


Kharlap S.N., Sivko B.V.

DEVELOPMENT OF HIGH-RELIABLE SYSTEMS BASED ON THE METHOD OF CROSS CHECK OF AXIOMATIC BASES

Based on the method of cross check of axiomatic bases, there have been developed reliable fail-safe microprocessor systems. They have been tested for reliability and fail-safety by using a set of simulation safety tests (SST). It has been revealed that reinforcement of the checking axiomatic basis allows us to improve fail-safety parameters. It has been shown that the cross check of axiomatic bases enables to develop fail-safe and reliable systems with target properties.

Keywords: *functional safety, formal methods, fail-safety, failure detection, IT critical objects.*

Introduction

Modern safety critical systems face enhanced requirements as to both reliability and functioning safety. To ensure a respective quality, certain measures are required to be taken to improve fail-safety indices. One of the purposes of such measures is to detect failures to be followed by a reaction in the way of transition into safe state or restoration process. As nowadays this problem does not have one solution, it remains pertinent to develop methods and tools for efficient detection of failures [1].

This paper describes peculiarities of the development and testing for functioning safety of hardware and software complex (HSC) constructed on the basis of cross check of axiomatic bases [2].

To estimate the efficiency of the method, it is necessary to check the results of its application in practice. At present simulation modeling is used for this purpose as a practice enabling to carry out a total cycle of development and verification with minimum expenditures in a lab environment. In this regard a program set of simulation safety tests (*SST*) was chosen as a simulation tool intended for simulation tests for functional safety in accordance with IEC 61508, EN 50126, OST 32.146 of microprocessor systems of control of safety related processes [3]. By means of SST it is possible to introduce different failures of hardware, as well as to analyze further behavior of the system under consideration.

The test object is HSC, with a microcontroller PIC16F877A within its scope.

When developing fail-safe systems, the recommended practice is the protection against certain types of failures which are known and typical for this hardware. In this case, the choice of the variety of failures to be protected against is due to the requirements of IEC 61508-2:2010. Therefore, we consider failures of used memory cells: stuck-at faults as well as bridging faults.

Based on the method requirements, two bases are initially required which are independent and diverse [4]. The first basis A is proper operation in the context of failures of one byte of random access memory (X). The second basis B is similar conditions for a microcontroller's accumulator (W) and a carry flag (C). Cross check is shown in Fig. 1, where A basis is checking B basis by a function s_1, and B basis is checking A basis by a function s_2. The system

performing this task contains a target function *f*, which is performed on both bases.

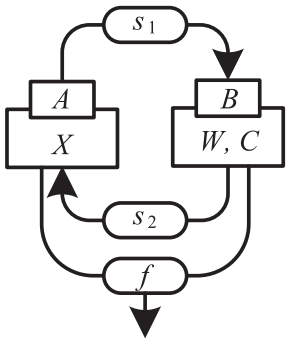

Fig.1. Cross check of bases

Development of a reliable system

As a failure detection signal as the result of basis check, it is convenient to choose two digital lines (let us call them *SAFE_A* and *SAFE_B*), which can be in states of logical 0 or 1, or refer to the microcontrollers ports. In case of a successful operation of the system, a variable signal (0 or 1) is sent to *SAFE_A* and *SAFE_B*. When one of the bases is disrupted the respective output gets a constant value 0.

An external signal of the described type is easy to realize and process, and based on this signal one can use external hardware enabling to solve the situation in case of failure, for instance, to transfer the system into safe state.

The algorithm is as follows:

1. To set *SAFE_A*=0.
2. To check B basis by means of A basis.
3. If *B* is performed, *SAFE_A*=1 should be set.
4. To set *SAFE_B*=0.
5. To check A basis by means of B basis.
6. If *A* is performed, *SAFE_B*=1 should be set.
7. Pass to step 1.

Thus, the considered system performs only constant check of bases and depending on the results, it forms output signals *SAFE_A* and *SAFE_B*. Every signal can be either variable (~), or constant (0), based on which it is possible to detect failures (Table 1).

