INTEGRATION OF PLC'S IN TANGO CONTROL SYSTEMS USING PyPLC

S.Rubio-Manrique, M.Broseta, G.Cuní, D.Fernández-Carreiras, A.Rubio,
J.Villanueva, ALBA-CELLS, Cerdanyola del Vallés, Barcelona, Spain

Abstract

The Equipment Protection Systems and Personnel Safety Systems of the ALBA Synchrotron are complex and highly distributed control systems based on PLC's from different vendors. EPS and PSS not only regulate the interlocks of the whole ALBA facility but provide a large network of analog and digital sensors that collect information from all subsystems; as well as its logical states. TANGO is the Control System framework used at ALBA, providing several tools and services (GUI's, Archiving, Alarms) in which EPS and PSS systems must be integrated. PyPLC, a dynamic Tango device, have been developed in python to provide a flexible interface and enable PLC developers to automatically update it. This paper describes how protection systems and the PLC code generation cycle have been fully integrated within TANGO Control System at ALBA.

INTRODUCTION

ALBA[1], member of the Tango Collaboration[2][3], is a third generation Synchrotron in Barcelona, Spain. It provides light since 2012 to users through its 7 beamlines,with 2 more under construction. The ALBA Control Section (ACS) is currently formed by 16 engineers devoted to the development of our Tango-based SCADA frameworks (Taurus[4], Sardana[5][6] and PANIC[7]) and PLC systems.

The ALBA Equipment and Personnel Protection Systems[8][9] (EPS and PSS) are distributed PLC-based systems autonomous from the Tango Control System. Both EPS and PSS are homogeneous systems based on single vendors (B&R and Pilz respectively). While the tasks to be done by the PSS are clearly specified and delimited by the ALBA Safety Group, the EPS PLC's became instead a versatile system that has been adapted for interlock, acquisition and motion control in both accelerators and beamlines.

Although other PLC based systems are used in ALBA to control the RF circulators, bakeout controllers and water or air cooling systems; the EPS is the most complex system managed by PLC's, using 58 B&R CPU's and 110 periphery cabinets to collect more than 7000 signals. In addition to the main purpose of protection, several hundreds of signals distributed across the whole system are acquired for diagnostics and control of pressures, temperatures and movable elements

The integration of the management of an independent system like EPS in the Tango Control System required of several phases, starting from the collection of cables from the Cabling Database to the final auto-generation of GUI's for both EPS Expert GUI and operator users (Taurus) .

GENERATION OF CODE VARIABLES

The Cabling and Controls Database

Every cable and equipment installed in the ALBA Synchrotron is registered in our Cabling and Controls Database (CCDB). Developed in 2007[10] by our Management and Information Software section (MIS) using MySQL and web technologies, it was the main support tool for the design and construction phase and now it is still kept updated as the main repository of equipments and configurations in our Accelerators and Beamlines. As of 2015 it lists 385 racks with 7131 equipments of different 874 equipment types. These equipments are connected using 20053 cables of 487 different cable types with a total length of 172.26 Km.

The CCDB provides, for each of our PLC CPU's or remote peripherals, the full list of connected devices, its equipment types, cable configurations, terminal used and the distribution of hardware in racks, including the routing of cables between hardware and control devices (Fig. 1). It also provides fast access to all available documentation for each type of equipment.

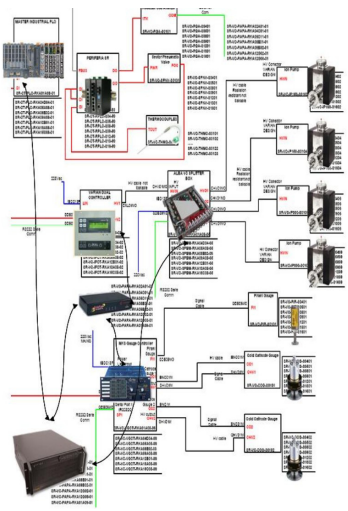

Figure 1: Diagrams of every subsystem have been produced prior to its introduction in our cabling database.

The PLC Auto-Generation Tool

The CCDB python API[11] provides full access to the Cabling Database from our control system tools. The API methods allow to search for equipments and get lists of connections, names and network information. These links are used to enable our Auto-Generation coding tool correlating the information of the equipments from the CCDB with the logics defined for them in the EPS.

