Low-Cost Solution for Adding Safety Functions to Programmable Logic Controllers (PLCs)

EFSTATHIOS THEOCHARIS[1], MICHAIL PAPOUTSIDAKIS[1], ANDREW SHORT[1],
KONSTANTIA ZISIMOU[2]
[1]Deptartment of Industrial Design and Production Engineering,
University of West Attica,
Thivon 250 & P. Ralli, Egaleo,
GREECE

[2]Programming Teacher in a High School,
10th Gymnasio Nikaias, Artemidos 25,
Nikaia,
GREECE

Abstract: - Increasing requirements in automation and production make control systems more complex and vulnerable to failure. Breakdowns can cause delays in production, material damage, and above all, work-related accidents. For this reason, directives and legislation have been created at the country, European Union, and global levels that define the essential safety and requirements of industrial equipment. Guidelines must be observed by those involved in the design, supply, purchase, or use of industrial equipment in the European Union and several countries outside the European Union. Some guidelines (CAT, SIL) are created to ensure their safe operation in case of failure of both the hardware and the software. For a reliable operation of a fail safety system together with a system operating at SIL2 or SIL3, it must have Safety hardware and software.

Industrial equipment manufacturers are incorporating security features into a variety of devices. Depending on the Safety Integrity Level (SIL) requirements, these features can be used during the design phase to increase safety in the event of failures or malfunctions. With proper design, the process as well as its environment (including people) can be protected by entering a controlled safe state. Manufacturers have approached this problem in a number of ways, including adding redundant Central Processing Units (CPUs), using special hardware to interface input and output signals, and developing secure network protocols for communication. Unfortunately, these features cannot be added to existing machines, at least without upgrading some hardware. As the associated costs lead to slower adoption, manufacturers rely on previous work to support certain security features, notably CPU debugging. This is implemented in the form of software libraries that operate at a low level (logic gate), designed to run on older hardware (PLC) so that they can offer an increased level of security. This study analyzes the required guidelines and legislation that must be followed for the safe operation of a production unit. It describes the mechanisms that basic and specialized PLC systems utilize to ensure the safety of an automation system. The authors have developed algorithms to record behavior measurements of electronic equipment. By further analyzing the results, it is concluded that basic equipment can be reused in these systems to provide safety functions, at the same time conserving both cost and time.

Key-Words: - Mean Time Between Failure (MTBF), Safety Integrity Level (SIL), Safety Relay (SR)

Received: November 25, 2022. Revised: July 23, 2023. Accepted: August 20, 2023. Published: September 26, 2023.

1 Introduction

Machine manufacturers are increasingly focusing on improving the safety aspects of machinery and at the same time many countries strengthen the legislation and hold them accountable in case of injuries. For this reason, a plethora of ISO and IEC standards are available today. These standards simplify trade as they usually conform to the legislation of most countries.

In the field of industrial automation systems, Programmable Logic Controllers (PLCs) are commonly used to coordinate complex tasks and control machinery in an autonomous fashion. PLC manufacturers are building specialized products to allow their use in safety-critical applications or in industries where downtime is very expensive. These new products increase their reliability by adding redundant systems to help self-diagnose a malfunction among others in a sensing device (e.g., a peripheral input device), an external safety component (such as an emergency stop button), or the CPU itself.

These technologies are still relatively new, novel, and expensive. For this reason, they are only adopted in newer, more expensive equipment. Older industrial machinery could have possibly been built with more relaxed safety standards. It is also a possibility that current owners of older but still expensive machinery are reluctant to upgrade their equipment with newer hardware due to the associated costs with the hardware, programming, design, and installation.

The authors believe that current PLCs used in older equipment (and do not currently conform to newer safety standards) can support at least a subset of the newer features such as diagnosing routines with no hardware modifications. This research is motivated by the fact that current owners of older machinery would be willing to improve their safety scores by only applying minor software updates.

As part of ongoing research in this area, the authors have already proposed a method that allows current non-fail-safe devices to detect external sensor failure by employing techniques in software and have shown that the advantages are comparable to those of the newer fail-safe devices. Extending this research, the paper proposes a software algorithm that will allow current legacy software to be able to detect hardware failure at the CPU level, for the executing program to be able to shut down in a safe manner, and to predict hardware faults of PLC itself.

Researchers are actively figuring out techniques that allow PLC-operated machinery to function in more safe and reliable ways. The authors of a paper have proposed a solution for formal verification that uses mathematical models of the specific application scenario to offer improvements in both fail and non-fail-safe PLCs, [1], [2].

Another approach discussed by researchers aims to detect safety violations (caused by faults or attacks) by comparing data sets of event sequences and the time of occurrence with data traces collected beforehand in Industrial Control Systems (ICS).

Furthermore, researchers are studying the advances of the new safety devices, in terms of the diagnostic capabilities, implementation strategies, and metrics such as response times of these routines, [3], [4].

Although the motivation of the research presented in this paper is in line with other work presented above (i.e., to study and improve upon the safety operation of devices), the approach discussed here differs in the following ways:

a) The solution extends previous work to directly port features of newer (fail-safe) PLCs to older legacy hardware.

b) The algorithm does not depend on each specific application scenario.

c) An application use case has been included to present the function of the algorithm.

The rest of this paper is organized as follows: Section (2) analyses current strategies for improving safety and downtime records in terms of relevant standards. Section 3 describes the approach that current PLC manufacturers are using to detect hardware failures in modern fail-safe equipment. Section 4 proposes a solution in the form of an algorithm that is able to run on legacy devices and present similar advantages as the newer more expensive products. The section also describes the experimentation setup. Finally, Section 5 summarizes the results of the presented approach.

