Practical Applications of IEC 61131 in Modern Electrical Substations

Mark S. Weber
Schweitzer Engineering Laboratories, Inc.

Presented at the
14th Annual Western Power Delivery Automation Conference
Spokane, Washington
March 27–29, 2012

Practical Applications of IEC 61131 in Modern Electrical Substations

Mark S. Weber, *Schweitzer Engineering Laboratories, Inc.*

***Abstract*—The open international standard IEC 61131 describes a powerful programming language that is quickly gaining acceptance for modern electrical substations. Proprietary logic solutions often have unique advantages. However, the new open programming language offers benefits for broader application and potential cross-platform use. The advantages grow as the IEC 61131 programming language merges with open industry standard protocols, such as IEC 61850 and DNP3. The flexible programming interface enables logic solutions ranging in complexity from simple I/O processing to more advanced, deterministic high-speed automation and control.**

This paper provides examples to illustrate basic programming techniques. IEC 61131 logic processing examples are shown using data types from IEC 61850 manufacturing message specification (MMS), Generic Object-Oriented Substation Event (GOOSE), and DNP3 protocols. Additionally, innovative IEC 61131 logic solutions are shown that enhance secure access control of North American Electric Reliability Corporation Critical Infrastructure Protection (NERC CIP) critical cyberassets in any electrical substation.

In this paper, IEC 61131 programming language performance considerations and comparisons are provided and techniques are shown to ensure deterministic control through task prioritization and scheduling.

I. Introduction

The international standard IEC 61131, originally published in 1993 and updated in 2003, establishes a widely accepted guideline for uniform programming of programmable logic controllers (PLCs) and embedded automation computers. IEC 61131 programming is commonly applied to PLCs for highly complex industrial automation systems.

In addition to industrial automation, the methods described in the IEC 61131 standard are rapidly gaining acceptance as a powerful programming language for modern electrical utilities, water/wastewater, metals/mining, petrochemical, and other mission-critical facilities and applications. Ambitious industry effort toward system automation and wide-area monitoring and control underscores the need for a common programming language for modern protection, control, and monitoring (PCM) applications. Compared to proprietary logic solutions operating on specific products, the IEC 61131 methods offer benefits toward broader application and potential cross-platform use.

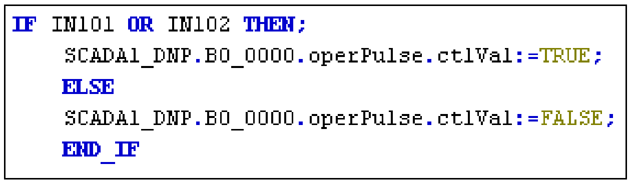
The advantages grow if we effectively apply open communications interfaces and standardized utility protocols, such as those described in the IEC 61850 standard and DNP3, to simplify data gathering for wide-area logic processing. IEC 61131 programming enables a broad range of flexibility easily scaled for simple I/O processing and easily extended to manage the most advanced requirements for deterministic, high-speed automation and control.

II. IEC 61131 Fundamentals and Definition of Terms

Part 3 of IEC 61131 (IEC 61131-3) defines both text-based and graphical programming language standards [1], as follows:

- Structured text (ST), text-based (example shown in Fig. 1)
- Instruction list (IL), text-based
- Function block (FBD), graphical (example shown in Fig. 2)
- Ladder diagram (LD), graphical (example shown in Fig. 3)

Additionally, a sequential function chart (SFC) may be used to graphically describe the parallel and sequential execution of a combination of IEC 61131-3 programs, functions, or function blocks.



```
IF IN101 OR IN102 THEN;
    SCADA1_DNP.BO_0000.operPulse.ctlVal:=TRUE;
    ELSE
    SCADA1_DNP.BO_0000.operPulse.ctlVal:=FALSE;
    END_IF
```

Fig. 1. Example ST programming.

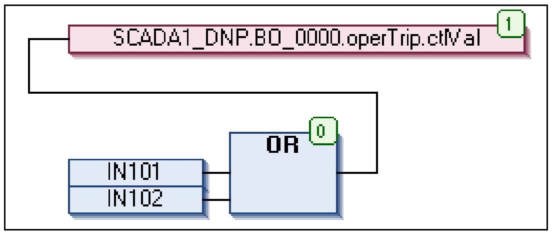


Fig. 2. Example continuous function chart (CFC) programming.

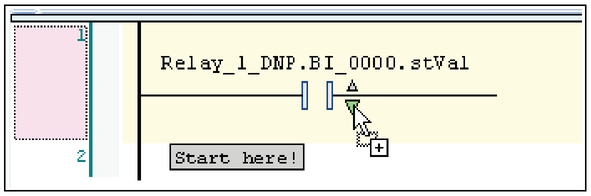


Fig. 3. Example LD programming.

