Interdisciplinary Description of Complex Systems 17(3-A), 492-496, 2019

LOGGING THE OPERATION AND ENHANCING THE RELIABILITY OF SAFETY-CRITICAL EMBEDDED SYSTEMS USING SELF-TEST

Zsolt Molnár*

Óbuda University, Kálmán Kandó Faculty of Electrical Engineering
Budapest, Hungary

DOI: 10.7906/indecs.17.3.8
Regular article

Received: 7 February 2019.
Accepted: 31 August 2019.

ABSTRACT

There are several solutions to increase the reliability of safety-critical embedded systems (e.g. redundant systems). Where appropriate, achieving the highest possible reliability is always an important goal. The present article also aims to describe a solution for this purpose.

One of these reliability enhancement options – besides redundancy – is the development of a self-testing system that can detect any malfunctions in downtime (during inactivity) or during normal operation. If there is no error, then this self-testing system reports that the system is error-free. The self-testing and event logging system described in this article provides an additional advantage over other solutions. In addition to increased reliability, the root causes of the stored events and information can be discovered and eliminated in case of an error, even, if necessary, by hardware or software changes.

The system outlined in this article – of course – requires further considerations and additions, and the details of circuit and software implementation should be elaborated, but its use in safety-critical systems is clearly beneficial.

KEY WORDS

safety-critical embedded system, self-test, operation and event logging, enhancing reliability, smart city

CLASSIFICATION

ACM: 10010583.10010737.10010744.10010741

JEL: D81

*Corresponding author, η: molnar.zsolt@kvk.uni-obuda.hu; +3616665174;
Becsi u 96/b, Budapest, Hungary, H-1035

SAFETY-CRITICAL SYSTEMS

Embedded systems are systems which include a computer but are not generally used for computing. Safety critical systems are systems whose failure can result in loss of life, significant property damage or damage to the environment [1]. Such systems generally have failure rate requirements ranging from 10^{-5} to 10^{-9} failures per hour or other suitable time period [2], with reliability encompassing the notion that the system is continuously operational and that it is operating with no functional defects during that time. Traditional safety critical domains are the aerospace, medical, chemical processing and nuclear industries. These domains have been conservative, slowly moving to rely upon software systems due to the difficulty of being able to prove that software-based systems will meet their desired operational reliability requirements. Newer non-traditionally safety critical domains including automotive, home automation and civil infrastructure do not always have the experience or the safety culture to help them accurately evaluate the benefits and risks of computer-based controls. Better technologies, processes and standards that improve or ease the use of software in safety critical domains are imperative to protect these domains where cost and functionality concerns may put pressure on safety principles. Regardless of the domain, acceptable mission critical systems are unlikely to be built without good system engineering processes [3].

One of the newest solutions, using a multitude of security-critical embedded systems, is the smart city. There are various elements of smart cities, but the following key areas are typically identifiable:

- smart mobility,
- smart energy,
- smart urban environment,
- smart lifestyle,
- smart governance, city administration,
- smart infocommunication infrastructure common to previous areas, which provides an integrated IT and communication background [4].

Each of the above areas requires numerous embedded system applications that need the use of a safety-critical embedded system. As described above, the presented solution increases reliability and highlights its importance.

ENHANCING RELIABILITY

To achieve the expected reliability of the elements of critical embedded systems (hereinafter referred to as the target units), and to facilitate the post-mortem detection of the events, additional elements are required. So, if the target units contain additional elements outside the security-critical (decision-making) units that implement the following additional activities:

- event and operation logging,
- self-testing (regularly or initiated by external command).

then these elements increase the transparency of the system operation and the resulted safety. The design must be such that, with their normal operation or in case of their failure, these additional elements could not affect the safe operation of the decision-making elements.

Event and Operation Logging

The additional elements collect, store, and make the following information, items, and events available in a suitable form for external request:

- commands, status signals and messages arriving at the target units' information boundary surface in their original (in unencrypted, or possibly in corrupted/damaged) form,
- signals, commands, status signals and messages sent from the target unit as a result of decisions made by the target units, in their realized form and with their parameters (signal level, duration),
- data inputs, configuration activities and other interventions through the controls of the user interface,
- significant changes occurring in power supply and temperature (and other environmental conditions, such as humidity, vibration, etc),
- the activities and decisions of the self-monitoring (self-testing) system,
- events related to event memory units (reading, deleting).

