Automated testing and fault diagnosis of the microcontroller system

D A Pankov and L A Denisova

Omsk State Technical University, 11, Pr. Mira, Omsk, 644050, Russia

E-mail: pankovDDD@yandex.ru, denisova@asoiu.com

Abstract. The article deals with the problem systems automated testing of microcontrollers. A system architecture proposed for fault diagnosis and determining the consequences of device failures. Highlighted separately intermittent failure. The issue of integrating software and hardware failure simulation tools into the testing architecture investigated. Model studies of the testing system performed and conducted diagnostics of simulated faults. The efficiency technology of automated testing confirmed with simulation known failures types (injection) for the components of the performed microcontroller device is proposed.

1. Introduction

Currently, there are stacks of technologies and methods searched faults [1] in hardware and software systems. However, their methods use is very limited. This is because most solutions don’t consider intermittent failures and stable failures together. It’s against the effective FMEA [2] military standard. Therefore, in work [3] the concept of fault tolerance considered how stability work with only intermittent failure. Work [4] is devoted to diagnostics of stable failures and discarded intermittent failure. In practice, stable and intermittent failures occurred jointly in the process of system work.

Note the set of measures for resistance to all types failure will vary significantly depending on the scope of the device. For devices used in atomics and cosmos segment applied the most stringent testing methods than for devices in communication segments [5]. To provide a check intermittent and stable failures in relation to the device that contains the set of microcontrollers is required to define the information controlled system portrait. Information portrait is the result of the analysis of reliability behavior of systems under simulated failures. It contains processed statistics based on that evaluate control system parameters. In the process of operation, the hardware-software system experienced the consequences of failures. For this reason, to assess the effectiveness of fault detection (identification - stable or intermittent failure) before the release of serial devices is searched for critical signatures (signs).

Fault simulations to assess the performance of functional controls has become a classic tool for fault-tolerant system developers. Since modern development tools use automated testing systems, it is most advantageous to integrate a fault simulation system together with system testing. The simulation process should be as automated as possible to speed up the device design time. This work is devoted to the system architecture of the automated testing for microcontrollers (SATM). The architecture is required to identify with the necessary accuracy both simulation stable and intermittent failures.
2. Analysis of SATM requirements
The architecture of the SATM should provide verification of the correct functioning of the subsystems by fixing the state of the device in various variants of normal operation. The complexity of the problem is due to the fact that, unlike hardware fault tracing, there are three main groups of problems that complicate software testing: the lack of a uniform methodology for creating fault-tolerant software; lack of a unified methodology for testing fault-tolerant software; lack of a unified approach to the analysis of the problem area.

To obtain a systemic solution to this problem, two European standards [6, 7] should be considered, which were relatively recently translated as domestic GOST: IEC 61508 (functional safety of electrical / electronic / programmable electronic systems related to security), EN 50128 (Automated software testing). Since this standard was introduced as GOST R IEC 61508, the use of fault simulation stable and intermittent failures is a main development requirement for control systems and information systems in general. The automated testing system contains a set of functional characteristics of the testing methods, which are discussed in Figure 1. The main objects are used in the figure, which are appropriately taken into account for the construction of testing methods for the microcontroller system:

![Figure 1. Diagram of functional characteristics of testing methods SATM.](image)

- Intermittent failure - an event in which a fault occurs for a short time, then the system self-repairs and works reliably. Gradual failure - the degradation of the system component over time. Combined failure - failure, the cause of which has a software and hardware component.
- Failure is a malfunction in the system. A sudden catastrophic failure is a failure that leads to the failure of controlled systems of increased danger and the economic costs of recovery. Permanent failures - failure, resulting in the final component failure.
- Adequacy assessment — a control function that is mapped or not defined with respect to fault or failure.
- Evaluation parameters - the control function at the software or hardware level, which is characterized by the physical property of the experiment, the method of determination and has the accuracy determined by the function of assessing the degree of adequacy.
For the microcontroller system, the following version of the architecture of automated testing for failures is proposed (include intermittent failures), which conforms to IEC 61508 and EN 50128 standards: the architecture consists of two subsystems - a control object (test unit) and a control system (control station – CS) aimed at imitation and preparation fault finding results. Figure 2 shows a functional diagram for assessing the consequences of imitation of fault and failures in the system under study.

![Diagram of the SATM.](image)

Imitation of faults and failures for SATM has a specificity related to the selection of the necessary components among all methods and tools applied to microcontroller devices. Therefore, a set of measures in such systems consists of another set of tools that can be distinguished into a specialized architecture. Highlight the main levels of imitation of faults: interface, processor and program level.