***Table 1* – States at the detection of failures**

No	*SAFE_A*	*SAFE_B*	Description
1	~	~	No failures, Both bases are performed
2	~	0	Basis *B* is performed, Basis *A* is disrupted
3	0	~	Basis *A* is performed, Basis *B* is disrupted
4	0	0	One basis is disrupted, or both of them are disrupted

To comply with safety requirements in case of disruption of *A* basis, the system shall reliably transit to state 2 and 4 as per Table 1. Similarly, when *B* basis is disrupted there should be a transit to state 3 or 4.

The system is considered to be fail-safe for states 1, 2 and 3 of Table 1. In this case one of the bases is performed, and the functionality realized on the respective basis is considered as fault free.

Table 2 presents a subprogram in the assembler language of microcontroller PIC16F877A [5] performing the check of *A* basis based on *B* basis. This subprogram is the implementation of steps 5 and 6 of the algorithm. As the execution of the subprogram takes finite time, which is for sure more than a specified limit, then in case the bases are correct, a variable signal will be received at the output of *SAFE_B*.

Table 2 – Program check of A basis

Action	Program
Check of stuck-at-1 fault	clrf X movf X,0 addlw 0x0ff btfsc STATUS,C return
Check of stuck-at-0 fault	movlw 0x0ff movwf X movf X,0 addlw 0x01 btfss STATUS,C return
Check of wired AND/OR failure for near-by bits (*bridging fault*)	movlw 0x0aa movwf X movf X,0 addlw 0x055 addlw 0x01 btfss STATUS,C return
Signaling of success of basis check	bsf SAFE_B return

To check the capacity of the developed program for failure detection, the considered HSC was realized in SST, where simulation tests were carried out. According to the bases and failures detected, there was composed the list of failures to be checked:

– *SA*0(*W*), *SA*1(*W*), stuck-at-0 and 1 of accumulator, 16 failures;

– *B*(*W*), wired AND/OR of accumulator (14 failures);

– *SA*0(*C*), *SA*1(*C*), stuck-at-0 and 1 for *C* flag (2 failures);

– *B*(*C*), wired AND/OR of *STATUS* register, affecting *C* flag (2 failures);

– *SA*0(*X*), *SA*1(*X*), stuck-at-0 and 1 of *X* memory cell (16 failures);

– *B*(*X*), wired AND/OR of *X* memory cell (14 failures).

Test program included the registration of every single failure, as well as the check of operation without failures. Based on the received graphs of the signals *SAFE_A* and *SAFE_B* failure response of the system was defined. The

results of simulation modeling based on the given failure list is provided in Table 3 (tests 1). Tests 2 of Table 3 were carried out with software (SW) that had been modified to increase fail-safety (it will be discussed later).

The tests showed that when entering most of the described failures, both signals *SAFE_A* and *SAFE_B* passed to stuck-at-0. First of all, the important result is the fact that the check of a respective basis was always successful, i.e. if any basis checks the second one, and the latter is not performed due to a failure, the first one always makes signals about the problem. In other words, initially it was mathematically proven that in case of the registration of one of the considered failures, the system shall always pass into the state signalling about the problem of a respective basis. It was confirmed in practice of simulation modeling when *A* basis was disrupted, signal *SAFE_B* always passed to 0 state (and when *B* was disrupted, signal *SAFE_A* also passed to 0). By that it was shown that in case of failures the system always passes into safe state that gives an evidence of a successful application of the methods in terms of the development of a reliable system.

Table 3 – Results of tests of a reliable system

Failures	States of output signals			
	Tests 1		Tests 2	
	SAFE_A	***SAFE_B***	***SAFE_A***	***SAFE_B***
–	~	~	~	~
$SA1(W_{1,3})$	0	0	0	~
$SA1(W_{0,2,4..7})$	0	0	0	0
$SA0(W)$	0	0	0	0
$B(W)$	0	0	0	0
$SA0(C)$ $SA1(C)$	0	0	0	0
$B(C)$	0	~	0	~
$SA0(X)$	0	0	0	0
$SA1(X)$	0	0	0	0
$B(X)$	0	0	0	0

Note:
0 – pass to stuck-at-0;
~ – a variable signal as per the requirements;
W_i – the *i*-th bit of *W* register.

