INTEGRATING SIEMENS PLCS AND EPICS OVER ETHERNET AT THE CANADIAN LIGHT SOURCE

R. Tanner, S. Hu, G. Wright, CLSI, Saskatoon, Canada
E. Matias, Mighty Oaks, Victoria, Canada

Abstract

The Canadian Light Source (CLS) is a 3rd-generation synchrotron light source on the University of Saskatchewan Campus in Saskatoon, SK, Canada. The control system is based on the Experimental Physics and Industrial Controls System (EPICS) toolkit [1]. A number of systems delivered to the CLS arrived with Siemens, PLC-based automation. EPICS integration was initially accomplished circa 2003 using application-specific hardware; communicating over Profibus. The EPICS driver and IOC application software were developed at the CLS. The hardware has since been discontinued. To minimize reliance on specialized components, the CLS moved to a more generic solution, using readily-available Siemens Ethernet modules, CLS-generated PLC code, and an IOC using the Swiss Light Source (SLS) Siemens/EPICS driver [2]. This paper will provide details on the implementation of that interface. It will cover detailed functionality of the PLC programming, custom tools used to streamline configuration, deployment and maintenance of the interface. It will also describe handshaking between the devices and lessons learned. It will conclude by identifying where further development and improvement may be realized.

SYSTEM OVERVIEW

The general system overview is shown in Fig. 1. Siemens PLCs from the S7-400 and S7-300 families have been fitted with Industrial Ethernet modules.

These PLCs have been delivered by vendors. There are two S7-300s for the storage ring SRF (Superconducting Radio Frequency) cavity, one in the BR (Booster Ring) RF controls, and S7-400s for the Linde Kryotechnik-supplied cryogenics plant and SR (Storage Ring) Thales RF amplifier.

At the time of this writing, EPICS integration of the TUV-certified, failsafe PLCs used in the CLS Access Control and Interlock System (ACIS) and O2 monitoring is still under review.

Because the PLCs perform many critical functions, a separate Virtual Local Area Network (VLAN) had been dedicated for them (referred to as a "plantbus" by the manufacturer), the EPICS IOC (Input-Output Controller) and Engineering Stations ("ES"). Access to the PLCs is meant to be accomplished only via the development stations or the IOC. The ESs have an industrial Ethernet adapter for connecting to the plantbus and commonly – available adapter for connecting to the "office" network. The IOC connects using the default VMWare Ethernet adapter and is configured to be on the VLAN, as well.

Any other services are configured to be available to the VLAN rather than accessible via a gateway or router (the IOC also provides NTP, for example). Traffic is restricted on the VLAN for performance and security considerations.

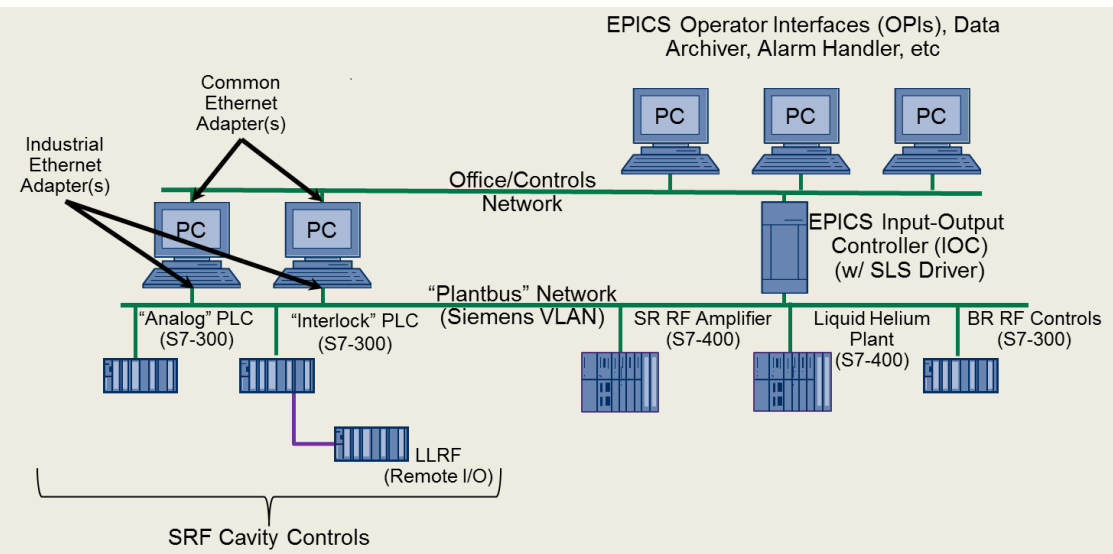

Figure 1: CLS Siemens PLC Network Topology

ISBN 978-3-95450-146-5

IOC

The IOC is running as an Intel-based, virtual machine (VM) on an ESXI VMWare. The O/S is Red Hat Linux 7.2 (Enigma).

