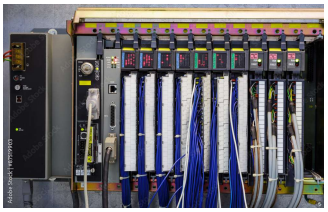
Cybersecure PLC and RTU Programming



© Adobe Stock

MLM-035-B

Industry	– Process
Principal Role	– Owner
Professional Role	– All
Enterprise Phase	– Master Planning

Turn on your audio and click start to begin video

START

PRINT VERSION OF NARRATIVE

This Micro-Learning Module provides Guidelines for creating or modifying PLC software to control plant equipment.

The intended audiences for this MLM are engineers, PLC Programmers, and teams dealing with cyber defense, consultants, systems integrators, and managers making decisions about how to achieve industrial cyber security.

Top 20 PLC programming Project

- **The aim of the Top 20 Project**
  - Provide guidelines to engineers who are creating IACS programs.
  - Methodology for all IEC 61311–based programming languages.
  - Help improve the security posture of IACS architectures.
- **Leverage available functionality in the PLCs/RTUs/DCS.**
  - No additional software or hardware are needed to implement these practices.
  - Fit into the normal PLC/RTU programming and operating workflow.
- **Provides more than cyber-security benefits**
  - This process may also help improve reliability and maintenance procedures.
  - You may utilize your accumulated expertise related to programming to protect the PLC and RTU code.

PRINT VERSION OF NARRATIVE

The "Top Twenty programming" project aims to provide meaningful guidelines and practices for engineers creating Industrial Automation and Control Systems to control equipment. Note that the IACS acronym is often pronounced as “eye-axe”.

The listed guidelines are practical for all I E C 61311 programming languages, including Ladder Logic, Function Blocks, Structured Text, and Sequential Function charts.

These programming practices may be used with commercially available PLC, RTU and DCS programming tools. No additional software tools or hardware are needed. Consequently, these practices fit into regular programming and operating workflows and apply to a broad range of PLC, RTU and DCS verticals.

These practices also help improve reliability and maintenance by leveraging available PLC programming and code protection knowledge.

To protect the code, you may utilize your accumulated in-house expertise in PLC, RTU and DCS programming.

Secure PLC programming : Top 20 Principles

1. **Modularize PLC Code**

   Split PLC code into separately tested function block modules

2. **Track operating modes**

   When PLC is not in RUN mode, activate an alarm to the operators.

3. **Leave operational logic in the PLC**

   Leave the operational logic as directly in the PLC as possible.

4. **Use PLC flags as integrity checks**

   Put counters on error flags to capture math problems.

5. **Use cryptographic and checksum integrity**

   Check PLC code integrity with cryptographic hashes or another way.

PRINT VERSION OF NARRATIVE

This MLM primarily describes PLC, programming, but similar programming practices also apply for RTUs, DCSs, etc.

Principle 1- Modularize PLC Code. We must split the code into modules and test these modules independently.

Principle 2- Track operating modes. The programmer should keep the PLC in RUN mode, and when it is not in RUN mode, the program must activate an alarm to the operator. This is critical for detecting unauthorized intervention.

Principle 3- Leave operational logic in the PLC wherever feasible. Thus, most of the calculated values will remain in the PLC.

Principle 4- Use PLC flags as integrity checks. The PLC programmer should put counters on PLC error flags to capture any math problems.

Principle 5- Use cryptographic and checksum integrity checks to ensure program integrity. If cryptographic hashes are unavailable, check PLC code integrity in another way and raise an alarm when a change occurs.

Secure PLC programming : Top 20 Principles

6. Validate Timers and counters

The PLC should validate timers and counters. The process shall detect unusual operations.

7. Validate and alert for paired inputs/outputs

Ensure that both signals are not asserted together.
When input/output is not feasible, create an alarm
Where paired/related signals are entered, insert delay.

8. Validate input variables at the PLC

PLC variables must be restricted to a valid operational range
Cross-checks should be added to the PLC program.
Prevent values outside of the acceptable ranges.

PRINT VERSION OF NARRATIVE

Principle 6- Validate Timers and counters. If timer and counter values are used by the PLC program, they should be validated ;for example, the program should detect and alarm if backward counts are below zero or if integer additions overflow.

Principle 7 - Validate and alarm if paired input and output signals are not asserted. The program shall alert the operator when input and output states are not physically feasible. To comply with that requirement, consider making paired signals independent by adding delay timers.

Principle 8 - Validate input variables at the PLC, not only at the Human-Machine Interface (or HMI). HMI access to PLC variables should be restricted to a valid value range defined in the HMI. Further cross-checks in the PLC should be added to comply with this requirement. The goal is to prevent or alert on values outside acceptable ranges programmed into the HMI.

Secure PLC programming : Top 20 Principles

9. Validate indirections

To detect / avoid buffer overflow, validate indirections to catch errors caused by abnormal conditions.

