DIGITAL INDUSTRIES SOFTWARE

How to Design Safe and Secure Industrial Devices

Executive summary
There is no question that safety and security cannot be emphasized enough in today’s world of Industrial IoT (IIoT), but you can build safe and secure RTOS, Linux®, and Android environments. IEC 61508, IEC 62062, ISO 13849, IEC 61511, and ISO 10218 are all safety standards in place today to maximize safety and minimize risk in industrial devices with embedded software. These standards help define a systematic approach to safety management with the incorporation of safety thought-processes in the product development process all the way from initial concept to end-of-life, and need to be applied not only at the system level, but also at the node and edge device levels. This white paper discusses the various steps your organization can take to follow and adhere to these safety standards in place today to greatly decrease the risk of economic damage, environmental harm, and risk to human life.
The importance of functional safety, as an inherent feature built into your IIoT enterprise or ecosystem, cannot be overstressed. As the CEO, company executive, or plant manager, it is critical that your industrial enterprise conforms to the latest functional safety requirements – not just at the system level, but down to every end node and edge device. Employing safety standards not only allows you to meet the latest industry regulations, but helps lower the risk of system malfunction or injury to personnel should an event occur. Another advantage with an orchestrated safety protocol in place is that your IIoT network (along with your customers) will be less susceptible to prolonged periods of downtime should a breach happen.

This paper discusses the various steps your organization can take to follow and adhere to the numerous safety standards in place today. These include IEC 61508 - Industrial; IEC 62062 - Machinery; ISO 13849 - Safety Related Parts of Control Systems; IEC 61511 - Process Control; and ISO 10218 - Robotics.

The issue of security requirements for IoT/IIoT devices will also be discussed later in this paper.

Factors addressing IIoT today
To stay competitive in a crowded marketplace, industrial automation manufacturers and IIoT leaders must address the following forces:

- Global Market Competitiveness: The global market is driving the focus on cost reduction for both capital and operational expenses.
- Safety Requirements: Costs related to the possibility of injury within an industrial automation setting are driving demands for increased safety requirements.
- Security Issues: Due to increased connectivity between devices and local access networks, and the increased value of the data being exchanged, corporate espionage, data theft, and cyber-terrorism are very real concerns, driving a major focus on a variety of security issues.

A reliable solution for industrial automation systems must not only address the trends mentioned on the previous page, but it must also enable industrial automation manufacturers to increase the competitiveness of their offerings, reduce product development time, and of course, minimize effort and cost with the start-up of an IIoT system. Achieving these goals will ultimately lead to faster and more efficient time to market. Further, when moving from a legacy system to the IIoT, companies are looking to consolidate their current embedded designs to optimize software reuse and extend industrial product lines while reducing size and power consumption.

Of course, tantamount to all these challenges, is building or maintaining an IIoT system that addresses safety issues and adheres to all possible security requirements. In doing so, you will not only save costs and run a more efficient operation, but you might increase the positive reputation of your company as well. You will also reap additional benefits such as lower maintenance costs and increased quality, which leads to higher customer satisfaction.
The difference between “safety” and “security”

Safety is about protecting people and equipment (yours, your customer’s, and/or the public’s at large) from a device and any system malfunction that might occur. Security is about protecting your device and network from the negative influences of the outside world.

In the industrial market, safety systems are designed to detect potentially hazardous conditions, and then respond in a way that will minimize the damage (both personal and economic) that might occur when these hazardous conditions occur. The proper behavior when these hazardous conditions occur will be dependent upon the type of hazard, the overall capabilities of the device, and other factors. In many cases, the best response will be to simply stop operation, but this may require that other activities also be performed beforehand (for example, stopping a chemical reaction that may already be taking place).

Secure systems are analogous in many ways; in that they are designed to detect or prevent potential devastating threats, and respond in a way that will minimize damage. The methods to minimize these threats include making it difficult for external agents to infiltrate the device (using techniques like authentication, encryption, and others); logging attempts to infiltrate when they occur; and also by making the application as difficult to exploit as possible.