In addition to that in most cases when *A* basis was disrupted, signal *SAFE_A* also passed to 0 state. This behavior comes from the fact that the check of *B* basis by a disrupted *A* basis turned out to be so sensitive that it sent the signals about s disruption of *B* basis despite the latter stayed correct. Exclusions were the failures of $B(C)$, which did not affect the output signal, as this resource (a nearby bit to *C* flag of *STATUS* register) was not involved in the check logic.

An additional conclusion from the results is an increase of fail-safety that holds when one of the signals *SAFE_A/ SAFE_B* is stuck-at-0 (0), and the other one is (~). The test results showed no increase of fail-safety in most cases, which comes from sensitivity of bases check, and from that fact that there were no additional measures taken to improve the fail-safety.

Improvement of fail-safety by basis enhancement

To improve the fail-safety, one can construct the software in such a way that the system could be a subject to failures in a less degree, i.e. the basis should be enhanced. This is possible when software uses fewer amounts of resources. For this purpose a program code for a check of *A* basis was changed to stuck-at-1 (Table 4).

The modified SW was a subject to the second tests which results are listed in table 3 (tests 2). SW was modified in such a way that only a high nibble of an accumulator that was used as a resource, and a low nibble was not involved. As a consequence, for two failures $SA1(W_1)$ and $SA1(W_3)$ the system started detecting the problem only at one of the bases that proves the improvement of fail-safety.

Table 4 – Modified check of A basis at SA1

Action	Program
Check of stuck-at-1 fault	clrf X movf X,0 addlw 0x0f0 btfsc STATUS,C return swapf X,0 addlw 0x0f0 btfsc STATUS,C return

Failure diagnostics changed for two bits of an accumulator and for *SA*1 failures, despite the fact that only one half-byte (4 bits) is used for check, and in other further checks – all bits of the register. Here it is explained by two reasons. Firstly, a further check for *SA*0 failure checks only the passes from 1 to 0, which is why it is sensitive for *SA*1, but is independent of *SA*0. Secondly, a check for $B(W)$ uses half of bits as 0, and the second part is used as 1. Consequently, $SA(W_0)$ and $SA(W_2)$ failures war detected as the failures of wired AND/OR.

In terms of cross check of axiomatic bases, a program modification brought the system to the state shown in Figure 2.

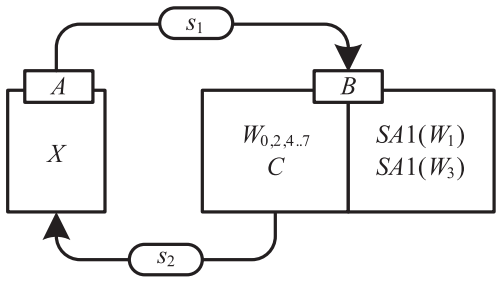

Fig. 2. Consolidation of basis to improve fail-safety

A check function s_2 got based on a smaller basis (which is stronger) and it made the system immune to the respec-

tive failures. Therefore, enhancement of bases enables to improve the system fail-safety, which means that at the development one should choose a stronger basis which is subject to failures in a less degree.

It should be noted that the basis enhancement has no influence upon reliability of the system in case of failures. Thus, the above described method could be used to improve fail-safety of systems without loss of reliability.

Development and testing of a fail-safe system

In practice the system functioning requires more resources (memory, input-output devices, etc.), than the checking operations. That is why enhancement of bases for improvement of fail-safety is a complicated task which is not always feasible. In such situation in terms of axiomatic bases to improve fail-safety it is possible to split the tasks of reliability and fail-safety, with further diversification, as it is shown in Figure 3.