2 Redundant Systems

The redundant automation systems are commonly used to offer greater availability. The objective of these systems is to reduce the possibility of production interruptions, the protection of individuals, the protection of the surrounding environment, and the safe termination of production. In very critical applications such as refineries, airports, and nuclear plants, such systems are required not only to avoid the cost associated with stopping production but also to prevent accidents.

<u>Software Redundancy</u>

In many applications, the requirements for fault-tolerant cannot be justified. Simple software mechanisms are sufficient to allow a failed process to continue on a substitute system if an error occurs. These mechanisms can be applied to control processes that can tolerate larger transition delays to a surrogate system, e.g., in waste-water plants, water treatment plants, or traffic streams.

<u>Hardware Redundancy</u>

Hardware Redundancy consists of two subsystems that are synchronized via fiber optic cables.

Both subsystems create a fault-tolerant automation system that works with two channels and is based on the principle of active redundancy. Active redundancy means that all redundant resources are continuously running and simultaneously participating in the execution of the control task, This means that the programs on both CPUs are identical and are executed synchronously by the CPUs.

To separate the two subsystems, the traditional expressions "master" and "reserve" are used for two-channel fault-tolerant systems. The reserve always processes events in sync with the master and does not wait for any errors from the master to start processing. The distinction made between the master and the reserve CPU is very important to ensure reactions in case of errors. For example, the reserve CPU can go into STOP when the backup link has an error, while the primary CPU remains in RUN mode.

The higher the risks and the cost of abruptly terminating the production, the more beneficial it is to use such systems, [5].

The evaluation of the Redundant systems is usually based on reliability and availability parameters. A commonly used reliability measurement is the MTBF (Mean Time Between Failure). This can be statistically analyzed on the basis of the parameters of the ongoing operation systems or by calculating the failure rates of the individual components.

The Mean Detection Time (MDT) of a system is determined by the times below:

- Time required to detect the error.
- Time is required to find the cause of the error.
- Time is required to encounter the problems and restart the system.

The MDT system is calculated based on the MDT of the individual components in the system. The structure in which the components form the system is also a part of the calculation.

2.1 Correlation between MDT and MTBF

The MDT value is of high importance for maintaining the quality of the system. The most significant factors are:

- Qualified personnel
- Efficient logistics
- High-performance tools for debugging and recognizing identification
- A good repair strategy

Figure 1 shows how the MDT depends on the factors mentioned above.

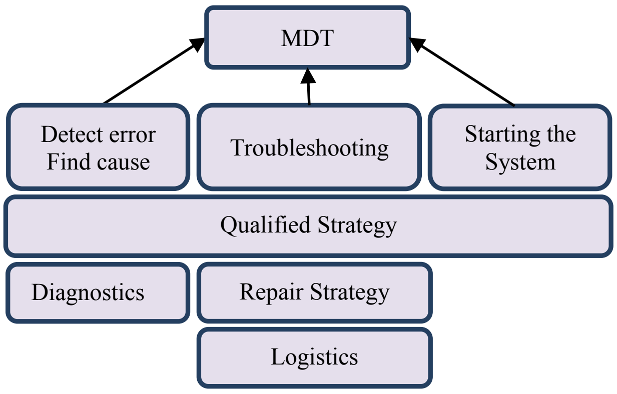


Fig. 1: MDT Relationships

The MTBF, or Mean Time Between Failures, is a critical metric used to assess the reliability of a system. It is calculated by dividing the total operational time by the number of failures that occur during that time period. Figure 2 shows the parameters included in the MTBF calculation of a system, [6].

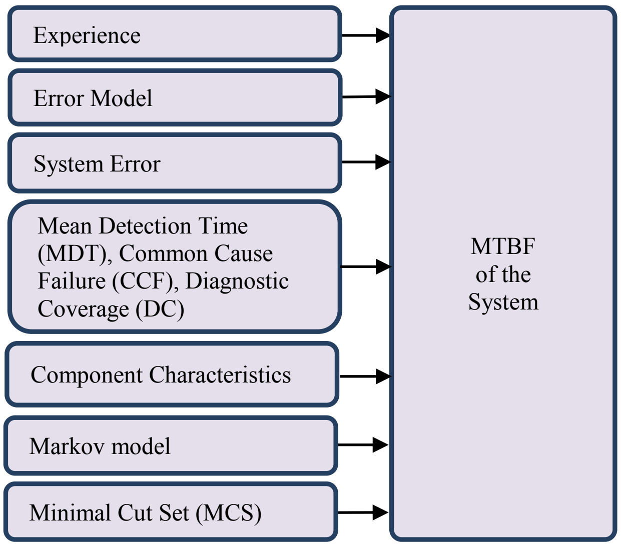

Experience
Error Model
System Error
Mean Detection Time (MDT), Common Cause Failure (CCF), Diagnostic Coverage (DC)
Component Characteristics
Markov model
Minimal Cut Set (MCS)
MTBF of the System

Fig. 2: MTBF

The analysis is based on the following assumptions:

- The failure rate of all the components and all the calculations is based on an average temperature of 40°C.
- All the spare parts are locally available, to prevent extensive repair times due to the lack of spare parts. This maintains the MDT element to its minimum.
- The MDT of individual spare parts is 4 hours. The MDT of the system is calculated based on the MDT of the individual components plus the structure of the system.
- The MTBF of the components meets the standards:

- SN 29500 (This prototype is compatible with MIL - HDBK 217 - F)
- IEC 60050, [7]
- IEC 61709, [8], (These calculations are made by using the diagnostic coverage of each component)

- Depending on the formulation of the system, the CCF coefficient ranges between 0,2% and 2%.

