PLCopen®
for efficiency in automation

PLCopen Safety - Integrating Safety Functionality into the Development System

Taking IEC 61508 & IEC 62061 to the Software Level

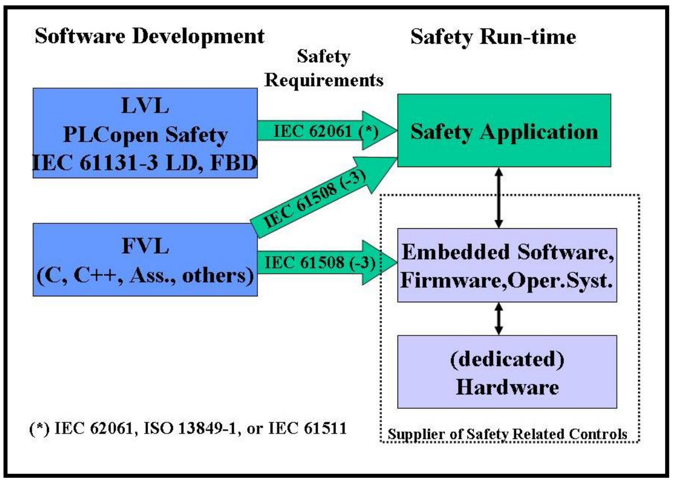
Overview

The changing environments facing the machine building industry ask for new solutions. The changes include:

- The availability of many safety standards, including IEC 61508 and IEC 62061;
- Additional governmental requirements increasing the liability issues;
- The availability and acceptance of digital networks with safety functionality built-in;
- The inherent move from hardwired safety functionalities to software solutions;
- The increasing importance of safety related issues regarding personnel and machines.

The solution includes standardization of the safety functionality on the software level, and integrating this in the development environment. This combination helps developers to integrate safety related functionality with more ease in their systems, even from the beginning of the development cycle. Also, it contributes to the understanding of safety aspects, as well as to reducing the certification time and costs by relevant organizations.

Positioning of the work of PLCopen


Software Development
Safety Run-time
Safety Requirements
LVL PLCopen Safety IEC 61131-3 LD, FBD
IEC 62061 (*)
Safety Application
IEC 61508 (-3)
FVL (C, C++, Ass., others)
IEC 61508 (-3)
Embedded Software, Firmware,Oper.Syst.
(dedicated) Hardware
(*) IEC 62061, ISO 13849-1, or IEC 61511
Supplier of Safety Related Controls

IEC 61508 and IEC 62061, and companion and related standards, describe safety requirements at different levels. However, they are open on the implementation of this functionality in software development environments for the creation of the application software. Also, both standards are targeted to a somewhat different level: run time environment or application level. For both different tools are used.
The new safety related standards come with a new nomenclature. Some related terms are:

- FVL : Full Variability Language. Application independent languages used by component suppliers for the implementation of (safety) firmware, operating systems, or development tools. Rarely used for the safety application itself. Typical languages are C/C++, Java, and assembler.
- LVL : Limited Variability Language. Aimed at users to create their safety application functionality. Typical languages used are Ladder Diagram and Function Block Diagram.
- SRAS : Safety Related Application Software
- SRES : Safety Related Embedded Software