10. Assign designated register blocks by function

The purpose is to validate data and block unauthorized external writes to protect the controller data.

11. Instrument for plausibility checks

The process shall allow for plausibility/ credibility checks by cross-checking different measurements.

12. Validate inputs based on physical plausibility

Operators can only input what's physically feasible

Create an alert upon an unexpected inactivity or condition-detect/avoid

5

PRINT VERSION OF NARRATIVE

Principle 9 - Validate indirections. To prevent buffer overflows, "poison" array ends that could thwart an efficient binary tree or hash function implementation.

Principle 10 - Assign designated register blocks by function. Using function blocks like read, write, and validate can prevent unauthorized writes and protect the controller data. It also helps to avoid memory-reset errors due to out-of-bound execution or malicious programs.

Principle 11 – Instrument for plausibility checks. This can be done by cross-checking different measurements or by integrating or differentiating time-dependent values over time and comparing them to time-independent measurements.

Principle 12 – Validate inputs based on physical plausibility. Ensure that operators can only input data that is physically feasible in the process. A timer can also be set for the maximum or minimum duration the activity or condition should physically take.

Secure PLC programming : Top 20 Principles

13. Disable unused ports and protocols

PLC controllers and network modules support multiple data ports and protocols.
Disable unused physical ports and block protocols not required for a specific IACS program.

14. Restrict third-party data interfaces

Restrict connections if not defined as mandatory for the process. Allow only read/write capabilities according to authorization.

15. Define a safe process state in case of a PLC restart

In case of PLC restarts, ensure operation safety (e.g., energize/de-energize contacts or keep the previous state).

16. Display PLC cycles on the HMI

Summarize PLC cycle time every 2-3 seconds and report the data to HMI for visualization on a graph.

PRINT VERSION OF NARRATIVE

Principle 13 - Disable unused ports and protocols. Programs that manage PLC controllers and network interface modules typically support multiple protocols and physical ports.
Most protocols will not be needed for a given application and should be disabled. As part of hardening, turn off physical ports that are not used.

Principle 14 - Restrict third-party data interfaces.
The program should restrict non-essential connections. Data interfaces should be well-defined and allow only read/write capabilities for the required data transfers.

Principle 15 - Define a safe process state in case of a PLC restart. The program should define safe conditions for the process, for example, energize or preferably de-energize contacts to recreate the “default” state.

Principle 16 - Summarize PLC cycle times and show them on the HMI.
The program should summarize PLC cycle time every 2-3 seconds and report this to the HMI for visualization on a graphic display.

Secure PLC programming : Top 20 Principles

17. Log PLC uptime and trend it on the HMI

Trend and log uptime on the HMI for diagnostics purposes.

18. Log PLC hard stops and trend them on the HMI

Store PLC hard stop events caused by faults or shutdowns for retrieval by HMI. Create an alarm before the PLC restarts

19. Display the PLC memory usage on the HMI

Measure and provide a baseline for memory usage for every controller in the production environment

20. Trap the false negative & and positive alerts

Identify critical alerts and program and trap those alerts. Set the trap to monitor and trigger on alert for any deviation