By using the Redundant units, the MTBF is greatly increased. They contain high-quality self-monitoring features that allow the identification of almost all errors with a reliability of at least 90%. Reliability in autonomous operation is defined by the corresponding failure rate. Reliability in Redundant operation is defined by the failure rate of the relevant components or otherwise "MTBF". The combinations of any failed components that cause a system failure are described and calculated using Markov models.

Availability is the probability that a system is operating at a given point in time. It can be improved by using Redundant I/O units. The Redundant components are arranged so that the functionality of the system is not affected by the failure of an individual component. The availability of a system is expressed as a percentage and is defined by the average time between failure (MTBF) and the average MDT repair time, [9]. The availability of a two-channel fault tolerance system (1 out of 2) can be calculated using the following formula as seen in Figure 3.

$$V = \frac{MTBF_{1v2}}{MTBF_{1v2} + MDT} * 100\%$$

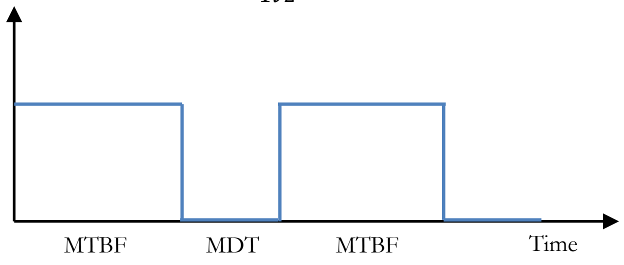


Fig. 3: Availability

A control system can have CPU Redundancy, Network Redundancy, I/O Redundancy, Sensor Redundancy, and PID bumpless Switchover capabilities. There are two similar CPUs in a Redundant system that execute the same code at the same time. These two CPUs synchronize through a special interface, often by means of a redundant connection. The system includes synchronization units that support Hot-Swapping, so in case one of the CPUs fails, the other CPU retains the procedure control and the production process can resume without any anomalies. For security and functionality purposes, a CPU can be up to 10 meters away when the synchronization units communicate through copper and up to 10km when they communicate through optical fiber cables. A standard configuration of a Redundant system is shown in Figure 4.

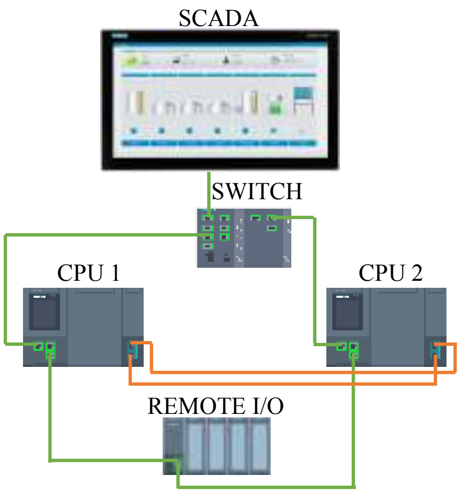


Fig. 4: Redundant Systems PLC, [10], [11]

The system consists of one Supervisory Control and Data Acquisition (SCADA), used as the interface for the machine operator, two CPUs with Redundant functions, an interface module (usually Inputs and Outputs) to operate the machine, and the required contact cards. The operation control of the machine is implemented as follows:

The two CPUs communicate with each other with a double connection (for Redundant purposes) through optical fibers. The communication is required so that one CPU can sense the state in which the other CPU is and also for their synchronization. That means that the execution of the machine's control code is always synchronized. One CPU is always set and operated as Master and the other one as Standby. The Master CPU is the one which controls the elements of the machine through output cards. Both CPUs scan the input cards, process them individually, and then determine what step of operation the machine is in. When the Standby CPU detects an error on the Master CPU, then it automatically goes into Master mode and assigns the other one into Standby mode. The control-switching time from one CPU to the other is usually less than 5ms. Although the SCADA communicates directly with both CPUs, any updates and operations are implemented by the Master CPU, [12], [13], [14].

3 Safety Technology

Safety Technologies have been developed for the safer operation of the machine, [15], [16], [17], as well as to avoid any accidents and deterioration, with the most basic ones being: (1) Contractual Safety Technology, and (2) Integrated Safety Technology.

3.1 Contractual Safety Technology

The safe operation of the machine is being monitored by a Safety Relay. Contacts of the Stop Emergency buttons, Safety Doors, etc., are wired to the Safety Relay and regardless of the commands given by the PLC automation system, the machine switches to Safety mode once a Stop Emergency has been pressed or a Safety Door has opened.
The wiring as well as the architecture of the protection functions are applied in accordance to EN 61508, [18], in SIL 3 or according to EN 954 in Cat. 4: The Stop Emergency button and the Safety Door positional switches are connected by two channels in the Safety Relay, [19], [20]. Two contacts are connected in series and are used to monitor the safe operation of the machine, [21].

3.2 Integrated Safety Technology

In complex automation systems, the safety functions of machinery are controlled by a Safety PLC (instead of a Safety Relay) that usually dictates an alternative way of safely stopping production, according to the issue arising each time, [22], [23], [24]. There is a separate safety program and special Safety Input and Output units in the Safety PLC, [1]. As soon as a wiring error occurs the emergency button is pressed or the safety door opens, the machine goes into Safety mode. The wiring of the actuators and the sensors related to safety are being monitored by these special input and output units of the PLC.