This paper references the term *program organizational unit* (POU), which is the IEC 61131 standard method of describing a program, function, or function block.

A calling logic block has the ability to call, or activate, a function. A function is a routine that a program or calling logic block can call to perform repetitive tasks. Square root (SQRT) is a good example of a standard function. A function is called by a program, and the result is immediately processed based on the pre-call variables provided by the program; the return value is not retained by the function. Other example numerical functions include sine (SIN), cosine (COS), tangent (TAN), or logarithm (LOG). Example string functions may include concat (CONCAT), find (FIND), insert (INSERT), left (LEFT), right (RIGHT), length (LEN), and midstring (MID).

IEC 61131 programmers may build their own customized functions to simplify otherwise complex equations. For example, consider the common requirement to calculate power factor from substation intelligent electronic device (IED) metering quantities. Typically, an equation similar to (1) is used.

$$\text{L1\_PF} := (\text{COS}(\text{ATAN}\left(\frac{\text{L1.MVar.instMag}}{\text{L1.MWatt.instMag}}\right) \tag{1}$$

The power factor equation in (1) requires the use of complex math functions, and the programmer must account for potential divide-by-zero errors that may result from the IEC 61131 compiler under certain conditions.

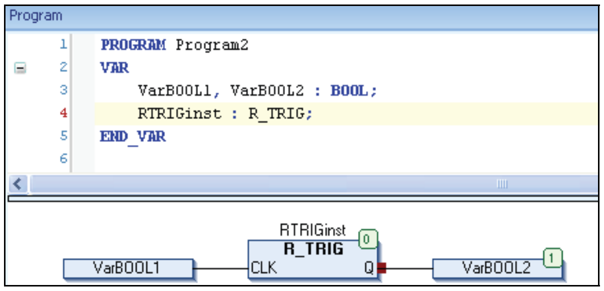
By developing a customized function within the IEC 61131 logic engine, the equation may be reduced to a simple function (which can later be called by a program) that only requires the entry of the IED megaVAR and megawatt quantities, similar to (2).

$$\text{L1\_PF} := \text{PF}\left(\text{L1.MVar}, \text{L1.Mwatt}\right) \tag{2}$$

Note the complexity of the power factor calculation. Handling of the divide by zero is now both accomplished and hidden by the embedded function.

In addition to complex math calculations, functions are useful for other purposes, such as real-time comparison of analog quantities to confirm the health and validity of measurements from multiple data sources. Voting functions are easily implemented to accept input quantities only if the values are confirmed to match multiple data sources within an acceptable range. When measurement mismatches are detected, alarm signals are easily generated to send an immediate notification to maintenance personnel to indicate that equipment is out of service or calibration is required.

A function block is a routine for which a program or calling logic block has definitions for multiple instances to perform specific tasks. Each instance is independent and must have a unique name in a project. Fig. 4 shows example code to instantiate a CFC function block. A function block retains output variable status between processing intervals. Function blocks typically do not directly access and act on global system variables. An example of a function block is a timer block (TON). A TON is a calculation value that must be retained between each processing cycle to ensure POUs operating in parallel properly read the present status of the timer variable. Each function block includes a logic input pin to determine the reset condition.



```
PROGRAM Program2
VAR
    VarBOOL1, VarBOOL2 : BOOL;
    RTRIGinst : R_TRIG;
END_VAR
```

Fig. 4. Declaring an instance of a function block.

A program has the most capability or the highest order of functionality in an IEC 61131 system given its ability to read and write I/O and system variables and call functions or function blocks. Programs can be standalone units. They contain logic to perform a single task, perform multiple tasks, or invoke functions and function blocks.

III. Logic Processing in Modern Utility Substations

Many conventional, hardware-centric, PLC-based automation systems depend on architectures with highly concentrated hard-wired I/O. Logic functions may simply read and compare the present value of these points at any instant in time to make logical comparisons. Regardless of the speed of reading the I/O, the data within the logic functions are considered coherent because they are detected as part of the same read function and are considered to be detected at the same instant in time. Because the points are monitored in real time, data latency when reading point status can often be ignored. While this design approach may appear to simplify logic programming, due to the practical limits of I/O wiring distances, this approach complicates the administration of safe, cohesive wide-area controls for geographically distributed systems. The cost and complexity of these copper terminations force designs to rely only on the data locally terminated and then process data to be sent to other locations via control messaging. This additional step reduces the amount, type, and speed of data to be distributed among PLCs. Information simultaneously presented to the programming interface is actually detected, calculated, and received via digital communications at unrelated times. During this continual asynchronous data acquisition, the PLC is constantly attempting to represent the present state of the system. However, this representation is never the actual state, but rather it is the present state of the data in the most recent responses to I/O reads, calculations, and message reception. Also, this requires that systems depend on robust, secure communications to transfer control signals between widely dispersed locations.

