ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH3001

AN EPICS CHANNEL ACCESS IMPLEMENTATION ON SIEMENS PLCs

M. Boros[†], R. Fernandes, European Spallation Source, Lund, Sweden
B. Péceli, G. Singler, evopro Innovation Kft, Budapest, Hungary

Abstract

At the European Spallation Source (ESS), a neutron research facility in Sweden, most of the controls are based on PLCs and layered in the following (traditional) way: field equipment ↔ PLC ↔ EPICS IOC ↔ high-level applications. In many situations, the EPICS IOC layer will not implement control logic *per se* and is only used for converting PLC tags into EPICS PVs to enable the usage of high-level applications such as CS-Studio, Archiver Appliance, and Alarm System.

To alleviate this (traditional) way of doing controls, we propose a simpler approach: implementation of the Channel Access (CA) protocol in the PLC layer for the latest family of Siemens PLCs to remove the EPICS IOC layer. We call it S7EPICS. S7EPICS respects version 13 of the CA protocol specification, and supports multiple EPICS-based client connections at the same time – e.g. CS-Studio, Archiver Appliance – without a noticeable service degradation (i.e. delays).

In this paper we introduce this implementation, its architecture and workflow, benchmarking results of tests performed, and future developments that could be pursued such as authentication & authorization mechanisms using, e.g., the Arrowhead Framework.

INTRODUCTION

Integrating a Siemens PLC with EPICS [1] high-level applications requires an EPICS Input/Output Controller, or IOC, properly configured with EPICS modules capable of communicating with the PLC using, e.g., s7plc-comms [2] or OPC UA [3]. Configuring an IOC, even for small projects, demands EPICS programmer skills, time and other valuable resources – e.g. a machine prepared to run the IOC. In some (control) use-cases, it would be simpler and more effective if the PLC could “talk” the Channel Access (CA) protocol [4] itself so that it could be interfaced directly with high-level applications.

In this paper, we describe an open source PLC code called S7EPICS [5], which allows a Siemens S71500 family PLC instance to declare and handle EPICS Process Variables (PVs) that EPICS aware applications may consume directly without an actual IOC running.

S7EPICS is a PLC code implemented as a TIA Portal library. When the library is imported into an existing TIA project and it is called in the main PLC cycle, the PLC becomes a fully-fledged CA server capable of receiving and sending EPICS CA protocol messages through UDP (to discover producers – i.e. servers – of PVs) and TCP packets (to exchange PVs data between producers and consumer – i.e. servers and clients).

DESCRIPTION

Typically, a common control system scenario encompasses one or more field equipment that are controlled/measured by a PLC. The PLC, eventually equipped with I/O cards, is located in a control cabinet and field equipment are connected to either the PLC CPU or to one of its I/O cards. The business logic is implemented at the level of the PLC, working as a self-contained and independent (hard) control system. In addition, field equipment’s signals – interfaced by the PLC – are usually consumed by high-level applications to solve domain specific issues.

In case of the (soft) control system is based on EPICS (and consequently high-level applications too – e.g. CS-Studio for OPI screens designing, Archiver Appliance for signals archiving), the integrator has to create and configure an EPICS IOC that 1) communicates with the PLC CPU or its I/O cards and 2) “converts” the PLC tags into EPICS PVs in a bidirectional communication. The IOC does not implement any control logic though, and (only) has as a main function to map PLC variables into corresponding EPICS PVs. To implement this mapping, the addresses and, sometimes, the offsets of all communication variables have to be specified in the EPICS IOC configuration, which is time consuming and prone to error. Moreover, the IOC has to be maintained and executed on a (physical or virtual) machine, adding an extra layer of complexity to the control system as a whole. Figure 1 illustrates the traditional PLC-EPICS integration involving an IOC layer between the PLC and EPICS high-level applications.

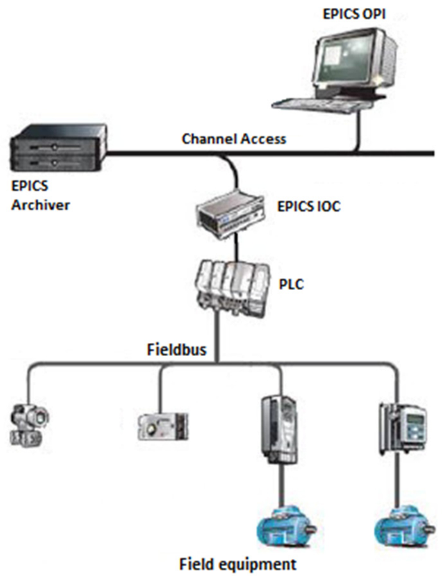


Figure 1: Traditional PLC-EPICS integration.