According to SIL 3 (EN 62061) Cat.4 (EN 954), the wiring as well as the architecture of the protection functions is similar to that of the Safety Relay. Stop emergency and safety doors are connected to the special PLC cards through two carriers, [24].

Safety Integrated Technology is the completely integrated security concept for automation and power units with proven technologies and automation systems used to safely operate the production. Additionally, it covers the entire security chain with sensors and actuators to the controller, including the security-related communication over the standard networks, [25]. Besides the automatic operation of the production, the PLC also undertakes its safe operation, [26]. In addition to ensuring reliable safety functions, Safety Integrated Technology also offers a high level of flexibility and productivity, [27].

The integration of a safe and automatic operation in a PLC has the following significant advantages:

- Greater flexibility than the electromechanical solutions
- Wiring reduction
- Only one CPU is required due to typical and safety program coexistence
- Simple communication between the typical and safety program
- Less implementation time of application

For dual operation (Safety and autonomous) of the system, the PLC utilizes a CPU that simultaneously executes the program for both the autonomous and safety operation (Figure 5). It provides standard Input and Output cards for the autonomous operation and special cards for the Safety functions. The main difference between Safety and standard cards is that the Safety cards have been constructed with two channels internally, two integrated processors monitor one another and automatically control the input and output circuit. In case of an error, they set the Safety card in safe mode. The digital Safety input cards acquire (by detecting) the signal status of sensors that are related to a safety status (e.g., emergency stop button pressed), perform short circuit and cross-circuit tests, as well as inconsistency analysis, and send the corresponding messages to the CPU. The digital Safety output cards are suitable for deactivation tasks and have additional capabilities to detect short-circuits of the actuator.

For the verification of the reliability of the outputs, tests are carried out at certain intervals. The tests are usually repeated every 1,000 sec when we don't want to have rapid wear of the actuators, and every 100 secs for fast fault detection. Retry time is the maximum time after the output is turned off that a feedback signal should be detected before it is declared as a short circuit fault. The repetition time must be set long enough, especially when switching capacitive loads, to allow the switching capacitance to discharge within the repetition time.

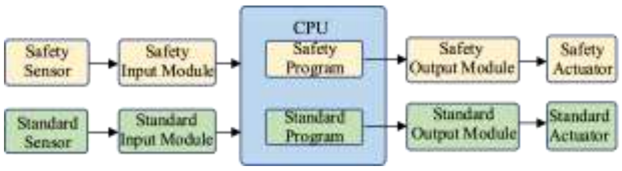


Fig. 5: Standard and Safety Operation in a PLC, [28]

The standard and safety programs are both developed in the same programming environment and because they co-exist in the same CPU, they can communicate with each other using variables without one affecting the other in case of an issue, [29].

3.3 Safety Program Operation

Safety-related CPUs work according to the principles of redundancy and diversification, which allow the implementation of safety systems with only one CPU and one processor. With the security program, the user programs the logic of the system's safe operation, and the operating system automatically creates additional blocks based on "differentiated" logic compared to the user's program as well as "differentiated" operators and functions.

The two parts of the security program are run sequentially and the results are compared. If an error occurs, the CPU reacts and puts the system into a safe state. The operating system also creates system blocks that can be used for example to manage Safety communication.

Error checking can help in the following circumstances:

- Old messages that have not been updated are sent again at the wrong time.
- A message is not received or recognized.
- A message referring to an unknown source is entered.
- The specified sequence (e.g. CRC, time references) of the messages of a particular source is incorrect.
- Messages can be corrupted due to errors at a bus node, i.e. errors in the transmission medium or due to mutual interference of messages.
- Messages can be delayed beyond the allowed arrival time, e.g. as a result of errors in the transmission medium, overloaded connection cables, mutual interference, or bus nodes (devices) sending messages in such a way that services are delayed or unrecognized.
- A message originating from a valid source is additionally inserted. Thus, a non-security-related message may be received by a security-related device, which then classifies it as security-related.
- FIFO (First-In-First-Out) error, the correct order of data is not observed.

In the following example, we will demonstrate the operation of a safety function while it is executing inside the CPU. The internals of the function are shown in Figure 6. A and B as used as the input signals. The objective is to run a logic operation on the inputs while at the same time detecting any CPU anomalies. If the result of the logic operation is (Boolean login) TRUE, and there are no faults detected, then the normal operation can resume. In any other case (i.e., the result of the logic operation is FALSE, or CPU malfunction is detected) the PLC should halt the execution of other instructions. In order to detect CPU faults, the logic operation is executed twice. First, the normal logic operation is carried out and its output is kept temporarily (variable C shown in Figure 6). Then, a suitable inverted logic operation is constructed which in turn results in output D. This inverted logic is implemented in safety by PLC manufacturers using certified mechanisms without the possibility of intervention by the user, [30]. The two intermediate outputs (C and D) are then compared and are expected to contain the same value. In the presence of a fault, it would be extremely unlikely that both functions would present the same erroneous response, therefore eradicating false negatives. In case the two outputs are different, the CPU enters safety mode status.

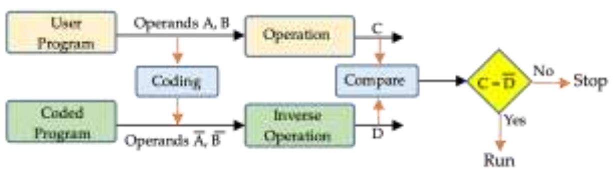


Fig. 6: Safety Operation

In the following example (Figure 7), we demonstrate how we can verify the output of a software OR gate using an inverse operation (AND gate). By comparing the output of two separate computation results, we add a layer of reliability and trustworthiness to the CPU results. Consequently, the CPU is able to detect whether it has malfunctioned and can disable itself considering that it is safer to stop a process rather than continue running it with faulty equipment. We assume that in this case, the user code intends to run an OR instruction and that A and B have values of 1001 and 1010 (in binary representation) respectively.