When planning to automate a utility substation as a mission-critical system example, we quickly recognize that many modern substations lack the hard-wired I/O points needed to support a conventional PLC system. We are left with a choice—add extra wiring and PLC equipment needed to gather the necessary I/O readings or determine the viability of utilizing presently installed equipment via digital messaging.

Many modern substations utilize microprocessor-based PCM IEDs that are fully capable of reading and exchanging the status of breakers, control switches, analog readings, and calculated logic values. For the most advanced automation and control systems, it is to our advantage to share the measurements, calculations, and decisions from these devices with other devices in the system and to do so without additional costly wiring. A communications solution must account for the inconsistency of data formatting that often results from nonstandardized protocols, normalization of data readings, latency of data reporting, quality and accuracy of measurements, and confidence in time stamps.

Once the data are gathered from substation IEDs and output values are calculated, we must ensure fast, deterministic, and secure control signaling. Most importantly, we must implement logic programming methods that meet or exceed the performance characteristics of traditional hard-wire-based automation systems. IEC 61131 programming provides the tools, performance, and flexibility for this purpose.

IV. Normalize IED Readings Through Data Structures

Substation IED data are often collected, aggregated, and reformatted through an information processor, PLC, or remote terminal unit (RTU) using a mix of protocols, such as those described in IEC 61850, DNP3, Modbus®, or the native protocols of IED manufacturers. While each protocol may have its own distinct advantage, when attempting to use logic to compare data tags with incompatible attributes, the formatting inconsistency creates challenges. Therefore, not only is the standardization of data formats important, it is also essential that a nonmanufacturer-specific and internationally recognized method be chosen for project functionality and longevity. For this reason, IEC 61850 data type definitions offer many advantages with complete definitions that include all of the necessary attributes needed for substation automation. IEC 61131 programming allows the creation of data structures that match the IEC 61850 data model. The following data types are easily embedded in an information processor that utilizes IEC 61131 programming:

- Single-point status (SPS)
- Double-point status (DPS)
- Protection activation information (ACT)
- Activation information directional protection (ACD)
- Binary counter reading (BCR)
- Measured value (MV)
- Complex measured value (CMV)
- Integer status (INS)
- Controllable analog set point (APC)
- Controllable integer (INC)
- Binary controlled step position (BSC)
- Integer controlled step position (ISC)
- Modbus coil control (MDBC)
- DNP3 controllable single point (DNPC)
- String (STR)
- Time (TIM)

Once the data types are embedded and defined in the information processor database, each protocol driver stores individual data tags according to standardized rules. This process of normalizing protocol data to be of the same format, regardless of the data acquisition protocol used, through standardized database structures simplifies subsequent logic processing, data movement, and reporting that takes place within the logic engine. When certain required attributes of the data structure are not supported by a data acquisition method, the information processor calculates and provides the missing details.

For example, protocols such as those described within the IEC 61850 standard and DNP3 include both status values and time-stamp information for data tags. Therefore, when properly configured, the entire data structure populated from these two protocols may be considered coherent relative to the IED-reporting capabilities and safely compared in the IEC 61131 logic engine. Fig. 5 shows a suggested method of mapping DNP3 data to an IEC 61850 MV data type.

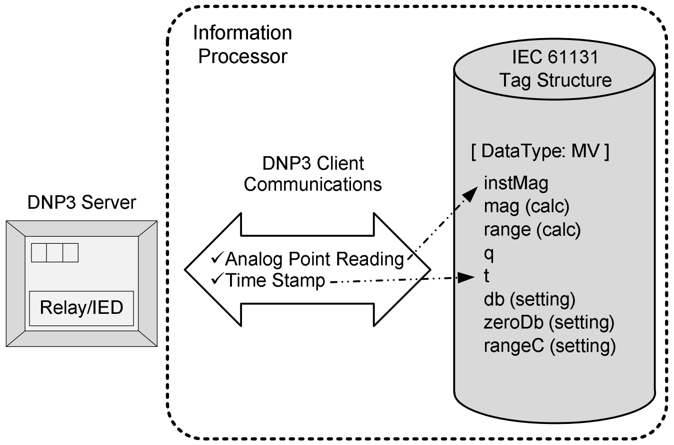


Fig. 5. Mapping a DNP3 point value to an IEC 61131 data structure.

However, Modbus is a protocol that inherently lacks the time-of-day information needed for the time-stamp attribute of the IEC 61850 data structure. When reading Modbus values, the information processor recognizes the missing time-of-day attribute and fills in the time stamp from the high-accuracy onboard system clock and appropriately marks the time quality. This process allows Modbus, IEC 61850, and DNP3 to utilize equivalent data types to support simple data movement between protocols.