Security issues can easily become safety issues. For example, a cyber terrorist attack might be designed to disable or otherwise subvert safety mechanisms in the device to devastating effect. Even if the infiltration is intended to be benign (hackers will infiltrate a device simply because they can), an inadvertent mistake on their part can lead to the same negative effects. Of course, safety issues can becomes security threats as well.

IEC 61508 – The defacto safety standard for industrial systems

With increased demand to improve onsite safety and IIoT functionality, the industry has initiated the need to implement functional safety standards with IEC 61508 as the lead standard. The International Electrotechnical Commission (IEC) is a global organization that publishes consensus-based standards and manages conformity of this standard. Following the IEC 61508 standard, users are able to design an IIoT network with inherent safety built-in from individual device node to cloud. Several subsectors of the broad industrial market have created specific extensions to IEC 61508, such as IEC 62061 for electrical installations in outdoor sites such as mines or quarries, or ISO 11161 for integrated manufacturing systems. In these cases, the more specific standards defer to IEC 61508 for general topics, extending those topics for the needs of a specific industry, or providing specific guidelines to the general topics for the same reason.

To conform to IEC 61508, one must first establish a model safety development process. Steps from the beginning of the process to completion and maintenance include:

1. Define the concept and scope.
2. Perform a hazard analysis.
3. Classify the hazards to Safety Integrity Levels (SIL).
4. Define safety requirements for each device to mitigate the identified hazards.
5. Plan for the realization for the device.
6. Develop the device’s components in terms of both hardware and software.
7. Integrate the device components.
8. Validate that all safety requirements are fulfilled.
Step #1: Define the concept and scope
In the language of IEC 61508, the concept of a project is to understand the equipment under control (EUC) with a goal of understanding the purpose of the device sufficiently so that the remaining safety activities have meaning, and that everybody who will be participating in the project has a similar understanding of what the project intends to accomplish. Once the concept is understood, the scope of the project is defined. The goal here is to determine the boundary of the EUC and its control systems so that this is all understood before any hazard analysis takes place.

Essentially, the objective in this phase is to make sure all participants know what is being created and how it will be used (both in terms of the environment and capabilities of the device) so that as the system is designed and developed, all participants have a shared understanding of the device. Note that this understanding, which is required by the safety standard, is also necessary for creating secure IIoT devices, understanding the concept and scope of the device are needed so that any kind of later threat analyses (and derivation of security requirements that come from the threat analysis) have meaning.

Step #2: Perform a hazard and risk analysis
Once the scope of the overall system is fully understood, the next step is to use that information to perform a hazard and risk analysis. The purposes of the hazard analysis are outlined as follows:

Define the hazards: Consider the hazards, hazardous events, and situations of the EUC. This analysis must be determined under reasonably foreseeable situations, including areas you might not normally consider such as misuse, unauthorized and/or malevolent actions. The standard requires that human factors also be considered, with a focus on infrequent modes of operation, since these modes are the most likely to be used incorrectly. It is also necessary to consider security threats.

Eliminate or reduce the hazards: As the threats are being defined, consideration should be given to if they can be eliminated as part of the overall design.

Understand the sequences leading up to the hazards: As the triggers to the hazardous events are understood, it might be possible to eliminate them through a combination of design and processes.

Determine the likelihood and consequences associated with the hazardous events: Knowing how likely the hazard is to occur in real-life, and what might happen when it does, is necessary to determine the SIL level of the hazard (see below).

IEC 61508 does not specify how the hazard analysis shall be performed; only that one shall be. There are many standardized hazard analysis methodologies including these standard methods:

Failure Mode and Effect Analysis (FMEA) – A qualitative analysis of the kinds of failures that can occur to the components in the system that could conceivably affect the safety of the system. Each of these potential failures is numerically rated to the severity of the effects resulting from the failure, how likely the failure is to occur, and how likely the system is to detect the failure (all of which are generally rated on a scale of 1-10). These ratings are then multiplied together to generate a Risk Priority Number (RPN), the higher the RPN, the more critical it is to mitigate the risk in the design of the system.