The Rationale of a New Safety Standard

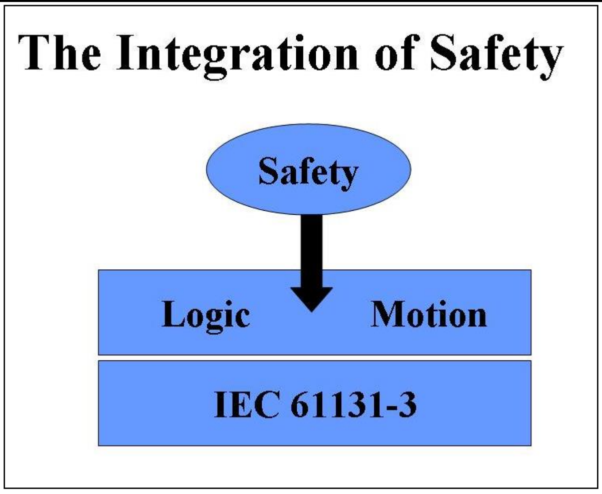
Machine builders are faced with a large set of safety-related standards. This makes it expensive and in some cases unfeasible for machine builders to understand them all fully. Yet in the end they are still responsible for their products and related safety aspects. This risk situation is not very healthy, especially since legislation imposes greater constraints on the equipment suppliers. And their liability increases.
Nowadays there is often a clear separation between the safety-related part and the functional application part. This separation can be made by using different systems for the environments, different tools, and even different people can be involved. This separation often results in the safety aspects being included at the end, and not integrated into the whole system philosophy from the beginning, and often with only limited tests performed. This clearly does not contribute to the overall safety aspects.
Also, the on-going technological innovation now provides safety-approved digital communication busses. This supports the trend away from hard-wired systems towards software-oriented solutions. A parallel can be drawn with the movement away from hard-wired relay logic towards programmable logic controllers, PLCs. Such a trend, of course, involves a change in the mindset. This type of change requires time, widespread support from the industry as a whole, support from educational institutes as well as from certification bodies.
In addition, governmental requirements add to the complexity. For instance, the US-based FDA, Food and Drugs Administration, has set strict regulations that must be complied with. Non-compliance can result in heavy financial penalties, again weakening the sustainability of the organization.

Standards enhance the safety aspects

With so many standards already available, one needs to help the users to implement them, without inhibiting their functionality and performance, and without adding costs. Standardization in functionalities and the integration and support from the software tools should help the programmers to integrate safety functionality in their applications more easily.
The common basic requirements of a safety application for machine builders within all applicable safety standards are:

- Distinction between safety and non-safety functionalities;
- Use of applicable programming languages and language subsets;
- Use of validated software blocks;
- Use of applicable programming guidelines;
- Use of recognized error-reducing measures for the lifecycle of the safety-related software.

In order to support this, the independent association PLCopen, together with its members and external safety related organizations, defined safety related aspects within the IEC 61131-3 development environments. With this, the safety aspects are transferred to a software tool, which is integrated in the software development tools. This helps developers to integrate safety-related functionality into their systems, even from the beginning of the development cycle.
Also, it contributes to the overall understanding of safety aspects, as well as reducing the certification time and costs by relevant organizations.



Standardization in the look and feel

In order to help developers use safety-related functionalities, the comfort zone of users must be improved, thus making it easier to accept this way of working. This can be done by standardizing the look and feel of the safety functionalities. In this way the safety functionality can be better recognized and used independently of the applicable system. Re-training is not necessary and the tendency to create dedicated safety functionality is reduced. This complete approach provides the user with a harmonized view to the total application within one environment. With this, a major contribution to the acceptance and usage of safety related functionality is made. This will take away several hurdles as they now exist, especially in the machine building industry.
In addition, this assists the certification bodies. Specifying and checking the safety software becomes much easier, and therefore quicker, less risky, and less costly.

Providing Function Blocks at a higher level makes them less dependent on the underlying hardware architecture. Architectures such as hard-wired systems, systems containing safe input and output modules, and network-based systems can be supported with the same Function Blocks. With this higher-level solution the implementation details can be hidden from users, making the implementation of safety-related software much easier and less costly. This also improves the comfort zone of users.

To support this, the PLCopen Technical Committee defined a multi-level approach:

A. Support in programming environment, including languages and functionality, combined with programming guidelines for the users of the safety related development environment, to be able to create safety related application programs;
B. The definition of a set of Functions and Function Blocks with safety related functionality, including a style guide for additional functionality;
C. Error handling and diagnosis.

Ad A: Support in programming environment

Within the programming environment there are three levels of users identified, with different sets of experience and authority to manage the safety related aspects:

1. Basic level – focused to the safety-application programmer using the specified function blocks;
2. Extended level – extended functionality giving the ability to define own extensions to the specified Function Blocks.
3. System level – focused to the implementation by the suppliers of the (specified) function blocks. This level is not further described in the document.