If needed, IEC 61131 programming provides easy rejection of values due to unacceptable data latency, time quality, or data quality. Programmers applying the IEC 61131 standard should properly account for time-stamp latency in their logic functions.

By creating complete and standardized data structures within the information processor, each protocol fills in the available field as indicated from the IED. Table I shows a suggested method of populating data structures given the available attributes of popular utility protocols. The data structure includes a protocol superset of all available fields, primarily based on the IEC 61850 standard. For example, an IEC 61850 measured value (MV) data type indicates an IEEE signed 32-bit float value, thus nearly every 32-bit reading from an IED, regardless of protocol, is stored in the database as an MV data structure. Other formats can be converted and stored as IEEE signed 32-bit float values. Other standard data types are easily supported through IEC 61131 programming.

TABLE I
DATA ATTRIBUTES ASSIGNED BY PROTOCOLS

Data Structure Attribute	IEC 61850	DNP3	Modbus	Fast Message Protocol
stVal (Status value)	Protocol	Protocol	Protocol	Protocol
quality_t (Data quality)	Protocol	Protocol	Protocol	Protocol
timeStamp_t	Protocol	Protocol	Information processor	Protocol
timeQuality_t	Protocol	Information processor	Information processor	Information processor

Sharing data tags between protocols is greatly simplified because the IEC 61131 programmer may then use basic copy statements to exchange data values between functions, programs, or devices. Additionally, large data blocks are easily exchanged between devices through standard communications protocols, such as IEC 61850 MMS, IEC 61850 GOOSE, or DNP3, and the tables may include the time stamp and data quality within each data tag.

Binary status and control tags are also stored using IEC 61850 data structures and standardized for all protocols. Similar to analog tags, these data structures are easily shared among the various protocols, including high-speed protocols such as IEC 61850 GOOSE and MIRRORED BITS® communications. Additionally, the IEC 61131 logic engine supports multiple processing threads, priorities, and task intervals to support execution of high-speed logic expressions operating independent from lower-priority processing threads and background functions. This separation of tasks guarantees deterministic control for high-speed protocols in support of teleprotection, interlocking, and high speed automation. These automation applications include substation automation, automatic system reconfiguration, fault restoration, fast bus trip, load shedding, distribution automation, volt/VAR control, dynamic feeder optimization, and automatic generator control. High-speed processing rates ranging from 1 to 4 milliseconds are typical for these critical automation tasks.

V. SIMPLIFY PROGRAMMING BY COMBINING IEC 61131 AND IEC 61850 STRUCTURES

By designing the IEC 61131 programming interface to accept standardized IEC 61850 data structures, the system is designed to immediately process incoming data without further analysis or tag manipulation. Because each structure carries the stVal, time stamp, and quality, the user easily passes the data tag through the system to another protocol or optionally uses the internal logger to record the present value and time stamp. The example shown in Fig. 6 illustrates how a CMV data type served from an IEC 61850 device is easily passed to the database with minimal analysis or manipulation by the information processor.

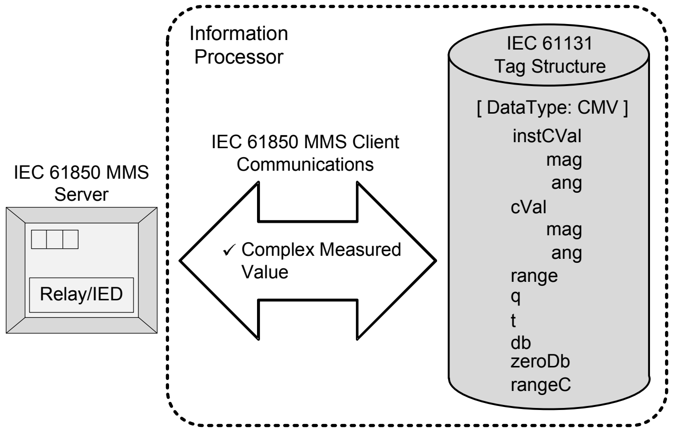


Fig. 6. IEC 61850 data structures transfer directly to IEC 61131 data tags.

Once the tags are normalized within the database, they are easily transferred to another protocol, as shown in Fig. 7.