Fault Tree Analysis – A quantitative analysis first looking at all undesirable outcomes, and then working backwards, identifying how those outcomes may occur. The analysis is generally performed graphically using Boolean logic, and allows for understanding of the prioritization of contributors to each of these faults.

Figure 1: Overall safety lifecycle - IEC 61508-1 (Edition 2).
• Hazard and Operability Study (HAZOP) – A more formally structured method generally performed on physical systems, in which systematically investigates each element of a system for every way in which important parameters can deviate from their intent, and cause resulting hazards and operability problems.

Once the hazards are understood, a second analysis will be performed to verify which of the hazards are independent, and which might be linked in some way (i.e. be common cause failures). From this analysis, a collection of independent hazards (or collections of linked hazards) are created.

**Step #3: Classify the hazards to SIL levels**

Once the hazards are understood, they must be classified. IEC 61508 classifies hazards according to their Safety Integrity Level, or SIL. Once the dangerous hazards are determined, a SIL level is assigned. The standard does not provide explicit guidance to the assignment of a SIL level to the hazard, since in the opinion of the standard, this is somewhat subjective; the guidance of the standard is that the SIL level is determined to provide a “tolerable risk” of how often the hazard will actually occur.

Instead of defining how the SIL level is calculated, the standard specifies how much protection a safety measure must provide to the system once the SIL level is determined. The determination of SIL level of a safety measure is set probabilistically, where the higher the SIL, the better the safety mechanism will perform in preventing the dangerous occurrence from happening.

The mapping of the SIL level to the probability of failure is based on the operation of the safety function as determined by the various modes below:

- **Low Demand Mode:** systems where the safety function is performed on demand, and the frequency is no more than once per year
- **High Demand Mode:** systems where the safety function is performed on demand, but is more than once per year
- **Continuous Mode:** where the safety function retains the RUC in a safe state as part of operation

For systems with a Low Demand mode, the SIL level is set based on the following table (Table 1), where the SIL level is based on the probability of a dangerous failure per operation. An example might be the transition of the device to an annual maintenance model.

<table>
<thead>
<tr>
<th>Safety Integrity Level (SIL)</th>
<th>Average probability of a dangerous failure on demand of the safety function (PFD_{avg})</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$\geq 10^5 \text{ to } &lt; 10^4$</td>
</tr>
<tr>
<td>3</td>
<td>$\geq 10^4 \text{ to } &lt; 10^3$</td>
</tr>
<tr>
<td>2</td>
<td>$\geq 10^3 \text{ to } &lt; 10^2$</td>
</tr>
<tr>
<td>1</td>
<td>$\geq 10^2 \text{ to } &lt; 10^1$</td>
</tr>
</tbody>
</table>

Table 1: Safety Integrity Level Mapping for Low Demand Mode.

where the safety mechanism might be that the system must first determine that no potentially dangerous activities are taking place, and if they are, then these are safely terminated before the transition to maintenance mode. Since the activation is manual, at the SIL 3 level, we might expect some kind of hazardous failure every 500 years or so, although this is spread out over the number of systems produced; if there are 500 such units, then one hazardous failure per year across these 500 systems will be expected. It is up to the system designer to determine if this risk is “tolerable” based on how severe the failure might be.

For other systems, the SIL level is based on a different table (Table 2), where the SIL level is based on the probability of a dangerous failure either per operation (High Demand mode) or per hour (Continuous mode).

As an example, a SIL 3 system running in continuous mode would be expected to have a dangerous failure once in every 107 to 108 hours (or once every 1,140 to 11,400 years). Again, this is spread out over the run of devices, so if there are 1,000 devices created, the manufacturer would expect a dangerous failure somewhere between once a year and once a decade, and the hazard is such that this risk is “tolerable.”