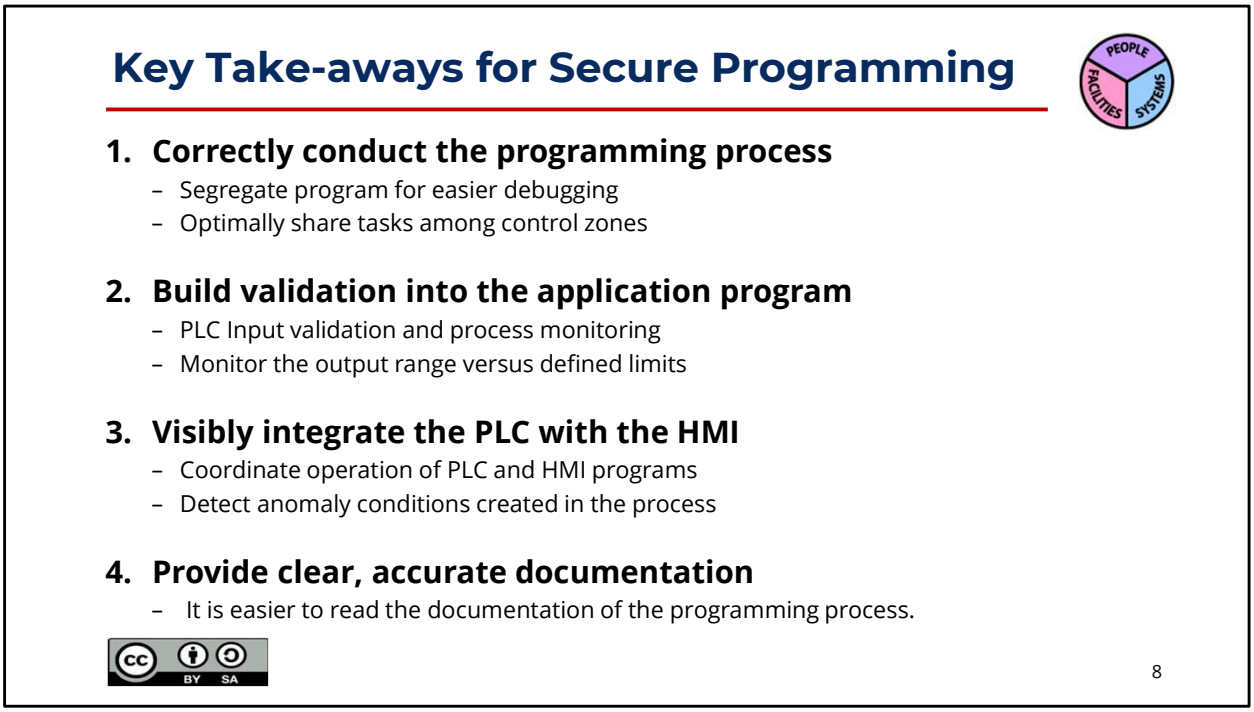
PRINT VERSION OF NARRATIVE

Principle 17 - Log PLC uptime and trend it on the HMI. The program shall know when it's been restarted. Conduct trending and log uptime on the HMI for diagnostics processes when required to be performed.

Principle 18 - Log PLC hard stops and trend them on the HMI.
The program shall store PLC hard stop events caused by faults or shutdowns for retrieval by HMI alarm systems to consult before PLC restarts.
Furthermore, it is essential to regularly conduct time synchronization to allow more accurate data to be recorded.

Principle 19 - Display the PLC memory usage on the HMI. The program shall measure and provide a baseline for memory usage for each controller deployed in the production environment.

Principle 20 - Trap false negatives and positives for critical alerts.
The program shall identify critical alerts and program handling for those alerts. The program shall set the trap to monitor the trigger conditions and alert any deviations.



PRINT VERSION OF NARRATIVE

The following are some key take-aways for Secure PLC and RTU programming.

1. Correctly conduct the programming process:
   Use segregated programs for easier debugging and optimally share tasks among control zones.

2. Build strong validation into the application program: The program should perform PLC Input validation ,process monitoring, and monitor output ranges against defined limits.

3, Visibly integrate the PLC with the HMI. This can be achieved through the coordinated operation of PLC and HMI programs. This facilitates detection of anomalous process conditions created in the process.

4. Finally, remember that thorough and accurate documentation is critically important. It makes it easier to understand the program and the intended process control.strategy.

Further Information and References

- **Related MLMs**
  - MLM-035-A Cybersecure PLC and RTU Principles and Definitions
  - MLM-035-C Cybersecure PLC and RTU Maintenance
  - MLM-035-D Cybersecure application of IIoT Edge Devices

- **References and Further information:**
  - https://www.youtube.com/watch?v=TX2m7QvEfNU&ab_channel=InternationalSocietyofAutomation-ISA
  - https://plc-security.com/content/Top_20_Secure_PLC_Coding_Practices_V1.0.pdf

Note: Since a substantial part of this MLM was extracted from "Top 20 Secure PLC Coding Practices", as required, we have included the Admeritia copyright statement at the end of this MLM.

PRINT VERSION OF NARRATIVE

Related MLMs:

This MLM is one of a set of 4 on the subject of cybersecure PLCs and RTUs. These MLMs address "Principles and Definitions", Programming, Maintenance, and Modification and Upgrades.

References and Further Information:
This MLM was created based on a project conducted in 2018 by a team of leading experts: Sarah Fluchs, Isiah Jones, Jake Brodsky, and Vivek Panada, and it was published as Secure PLC programming Practices, Top Twenty List.

For more detailed information and to read the complete document, please refer to the blue links on this slide.

Author

Daniel Ehrenreich

Daniel has over 33 years of experience with control of industrial operations and integration of cyber security solutions.

He is a control engineering consultant, workshop lecturer, and an expert in cyber secured operation for IACS.

Daniel is contributing his knowledge and expertise to multiple ISA 62443 workgroups.

Since 2016, acting as the Chairman of the annual ICS-Cybersec Conference taking place in Israel

Please click here to provide feedback on this MLM.

PRINT VERSION OF NARRATIVE

Daniel Ehrenreich has over 32 years of experience with control of industrial operations and integration of cyber security solutions.

He is a control consultant, workshop lecturer, and a consultant at Secure Communications and Control Experts. Daniel is also contributing his expertise to multiple ISA 62443 workgroups and conducting free of charge podcast sessions for educating engineers worldwide. Since 2016, acting as the Chairman of the annual ICS-Cybersecurity Conference.

Please click this link to provide feedback to the author.

Admeritia Copyright Terms and Conditions

11

This Admeritia copyright statement is in addition to the standard PERA website "Creative Commons copyright, and supersedes it where they conflict.