The previously listed events and information must be stored in non-volatile memory units for each event type. Each event has a stored time stamp that indicates the beginning/end of the event with an appropriate (e.g. 1 ms) resolution, with a considered time value for each event. For example, information from a control (on the user interface) or a temperature sensor is unnecessary to sample, to assign time stamps, and to store in every e.g. 10 ms, because the changes at these points are slow. Probably it is sufficient to have intervals of 50 to 100 ms, or even larger, but the event-driven data collection could be used, too.

However, a controlling signal at a microprocessor output, may require storage in the resolution of μs or even denser. It is worthwhile to use a real-time clock (RTC) to create the timestamp, so the absolute and relative time of the events can be determined later, and their timeliness can be examined. If the size of the event-storing memory unit is large enough, it is possible to store the events with the highest density required. When the stored events become obsolete, they can be overwritten, so a circular memory management can be used. The time of data obsolescence should be determined carefully, as an event may affect operation even after a long period of time, therefore, it must be made retrievable much later.

The auxiliary elements allow the reading of the information stored in the event memory units for a unit above the operating hierarchy, and the external (scanner/reader) device connected to it by a special interface. The event of the deletion by command of the event memory unit contents must also be logged, but these events cannot be deleted. Such events should be stored in a separate, protected memory area or memory device.

Self-Test of the Target Unit

The auxiliary elements must be capable of the followings:

- must be able to replace the expected signals, messages, commands, operator interventions at normal operation mode, on the standard boundary surfaces, with predefined test events/signals,
- must be able to register that the decision-making unit responds to test events at what type of events/signals and by how much delay.

The response events/signals for the test events/signals do not get out of the target unit. After checking the functions of the target unit, the results are stored in the memory, and the system returns from the test mode to the normal operation mode. If necessary, an error message is sent to the unit above in the hierarchy.

Self-test can be initiated:

- when powering up the system,
- at regular intervals, provided there is no need for normal operation during the self-test period. The length of the self-test interval (e.g. 1 hour, 1 day) and the suspension of the normal operation mode requires further considerations,

- by the command of the unit above in the operation hierarchy,
- by the operator's intervention.

The identity of the initiator of the self-test, the start and end time, the result of the evaluation and the fact of the possible error result must be stored in a non-volatile memory.

Figure 1 shows the schematic connections between the critical embedded system (also named the target device), and the auxiliary elements for event logging and self-testing. The items presented in Figure 1 include:

- **Target device** blocks (black blocks),
- Signal transmission and signal switching required for normal operation and test mode is performed by **analog and/or digital switches** (rounded gray blocks),
- **Test signal generators** (generate analog and digital signals for self-test) (rounded blocks, some of them are signal converters),
- To the event logging and self-testing unit, the inputs and outputs of the target device are transmitted by **signal converters** (rounded blocks, some of them are test signal generators). Their primary function is level fitting e.g. between different voltage levels, input protection, and, in some cases, galvanic isolation,
- **The event logging and self-monitoring unit controller**, as well as the **associated element**s (memory unit, RTC, environmental sensors) provide the intelligence of the auxiliary elements (gray blocks),