[†] miklos.boros@esss.se

In contrast, when a PLC (of the Siemens S71500 family) is equipped with the S7EPICS library, it is capable of both answering PV name search requests and interchanging PV data with EPICS high-level applications, making the IOC layer redundant – assuming that the control logic is fully respected/implemented by the PLC. Figure 2 illustrates the (simplified) integration between the PLC and EPICS high-level applications when using S7EPICS.

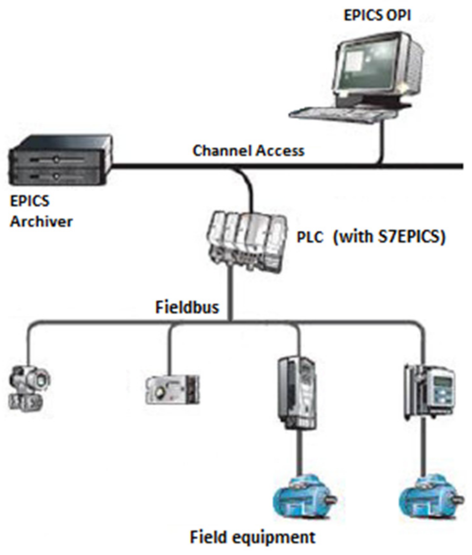


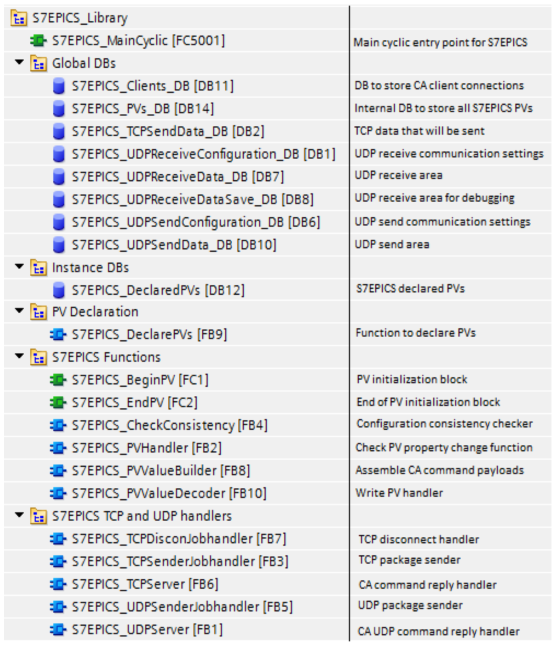
Figure 2: PLC-EPICS integration with S7EPICS.

Channel Access (CA)

The Channel Access (CA) is a communication protocol that all EPICS aware applications need to implement in order to communicate with IOCs or CA servers. A Process Variable (PV) is the addressable unit of data accessible through the CA protocol. Each PV has a unique name which is served by a single CA server. A CA server maintains two sockets:

- An UDP socket bound to the CA port listens for PV name search request broadcast messages. PV name search replies are sent as unicast messages to the source of the broadcast.
- A CA TCP socket listens an arbitrary port number. The exact port number is included in the PV name search reply. The socket is used to build a Virtual Circuit between a CA client and the server (upon the establishment of a TCP connection among the two components).

S7EPICS supports the CA protocol and was strictly implemented according to the CA specification manual [4]. Moreover, S7EPICS was thoroughly tested with the most popular EPICS CA clients (see subsection 'Compatibility with Existing EPICS Tools' for additional details). To be able to use the S7EPICS library the programmer has to call S7EPICSMainCyclic block in a cyclic organization block, e.g., OB1. The implementation of the S7EPICS library in the TIA Portal is (succinctly) depicted in Fig. 3.