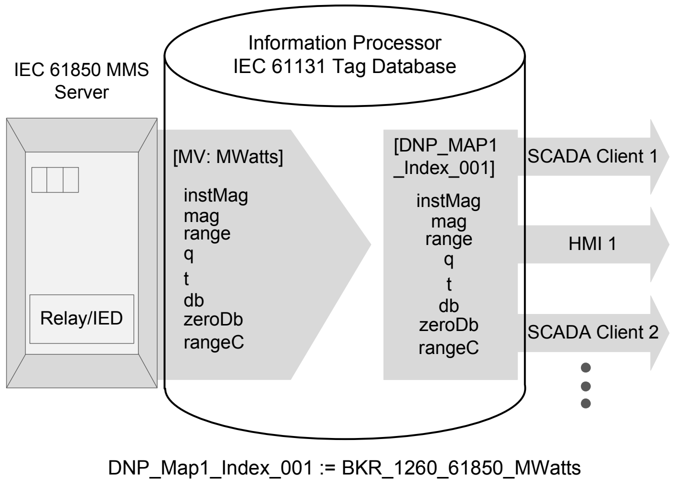


Fig. 7. Use simple IEC 61131 copy statements to easily transfer tag status to multiple clients.

IEDs programmed using the IEC 61850 standard are configured via Substation Configuration Language (SCL), Configured IED Description (CID), or IED Capability Description (ICD) file management. These files have a uniform, standardized template defined by the IEC 61850 standard, and many IEDs actually accept and use the file directly to configure in-service behavior. Using programming methods in this paper, the information processor data, regardless of the IED of origin, are mapped to IEC 61850 data sets, which are defined and stored in a template file and "pushed" to the remote IED. Also, once the CID and/or ICD files are specified, the tag lists are easily imported and interpreted by the IEC 61131-3 programming interface, allowing offline and automated template-based data mapping for supervisory control and data acquisition (SCADA), distributed control system (DCS), human-machine interface (HMI), and event notification. As shown in Fig. 8, a complete library of data types is important to allow simple transfer of all IEC 61850 device tags to the information processor database.

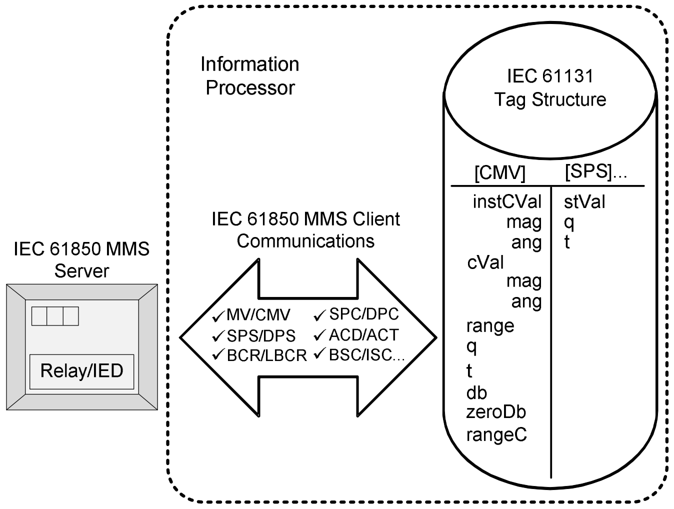


Fig. 8. Build IEC 61131 data structures to match any IEC 61850 data type.

VI. Use IEC 61131 Programming for Deterministic, High-Speed Control

When installed on a real-time embedded operating system, an IEC 61131 logic engine uses cyclic and scheduled high-speed processing to ensure deterministic control.

In most instances, multiple processing threads are offered to support user-configurable cycle times and thread priorities. By properly configuring thread prioritization, automation logic operates without compromise, while other functions operate at a lower priority, giving the user flexibility to specify primary and background processing threads. This separation of tasks guarantees deterministic control for high-speed protocols that are needed for substation automation, including automatic system reconfiguration, fault restoration, fast bus trip, load shedding, and automatic generator control. High-speed processing rates ranging from 1 to 4 milliseconds are typical for these critical automation tasks. The embedded real-time operating system creates a deterministic logic processing environment with little jitter and dramatically less susceptibility to malware.

An event-driven operating system, such as the Microsoft® Windows® operating system, allows preemption (or interruption) of tasks by other processes determined by the operating system to be of higher priority. An IEC 61131 logic engine operating on an event-driven operating system is susceptible to increased processing jitter as more processes are added to the operating system. Unfortunately, programming of another task, perhaps by another individual, may affect the performance of high-priority, mission-critical tasks without either designer knowing the problem exists.

VII. Port Access Control Using IEC 61131 Programming Logic

Advanced information processors offer built-in security features with programmable functionality enabled through IEC 61131 logic. To name a few options, detailed port statistics and control elements may be monitored, including online and offline indications, individual port enable and disable, data traffic monitoring and parsing, and pass-through connection enable.

In addition to centralized Lightweight Directory Access Protocol (LDAP) user account authentication, the information processor enables any combination of logic or control points to allow and/or block transparent communication to critical cyberassets, such as line protection relays. Fig. 9 shows a common substation communications architecture with a need for supervisory control of all transparent communications. This type of access control ensures only authenticated and proven active employees gain access to the substation IEDs based on end-user-defined employee and activity rules, roles, and responsibilities.

