# Realizing Safety Applications with the Industrial PC

An increasing number of modern automation systems are using high-performance Industrial PCs to execute control tasks. Until now, their performance capabilities could not be used for the execution of safety functions. The IPC's complexity prevented safety proofs in accordance to the internationally accepted safety standards if conventional control solutions were simply transferred to the IPC. By using mathematical codes, it is now possible to execute safety functions with the IPC according to SIL 3.

Enormous skepticism surrounded the introduction of the first PC-based control systems in the late 1980s. This was understandable as every user had negative experiences with regular office PCs, and did not want to introduce their (un-)reliability to manufacturing operations. Although the IPCs already exhibited a high standard of quality even at that time and have since become better and more powerful, a trace of this skepticism can still be felt today. Therefore, it may still astonish certain users that IPCs can now be used in safety applications.

**Why IPC-based safety?**

The use of IPCs for safety functions is mainly based on the high quality of modern IPCs and the firmware tailored to them. The successful development of fail-safe I/O terminals and the high standard of fail-safe communication, such as Safety over EtherCAT, contribute to their successful usage.

But how should calculations on a PC-based control system with provable safety be made? The tasks of fail-safe automation systems are to detect errors with proven high probability and to deploy a safe reaction via peripheral devices. In most cases, redundant hardware and software channels are used to detect errors. Results are regarded as correct and further used only if the channels supply consistent data.

Moreover, in order to comply with international safety standards, it must be proven that there is no single error that affects several channels in the same way, so that a comparison is successful and the error cannot be detected that way (so-called common cause). It is precisely this characteristic that must be validated very critically when using different cores of multi-core processors, since many resources are shared by some or even all of the cores. It is much more efficient to use mathematical methods that also support the use of multi-core processors in which the distribution of the sub-tasks to the individual cores and the type of usage of common resources are not safety features that need to be verified.

The combination of several software channels with different coding, which is easily possible on an IPC, increases the quality of the error detection and thereby considerably lowers the residual error probability (see fig. 1). Although the size of the necessary memory multiplies (longer data types, larger number of operations per channel, redundant channels) and the processing time potentially increases due to the fact that the software channels are actually processed successively – these potential disadvantages are insignificant if high-performance IPCs are used.

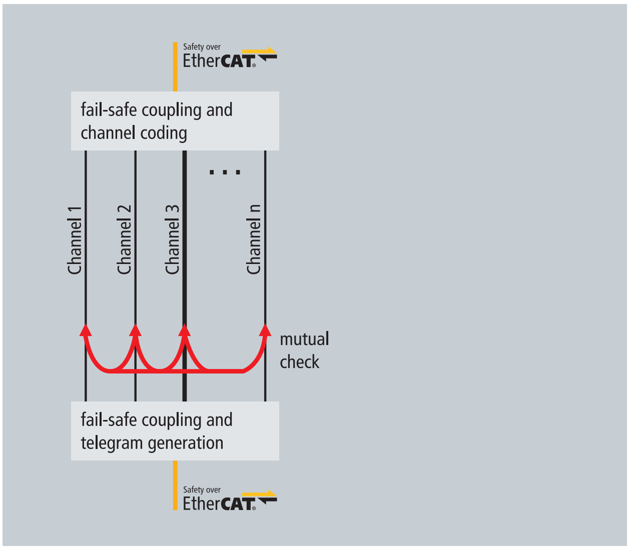

**Fig. 1: Architecture of the fail-safe IPC**

## Basic principle of the fail-safe Industrial PC

In addition to redundancy – multiple executions of identical or similar tasks – data redundancy plays an increasingly larger role. This kind of redundancy can also be considered as mathematical coding: Transformed data are generated from the original data in a new, much larger range of values by means of coding rules. If data assumes values that are inconsistent with the coding rules, even if they lie within the correct range of values, they do not belong to the code. The data are therefore invalid and can only be a result of errors in the storage or processing of the data.

The arithmetic codes used here are based on prime numbers and were first described in detail in [1]. In the publication, the original data are multiplied by a prime number in an upstream unit, e.g. in special safe sensors, and a specific offset is added. If no multiple of this prime number remains after subtracting the offset, an error is detected. Hence, [1] even describes a single-channel solution (single-channel hardware and single-channel software).

The value of a coded variable $x_c$ can thus be represented as a product of the original value $x_f$ and a prime number $A$ (see [1, 5]):

$$x_c = A \cdot x_f$$

Then, a variable-specific offset $B_x$ can be added to the coded value, so that

$$x_c = A \cdot x_f + B_x$$

applies. $B_x$ is also called the static signature. Now, there are no longer only multiples of the prime number $A$ located in the memory; the value $B_x$ , depending on the memory location, must be subtracted before a multiple of $A$ results.

An additional offset $D_t$ containing the respective processor cycle, called the dynamic signature, completes the coding:

$$x_c = A \cdot x_f + B_x + D_t.$$

With the help of the dynamic signature, the incorrect use of outdated values is now detectable. The error detection is explained below using an addition as an example.

