role of databases in each of their systems, it is important to guarantee that the information in the IOC records is consistent with the information in each FGC property. In order to do so, the IOC records should only serve as a simple abstraction layer, and processing them should always trigger the reading or writing of the underlying FGC properties. Record processing is automatically triggered by CA puts and by hardware interrupts, so both put and monitor commands are guaranteed to act upon the real FGC property value. However, getting a value from a record must be preceded by a processing order (i.e. using a CA put), otherwise it will use the "cached" value in the record without a guarantee that it is the same as in the FGC.

Command / Response Protocol

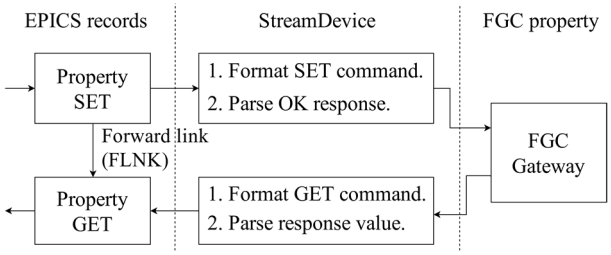

Figure 4: Command / response protocol integration with EPICS.

Figure 4 shows how the command / response protocol is implemented in EPICS for FGCs. For each FGC property that is exposed, there are usually two EPICS records: one for setting and the other for getting the value. Exceptions apply for properties which are only settable or gettable.

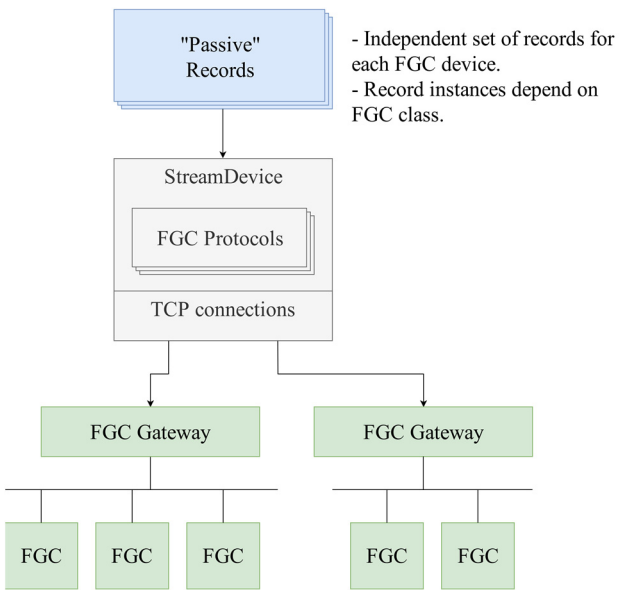

Figure 5: Command/response protocol integration with EPICS.

As illustrated in Fig. 5, StreamDevice is used to translate an EPICS record put/get into an FGC command/response protocol interaction. If the response for setting/getting this new value is an error, then StreamDevice will recognize it as such and invalidate the corresponding record. This will give clients the necessary feedback of whether actions are successful. In addition, writing to a SET property record will also trigger the reading of the respective GET property record, through a forward link field (FLNK). This will guarantee that the GET property record will stay consistent with the new value in the FGC. However, it is important to note that if other FGC clients are changing these properties actively, it will always be necessary to explicitly process the record before reading it. This also applies to properties that are only gettable.

Since there are hundreds of FGC properties, we had to make use of record templates. On the StreamDevice level, basic protocol files were generated for handling string, integer and float type properties. More types can be added if necessary. Then, on the record level, several record templates were designed – for setting and getting FGC properties of different types – that take advantage of the protocols previously mentioned. Finally, each record template can be used to instantiate any number of records, in order to cover all the exposed FGC properties, using instantiation files. These files are auto-generated, each covering the set of properties of one FGC class. These instantiation files should be imported in the IOC start-up script, for each of the FGCs connected to the IOC.

FGC Published Status Decoding (UDP)

Figure 6 shows the FGC status publication integration with EPICS. These messages are periodically unicasted by the FGC gateway to a set of pre-defined IP addresses (in this case the IOCs), where they must be received, decoded and used to update the corresponding records. Each message consists of a single UDP datagram, containing enough slots of data for 65 devices (FGC Gateway + a maximum of 64 FGCs) in binary format. Each device may have a different set of properties encoded, depending upon its class.

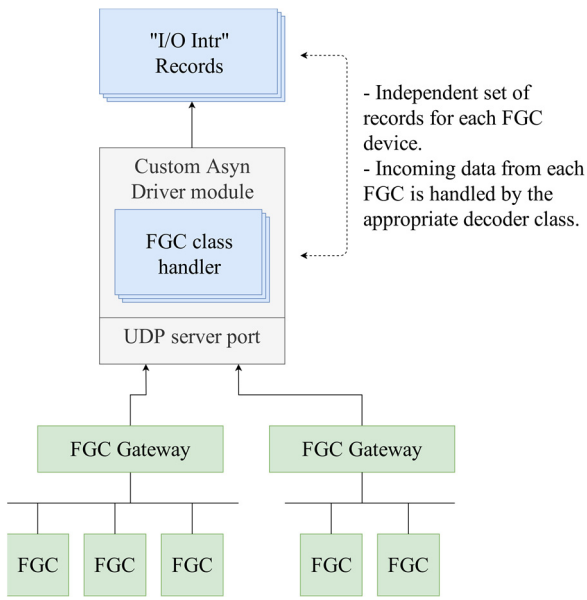

Figure 6: UDP status publication integrated with EPICS.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

However, since this data is not self-describing, the IOC must know in advance how to decode the published properties of each device.