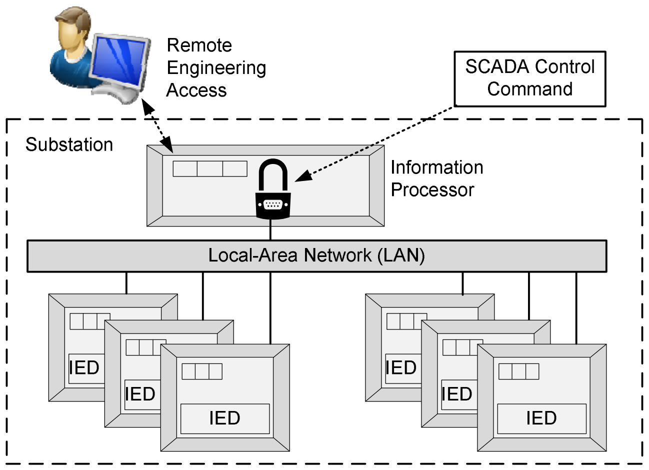


Fig. 9. Keep transparent port access restricted until control logic grants authority (i.e., SCADA).

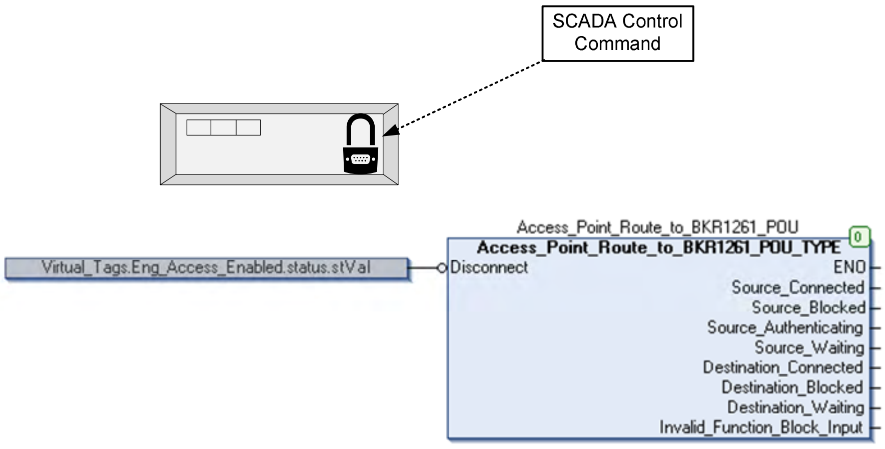


Fig. 10. Use a simple IEC 61131 logic expression to enable transparent communications to only one IED.

These security-based control tags may be controlled and monitored using numerous methodologies to add another layer of protection to engineering access connections. Fig. 10 shows an example where a SCADA protocol sends control commands to an IEC 61131 CFC block, allowing strict supervision of all transparent communications to a selected destination device. SCADA may keep the disconnect pin asserted at all times until SCADA properly authenticates a requesting user for access. This pin is used in logic as a permissive to deny access to the communications channel until the disconnect pin is deasserted.

Numerous IEC 61131 pickup and dropout timers and latches are available that are often used to customize and group communications diagnostics information. This is another area where shared data structures simplify the programming necessary to share this information with other connections, such as SCADA.

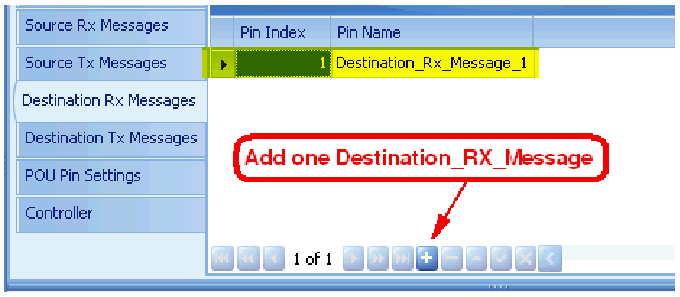
VIII. User Access Control Through IEC 61131 Programming Logic

Advanced IEC 61131 programs exploit extended IEC 61131 programming functions found in the information processor to monitor traffic on communications channels. A commonly used case involves monitoring traffic on a port connection for the purpose of detecting undesired access or entry into an unacceptable access level.

When monitoring and detecting the entry into an undesired access level to an IED, static feedback from the IED tells the level of user access. For example, a command prompt may be displayed differently for a Level 2 command line prompt compared with a Level 1 prompt. In most cases, this prompt is a static response from the IED and may not be altered by the user. As a result, the fixed returned character strings are used to monitor the user access level within an engineering access connection.