In addition to the coded variable $x_c$ , the variable $y_c$ , with

$$y_c = A \cdot y_f + B_y + D_t$$

is used. The result $z_c$ of the coded addition follows the corresponding composition

$$z_c = A \cdot z_f + B_z + D_t.$$

The coded addition must be executed as follows:

$$z_c = x_c + y_c + (B_z - B_x - B_y) - D_t.$$

The value $(B_z - B_x - B_y)$ is a constant that contains information about the variables involved and the kind of operation. For example, if the offsets $B_x$ , $B_y$ and $B_z$ are specified as 1093, 5012, and 8913 respectively, then 2808 is to be found at the appropriate place in the coded program. The dynamic signature must be subtracted here since each coded variable $x_c$ and $y_c$ contains exactly one $D_t$ and the correct coded sum $z_c$ also contains exactly one $D_t$.

A simple, single-channel check of correctness for $z_c$ tests only the validity of the value:

$$(z_c - B_z - D_t) \bmod A == 0?$$

Falsifications of the values $x_c$ and $y_c$, the reading of incorrect memory locations, the use of outdated values and errors of the arithmetic unit can thereby be detected. The probability that an error is not detected (i.e. the residual error probability) is calculated by $1/A$ (see e.g. [3]). The distance between all valid values is $A$. All values lying in between are identified as being erroneous (see fig. 2). $A$ must therefore be selected as large as possible.

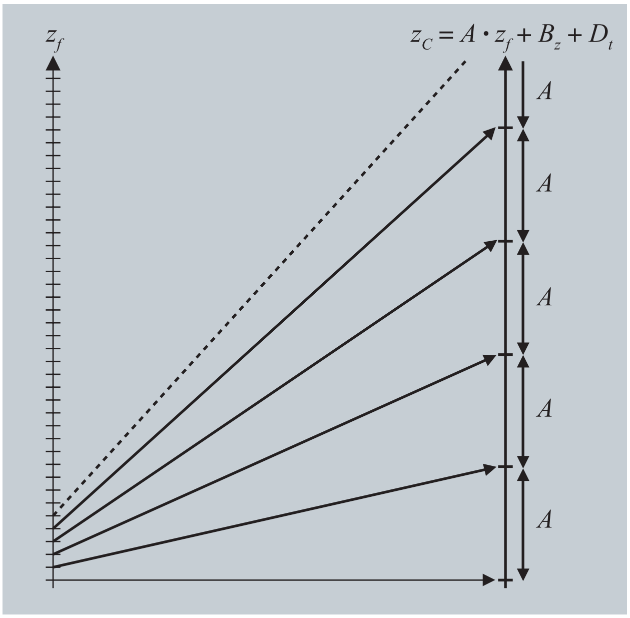

**Fig. 2: Range of values of coded variables**

A substantial improvement of the detection of errors can be achieved by checking with the help of coded data from various software channels [4]. For example, in the case of two coded channels the check is:

$$A_2 \cdot (z_{c1} - B_{z1} - D_{t1}) == A_1 \cdot (z_{c2} - B_{z2} - D_{t2})?$$

The residual error probability reduces considerably if several channels are employed.

## Advantages of the use of mathematical coding

The fast innovation cycles for microprocessors mean that every hardware-based safety proof becomes outdated within a short time span [5], therefore continuously requiring new proofs. Due to the strict mathematical basis applied here, the proof does not need to refer to the respective processor and its environment. The characteristics of the mathematical code determine the residual error probability and finally the SIL according to IEC 61508.

**Prof. Dr.-Ing. Frank Schiller, Scientific Project Manager of Safety and Security, Beckhoff Automation**

In addition, the virtually simultaneous execution of safety-relevant and non-safety-relevant control programs on one IPC is possible, so that its available performance can be optimally utilized.

## Conclusion

The present capability of Industrial PCs permits the efficient use of arithmetic codes to execute safety-relevant control programs. The necessary increased memory and the large number of extensive coded operations do not present a problem for a modern IPC. The quality of Industrial PCs is of such a high level that continuous operation is ensured and the detection of an error does not continuously initiate the transition to the safe state (usually a non-productive state).

In the solution presented in this article, extended coding is used in order to obtain a high level of error detection in the data processing unit of an Industrial PC. The certifiability of the approach according to IEC 61508 has been confirmed in a report by TÜV SÜD. The necessary safe coupling to the process peripherals is enabled via Safety over EtherCAT.

**Literature**

[1] Forin, P.: Vital Coded Microprocessor Principles and Application for Various Transit Systems. IFAC Control, Computers, Communications, Paris, 1989, S. 79-84.

[2] Wappler, U., Fetzer, C.: Software Encoded Processing: Building Dependable Systems with Commodity Hardware. International Conference on Computer Safety, Reliability and Security, SAFECOMP 2007, LNCS 4680, Munich, Springer, 2007, S. 356-369.

[3] Ozello, P.: The Coded Microprocessor Certification. International Conference on Computer Safety, Reliability and Security, SAFECOMP 1992, Munich, Springer, 1992, S. 185-190.

[4] Oh, N., Mitra, S., McCluskey, E.J.: ED4I: Error Detection by Diverse Data and Duplicated Instructions. IEEE Transactions on Computers, 51, 2002, S. 180-199.

[5] Mottok, J., Schiller, F., Völkl, T., Zeitler, T.: A Concept for a Safe Realization of a State Machine in Embedded Automotive Applications. International Conference on Computer Safety, Reliability and Security, SAFECOMP 2007, LNCS 4680, Munich, Springer, 2007, S. 283-288.