Without going too much in detail, we zoom in on the first group. For this, the safety standard IEC 61508, Chapter 7, defines a reduction in the preferred programming languages for the different SIL levels (Highly Recommended, Recommended or Not Recommended). Based on this, the preferred languages within this group are the graphical languages FBD and LD, and even ST with a defined subset of these. The graphical languages provide a clear overview of the safety program itself, and the tool can support and guide the users much better. The textual language adds flexibility to the Extended Level.

Separation via the SAFE datatype

In order to differentiate clearly between safety-relevant and standard signals, a new data type with the designation "SAFE" was defined. This provides the basis for the development tool to identify safety-critical program parts, and guide the user with permissible connections, while preventing incorrect connections. Furthermore, because of this designation the data links can be verified automatically to detect any impermissible links between standard signals and safety-relevant signals. In this way, support can be implemented for the different levels of the various safety standards and errors in the application program are minimized. Additionally, when releasing the application program, the safety-relevant signals can be clearly recognized. This simplifies and shortens signal flow verification and eases the validation and certification purposes.
For instance, SAFEBOOL is a data type that is applicable within the safety-related environment and represents a higher safety integrity level. It differentiates between safety-related and non-safety-related variables. A SAFEBOOL acts as a BOOL within the system, but can contain additional information (attributes) necessary for the safety status and level (could include PL, SILs, PFD/PFH). Such information could be used to calculate the SIL with the programming tool.

Essentially there are (at least) two ways to get a SAFE(BOOL) variable in the application level:

1. The data is provided as a safe data type by the devices, either by the devices themselves or by the operating system or firmware. This can include a safe network.
2. The data is provided by combining safety inputs in the application itself (such as two safe single-channel inputs).

Reduction in data types and declarations (short form: user-defined data types and variable declarations not shown here)

Note: BL = Basic Level; EL = Extended Level

Description	**BL**	**EL**	**Comment**
ANY_INT, ANY_SAFE INT	X	X	*BL*: Arithmetic functions are not permitted. *EL*: Arithmetic functions are permitted.
ANY_REAL ANY_SAFE REAL	X	X	Same as INT, DINT Only valid values for REAL and SAFEREAL are allowed.
ANY_DURA TION ANY_SAFE DURATION	X	X	*BL*: Only as a constant FB input parameter and/or as outputs for diagnosis on the called FB. *EL*: no restrictions
ANY_BIT ANY_SAFE BIT	X	X	*BL*: Only as outputs for diagnosis on the called FB and as inputs for processing diagnostices codes from another FB. *EL*: no restrictions
ANY_DATE	X	X	
ANY_SAFE DATE	-	X	

Reduction in functions and function blocks

1. Standard functions: (See IEC 61131-3; Tables 22-30)

Description	BL	EL	Comment
AND	X	X	*BL*: Operation of both BOOL and SAFEBOOL. *EL*: Operation on ANY_BIT
OR	X	X	*BL*: Operation of SAFEBOOL only allowed. Extended Level: Operation of both BOOL and SAFEBOOL allowed, but not mixed. (S_OR and OR)
XOR, NOT	-	X	Operation for XOR only allowed with 2 inputs
ADD, MUL, SUB, DIV, MOD, EXPT, +,*,-,/,MOD,**	-	X	Operation of INT/ DINT/ REAL and SAFEINT/ SAFEDINT/ SAFEREAL permitted.
NEG -	-	X	Negation in ST (and FBD)
MOVE	-	-	
SHL, SHR, ROR, ROL	-	-	Shift functions are not required.
EQ, NE, =, <>	-	X	Restrictions apply
GT, GE, LE,LT >,>=,<=,<	-	X	Restrictions apply
SEL,MAX,MIN, LIMIT, MUX	-	X	Restrictions apply
Type conversion functions	X	X	Implicit & explicit Restrictions apply
String functions	-	-	No STRING available
Time Functions	-	X	Only ADD,SUB, DIV, MUL with type TIME or SAFE_TIME
Unary REAL functions	-	X	e.g. SIN, SQRT, LOG.

2. Standard function blocks: (See IEC 61131-3; Tables 34-37)

Description	UL	EL	Comment
TON, TOF, TP	X	X	
CTU, CTD, CTUD	X	X	
Bistable FB (SR, RS)	-	X	No semaphores (SEMA)
Edge Detection	-	X	

Structured Text Specifics.