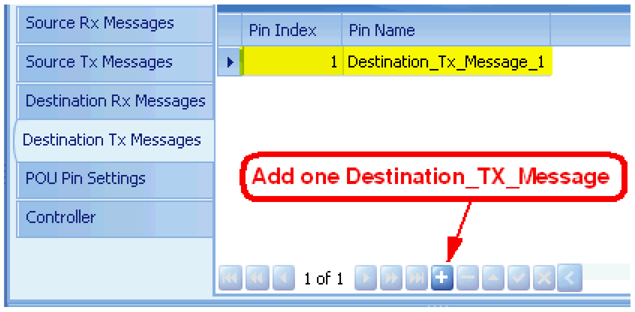
Further, IEC 61131 programming can also trigger and instruct the real-time traffic monitor to detect the unique IED command prompts and take appropriate logic actions if undesired entry is noted.

As shown in Fig. 11, we first define a trigger Destination_RX_Message_1 in the access point traffic monitor. In the access point router (APR), the destination device is the outgoing connection, thus the protective relay. Next, in Fig. 12, we define a Destination_TX message, which is a transmitted message that will be sent once the undesired prompt is detected.



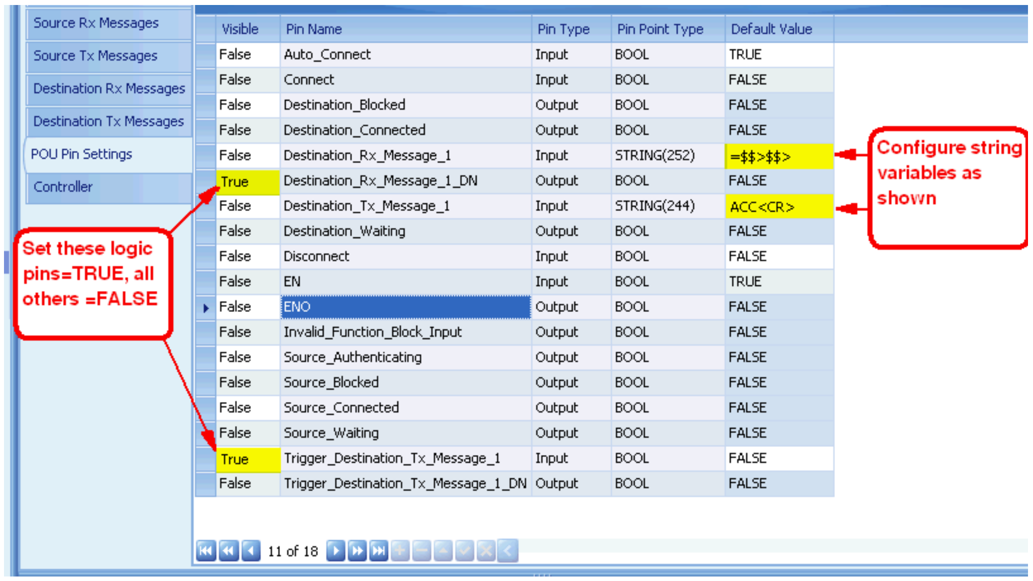
Pin Index	Pin Name
1	Destination_Rx_Message_1

Fig. 11. Add one Destination_RX message for monitoring the IED command prompt.



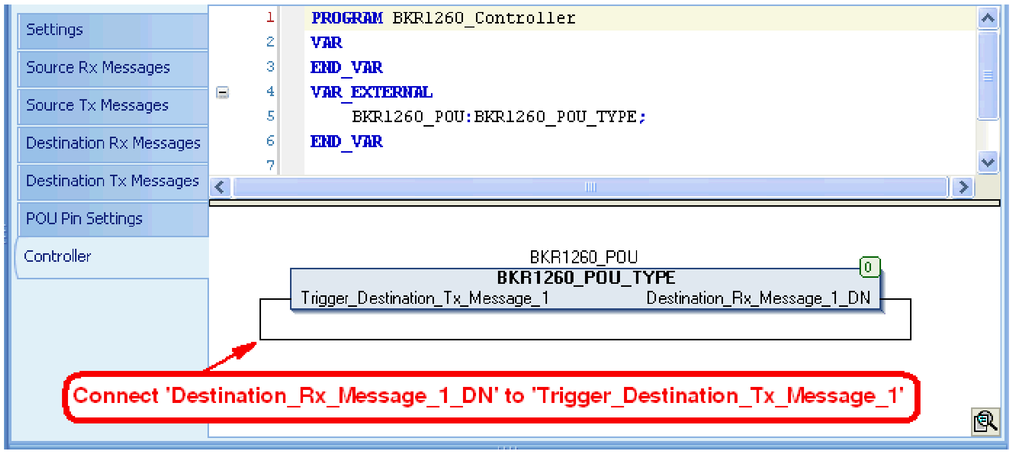
Pin Index	Pin Name
1	Destination_Tx_Message_1

Fig. 12. Add one Destination_TX_Message used for transmitting the control message upon detection of undesired access.