The expected output C of the OR operation should be 1011. In this scenario, the inverse operation used is an AND gate. The first step is to compute the inverse of A and B ($\overline{A}, \overline{B}$), which are 0110 and 0101 respectively. The second step is to compute the output of the AND gate D, which is 0100. Last, we need to compute the inverse of the output D ($\overline{D}$) which turns out to be 1011. In this case C = $\overline{D}$ so the CPU would decide to continue running. In any other case, it would enter a stopped mode.

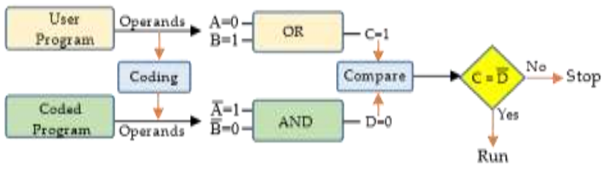


Fig. 7: Safety Operation Example

4 Development of a Fail Safety Algorithm for use in basic PLC Hardware

The solutions demonstrated in the previous section require specialized hardware and expert knowledge, factors that prohibit their wide adoption, especially for smaller projects. For these reasons, an algorithm was designed and developed which can be used with ease and without any particular expertise, that leads to the increased reliability of an automation system. The concept of the proposed solution is that by executing a logic operation multiple times may lead to the detection of a transient fault. Once the operation has finished executing multiple times, its output of each iteration is compared to the previous one. If the output of each iteration is not identical, the CPU would stop further code execution. Figure 8 illustrates the operation of the algorithm.

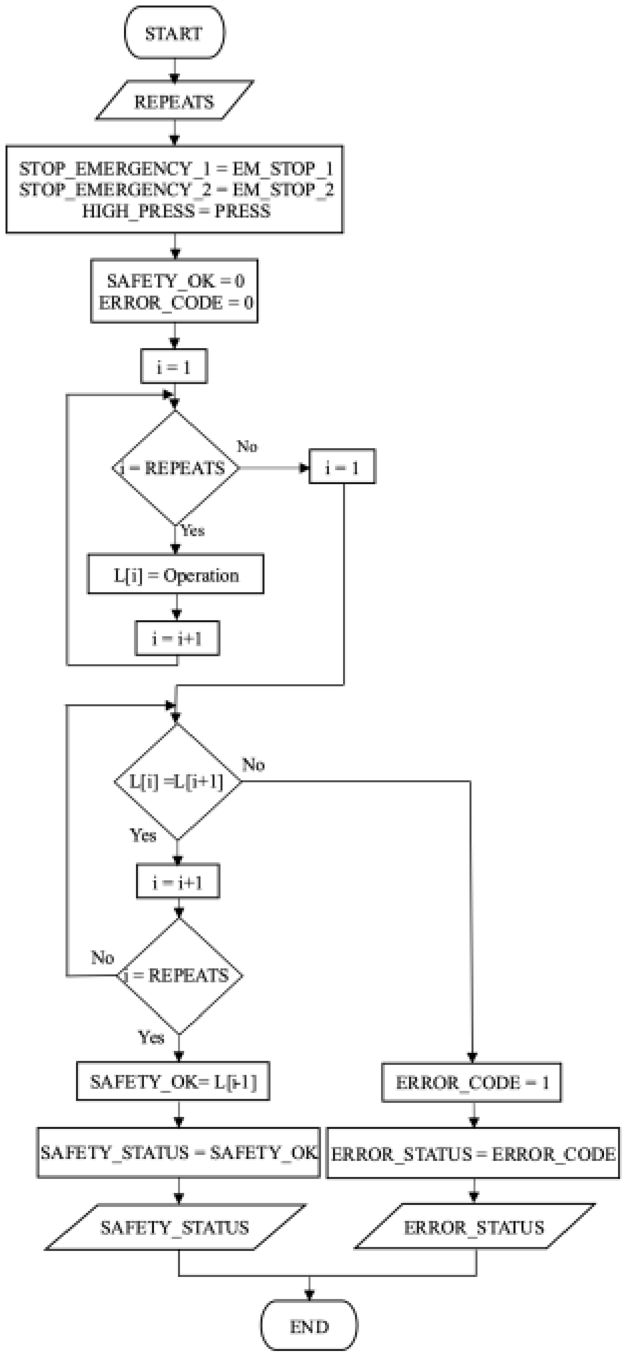


Fig. 8: Algorithm Operation

Algorithm Description
The operation of the algorithm is as follows:
1. The desired code execution iterations (n) is entered.
2. The variables (inputs and outputs) of the code are entered (IN1, IN2, OUT1, OUT2, etc.).
3. The code is executed 'n' times and in each execution the results of the code (Outputs) are stored in a table.
4. After the execution of the code, the results found in the table are checked and if they are the same then they are applied to the respective outputs. In

any other case, an error message is output and the system switches to safe mode.

4.1 Overview of the Experiment

For the evaluation of the algorithm, an experiment was carried out using industrial equipment as the automation (Figure 9).