To do so, all the common elements of our different Equipment Protection Systems have been standardized in

ISBN 978-3-95450-148-9

our EPS_LIBRARY project. This library, later exported to our different CPU sub-projects, provides the standard logic behavior for managing each equipment type connected to the PLC and its remotes. These standard procedures include managing in/out digital signals for valves, alarm/warning ranges for analog values, value conversions for thermocouples, linear correlation for 4-20 mA transducers and standard conversion scales for flowmeters amongst others.

For each CPU of the EPS, the PLC Auto-Generation tool (Fig. 2) extracts exhaustive reports of all equipments connected to the CPU and its remotes, parses the naming of these equipments and its logics depending on the cable and terminal connectors used in both sides (along with notations introduced by the Controls or Electronics engineers). This full cable report is later translated to a full list of the PLC variables with its matching PLC code structures.

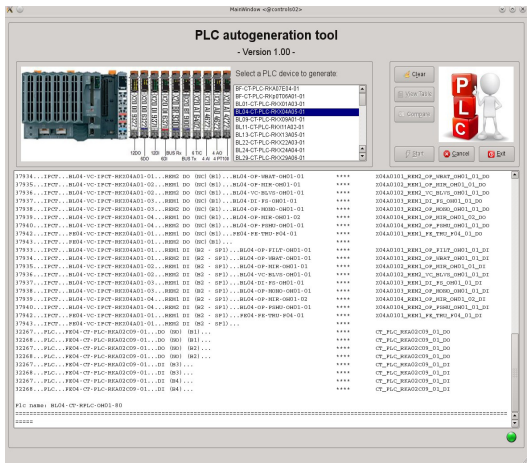

Figure 2: PLC Auto-Generation tool.

These steps does not produce the final code of the PLC, but a detailed guide of the structures to be generated. The control engineer then must coordinate logics of individual variables and program the most specific logics of every beamline or accelerators section, being capable of focusing in the most critical parts instead of the tedious and iterative variable naming.

The processes for auto-generation of code variables were originally developed using visual basic, which required manual extraction from database and generated static reports. Migrating these procedures to python allowed to develop tools that can be executed automatically, enabling the automatization of regular code review and error checking.

The final output of the auto-generation process is the variable modbus mapping spreadsheet, commonly known as the EPS CSV. This file will be used later to generate the EPS Expert GUI and the Tango Attributes used by User-Level GUI's.

PLC TANGO DEVICE SERVERS

PLC's at Alba are accessed using two protocols: Fetch & Write for Siemens PLC's (cooling systems) and Modbus (over TCP or RS485) for the rest of systems, including EPS.

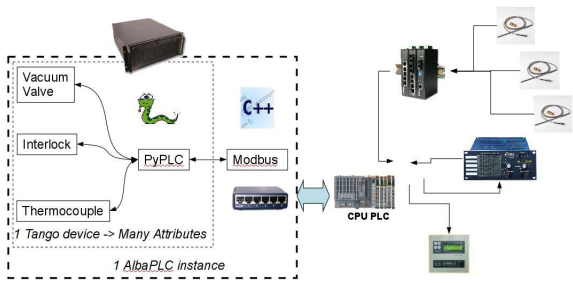

Figure 3: Structure of the PyPLC device server.

All Modbus-based PLC's are accessed using the PyPLC[12] device server. This python device server[13] (in use since 2008) have been developed over the Modbus C++ (Fig.3) device class to allow four different types of access:

- Directly executing Modbus commands in the PyPLC device server, direct actions mapped to specific bits (e.g., open Front-End).
- Reading the whole Modbus address space into array attributes, method optimized to get maximum update frequency, used by the Expert GUI.
- Exporting EPS variables as individual Tango attributes, thus enabling Tango features like ranges, alarms, archiving, labels, etc.
- Exporting EPS variables as individual Tango devices, typically valves with its own Open/Close commands.

The PyPLC exports the most standard Modbus commands (Read/Write Input/Holding Registers) and PLC variable types (Coil / Flag / Int / Long / Float / Ieee Float). This PyTango device server uses the Fandango Dynamic Attributes [14] template to generate new attributes at runtime based on user-defined formulas (Table 1). These dynamic generation of attributes boosted the initial prototyping of the devices, providing enough functionality for the most elementary systems (temperature controllers, stand-alone PLC's) and allowing to customize the attribute generation done by the EPS Auto-generation tools.