Within the textual language ST, supported only in Extended Level, the powerful statements like FOR, IF..THEN..ELSE, CASE OF are supported, as well as EXIT, RETURN and CONTINUE. WHILE and REPEAT are not supported due to the danger of an endless loop and non-constant loop count.

Ad B. Functions and Function Blocks

The PLCopen Technical Committee on Safety has identified 23 safety functions, which are represented by 33 Function Blocks. These provide the basis for developing certified Function Blocks within an environment. The specification itself provides a unified description of all the Function Blocks. Included elements are:

1. Applicable Safety standards, with reference to the sections of the relevant requirements;
2. Interface description, including name of the FB, and short description;
3. Functional description, including Safe State description, in both textual and graphical form, including description of normal operation and start behavior;
4. Error detection, with description of External signals, Internal signals, and External test signals;
5. Error behavior;
6. Function Block specific error and status codes.

The full list of blocks is:

Reset Button	Adds the trailing edge functionality to all FBs. Used to comply to EN ISO 13849-1:2015
Equivalent, Antivalent	Converts two equivalent or antivalent SAFEBOOL inputs to one SAFEBOOL output, including discrepancy time monitoring.
Mode Selector	Selects the mode of operation of the system, such as Manual, Automatic, Semi-automatic, etc.
Emergency Stop	Monitoring an emergency stop button.
ESPE	Monitoring Electro-Sensitive Protective Equipment (ESPE).
PSE	Pressure Sensitive Equipment like safety mats
Two-Hand Control Type II & Type III	Provides the two-hand control functionality. (cf. EN574, Ch. 4 Type II and Type III (with fixed time difference is 500 msec.)).
Testable Safety Sensors	Can be used for external testable safety sensors (like ESPE:).
Sequential Muting, Parallel Muting, Parallel Muting with 2 Sensors	Muting is the intended suppression of the safety function (e.g. light barriers). Specified for different configurations: 4 sensors and 2 sensors.
Enable Switch Enable Switch 2	Evaluates the signals of an enable switch with three positions. Also Version 2 without detection of panic position.
Safety Guard Monitoring	Monitors relevant safety guard (for two switches coupled with a time difference for closing the guard).
Safety Guard Interlocking with locking (two versions)	Controls an entrance to a hazardous area via an interlocking guard with guard locking ("four state interlocking"). Two different versions: Version 2, and one for switches with serial contacts.
Override	To move a product in the production line even when the sequential muting was aborted due to an error. This FB is only applicable in combination with a muting FB.
Safety Request	Provides the interface to a generic actuator, e.g. a safety drive or safety valve, to place the actuator in a safe state.
Out Control	Control of a safety output with a signal from the functional application and a safety signal with optional start-up inhibits.
External Device Monitoring	Controls a safety output and monitors controlled actuators, e.g. subsequent contactors.

Part 2 of the specification provides user guidelines to support the user with examples of these functionalities.

The FBs in Part 4 - Safety for presses are the following:

Foot Switch	Evaluates the signals of a foot switch with three positions.
Press Control	Controls the safety related process and enables the signal for the safety related valves, depending on the operation mode.
Single Valve Monitoring	Monitors the switching behavior of fluidic single valves, which confirm the safe state within a monitoring time by a static feedback signal (spool position monitoring).
Double Valve Monitoring	Idem for double valves (press safety valves). Both valves used should be of the same type.
Single Valve Cycle Monitoring	Idem for a cartridge valve in that manner, that per machine cycle a signal change of the spool monitoring signal must be recognized.

Valve Group Control	Summarizes all the connected valves to a group. In case of an error of one valve all valves are switched off.
Camshaft Monitor	This function block provides the camshaft monitoring functionality. There must be a number of signal changes in a specified time period.
Cycle Control	Controls the cycle mode (1 to n cycles) of an ESPE.
Cam Monitoring	Provides the cam monitoring functionality.
Two-Hand Multi Operator	Controls a press with two operator control stations. On each control station is one two-hand control device.
Two-Hand Con-trol Type III C	Provides the pluggable two-hand control functionality with fixed specified time difference of 500 ms.