PLC

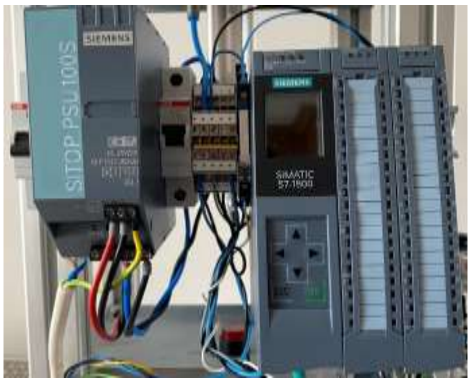


Fig. 9: Experiment Equipment

The CPU consists of a SIEMENS CPU 511, one of the smallest (in terms of CPU capabilities) of the mid-range provided by SIEMENS, and has built-in input and output signals. The safety function was applied to an automation system used for pallets filled with boxes. The proposed algorithm in the previous section was incorporated into a pre-existing pallet stacking process. The safety functions of the system utilize two emergency stop buttons (Normally Closed) as well as the pressure feedback from the piston (by means of a pressure switch). In case of an obstacle in front of the piston area, the pressure will increase above a pre-set limit and will in turn activate a digital input (Normally Closed). As long as the emergency stop button is not pressed and the pressure switch has not been activated, the safety algorithm should allow the execution of the code responsible for pallet stacking and should energize the system's outputs responsible for actuators. In any other case, the actuators should become inactive and the machine operation would consequently stop. Figure 10 shows the machine's components.

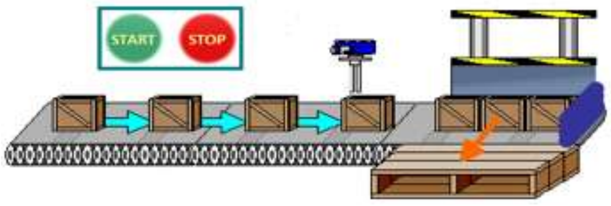


Fig. 10: Disposition of the Machine

For the implementation of the experiment, the TIA V16 program was used from which the FB10 routine (Function Block) was developed in SCL language. The code of the routine and its explanation are given below.

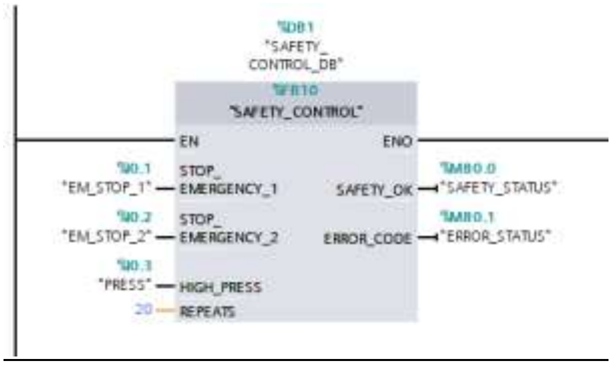


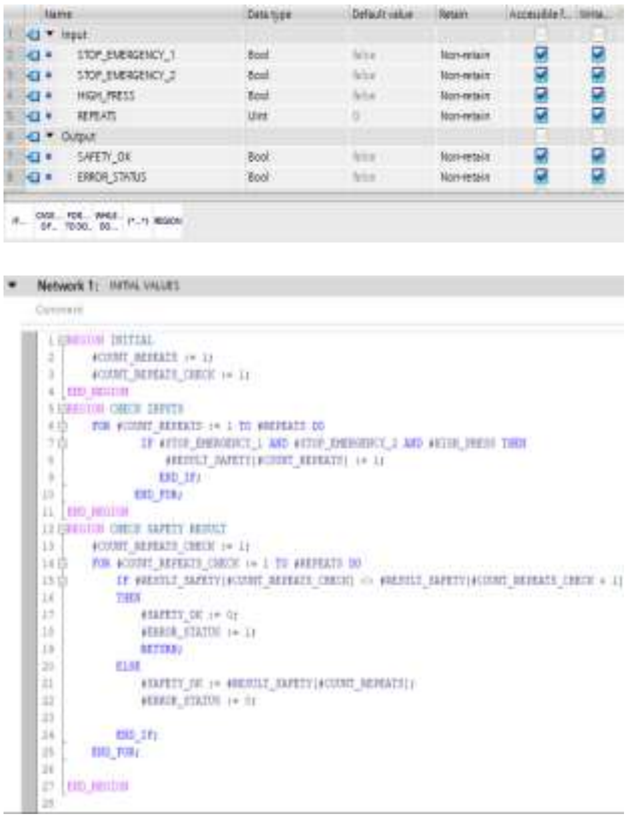
Fig. 11: OB1

OB1 is the Main Block which is executed by a CPU and contains the user code. The newly added function (FB10 shown in Figure 11) is declared along with the addresses of the actual Inputs and Outputs.

Please note that in the following algorithm, all variables other than the ones explicitly mentioned are temporary, have a start value of 0, and are released from system resources when the algorithm ends its operation. It is also worth mentioning that the algorithm is not able to determine the origin of the fault (either of the digital inputs or the CPU) but only the presence of the anomaly.

The number of repetitions can be configurable. A higher number of repetitions increases the reliability at the cost of processing overhead. In terms of the "Big O" notation, the algorithm executes in linear time O(n), where n relates directly to the number of repetitions.

Algorithm 1 – Checking for the existence of CPU faults or Input data mismatch.
Inputs – STOP_EMERGENCY_1, STOP_ EMERGENCY_2 are inputs from the external safety switch (Boolean logic), HIGH_PRESS is the output of the pressure switch on the piston (Boolean logic), REPEATS configures the number of repetitions (configurable).
Outputs – SAFETY_OK = 1 when the safety checks are successful. Similarly, ERROR_STATUS = 1 under faulty scenario.