The control object is a set of tested microcontrollers (MCU) with a common input and a common output (the number of MCU is N). On microcontrollers, a test program is executed that processes the input data sequence and performs a series of actions required to detect faults within the test. Data on the results are removed using an external control device and recorded in a database (DB).

An imitation device block (ID) is a device with a running program for the classification of signatures, which selects data from block 2 and stores on this block information about the sequence of output from the block of time. The external control unit controls the parts of the test unit, the list of which is corrected from the simulation station unit and is associated with the CS unit, which, when commanded from the PC, launches test sequences. CS is a simulation station that organizes the installation of signals inside the test block. Separately, it is necessary to consider the technical means of imitation of faults, which is represented in figure 2 by the block of the imitation device. To simulate faults, it is necessary to implement the introduction (injection) of faults into the hardware-software system. To organize the simulation unit must have a set of characteristics shown in figure 3. Basically, the unit uses 3 basic concepts for post-injection testing [9].

The fault injector consists of an injector preparation algorithm that is implemented at the control station. This algorithm implements the following types of testing strategies:

- **Heuristic injector.** Testing for faults based on knowledge of the software and hardware features of the system.
- **Data Injector.** Fault testing based on input changes that exploit the robustness problem of the software.
- **Mutation Injector.** Testing for faults based on pseudo-random source code changes and checking the response to these changes in the automated test suite.
Figure 3. Diagram of imitation devices (fault injector).

3. Software and technical implementation of hardware control functions
Detection of hardware failures requires the use of hardware control modules that define technical malfunctions of components. One of the effective approaches is the comparison of signals at the inputs and outputs and the parametric regulation of the module. If the module deviates from the established order, a fault is fixed, after which the information is transmitted to the workstation. The use of such modules is relevant for the main subsystems of the microcontroller: internal memory management systems, data buses, I/O, hardware timer. The modules work for test programs that change the values of parameters that the module controls. Table 1 lists the main modules that can be used to identify emerging intermittent and stable system failures [10].

| Table 1. Control system module. |
|---------------------------------|
| 1. The analysis module of the time. |
| **Input:** $x$ – interface data input $I$; $S_i$ – the sign of failure of the data transfer interface. **Output:** $y$ – the time of transmission of a data byte; $S_y$ – sign of time deviation from acceptable values. |
| **Setup option:** $I$ – the number of the transmission interface; $U$ – adjustable transmission speed. |
| 2. I/O control module |
| **Input:** $x$ – input signal; $S_i - S_n$ – I/O port access denied sign. **Output:** $y$ – changing the output signal from the input signal; $S_y$ – I/O port failure indication. |
| **Setup option:** $t_{max}$ – maximum I/O port response time, $n$ – number of ports to check input-output. |
3. Change control module

**Input:** $x$ – the signal changes; $S_y$ – sign of denial of access to the address.

**Output:** $y$ – the detection of changes; $S_y$ – a sign of the correctness of the response; $f_y$ – the sign of failure of the interface.

$$
x = \begin{cases} 0 & \text{if } S_y = 0, f_y = 0 \\ 1 & \text{if } S_y = 1, f_y = 1 \\ 2 & \text{if } S_y = 1, f_y = 2 \end{cases}
$$

$$
y = \begin{cases} 1 & \text{if } x = k_p \\ 2 & \text{if } x = k_i \end{cases}
$$

**Setup option:** $k_p$ – indication of a change in processor memory; $k_i$ – sign of a change in the interface.

4. Experimental results

For the microcontroller system, a SATM was created and an experiment was performed to simulate faults by changing the input data of the functions of the software being executed in various modes of operation of the system. Based on the data obtained, a characteristic of detecting intermittent and stable failures resulting from the injection of corrupted data during the operation of the device was constructed.

An experiment using the hypothesis of introducing faults captures a greater number of faults and failures by introducing test data into the most vulnerable software functions in terms of execution time and hardware dependency. It is noticeable that in table 2 compared with table 3, the number of actual faults and failures varies depending on the number of experiments, and the number of detected intermittent and stable failures with increasing number of tests reaches a stable value for the microcontroller device.