The tool selected for this device support module was asynDriver. It provides a structured environment for developing support modules for hardware devices, taking care of most of the communication with other EPICS layers. The user only has to select which interfaces to make available and implement how they communicate with the device. These interfaces will affect how the records interact with the asynDriver parameters.

We identified the types of properties exposed by the FGCs and matched them to asynDriver parameters and EPICS records, as shown in Table 1.

Table 1: FGC Property and AsynDriver Parameter Types

FGC published property type	AsynDriver parameter type	Record type
Int (8, 16, 32 bits)	asynParamInt32	longin
Float (32 bits)	asynParamFloat64	ai
Bitmask (8 or 16 bits)	asynParamUInt32Digital	bi (each bit separately)
Enum (8 bits)	asynParamUInt32Digital	mbbi

Internally, asynDriver uses several auto-generated C++ classes, all sharing a common abstract interface. Each one is responsible for parsing the results of a different FGC class and exposing them as asynDriver parameters.

When the fgcudp device support module is instantiated by the IOC startup script, it should be configured with the following information:

- UDP port number to listen to.
- Which gateways will be sending their publication data to that port, and the data ID (this allows the data from multiple gateways to be received in one port). It is also possible to set a reception timeout.
- For each gateway, the existing FGC device numbers, names and classes. Internally, the device support module will instantiate the appropriate C++ class to handle each FGC.

Accordingly, for each device, a set of records should be instantiated depending upon the FGC class. These are I/O-triggered records, updated when the asynDriver receives a new value. They can also be set as invalid if the received data format is unknown, or if there is a timeout while waiting for new UDP datagrams.

As in the command / response protocol, there is a different record instantiation file for each FGC class. These records will interact with the parameters exposed by asynDriver. Both C++ and record files are auto-generated simultaneously to ensure that they are consistent.

THE TANGO CONTROLS MODEL

TANGO is an object-oriented control system started as a collaboration between ESRF and SOLEIL laboratories. It uses a combination of CORBA and ZeroMQ for network communication.

Several features are exposed per device, such as: device name, properties (persistent storage items), attributes (device data fields), commands (device actions), events and others.

In order to provide the above features, TANGO devices may use different services provided by the framework. An important one is the TANGO database (MySQL) which is used as persistent storage for device properties and device names.

Several tools are provided, such as Jive – a graphical tool for browsing and editing the TANGO database and ATKPanel – used to control any device.

These modules, together with existing TANGO bindings to C++, Java and Python, make it simple to develop, test and release new device classes.

TANGO IMPLEMENTATION

This work is currently under development. It may not represent the final product. For this implementation, we decided to use pyTango, which allows us to use several Python APIs developed at CERN, such as pyFGC.

Each TANGO attribute will expose a single FGC property. The attribute read/write methods make use of pyFGC to translate a write into a SET command and a read into a GET command. Then, the respective FGC responses will be parsed and used to update the attribute value or throw an exception in the case of an error. Alternatively, TANGO commands can be used to expose FGC write-only properties.

In case of UDP status properties, a separate thread will use pyFGC to wait for new UDP packets. It will then parse them and update the respective attributes.

Finally, naming data will be exposed as TANGO database properties, which are loaded by each device.

We will make use of Python metaclasses to generate each TANGO FGC class dynamically, based on a JSON file containing a list of FGC properties. This will make it very simple to add or remove FGC properties, since only the JSON file will need to be modified. This task can even be done automatically, before the release of each new version.

FUTURE PROSPECTS

CERN power converter controls are now in use at three other labs via knowledge transfer agreements. The EPICS reference implementation for FGCs and ongoing development of an equivalent for TANGO will ease their integration into control systems at a wider range of labs and several organisations have expressed interest. We look forward to establishing new collaborations and to further developing our offering to benefit a wider scientific community.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA105

REFERENCES

[1] https://kt.cern/technologies/magnet-power-supplies

[2] B. Todd, "Function Generator/Controller 3.1 (FGC3.1)", https://edms.cern.ch/document/1377659

[3] S. T. Page, C. Ghabrous Larrea, Q. King, B. Todd, S. Uznanski, and D. J. Zielinski, "FGC3.2: A New Generation of Embedded Controls Computer for Power Converters at CERN", presented at the 17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOPHA106, this conference.

[4] S. Carrozza *et al.*, "The New 20 kA 80 V Power Supply for the 520 MeV H- Cyclotron at TRIUMF", in *Proc. 9th Int. Particle Accelerator Conf. (IPAC'18)*, Vancouver, Canada, Apr.-May 2018, pp. 3792-3795. doi:10.18429/JACoW-IPAC2018-THPAL062

[5] Q. King, "Using CERN Power Converter Controls with EPICS and TANGO", https://edms.cern.ch/document/2002516

[6] https://cern.ch/service-powerspy