	Name	Data type	Default value	Retain	Accessible f...	Writa...
1	▼ Input					
2	STOP_EMERGENCY_1	Bool	false	Non-retain	✓	✓
3	STOP_EMERGENCY_2	Bool	false	Non-retain	✓	✓
4	HIGH_PRESS	Bool	false	Non-retain	✓	✓
5	REPEATS	Uint	0	Non-retain	✓	✓
6	▼ Output					
7	SAFETY_OK	Bool	false	Non-retain	✓	✓
8	ERROR_STATUS	Bool	false	Non-retain	✓	✓

IF... CASE... OF... FOR... TO DO... WHILE... DO... (*...*) REGION

Network 1: INITIAL VALUES

```
REGION INITIAL
    #COUNT_REPEATS := 1;
    #COUNT_REPEATS_CHECK := 1;
END_REGION
REGION CHECK INPUTS
    FOR #COUNT_REPEATS := 1 TO #REPEATS DO
            IF #STOP_EMERGENCY_1 AND #STOP_EMERGENCY_2 AND #HIGH_PRESS THEN
                #RESULT_SAFETY[#COUNT_REPEATS] := 1;
               END_IF;
          END_FOR;
END_REGION
REGION CHECK SAFETY RESULT
    #COUNT_REPEATS_CHECK := 1;
    FOR #COUNT_REPEATS_CHECK := 1 TO #REPEATS DO
        IF #RESULT_SAFETY[#COUNT_REPEATS_CHECK] <> #RESULT_SAFETY[#COUNT_REPEATS_CHECK + 1]
        THEN
            #SAFETY_OK := 0;
            #ERROR_STATUS := 1;
            RETURN;
        ELSE
            #SAFETY_OK := #RESULT_SAFETY[#COUNT_REPEATS];
            #ERROR_STATUS := 0;

        END_IF;
    END_FOR;

END_REGION

```

Fig. 12: FB10

The implemented source code in Structured Controled Language (SCL) is depicted in Figure 12. The routine parameters are initially stated in the parameter table. Additionally the code is categorized in the following logical sections:
Initial: This is where the iteration variables are initialized.
Check Inputs: This is where the operation of production is checked if it is safe or not.
Check Safety Result: This is where we check if the Safety status was implemented correctly.

5 Conclusion

The paper builds upon earlier research regarding methods for executing safety routines on legacy hardware and describes the operation of current fail-safe systems in the context of PLCs. More specifically, a high-level approach has been presented that allows monitoring the PLC itself while it is running safety critical routines inside the user program. The routines allow for the creation of PLC algorithms that replace low-level logic operations that in addition to the calculation output are also able to decide whether the CPU is faulty and consequently safely stop execution. The operation relies on an approach of calculation repetition and inverse operations, in effect sacrificing some CPU resources for the ability to be able to self-diagnose some types of CPU-related hardware failures. Furthermore, the proposed solution has been implemented in an actual pallet stacking machine to collect results and provide implementation information. The research concludes that legacy PLC hardware can make use of our proposed approach to offer advantages such as fault detection capabilities and consequently increased safety during the automation process without any additional hardware.

Most importantly, due to the minimum impact of changes involved, it is anticipated that the proposed solution can increase adoption and therefore the safe operation of existing machinery since it requires less costs and downtime. As further work, the authors of the paper will examine the impact of the execution of such routines in terms of CPU performance overhead as well as the feasibility of the execution of distributed systems (for systems requiring a higher level of redundancy).

Acknowledgments:
All authors would like to thank the University of West Attica and specifically the Post Graduate Program of Studies (MSc) "New Technologies in Shipping and Transport", for the support provided to them to undertake this research project.

References:

[1] K. Rástočný, J. Ždánsky, J. Balák and P. Holečko;, "Effects of diagnostic on the safety of a control system realised by safety PLC", vol. 7512118, 2016.

[2] Dániel Darvas, István Majzik and Enrique Blanco Viñuela, "Formal Verification of Safety PLC Based Control Software", *Springer*, vol. 9681, 2016.

[3] Hiroo Kanamaru, Tsuyoshi Mogi and Naoki Aoyama, "Functional safety application using safety PLC", *SICE Annual Conference*, vol. 4421408, pp.2489-2492, 2007.

[4] Rástočný, J. Ždánsky and K., "Influence of safety PLC parameters to response time of safety functions", in *International Conference on Applied Electronics*, 2013.

[5] Allen Bradley, "literature.rockwellautomation.com," [Online]. Available: https://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um523_-en-p.pdf. [Accessed 12 2021].

[6] International Standardization Organization, "www.iso.org," Standardization, International Organization, 12 2015. [Online]. Available: https://www.iso.org/standard/69883.html. [Accessed 12 2021].

[7] International Electrotechnical Commission, "International Electrotechnical Commission (IEC) 60050-114:2014/AMD1," 2017.

[8] International Electric Components Standard, "webstore.iec.ch," 17 02 2017. [Online]. Available: https://webstore.iec.ch/publication/59985. [Accessed 12 2021].

[9] Attila Hilt, Gabor Jaro amd Ostvan Bakos, "Availability Prediction of Telecommunication Application Servers Deployed on Cloud P", *Periodica Polytechnica, Electrical Engineering*, DOI: 60.72-81.10.3311/PPee.9051., 2016.