S7EPICS_Library	
S7EPICS_MainCyclic [FC5001]	Main cyclic entry point for S7EPICS
Global DBs	
S7EPICS_Clients_DB [DB11]	DB to store CA client connections
S7EPICS_PVs_DB [DB14]	Internal DB to store all S7EPICS PVs
S7EPICS_TCPSendData_DB [DB2]	TCP data that will be sent
S7EPICS_UDPReceiveConfiguration_DB [DB1]	UDP receive communication settings
S7EPICS_UDPReceiveData_DB [DB7]	UDP receive area
S7EPICS_UDPReceiveDataSave_DB [DB8]	UDP receive area for debugging
S7EPICS_UDPSendConfiguration_DB [DB6]	UDP send communication settings
S7EPICS_UDPSendData_DB [DB10]	UDP send area
Instance DBs	
S7EPICS_DeclaredPVs [DB12]	S7EPICS declared PVs
PV Declaration	
S7EPICS_DeclarePVs [FB9]	Function to declare PVs
S7EPICS Functions	
S7EPICS_BeginPV [FC1]	PV initialization block
S7EPICS_EndPV [FC2]	End of PV initialization block
S7EPICS_CheckConsistency [FB4]	Configuration consistency checker
S7EPICS_PVHandler [FB2]	Check PV property change function
S7EPICS_PVValueBuilder [FB8]	Assemble CA command payloads
S7EPICS_PVValueDecoder [FB10]	Write PV handler
S7EPICS TCP and UDP handlers	
S7EPICS_TCPDisconJobhandler [FB7]	TCP disconnect handler
S7EPICS_TCPSenderJobhandler [FB3]	TCP package sender
S7EPICS_TCPServer [FB6]	CA command reply handler
S7EPICS_UDPSenderJobhandler [FB5]	UDP package sender
S7EPICS_UDPServer [FB1]	CA UDP command reply handler

Figure 3: S7EPICS implementation in TIA Portal.

Declaring EPICS PVs

Declaring PVs in an EPICS IOC is done through the specification of records that are stored in one or more .db files – these are loaded in the IOC start-up script (i.e. st.cmd file). Due to being a TIA Portal library, S7EPICS declares PVs in a different way: instead of using the traditional record based declaration, S7EPICS implements a dedicated PLC block called S7EPICS_DeclarePVs for declaring PVs. This block is composed of several input parameters that are used to specify the following PV fields:

- VAL (value of the PV in the selected DBR type)
- STATUS (is 1 for a normal, connected PV)
- SEVERITY (alarm state)
- UPPER_DISCR (upper discrepancy)
- LOWER_DISCR (lower discrepancy)
- UPPER_ALARM (upper alarm limit)
- LOWER_ALARM (lower alarm limit)
- UPPER_WARNING (upper warning limit)
- LOWER_WARNING (lower warning limit)
- UPPER_CONTROL (upper control limit)
- LOWER_CONTROL (lower control limit)
- EGU (engineering unit)
- TIMESTAMP (calculated from the PLC CPU time)

The S7EPICS code structure makes it easy to access PVs across the PLC code since the index of the PV in the PLC array is hidden – in other words, the PLC programmer may refer to a PV simply by its name, not by its position in the array, in the PLC business logic. For example, with S7EPICS, to assign the value 12 to a PV called Cryo:SetPower of type integer looks as follows: "S7EPICS_PVs".PVs["S7EPICS_DeclaredPVs". "Cryo:SetPower".PVArrayNumber].ValueDBRIntOrShort := 12;

Scalability

S7EPICS relies on certain (system) constants that users may assign values to 1) configure the logic of the S7EPICS (running in the PLC) properly and 2) scale the system as whole according to (specific) needs (see Table 1). These constants dictate the behaviour of S7EPICS such as the UDP port to listen for PV name search request. The maximum allocated PV count and allowed client connections are also configurable through these constants. Modifying these constant values automatically allocates the necessary memory in the PLC code (see Table 2).

Table 1: S7EPICS Constants

Name	Default Value
EPICS_TCP_Send_Timeout	T#3s
EPICS_Server_Port	5064
EPICS_Server_Version	11
EPICS_UDP_Server_Port	5064
EPICS_Max_PV	100
EPICS_Max_Client	16

S7EPICS can handle a maximum of 200 simultaneous EPICS connections. This means that multiple CA clients can be connected to the same port of the PLC. Channel Access IDs like SID, CID, IOID and MonitorID are registered by the S7EPICS PLC code for every client. Defining an adequate value for the EPICS_Max_PV constant is crucial to keep the PLC code at an optimal size.

Table 2: S7EPICS Memory Requirements

EPICS_ Max_PV	Load	Work Data
1	3 886 bytes	616 bytes
10	7 854 bytes	4 576 bytes
50	25 461 bytes	22 176 bytes
100	47 456 bytes	44 176 bytes
500	223 461 bytes	220 176 bytes
1000	443 460 bytes	440 176 bytes

Compatibility with Existing EPICS Tools

S7EPICS was methodically tested against widely used EPICS tools and high-level applications such as caget, caput, camonitor, CS-Studio and Archiver Appliance. Both reading and writing of PV data between the S7EPICS (i.e. CA server) and tools/high-level applications (i.e. CA clients) have been validated, and are correctly performed among the two components.