Visible	Pin Name	Pin Type	Pin Point Type	Default Value
False	Auto_Connect	Input	BOOL	TRUE
False	Connect	Input	BOOL	FALSE
False	Destination_Blocked	Output	BOOL	FALSE
False	Destination_Connected	Output	BOOL	FALSE
False	Destination_Rx_Message_1	Input	STRING(252)	=$$>$$>
True	Destination_Rx_Message_1_DN	Output	BOOL	FALSE
False	Destination_Tx_Message_1	Input	STRING(244)	ACC<CR>
False	Destination_Waiting	Output	BOOL	FALSE
False	Disconnect	Input	BOOL	FALSE
False	EN	Input	BOOL	TRUE
False	ENO	Output	BOOL	FALSE
False	Invalid_Function_Block_Input	Output	BOOL	FALSE
False	Source_Authenticating	Output	BOOL	FALSE
False	Source_Blocked	Output	BOOL	FALSE
False	Source_Connected	Output	BOOL	FALSE
False	Source_Waiting	Output	BOOL	FALSE
True	Trigger_Destination_Tx_Message_1	Input	BOOL	FALSE
False	Trigger_Destination_Tx_Message_1_DN	Output	BOOL	FALSE

Fig. 13. Add command prompt string and transmit string.



```
PROGRAM BKR1260_Controller
VAR
END_VAR
VAR_EXTERNAL
    BKR1260_POU:BKR1260_POU_TYPE;
END_VAR
```

Fig. 14. Program IEC 61131 logic to transmit messages upon detection of Destination RX_Message_1.

As shown in Fig. 13, the STR variables are then programmed based on the expected unique string combinations and desired transmit message.

Fig. 14 shows the CFC logic that triggers the Destination TX_Message_1 whenever the Destination RX_Message_1 is received during any communication with the protective relay.

After downloading the IEC 61131 code to the information processor, the logic engine continually monitors the connection for a received character set matching =>>. Once detected, the logic engine immediately transmits **ACC<CR>** to the IED, which, in this case, forces the relay immediately to Access Level 1. (Note that in this case, due to momentary Level 2 access, the protective relay alarm contact momentarily pulses, indicating unwanted intrusion.) Because the intruder has no option to modify the command prompt in IEDs with static feedback prompts, this IEC 61131 logic permanently blocks Level 2 access from anyone entering through this communications path.

IX. Conclusion

Standardized, nonproprietary IEC 61131 programming simplifies project configurations for substation automation needs. Based on common application requirements, engineering teams may efficiently produce standardized IEC 61131 automation libraries and streamline programming procedures for potential repeat projects. The benefits from reduced engineering costs, reduced training cost, and overall increased productivity can be substantial.

With proper definition and utilization of data structures within the embedded information processor, it is possible to configure an integrated IEC 61131 logic engine to process data tags and control commands deterministically at the programmed task interval of the IEC 61131 logic engine. Advanced embedded information processors often allow cycle times at or below 4 milliseconds. This level of performance is not achievable using an external, nonintegrated logic engine because multiple processing intervals are necessary to read and write logic variables and subsequently output control commands.

High-speed, peer-to-peer protocols are critical for maximizing the level of determinism possible through the use of an IEC 61131 logic engine. By using IEC 61131 programming to trigger control commands via protocols such as IEC 61850 GOOSE or MIRRORED BITS communications, the outgoing control signal transmits within one processing interval and is typically received by the remote device in less than 10 milliseconds (4-millisecond processing interval at each end).

Multithread IEC 61131 logic programming offers additional potential and promotes further application extensions. Multithread support typically includes adjustable processing rates along with configurable thread priorities, allowing optimal utilization of the information processor. Multithread prioritization and IEC 61131 programming flexibility offers significant performance benefits for modern substation automation applications.

As demonstrated through examples in this paper, IEC 61131 programming also offers flexible programming options to enhance secure access control policies. Users may combine powerful IEC 61131 logic controls with advanced and integrated security functions to develop customized solutions to fit their needs.

X. REFERENCE

[1] K.-H. John and M. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids*, 2nd ed., Springer, 2010.

XI. BIOGRAPHY

Mark S. Weber received his AAS in Electronics Engineering Technology in 1985. In 1986, he joined Schweitzer Engineering Laboratories, Inc. (SEL), where he specialized in testing digital protective relays and communications equipment. He later moved to a product support role, providing applications assistance and training to SEL customers. In 2006, he began a supervisory role with SEL in the automation and integration engineering group, where he focused on the design, specification, and application of SEL automation products. Mr. Weber is presently the research and development manager for automation controllers at SEL. He has authored and coauthored several technical papers and application guides.