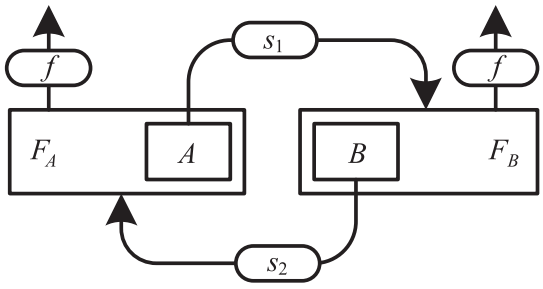

Fig. 3. Fail-safe diverse system

Four bases are analyzed here – A, B, F_A and F_B. A basis is stronger than F_A basis (a disruption of A basis results in disruption of F_A basis) and this basis is used to check F_B basis by means of the function s_1 (similar for the bases B, F_B, A and the function s_2). Two implementations of the function f are based on the bases F_A and F_B which are independent of each other. If the check of basis is successful (for instance, the result is s_1), then the function f based on the respective basis (F_B) can be performed safely.

For the described realization, failures disrupting A basis or B basis bring the system to safe state, and when F_A or F_B are disrupted when performing A and B , the system stays fail-safe.

To test the mentioned method of fail-safety improvement, the reliable system already tested (tests 1) has been improved by adding of the functionality to generate a digital signal from a microcontroller in form of sequence of zeros and ones (01010101 and 11101000 respectively). That means that SW operating on an inner cycle changes an output signal to 0 or 1 on every cycle (in accordance with the specified sequence). For this purpose the memory keeps a word of one byte over which a cyclic shift is made, and a higher bit goes to an external port. Operation of the system with no failures is shown in Figure 4.

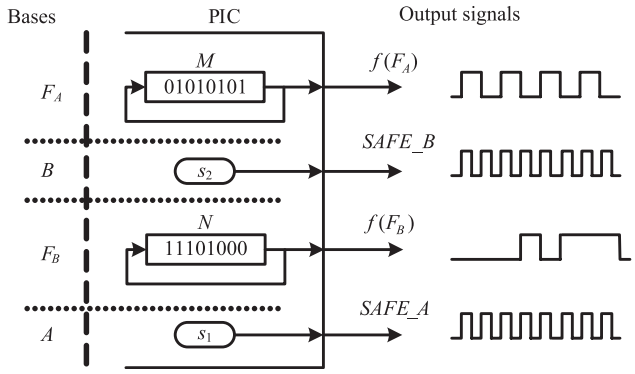

Fig. 4. Operation of the system with no failures

For realization of the described functionality every basis needs an additional memory byte, which may become subject to failures (byte M for A basis, and byte N for B basis). The sequence of algorithm actions, which bases they are based on, as well as the description are given in Table 5.

Distribution of operations between bases follows from the aspects of bases check and the resources used. As the basis check implies a change of memory cells, and their content is required to be kept to accomplish a task, then during a check it is necessary to make a copying, which is performed on the other basis, then a check is. All actions related to the calculation of the function f are realized on the respective basis (F_A or F_B), which ensures fail-safety of one them in case the other one is disrupted.

A necessity to distribute actions between the bases can be considered in a form of the rule according to which it is necessary to use the basis to be checked, for service operations during preparation of check procedures. For example, the check F_A is performed based on B, and all service operations are required to be performed on F_A.

For the described system a test program was carried out, the results of which are shown in Table 6. The tests have shown that the system holds its property of reliable

Table 5 – Algorithm of a fail-safe system

No	Basis	Action
1	B	Check X
2	F_A	Copy from M to X
3	B	Check M
4	F_A	Copy from X в M
5	F_A	Implementation of function f
6	A	Check W, C
7	F_B	Copy from N to W
8	A	Check N
9	F_B	Copy from W to N
10	F_B	Implementation of function f
11		Pass to clause 1