EPICS bundled (console) tools caget and caput send a [FIN, ACK] TCP signal before closing down the CA TCP connection. Industrial PLC devices, such as the Siemens S71500, have a different reaction implemented than what caget/caput tools expects: when a PLC receives a [FIN, ACK] signal it closes the TCP connection immediately, which (unfortunately) makes these tools display an error message. This behaviour, however, does not affect the success of PV communication and data exchange.

CS-Studio registers DBR_TIME and DBR_CTRL monitors for PVs. These monitors are fully implemented and supported in S7EPICS.

Archiver Appliance reads PV fields like RTYP, SCAN and NAME$. These fields are not part of the basic CA fields; therefore, these need to be declared as new PVs, e.g., [pvname].RTYP. Additionally, the programmer has to make sure that new PVs are handled properly by the PLC.

Finally, to further validate the S7EPICS compatibility with EPICS high-level applications, an existing IOC was converted to the format of the S7EPICS_DeclarePVs PLC block, thanks to a tool that reads an IOC .db file and outputs an external source .scl file for TIA Portal (generating the same PV names and data types as declared in the .db file). After the code was transferred to the PLC, the IOC was stopped and the PLC using S7EPICS took its place. CS-Studio successfully connected to all the PVs and the same functions were accessible as with the IOC.

Performance

To evaluate the performance of the traditional PLC-EPICS integration versus the alternative proposed in this paper (i.e. S7EPICS), two tests were performed: in the first test (A), a CS-Studio OPI screen reads PVs from an EPICS IOC which, in turn, reads values from a PLC using s7plc (in other words, PLC ↔ EPICS IOC (s7plc-comms) ↔ OPI screen). In the second test (B), the same CS-Studio OPI screen reads PVs directly from a PLC using the S7EPICS library (in other words, PLC (with S7EPICS) ↔ OPI screen).

Both tests were performed using the same PLC (Siemens S71516-3 PN/DP), business logic, CS-Studio (version 4.6.1.26), OPI screen, and with 100 PVs combining all supported data types, namely:

- SHORT/INT (16 bit signed)
- LONG (32 bit signed)
- FLOAT (32 bit single precision)
- DOUBLE (64 bit double precision)
- CHAR (8 bit single character)
- STRING (string)

The (key) parameter being studied in both tests was the time required to load the OPI screen, measured between the first UDP PV name search request sent by CS-Studio and the last TCP response sent by IOC. To perform the measurements, Wireshark [6] was used.

A tool called PLC Factory [7] was used to build the IOC for test A. This tool, developed at ESS, retrieves the configuration model from the Controls Configuration Database (CCDB) [8] and generates (i.e. outputs) the following:

- ESS EPICS Environment IOC start-up (st.cmd file).
- EPICS record based declaration file (.db file).
- PLC external source code (.scl file) containing all the necessary blocks (modbus block, TCP socket blocks and array mappers) to communicate with the IOC.

In test A, the communication time between the PLC and the IOC was not considered due to the OPI screen loading time being the phenomena of interest to measure. The variable values were not important. Repeating test (B) with different PLC loads showed that the response speed of S7EPICS highly depends on the PLC free computation capacity available per cycle. The tests were performed under a PLC cycle of approximately 2-6 milliseconds.

Under these conditions, the average result of performing test A (EPICS IOC ↔ OPI screen) was 0.5352 seconds, while for test B (PLC with S7EPICS ↔ OPI screen) this was 0.5960 seconds.

IOT AUTOMATION WITH S7EPICS

The S7EPICS library is an effective solution to the interoperability issue of turning PLC tags into EPICS PVs. However, beyond interoperability, further issues arise when dealing with large-scale control systems – such as the one under development at ESS – that are organized in a traditional, hierarchical (control) structure. In this last section we identify some of the most acute issues and propose a framework from the IoT Automation domain as a promising candidate to eliminate (or at least) mitigate those. The aim of the discussion below is only to designate the direction of future work.

Considering the common specifications of large number of I/Os, regularly changing experimental setup, various communication protocols, etc., the following issues should be tackled:

- User access management: proper authentication and authorization of the operational and re-search personnel.
- Cyber security: machine-to-machine access management.
- Interoperability: translation between data formats and semantics.
- Scalability and flexibility: quick response times to planned and unforeseen changes.

This listing resembles the main challenges of state-of-the-art IoT Automation [9], which urges to make use of Internet technology in order to come up with solutions enabling the next generation of production automation systems. The Arrowhead framework [10] [11] is one of the largest projects in the field. It proposes a service-oriented control architecture (in contrast to the traditional, hierarchical approach) for industrial facilities with strict requirements on quality and security. Automation systems are organized into so called “local clouds”, which create secure boundaries protecting the local automation operations from any hazardous external activity (see Fig. 4). The local cloud must implement mandatory core services (service registry, authorization, orchestration) in order to become self-contained and interoperable towards other Arrowhead networks. Local clouds may interact with each other if local servicing becomes inefficient or impossible. This follows the system-of-systems design principle [12].