<table>
<thead>
<tr>
<th>Safety Integrity Level (SIL)</th>
<th>Average frequency of a dangerous failure of the safety function [h^{-1}] (PFH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$\geq 10^9 \text{ to } &lt; 10^{10}$</td>
</tr>
<tr>
<td>3</td>
<td>$\geq 10^8 \text{ to } &lt; 10^9$</td>
</tr>
<tr>
<td>2</td>
<td>$\geq 10^7 \text{ to } &lt; 10^8$</td>
</tr>
<tr>
<td>1</td>
<td>$\geq 10^6 \text{ to } &lt; 10^7$</td>
</tr>
</tbody>
</table>

Table 2: The Safety Integrity Level Mapping for High Demand or Continuous Mode.
For example, the risk of a failure that might result in human death might not be tolerable if it were to happen once per year across all of the created devices if the device were to be used inside people’s homes, but might be acceptable if the device is intended to be used in an already dangerous workplace (especially if this rate is lower than whatever device the new system might be replacing).

**Step #4: Define safety requirements for each device**

Now that the hazards have been identified and assigned SIL levels, the next step is to devise safety measures to meet the SIL requirements for those hazards, and these measures must then be turned into safety integrity requirements. The safety measures will be determined by your systems engineers, considering the hazards and the best ways to mitigate them in the context of the EUC. The requirements to satisfy these safety mechanisms should be stated in the same way as the functional requirements are stated i.e., to the same level of specificity, completeness, and non-ambiguity.

Once the safety requirements are specified, they are allocated to specific systems in the overall device. Note that at this point, the SIL levels may be reconsidered. For example, it’s possible to assign a single safety requirement to multiple systems, not all of which will be assigned the SIL level of the original requirement. As a more specific example, it is commonplace to handle a SIL 4 requirement through redundant systems; if one of these systems fail, then the system will still be safe as long as the redundant system(s) continue to operate properly. In this case, it is often appropriate to consider each of the redundant systems to be SIL 3.

**Step #5: Plan for the realization for each device**

While considering the system architecture and design – safety validation, operation, maintenance, installation, and commissioning also need to be considered. Since these plans may influence and modify each other, they should be considered as close to each other as possible. Once these plans are complete, the system can be decomposed into its overall design requirements and its overall system architecture, consistent with these plans.

**Step #6: Develop the device’s components in terms of both hardware and software**

This part of the process is something that your engineering teams are probably already pretty good at. At this stage, it should be fully understood what needs to be created, and the planning for its detailed design, implementation, and unit verification should be well understood.

Of course, how the engineers go about these important activities is extremely important if the end result is to be safe and secure. All of the understanding of hazards, definition, and planning is not very useful if the implementation is done haphazardly, or if there is an easily exploitable software flaw (like a buffer overflow exploit) that allows hackers to take over operation of the device. Because of this, it is vital that both hardware and software development be performed using a high quality, repeatable, and formalized process (see the next step below).

![Figure 2: "Software system capability and the development lifecycle" from ISO 26262-3](image-url)
Once the components have been created, they should be validated stand-alone at the unit level (figure 2). This allows for a deeper and more complete level of verification for each component then they will have when integrated, and increases the likelihood that latent defects will be uncovered. Note that verification at this level does not just mean unit testing, but also includes activities like static code analysis and inspections.

**Step #7: Integrate the device components**

Once the system’s components have been individually verified, they must then be integrated. At this point, special attention must be given to components that were not developed by your internal development teams and processes. For life critical systems, you cannot assume that you can safely use third-party software and hardware; instead you need to use trusted suppliers, with well-defined processes, and preferably with certified/certifiable solutions to support safety requirements. Note that this does not preclude the use of open source or un-certified components, but especially at high SIL levels, it might mean that these may not be appropriate to handle the safety functions of the device.