[10] SIEMENS, "Siemens S7-1500 Technical Slides," SIEMENS, 2019. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:8ad3e246-011b-4206-aa85-1ec60aac51ac/version: 1562856005/simatic-s7-1500-redundant-systems--techslides-2018-11-12-en.pdf. [Accessed 05 2022].

[11] Schneider-electric, "https://download.schneider-electric.com," Schneider-electric, 11 12 2018. [Online]. Available: Redundant Control and Communication for Power Control Systems (whitepaper) - https://download.schneider-electric.com/files?p_enDocType=White+Paper&p_File_Name=asc-pcm-wp-redundant-control.pdf&p_Doc_Ref=PCM-WP-REDCONT. [Accessed 6 2023].

[12] Nguyen Tuan Hung and Truong Dinh Chau, "An Application Solution for PLC Redundancy in Distributed Control System", *International Symposium on Industrial Electronics*, 2011.

[13] G. Gabor, D. Zmaranda, C. Gyorodi and S. Dale, "Redundancy method used in PLC related applications", *3rd International Workshop on Soft Computing Applications*, DOI: 10.1109/SOFA.2009.5254867., pp.119-126, 2009.

[14] D. J. Rankin and J. Jiang, "A Hardware-in-the-Loop Simulation Platform for the Verification and Validation of Safety Control Systems"," *IEEE Transactions on Nuclear Science* , vol. 58, pp.468-478, 2011.

[15] D. Darvas, I. Majzik and E. Blanco Viñuela, "Formal Verification of Safety PLC Based Control Software," *Integrated Formal Methods*, vol. 9681.

[16] V.A. Gapanovich, E.N. Rozenberg and I.B. Shubinsky, "Some Concepts of Fail-Safety and Cyber Protection of Control Systems", DOI: 10.21683/1729-2646-2014-0-2-88-100, 2014.

[17] J. Ždánsky and J. Valigurský, "Application diagnostic of distributed control system with safety PLC", *ELEKTRO*, DOI: 10.1109/ELEKTRO.2018.8398311, pp.1-6, 2018.

[18] IEC Standards, "webstore.iec.ch," 30 4 2010. [Online]. Available: https://webstore.iec.ch/publication/5517. [Accessed 3 2023].

[19] Heidi Hartmann, Dr. Eric Scharpf and Hal Thomas, "Practical SIL Target Selection - Risk Analysis per the IEC 61511 Safety Lifecycle", ISBN: 978-1-934977-03-3.

[20] E. Theocharis, M. Papoutsidakis, A. Sort and C. Drosos, "Experimentation on the Electromechanical Behavior of Automation Safety Buttons Applied to an Industrial PLC" WSEAS Transactions on Systems and Control, ISSN / E-ISSN: 1991-8763 / 2224-2856, Volume 15, 2020, Art. #74, DOI: 15. 10.37394/23203.2020.15.74, 2021.

[21] Mitsubishi Electric, 01 2008. [Online]. Available: http://dl.mitsubishielectric.com/dl/fa/document/catalog/plc/l08117eng/l08117enga.pdf. [Accessed 2023 07].

[22] Feng Wang, Ou Yang, Ruibo Zhang and Lei Shi, "Method for assigning safety integrity level (SIL) during design of safety instrumented systems (SIS) from database”,", *Loss Prevention in the Process Industries*, 2016.

[23] D. Smith and K. Simpson, *The Safety Critical Systems Handbook”, Elsevier Ltd,* ISBN: 9780128207000., 2011 .

[24] K. Rástočný, J. Ždánsky and J. Hrbček, "The Problems Related to Realization of Safety Function with SIL4 Using PLC", *Cybernetics & Informatics (K&I)*, DOI: 10.1109/KI48306.2020.9039878, pp.1-5, 2020.

[25] J. Ždánsky and J. Valigurský, "Time response of safety function realised by decentralised SRCS with safety. PLC", *International Conference on Applied Electronics (AE)*, DOI: 10.23919/AE.2018.8501425, pp.1-4, 2018.

[26] G. Buja and R. Menis, "Dependability and

Functional Safety: Applications in Industrial Electronics Systems", *IEEE Industrial Electronics Magazine,* vol. 6, pp.4-12, 2012.

[27] Abdullah Al Farooq Jessica Marquard, Kripa George and Thomas Moyer, "Detecting Safety and Security Faults in PLC Systems with Data Provenance", *IEEE International Symposium on Technologies for Homeland Security,* 2019.

[28] plcopen.org, "https://plcopen.org," 2018. [Online]. Available: https://plcopen.org/system/files/downloads/ plcopen_safety_part_1_version_2.01.pdf (Accessed 12/2021). [Accessed 01 2022].

[29] M. M. a. J. Ždánsky, "Safety PLC Programming Based on UML Statechart," *ELEKTRO,* DOI: 10.1109/ELEKTRO49696.2020.9130307, pp.1-5, 2020.

[30] Younju Oh, Junbeom Yoo, Sungdeok Cha and Han Seong Son, "Software safety analysis of function block diagrams using fault trees, Reliability Engineering & System Safety", vol. 88, no. 3, pp.215-228, 2005.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- Efstathios Theocharis has constructed the demo unit and implemented the PLC - SCADA code.
- Michail Papoutsidakis carried out the simulation and the optimization of the experiments.
- Andrew Short has organized and executed the experiments.
- Konstantia Zisimou was responsible for the Reports and the requirement for their repetition.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
All authors would like to thank the University of West Attica for the financial support provided to them to undertake this research project.

Conflict of Interest

The authors have no conflicts of interest to declare.