The SLS Ethernet driver and IOC application were compiled against EPICS 3.14.6. All PLCs can be interfaced with a single IOC application, or a specific app written for each PLC or any combination thereof. The CLS will likely move to one IOCapp/PLC; which is more modular and fault-tolerant.

The IOC application was modified slightly to include a synchronization step with the PLC before it is allowed to set values in the PLC.

PLC CODE

The PLC code is written using Siemens PCS 7; which provides a number of development languages ranging from an Assembly-like option called Statement List (STL), a Pascal-type language called Structured Control Language (SCL) and graphical languages including a Ladder Logic offering (LAD), Function Block Diagram (FBD) as well as higher-level languages such as Sequence Function Chart (SFC) for state logic programming and Continuous Function Chart (CFC), the language used for safety system programming.

The PLC side of the EPICS interface was written using SCL.

Functions

All the code and variable definitions are contained in a single SCL file. Comments indicate where automatically-generated code is to be inserted.

The CLS PLC code consists of four functions; IE_COMMS, CHECKSYNC, PACK_TX_DB and UNPACK_RX_DB. The core functionality is handled by the IE_COMMS function.

IE_COMMS calls PACK_TX_DB which copies data from various internal memory areas to a buffer (IE_TX). The data consists of all the values meant to be sent to the IOC as well as the current values the IOC is interested in setting (readback values). IE_COMMS then calls the SEND function and waits for it to return. Upon return, it logs any errors and starts the process again. A watchdog timer will also restart the procedure if SEND times out.

IE_COMMS also monitors the RECEIVE function watching for the arrival of data. Before the IOC can manipulate data in the PLC it must synchronize with it by demonstrating it has the current values of the variables in question. When data first arrives, if the SYNC flag is not set, IE_COMMS calls CHECKSYNC. If CHECKSYNC succeeds, the SYNC flag is set and UNPACK_RX_DB is called which moves data from IE_RX to the internal PLC locations using a combination of pointer-based and explicit transfers as outlined below.

As with the SEND function, errors are logged before preparing for the next packet of data.

Data Transfers

Transfers are accomplished either by explicit assignments or pointer-based operations. Explicit assignments are simple Var1 = Var2 statements (i.e. "**IE_TX.DBD8 := DB4.DBD12;**" copies the double word at offset offset 12 of datablock 4 to IE_TX at offset 8).

In cases where a number of variables are adjacent in a contiguous section of memory, it is convenient and simpler to call the BLKMOV function (analogous to memcopy in C) which takes "ANY pointers" as source and destination parameters. ANY pointers are 10 bytes long and contain fields for the length of the data to copy, the memory area (i.e. input, output, global memory, datablock, etc), the datablock number (if applicable) and the *bit offset* in that memory area to start the copy. For example, if 16 boolean values, 4 integers and 6 floating point values were located in a single datablock starting at byte 4, they could be copied in a single statement using the pointer, "**P##DB2.DBX4.0 BYTE 34**".

In order to access the fields of the ANY pointer, a User Data Type (UDT) is defined (called AnyPoint) that specifies the pointer layout. The UDT is then associated with ANY pointers in a manner similar to a union{} in C as follows:

```
BUFFER_TX: AnyPoint; // UDT
ANY_SRC at BUFFER_TX: Any; //POINTER
```

For both PACK_TX_DB and UNPACK_RX_DB, a list of pointers is maintained in datablocks (DBs) named DB_TX_PTRS and DB_RX_PTRS, respectively. The pack and unpack functions iterate through the pointers performing the pack or unpack operations as required.

The IE_TX and IE_RX buffer are also datablocks. Datablocks (DB) are user-defined records analogous to a C struct{}.

SYMBOL MAP

The PLCs delivered as part of larger systems do not generally have internal symbols which adhere to the CLS naming convention. Therefore, a symbol map of each PLC is manually generated mapping the internal symbol name to the CLS-defined PV name. It also includes the internal (PLC) address of the variable, its offset in the TX or RX buffer, the description, type of transfer (Explicit or Pointer) as well as a number of EPICS-specific definitions required for the substitution files.