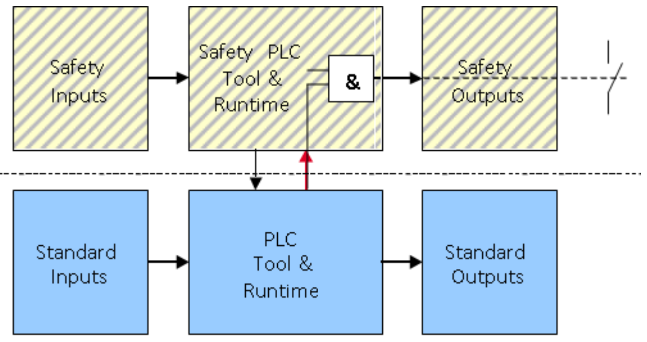
Implementations
The next step is implementing these Function Blocks on multiple platforms. At this level these FBs can be certified by independent organizations. With this, the standardized look and feel is supported on a broad set of software tools, reducing the tendency to create own functionality which can contain errors.

Ad C. Error Handling and diagnosis

For a transparent and unique diagnostic concept all safety-related Function Blocks have at least two error-related outputs: Error and DiagCode. These are provided for diagnostic purposes on the user application level, and not for diagnostics on the system/hardware level.
With a set of pre-defined diagnostics codes per Function Block it is guaranteed that uniform diagnostic information is available to the user, independent of the vendor's implementation. The internal status of the function block (State Machine) is indicated. An error is indicated via a binary output (Error). One can retrieve detailed information on the Function Block internal or external errors via the DiagCode.
Also the DiagCode can provide information when an operator interaction is needed, like a reset is needed or if the safety chain needs to be closed. For this the outputs ResetRequest and SafetyDemand are available in the relevant FBs.
A safety-related function has the highest priority, and after switching to the safe mode there is sufficient time for the diagnostics, either in the functional program or in the operator interface.

Combining the functional and safety application

By integrating the functional application with the safety application in the same environment, one can exchange data between both. This goes beyond the diagnosis functionality. It can include confirmation by an operator or it can give general status information. Of course different rules apply for going from the safety environment to the operational application then vice versa. Again, the safety related software tool helps to avoid errors here.



On the left side of the model, two sets of inputs are identified, and on the right side two levels of outputs. In the middle, the two environments are shown separately, both coupled to their related inputs and outputs.
Data exchange between the safety and the functional applications are shown in the middle. The functional application has unrestricted read access to the safety inputs. The non-safe signals can only be used in the safety application to control program flow and cannot be connected directly to the safe outputs. It is limited to the non-SAFE input parameters, such as a ‘reset’ and diagnosis, and limited to the AND function with a SAFEBOOL parameter for a safety related input. The same is valid for the two sets of outputs.

PLCopen Compliance information

For quick identification of compliant products, PLCopen has developed a logo for the Safety Specification:

Different levels of certification are applicable:
1. Certification of the software tool supplier, often part of the control supplier;
2. Certification of the application at the user / machine builder.

Ad 1: Certification of the software tool supplier
The development environment, including the safety related Function Blocks, as well as the underlying hardware, have to be certified by the relevant safety related bodies. In order to be able to be certified, certain rules, like described in IEC 61508 and related standards like IEC 61511, are applicable. The PLCopen specification provides a framework for this; however the overall requirements are beyond the scope of PLCopen, and have to be dealt with by external dedicated organizations.

Ad 2: Certification / Conformity of the application
Within an application, a certification includes the safety related software combined with the infrastructure, like sensors, switches and actuators, connection schemes, etc, like described in standards like IEC 62061. Certification of the application software is made easier; however the full application has to be dealt with by external dedicated organizations.

The use of the PLCopen logo does not give any guarantee about any compliance or fulfillment. The use of the logo just refers to the inclusion of the ideas and guidelines as described in this document, within the relevant software environment, and the availability of this information in more detail on the relevant section of PLCopen website www.PLCopen.org

Note:
Some motion related functionalities like SafeStop and SafelyLimitedSpeed are removed from Version 2.0 and moved to a separate specification dedicated to SafeMotion. This separate specification is also available from the PLCopen website.