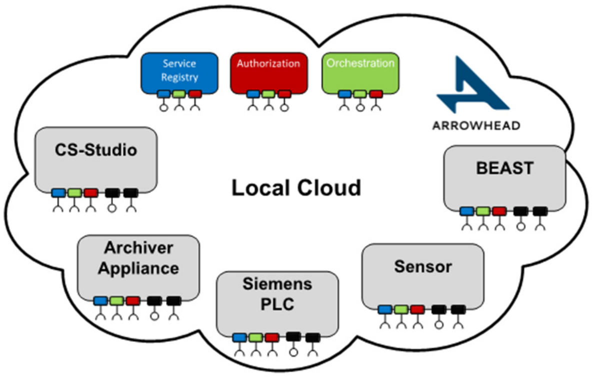


Figure 4: Local Cloud with Arrowhead.

The Arrowhead framework does not break down legacy protocols such as the EPICS CA, but move them into a state-of-the-art, loosely-coupled, service-oriented environment. Consequently, there is a significant potential in integrating EPICS-based automation operations at ESS with the Arrowhead framework, bringing benefits such as access management, security and flexibility.

FUTURE DEVELOPMENT

One of most important future development would be to implement the array support for S7EPICS, which includes handling long CA headers. The S7EPICS code already detects array requests and long CA headers, but a proper reply is not yet implemented.

Even if an EPICS IOC implements access security, currently, S7EPICS (and the traditional CA protocol) does not implement system authentication & authorization. One way to properly manage this matter could be through the adoption of the Arrowhead framework.

CONCLUSION

S7EPICS is an open source PLC code, implemented as a TIA Portal library, which enables EPICS high-level applications to connect directly to a Siemens S71500 family PLC without an actual EPICS IOC mediating the two components. Consequently, S7EPICS simplifies the control system as a whole and, eventually, helps reducing its costs and time to production.

The library may be instrumental for a (research) group that does not possess (much) EPICS competences and/or has substantial PLC development expertise, and when the business (control) logic is entirely implemented in the PLC.

To validate the S7EPICS library many tests were performed. These have successfully demonstrated that the library is compatible with generic CA clients (e.g. caput, caget, CS-Studio) and is capable of successfully handling (several) thousands of PVs without putting an unmanageable load on the PLC. Finally, this paper also proposes a software architecture based on the Arrowhead framework to find a solution for next generation challenges.

REFERENCES

[1] EPICS, "Experimental Physics and Industrial Control System," 2019. [Online]. Available: https://epics-controls.org/

[2] E. E. Modules, "s7plc_comms EPICS module," European Spallation Source. [Online].

[3] S. O. UA, "Siemens OPC UA," Siemens , [Online]. Available: https://new.siemens.com/global/en/products/automation/industrial-communication/opc-ua.html

[4] E. community, "Channel Access Protocol Specification," 2014. [Online]. Available: https://epics.anl.gov/base/R3-16/0-docs/CAproto/index.html

[5] M. Boros, "S7EPICS BitBucket repository," 06 10 2019. [Online]. Available: https://bmykeb@bitbucket.org/bmykeb/s7epics.git

[6] WireShark, "WireShark," [Online]. Available: https://www.wireshark.org/

[7] G. Ulm, F. Bellorini, D. P. Brodrick, R. N. Fernandes, N. Levchenko, and D. P. Piso, "PLC Factory: Automating Routine Tasks in Large-Scale PLC Software Development", in Proc. ICALEPCS'17, Barcelona, Spain, Oct. 2017, pp. 495-500. doi:10.18429/JACoW-ICALEPCS2017-TUPHA046

[8] R. Fernandes†, "Controls Configuration Database at ESS," 2017. [Online]. Available: http://accelconf.web.cern.ch/accelconf/icalepcs2017/papers/tupha156.pdf

[9] J. Delsing, "Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions," IEEE Industrial Electronics Magazine, pp. 8-21, 2017.

[10] J. Delsing, IoT Automation: Arrowhead Framework, CRC Press, 2017.

[11] Arrowhead consortium, "Arrowhead Framework," 2019. [Online]. Available: https://www.arrowhead.eu/arrowhead-framework [Accessed 15 August 2019].

[12] C. Keating, "System of Systems Engineering," pp. Engineering Management Review, IEEE. 15. 62 - 62. 10.1109/EMR.2008.4778760, 2003.