**Table 2.** Experimental results for randomly filing data.

| Test number | Count tests | Count real fault (include intermittent faults) | Count fault detect | Fault detection rate |
|-------------|-------------|-----------------------------------------------|--------------------|---------------------|
| 1           | 150         | 3                                             | 0                  | 0%                  |
| 2           | 400         | 10                                            | 1                  | 10%                 |
| 3           | 600         | 28                                            | 2                  | 7%                  |
| 4           | 900         | 35                                            | 1                  | 2%                  |
| 5           | 1200        | 41                                            | 3                  | 7%                  |
|             |             |                                               |                    | Average percent detect faults for all tests | 5% |

**Table 3.** Experimental results using the imitation hypothesis.

| Test number | Count tests | Count real fault (include intermittent faults) | Count fault detect | Fault detection rate |
|-------------|-------------|-----------------------------------------------|--------------------|---------------------|
| 1           | 150         | 9                                             | 3                  | 30%                 |
| 2           | 400         | 39                                            | 8                  | 20%                 |
| 3           | 600         | 60                                            | 11                 | 18%                 |
| 4           | 900         | 65                                            | 12                 | 18%                 |
| 5           | 1200        | 78                                            | 15                 | 19%                 |
|             |             |                                               |                    | Average percent detect faults for all tests | 19% |

5. The discussion of the results

Conducted research of the fault simulator using an automated testing system make it possible to notice that it is possible to model injections of failures (include intermittent faults) using modifications of the program data, and using the algorithm with the hypothesis of introducing a fault allows the injection to be performed more efficiently. It can also be noted that the number of all faults recorded for given
algorithms does not exceed half of the simulated faults. For an experiment using the imitation hypothesis, the average percentage of faults and failures is 19% versus 5% for pseudo-random fault insertion. It should be noted that the percentage of fault detection even when using the simulation hypothesis is not high. To improve the quality of detection, a statistical analysis based on machine learning methods is planned to be introduced into the system in the future.

Nevertheless, the SATM can be considered effective, since the occurrence of one fault or failure in actual operation can lead to failure, and the time of failure is no less than the MTBF indicator and this can lead to large economic losses for restoration. [1, 3, 11].

When using a pseudo-random method of introducing faults by modifying data, the choice of a program function is performed over time. Since the execution time of functions may vary depending on the mode of operation, the result of imitation is not stable. When using the imitation hypothesis, faults are introduced for functions that are most critical in terms of execution time, which gives a higher percentage of occurrence and detection type failures.

6. Conclusion

The architecture of the automated testing system of microcontrollers was built using the method of injection of faults during operation. An experiment on the injection stable and intermittent failures using data modification in the software of microcontrollers. As a result of the research, it was found that this technique is effective for general purpose microcontrollers. In the future, to expand the techniques and methods of injection of faults, it is planned to expand the algorithms for fixing failures. Since increasing the mean time between failures is important for ensuring the key functions of microcontroller systems, the search for the consequences of failures is relevant for any application.

References

[1] Avizienis A 1985 The N-Version approach to fault tolerant software IEEE Trans. Softw. Eng. 11(12) 1491–501
[2] Legg J M 1978 Computerized Approach for Matrix-Form FMEA IEEE Trans. Rel. 27 254-57
[3] Chekmarev S A and Khanov V Kh 2015 Fault injection via on-chip debugging in the internal memory of systems-on-chip processor IOP Conf. Ser.: Mater. Sci. Eng. 94
[4] Kopylov S A, Fedorenko R V and Gurenko B V 2014 Software package for detecting and diagnosing hardware failures in robotized marine moving objects Eng. Bull. of the Don 3 16-20
[5] Tai A, Meyer J and Avizienis A 1993 Performability enhancement of fault-tolerant software IEEE Trans. Rel. 42 (2) 227–37
[6] Medoff M and Faller R Functional Safety An IEC 61508 SIL 3 Compliant Development Process
[7] European Committee for Electrotechnical Standardization (CENELEC) 2014 Railway applications - Communication, signalling and processing systems
[8] Silva A, Martinez J, Lopez L and Redondo L 2007 A Fault Injection Tool for Distributed Heterogeneous Embedded Systems Proceedings of the Euro American Conference on Telematics and Information Systems p 17
[9] Manes V J, Han H, Sang H H, Egele M, Schwartz E J and Maverick W 2018 ACM Comput. Surv. Fuzzing: Art, Science, and Engineering
[10] Pankov D A and Denisova L A 2018 Control and diagnostics of faults in the hardware-software complex Omsk Sci. Bull. 2 129–33
[11] Kovalev I V 2014 Analysis of problems in the research area of software reliability: a lot of stages and architectural aspect Vestnik SibGAU 3 78–92