This applies equally to security. Traditionally, commercial embedded software did not concern itself with security concerns, especially as it might apply to potential exploits in the software itself. For many years, there was more financial incentive for hackers to exploit Windows or Linux systems, and there was some protection in the fact that these systems were not as well understood (security through obscurity). More recently, commercial operating systems like Windows 10 and Android have become harder to exploit, while at the same time hackers have found both economic and disruptive reasons to figure out these embedded systems and devices. Thus, it becomes even more important to use trusted vendors with high-quality and complete embedded system solutions.

**Step #8: Validate that all safety requirements are fulfilled**

While it’s important that the complete functionality of the device is verified, it is equally important that the verification of the safety requirements be complete and correct. For this very reason, it is essential that verification be considered early in the process (verification planning started when the safety requirements were defined, and refined as subsequent steps took place). Now that the device is complete, this step provides assurance that the hazards that were originally identified are properly mitigated in the final device.
Safety software development – A formalized process

As already discussed in the overall safety development process, safety software development has common themes across industries. Further, when considering the overall requirements for software development, the tenets of a safety development process essentially require a high-quality software development process. IEC 61508 does not specify the particulars of what the specific steps of what the development process entails, only that it takes on certain activities. These activities essentially are those one would expect to see in a quality software development process:

- Specification of software safety requirements to a level of specificity and completeness that they can be unambiguously implemented.
- Verification will be planned in a way that ensures the verification of these safety requirements without regard to how they are implemented.
- Complete documentation of the design that makes the data flow, timing, exception handling, etc. in a clear, unambiguous way.
- Implement the software based on the requirements, architecture and design, and that this code is reviewed by manual and/or automated means.
- The software units will be unit tested, and once the modules are integrated they will be tested together.

For each of these top level concepts, the standard breaks these into component parts, and provides guidance of the goals of a process that would be compliant to the standard, but does not provide guidance on how those goals will be met. As a result, it is the responsibility of your development organization to create its own developmental processes with these goals in mind.

So what makes a high-quality developmental process? When looking at this at a very high level, a quality development process takes on three basic tenets:

Following the three basic tenets:

#1: Say what you do
The developmental process must state how and when each activity will be performed, what tools will be used, what the expected outcomes are, etc. As a simple example, it is not sufficient to state that all code will be statically analyzed, instead, you should state what tools will be used, how it will be configured, which rules (if any) will be ignored, what happens when issues are flagged by the tool, etc.

#2: Do what you say
After you’ve gone to the trouble of documenting your processes, it’s important that the development, verification, quality, and testing teams follow those processes. Training is obviously important here, but so is auditing. Auditing does not have to be a heavy-weight processes, but if you aren’t tracking adherence to the processes you’ve defined, you can’t claim conformance to the high-quality development process you spent the time to create.

#3: Be able to show a 3rd party that you did the first two tenets
When constructing your quality system, consider that you will likely be audited by external agents, whether these are customers, 3rd party certifiers such as TÜV or UL, or other parts of your organization. These auditors will not understand your processes or the product you’re creating, but what they will be interested in is whether your development was done in a high-quality manner that is in alignment with the other two tenets. To be able to show this, you need to make certain that the results of each step of the process leaves some collection of artifacts that shows that you did what you said you’d do.
Going back to the static analysis example, when you run your static analysis tool, you’ll want to configure the tool so that it generates a report, and you’ll want to make sure that report is archived in a known location. You’ll also want to make sure that the decisions you made about the configuration of the tool are documented, likely as part of your verification plan. Finally, you’ll want to make sure that any issues identified are filed as issues rather than handled in a less formal manner. As long as all of this is considered as part of your process development, then the additional overhead to make sure that you’ve achieved conformance will be minimized.

Will following these tenets automatically mean that you’re fully conformant to IEC 61508 (or any other safety standard)? Not necessarily, but if your organization takes a best effort in defining the processes in accordance with the standard, then most auditors/certifiers will work with you to map what you’re doing to the standard rather than requiring an immediate update to the processes. Even if adjustments must be made, they can be made on a solid base and the improvements can be planned. Another benefit of these tenets will mean that you will find that your processes are likely nearly or completely conformant to quality standards such as ISO 9001 or SPICE, and that you did this in a way that improves quality rather than just generating paperwork. A win-win-win for your organization!