Table 1: PyPLC Attribute Formulas, an Array, a Writable Boolean Flag and Two Commands

`TEMPERATURES= DevVarLongArray(Regs(7800,100))`
`DIO_01= bool( READ and Flag(80,7) or WRITE and WriteFlag(81,7,VALUE))`
`Open_PNV01=(WriteBit(193,2,1),1)[-1]`
`Close_PNV01=(WriteBit(193,1,1),0)[-1]`

ISBN 978-3-95450-148-9

User-defined formulas allowed to create meta variables (e.g. BL_READY) or special commands to be used in PANIC Alarms [15] or our Sardana Experimental Framework to control pneumatic devices in the Beamline (Table 1).

As of 2015 both the automatic-generated variables and the user-defined variables are stored as properties in the Tango Database, being the new beamline MIRAS [16] the first in which all standard attributes will be generated from EPS CSV files instead; reserving Dynamic Attributes formulas only for special behaviors. This is done using the AlbaPLC, a PyPLC subclass that implements in the Tango side the data structures of the EPS_LIBRARY.

HUMAN MACHINE INTERFACES TO EPS

EPS GUI's

The current application developed to interact with the EPS system is the EPS Expert GUI (Fig. 4). This application loads all the modbus mapped variables from the PLC and organizes the whole memory structure in several tabbed panels, grouped by variable type.

From the EPS GUI application we can manage, after LDAP user validation, the Alarm/Warning levels of every analog signal, interlock status (active/latched/was_first) for analog and digital inputs, forced and disable states both for interlocks and digital outputs, and the status of the PowerLink word; synchronized in all the EPS cpu's to share interlock signals between them.

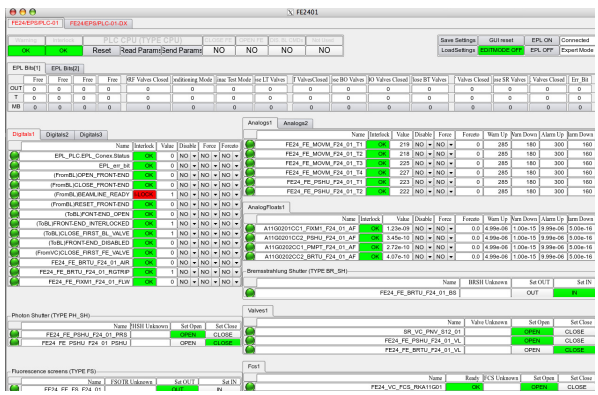

Figure 4: The EPS expert GUI.

The information regarding the several subsystems managed by the EPS (vacuum, magnets, RF, bpm's, front-ends) is later summarized in 2 user-level applications: the ALBA-EPS (that allows fast diagnose and acknowledge of interlocks in the accelerators side) and the Machine Protection System (with specific diagnostic tools in case of BPM or RF interlocks).

Tango, Sardana and PANIC

Many analog and digital signals in ALBA beamlines have been exported to our Tango-based experiment control framework, Sardana. The Sardana suite allows to add Tango controlled hardware as experimental channels, using them as I/O or experimental data (e.g. temperatures and vacuum pressures). In addition, limited motion control have been implemented using PyPLC and dedicated structures in the EPS_LIBRARY, which allowed direct control from Sardana macros or scans developed by scientists.

A similar approach to higher-level procedures involving several Tango devices are implemented using automated actions from PANIC Alarm System[7]. Those macros allow both scientists and PLC engineers to program automated actions on elements controlled by the EPS (valves, shutters) during experiments. These actions do not bypass the EPS, as it is always kept as an autonomous system ensuring the safe state of the installation, but allow the recovery of the system once all the permits conditions are matched.

Figure 5: The VACCA User Interface.

Taurus, VACCA and the EPS User GUI

All those high-level features were enabled by the Dynamic Attributes syntax, that allowed specific interfaces to express intermediate states like warning, external interlocks, complex error codes, attributes managed by multiple registers (e.g. multiple-stage pneumatic elements), etc. Those attributes use Tango Qualities to complement the raw attribute values, passing the information regarding Alarm/Warning/Moving limits and positions with each value sent to clients.

Qualities and state machines become visible in Taurus applications, being converted into colors easy to catch on either control panels or synoptic applications like VACCA [17]. All attributes exported to Tango from the PLC become available to all Taurus widgets (device panels, forms, trends) and can be displayed in tabbed panes as separated elements or in combination with those equipments to be controlled. Those same attributes also allow the user to customize how this information is displayed, either modifying the attribute label or its units and formatting.

This transparent interaction between the Tango and EPS control systems is pushing the need of an standard User-level GUI for the EPS, providing the same information than the Expert GUI but in a way that makes easier to locate and diagnose the causes of an interlock (and the best way to recover). This diagnose feature is being

ISBN 978-3-95450-148-9

implemented as a PLC source navigator integrated within our current Vacuum Control Application (VACCA), which has been already extended to manage both Archiving and Alarms services.