Scripts

Adding or inserting another process variable (PV) to the map requires a lot of manual editing for both the IOC and the PLC, which is painstaking and leads to errors. The symbol map was built in MS Excel which provides an interface for automation. VBA scripts were written to generate EPICS substitution files for the IOC as well as to

ISBN 978-3-95450-146-5

generate Siemens SCL code to be copied in to the environment for compilation.

The scripts are in a relatively primitive stage (i.e. not very flexible and with minimal exception-handling) but even so, have proven to save time and significantly reduce clerical errors.

LESSONS LEARNED

IE_TX and IE_RX have four-byte headers followed by the data. One has to be cautious of relative vs absolute offsets. Byte 0 of the PLC IE_TX data is byte 4 of the TX DB (and of the IOC input buffer).

The CLS implementation allows the PLCs to transmit as fast as they can (rather than on a set period). The S7-400s turned out to be too fast; flooding the data archiver and causing jarring updates on the display(s). An input parameter was added to allow for an additional delay interval.

Some systems come complete with an HMI. IE_COMMS has a Boolean input (LOCAL/REMOTE). If the parameter is set to LOCAL, IE_COMMS will not unpack IE_RX. It will however, keep the IOC updated via the readback values in IE_TX.

PLC timers are explicitly defined. One must be careful that timers used for the EPICS functions, are not assigned elsewhere.

The use of pointers is notoriously wrought with pitfalls and ANY pointers are no different. One must be diligent with respect to data types (integer vs word) because the PLC compilers are less flexible, forcing the developer to be precise regarding definitions.

The PLC user code/compilers allow direct access to digital values (i.e. the 3rd channel of a digital input card starting at address 4 would be “I 4.2”). To accommodate such addressing, the ANY pointer offset is a *bit offset*. To refer to byte 4, the ANY offset would be set to 32 (or rather 20, because it is a hex value).

As with data types, the format of some function parameters must be explicitly defined and are dependent on the language in which the function is being used. For example, a 3-second timeout in CFC is represented as “3s” whereas in a LADDER diagram it would be specified as “S5T#3S”.

The vendor-supplied PLC RECEIVE function expects periodic refreshes from its connection partner. This acts as a heartbeat to confirm that the other device is still connected and functioning. If it does not get them, it assumes an error. Because the IOC driver only sends data when there are updates, the PLC throws multiple errors/second. As a workaround, logging of that error has been commented out of the code.

FUTURE DEVELOPMENT

Currently, the SYNC flag is only reset when the watchdog timer expires or when the PLC is restarted. A slightly more robust realization could result if the EPICS driver sent periodic updates as expected by the RECEIVE function (rather than only when it has new data for the PLC). A loss of communication could reset the SYNC flag and logging of these events reintroduced to provide potentially useful insight in to how often that occurs.

Significant convenience would be realized by having the IOC populate the TX and RX pointer tables. This would completely remove all EPICS configuration from the PLC, limiting modification to the addition of the code and DBs required for communicating with the IOC. It would remove a lot of the manual work that is still required on the PLC end of configuration.

Having systems delivered with symbols already named in accordance with a naming convention would permit that table to be exported to the Symbol Map. Building the Symbol Map is the largest effort in the current work flow so improvements here have the biggest impact.

CONCLUSION

The use of SCL greatly simplifies integration with existing projects irrespective of the tools used to develop them. It similarly simplifies uploading an existing PLC binary, compiling the EPICS-specific blocks in to the code and downloading to the PLC, which is most often the case for vendor-supplied automation.

To date, four vendor-supplied PLCs have been set up to use the CLS-modified SLS driver for interfacing with EPICS with a fifth being targeted for our 2014 fall shutdown. A very small amount of data has been written to the PLCs from EPICs, with considerably more planned in the future. A larger amount has been read out from them. The SLS driver, IOC application and CLS-developed PLC code has not required any modification or debugging since installation. The combination has proven stable over more than a year of deployment.

The investment in development time has simplified future installations, eliminated the reliance on exotic hardware, reduced implementation time and clerical errors as well as enhancing performance.

ACKNOWLEDGMENT

The CLS gratefully acknowledges the contributions of its operating funding partners:

- University of Saskatchewan (UofS)
- Canadian Foundation for Innovation (CFI)
- Canadian Institutes of Health Research (CIHR)
- National Research Council (NRC)
- Natural Sciences and Engineering Research Council (NSERC)
- The Government of Saskatchewan
- Western Economic Diversification (WD)

REFERENCES

[1] http://www.aps.anl.gov/epics
[2] http://epics.web.psi.ch/style/software/s7plc/s7plc.html

ISBN 978-3-95450-146-5