From a security standpoint, what is interesting is that these tenets are still vital to the creation of secure software. Any formal security software development standard (such as Microsoft’s Security Development Lifecycle or the recommendation of the National Institute of Standards and Technology, as spelled out in NIST 800-82) will outline these tenets, as well as several of the specific recommendations for the elimination of latent threats such as static analysis and threat analysis. The main takeaway here is that once a solid safety process is in place, expanding that process to ensure security is straightforward. The main differences are at the beginning, where identifying the security threats is a practice that will be unfamiliar to system engineers who already understand identifying safety hazards, and at the end where different verification techniques such as penetration testing might be required. This gap can be closed with training, or by bringing on additional expertise (like hiring a security manager who works hand-in-hand with your safety manager).

**Security for IoT/IIoT devices**

While most of this paper has described safety practices, security requirements for IoT/IIoT devices will fall into three basic categories:

1. **Basic security functionality** that’s needed to protect the device from generalized external threats. This would include things like Secure Key Exchanges, the use of HTTPS instead of HTTP for any kind of web-based interfaces, the use of data encryption for data-in-transit, logging and alarming of suspicious activity, etc. These will come out of the hazard analysis, and are simply good security development practices for the current world.

2. **Basic security functionality needed to protect the device from specific external threats.** These also will come from the hazard analysis, but will target scenarios specific to the device being created. For example, large paper milling operations that handle sensitive operations (such as currency printing) will need to consider specific attack vectors that will either compromise the machinery or its output.

3. **Generalized protection of the application.** The concepts above concern themselves with functionality to protect the device from known threats. However, in the world today, we have to assume that regardless of how much security we build into the device that at some point in the future a hacker will manage to break through. Bringing in white-hat hackers to attempt to identify (and fix) zero-day vulnerabilities will help, but even that is not perfect. As a result, care should be given to make the underlying application as difficult to exploit as possible; the lessons we’ve learned from safety software development described in this paper will provide a level of protection from exploit as well.
Conclusion

Over the past decade, management of safety as part of system designs has gone from being something that we try to do the “best we can” to a much more formalized practice that can be used to greatly decrease the risk of economic damage, environmental harm, and risk to human life when things go wrong. This has been achieved through a systematic approach to safety management and incorporation of safety thought-processes in the product development process all the way from initial concept to end-of-life. This improvement has been difficult, but many subsectors of industry now have a common understanding on how to achieve this, and the economic and moral obligation to do so.

Over the next decade, the management of security threats will become just as important to manufacturers of devices in all marketplaces, but especially vital in the industrial marketplaces. Fortunately, the same thought process (and many of the same methodologies) that are part of the safety lifecycle can be re-applied to the security life cycle.

As thought leaders in your respective industries, it is vital that you drive this thinking into your organizations. It’s a challenge that should be relished; those who best solve the issues and concerns of customers and the public about industrial espionage and cyber terrorism will have a significant competitive advantage for the foreseeable future.

For more information on Siemens Embedded products and services please visit siemens.com/embedded.
About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Xcelerator, the comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software, helps companies of all sizes create and leverage a comprehensive digital twin that provides organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

About the author

Robert Bates is responsible for safety, quality and security aspects of Siemens’s embedded product portfolio targeting the Industrial, Automotive and Aerospace markets. In this role, Robert works closely with customers and certification agencies to facilitate the safety certification of devices to IEC 61508, ISO 26262 and other safety certifications. Before moving to this position in 2014, Robert was a Software Development Director at Wind River, where he was responsible for commercial and safety certified operating system offerings, as well as both secure and commercial hypervisors. Robert has 35 years of experience in the embedded software field, most of which has been spent developing Operating System and Middleware components to device makers across all markets and regions.