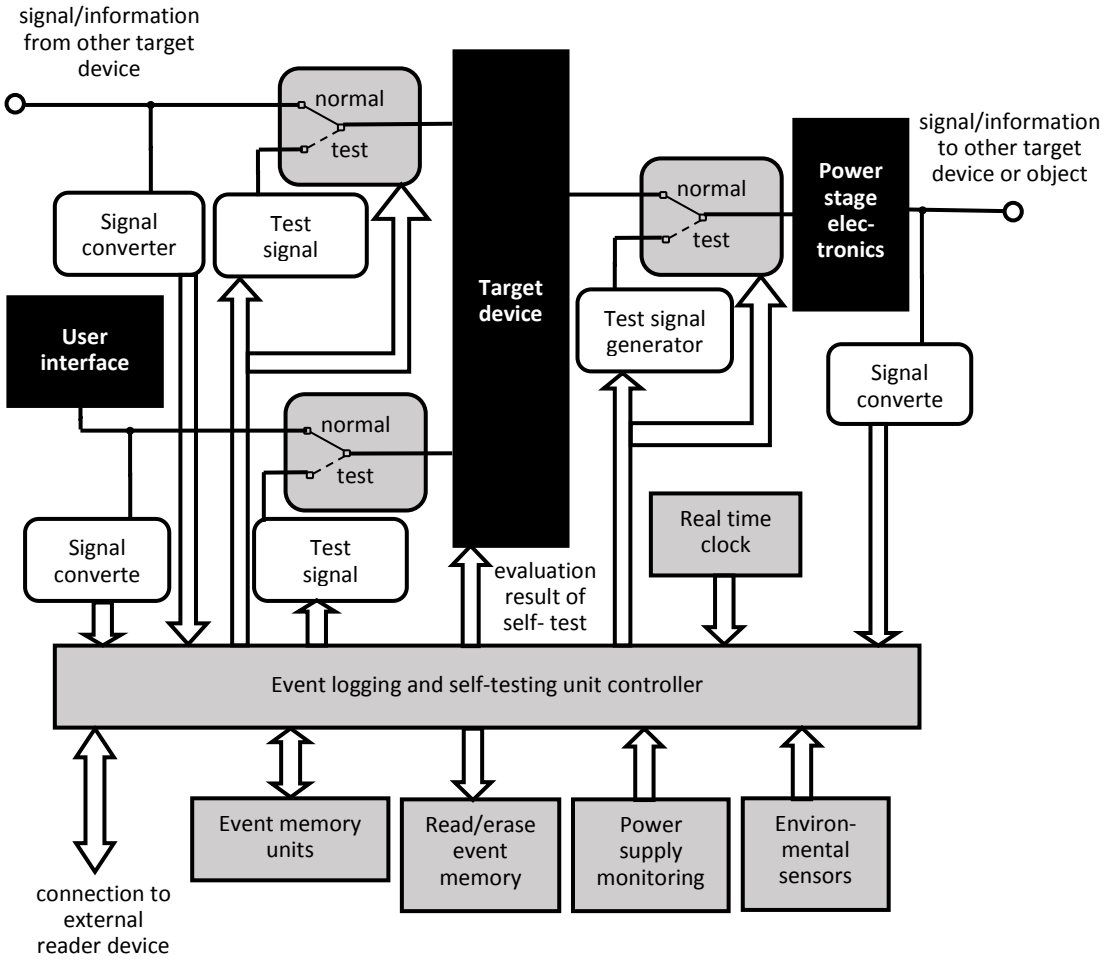

Figure 1. The connections between the target device and the logging and self-testing units.

- Single line connections represent analog/digital signals,
- Arrows represent information flow.

CONCLUSION

For those systems where the stated principle is planned to be applied, the principles of "Design for Testability" (DFT) should be followed. These principles should be considered from system design through the selection of circuit elements and circuit design to the design of the software, as the testability should be planned at system, sub-system and component level [5]. In order to implement the principle described in this article, the boundary scan test method ("digital" boundary scan – IEEE 1149.1 and mixed-signal boundary scan – IEEE 1149.4) can be applied at several points. One of my previous articles described the concept of an integrated circuit to support the self-testing of analogue circuits. [6] Such an integrated circuit is also applicable and can solve many problems during further developments in this field. The solution outlined in the present article can improve the reliability of many safety-critical systems, and it can help to detect the causes of failures [7].

ACKNOWLEDGMENT

The research on which the publication is based has been carried out within the framework of the project entitled "The Development of Integrated Intelligent Railway Information and Safety System", application number: GINOP-2.2.1-15-2017-00098.

REFERENCES

[1] Knight, J.: *Safety critical systems: challenges and directions.*
In: Proceedings of the 24th International Conference on Software Engineering. IEEE, Orlando, 2005,

[2] Leveson, N.G.: *Software safety: why, what, and how.*
ACM Computing Surveys **18**(2), 125-163, 1986,
http://dx.doi.org/10.1145/7474.7528,

[3] Kane, A.: *Runtime Monitoring for Safety-Critical Embedded Systems.*
Carnegie Mellon University, Pittsburgh, 2015,

[4] Bakonyi, P., et al.: *Smart City megoldások hat kulcsterületről.*
Budapesti Műszaki és Gazdaságtudományi Egyetem Egyesült Innovációs és Tudásközpont, Budapest, 2016,

[5] Vranken, H.P.E.; Witteman, M.F. and Wuijtswinkel, V.R.C.: *Design for testability in hardware-software systems.*
IEEE Design and Test of Computers **13**(3), 79-87, 1996,
http://dx.doi.org/10.1109/54.536098,

[6] Molnár, Zs.: *Analóg áramkörök beépített öntesztbe vonását támogató integrált áramkör.*
Óbuda University, Budapest, 2009,

[7] Flammini, F.: *Railway Safety, Reliability, and Security: Technologies and Systems Engineering.*
IGI Global, Rome 2012.