CONCLUSION

After 8 years of design, installation, commissioning and operation the EPS has become a mature control system by itself, achieving a high level of consistency and reliability.

Its integration within Tango using PyPLC allowed to integrate the PLC systems in all our Tango services. Not only in “passive” ones like Archiving or Alarms, but also as an active part in our experiment-control framework, Sardana.

The next step is the integration of the current tools in an automated cycle of continuous development and delivery. First steps in this direction have been achieved introducing all the PLC's code in SVN repositories, and later migrating to python all the auto-generation tools.

Now, after several years of operation, the two new beamlines under construction at Alba are bringing an opportunity to put all the recent improvements and develop a clean an optimized solution for them. Learning from the previous experience and aiming to refine the new procedures to be later reapplied to the old beamlines.

The current level of integration between Tango and EPS on older beamlines is still much lower than in the accelerators side. This is due to the bigger number of special cases and exceptions while integrating specific devices in the CCDB catalog or the EPS_LIBRARY. It is expected that the new tools as the Auto-generation and the Source navigator will help the PLC engineers to simplify the logics and naming of the existing systems, but there will be always a big percentage of beamline variables too specific to be standardized.

For those cases, enabling all EPS variables as Tango attributes will allow the users to setup and use their own Labels, while keeping the cabling-based tags as attribute names. This kind of mixed naming schema (attribute for control engineers, label for scientists) should reduce the number of naming conventions exceptions and provide enough consistency to close the auto-generation loop.

ACKNOWLEDGEMENT

Many former ALBA engineers have collaborated in the PLC-related projects in the last 6 years: A.Rubio, R.Ranz, R.Montaño, R.Suñé, M.Niegowski, M.Broseta and J.Villanueva.

The collaboration of Tango core developer, Emmanuel Taurel and former ALBA developer R.Suñé was fundamental in the development of Dynamic Attributes templates and debugging of PyPLC performance.

The work of the Generic Software Group at ALBA in the development of Taurus and the Taurus GUI framework has made possible the evolution of the EPS Graphical User Applications.

Last but not least, none of the elements in the auto-generation cycle would have work without the Cabling and Controls Database, developed by the Management and Information Systems section of ALBA.

REFERENCES

[1] ALBA website: http://www.cells.es
[2] TANGO website: http://www.tango-controls.org
[3] A.Götz, E.Taurel et al., “TANGO V8 – Another Turbo Charged Major Release”, ICALEPCS'13, San Francisco, USA (2013).
[4] C. Pascual-Izarra et al., “Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus”, ICALEPCS'15, Melbourne, Australia (2015).
[5] T.Coutinho et al., "Sardana, The Software for Building SCADAS in Scientific Environments", ICALEPCS'11. Grenoble, France (2011).
[6] Z. Reszela et al., “Sardana – A Python Based Software Package for Building Scientific Scada Applications”, PcaPAC'14, Karlsruhe (2014).
[7] S.Rubio et al., “PANIC, a Suite for Visualization, Logging and Notification of Incidents”, PcaPAC '14, Karlsruhe, Germany (2014).
[8] R.Ranz et al, “ALBA, The PLC based Protection Systems.”, ICALEPCS'09. Kobe, Japan (2009).
[9] D.Fernández-Carreiras et al, “Personnel protection, equipment protection and fast interlock systems”, ICALEPCS'11, Grenoble, France (2011).
[10] D. Beltran, et al. “ALBA Control And Cabling Database”, ICALEPCS'09, Kobe, Japan (2009).
[11] S. Rubio-Manrique et al. "A Bottom-up Approach to Automatically Configured Tango Control Systems", ICALEPCS'11, Grenoble, France. (2011).
[12] S. Rubio-Manrique et al., "PyPLC, A Versatile PLC-to-PC python interface", PCaPAC'14, Karlsruhe, Germany (2014).
[13] D.Fernández et al. “Alba, a Tango based Control System in Python”, ICALEPCS'09, Kobe, Japan (2009).
[14] S.Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS'09, Kobe, Japan (2009)
[15] S.Rubio, et al. “Extending Alarm Handling in Tango”, ICALEPCS'11, Grenoble, France (2011).
[16] MIRAS website: http://www.miras2.es/
[17] S. Rubio-Manrique et al., “Unifying All TANGO Control Services in a Customizable Graphical User Interface”, WEPGF148, ICALEPCS2015, these proceedings.

ISBN 978-3-95450-148-9