Table 6 – Results of tests of a fail-safe system

Failures		Results			
Type	Number	Safety		Correctness of the function performed	
		SAFE_A	SAFE_B	$f(F_A)$	$f(F_B)$
–	1	~	~	+	+
$SA1(W)$	8	0	0	– + – + – + – +	– – – + – + + +
$SA0(W)$	8	0	0	+ – + – + – + –	+ + + – + – – –
$SA1(X)$	8	0	0	– + + + + + + +	+ + + + + + + +
$SA0(X)$	8	0	0	– + + + + + + +	+ + + + + + + +
$SA(C)$	2	0	0	+ +	– –
$B_{OR}(W)$	7	0	0	– – – – – – –	+ + – – – + +
$B_{AND}(W)$	7	0	0	– – – – – – –	+ + – – – + +
$B_{OR}(X)$	7	0	0	– + + + + + +	+ + + + + + +
$B_{AND}(X)$	7	0	0	– + + + + + +	+ + + + + + +
$B_{OR}(C)$	1	0	~	+	–
$B_{AND}(C)$	1	0	~	+	–
$SA1(M)$	8	~	0	– – – – – – – –	+ + + + + + + +
$SA0(M)$	8	~	0	– – – – – – – –	+ + + + + + + +
$SA1(N)$	8	0	~	+ + + + + + + +	– – – – – – – –
$SA0(N)$	8	0	~	+ + + + + + + +	– – – – – – – –
$B_{OR}(M)$	7	~	0	– – – – – – –	+ + + + + + +
$B_{AND}(M)$	7	~	0	– – – – – – –	+ + + + + + +
$B_{OR}(N)$	7	0	~	+ + + + + + +	– – – – – – –
$B_{AND}(N)$	7	0	~	+ + + + + + +	– – – – – – –

Note: B_{OR} and B_{AND} – wired OR and AND failures respectively; (+/–) – successful or unsuccessful function implementation; Position (+/–) from a low bit of the register to a high one.

behavior in the presence of failures (in case of failure, the signal *SAFE_A/SAFE_B* corresponding to the basis passes to 0 state).

The second result is the acquired fail-safety. If *SAFE_A* has a constant 0, and *SAFE_B* is variable (~), then $f(F_A)$ is always performed. Similarly, if *SAFE_B*=0, and *SAFE_A* is (~), $f(F_B)$ is performed.

Conclusion

A choice of bases and development of check procedures have shown in practice that it is advised to enhance the basis as much as possible, which results in the improved reliability and fail-safety of the system. In case a large amount of elements possible to become subject to failures are involved, the procedures of bases check are getting complicated. Consequently, at the design of a reliable or fail-safe microprocessor system, an important task is to get a certain already tested and not very large (i.e. strong) basis. Such basis shall hold a significant level of operation flexibility, and it can be used to check the functionality different from the basis.

Thus, the method of cross check of axiomatic bases has been tested, that let us formally develop and verify reliable and fail-safe systems, capable of failure detection, and as a sequence of passing into safe state, or a problem detection without any loss of performance capability. Test results show that by means of the method of cross check of axiomatic bases we can move to a new level of formalization and quality in development and verification of fail-safe and reliable microprocessor systems.

References

1. **Bochkov K.A.** Microprocessor automation systems on railway transport: study guide / K. A. Bochkov, A. N. Kovriga, S. N. Kharlap // Gomel, BelSTU. – 2013.

2. **Sivko B.V.** Axiomatic and basis approach to the development of reliable and fail-safe systems / B. V. Sivko // Automatic equipment on transport: PSTU. – 2015. – No. 4, v. 1., P. 381–399.

3. **Bochkov K.A.** Methods and tools of evidence for functional safety of microelectronic systems of railway automation equipment / K. A. Bochkov, S. N. Kharlap, D. N. Shevchenko // – 2011. – No.2. – P. 73–81.

4. **Sivko B.V.** Diverse axiomatic bases for the development of reliable and fail-safe systems / B. V. Sivko // BelSTU Reporter: Science and Transport. – 2014. – No. 1(28). – P. 19–23.

5. **Bates M**. PIC microcontrollers: an introduction to microelectronics. / M. Bates // Elsevier. – 2012. – 441 p.