Investigation of the heat meters calculator applying a measurement devices software reliability method

V.KNYVA, M.KNYVA
Department of Electronics and Measurements Systems
Kaunas University of Technology
Studentu str. 50, LT-51368, Kaunas
LITHUANIA
phone: +370 37 300535; e-mail: vytautas.knyva@ktu.lt, mindaugas.knyva@ktu.lt

Abstract: Nowadays, parameters, characteristics and functionality of measurement devices basically depend on microprocessors software. Software used in measurement devices overtakes functions from a hardware part of the device. For example, data processing algorithms, control functions are implemented using software. Basically during verification of the device, just a hardware part of the device is verified. But software controls all the functions of the measurement device. Therefore, a method for metrological estimation of reliability of measurement devices software will be presented.

Keywords: software validation, metrology, measurement devices, verification and testing of software

1. Introduction

Most functions of a measurement device are controlled by a microcontroller and software thereof. Software clearly influences characteristics and functionality of measurement devices. During verification of the measurement device, just a hardware part of the device is tested. OIML recommendations or specific standards of measurement devices provide no information on measurement devices software verification methods or methodology. In this paper, a method for metrological estimation of reliability of measurement devices software will be presented.

2. Problems of metrological estimation of measurement devices software

Today, most functions of measurement devices are controlled by devices software. Therefore, analysis of influence made by measurement devices software faults on devices functionality shall be carried out. Errors of measurements devices functionality can be divided to the following:

- Critical, i.e. measurement devices functionality is disturbed for an unknown period of time;
- Minor, i.e. transient disorder of measurement devices functionality.

All disorders of measurement devices are called by known factors, i.e. faults in a software code, simplified protection algorithm or others. In literature, the following sorts of faults are mentioned:

- Coding faults;
- Faults of measurements converters;
- Inadmissible influences of a user;
- Hardware faults.

Software of measurement devices that measures the same object is different by means of its complexity. Complex software clearly can have considerably more faults compared to simple one. Therefore, measurement devices must be divided into groups by means of software complexity and functionality thereof. The groups can be described in the following way:

- Measurement devices with software data processing;
- Measurement devices with software control of measurement converters and data processing;
- Measurement systems.

Software faults influence metrological characteristics of the measurement device. The following characteristics may be distinguished:

- Characteristics of the measurement result – measurement converters, stability of the measurement device, fault detection, etc.;
- Characteristics of measurement precision – instrumental, random, method faults;
- Dynamical characteristics of the measurement device.

Not all characteristics are influenced by measurement devices software. The influenced ones are provided below:

- Measurement faults – a wrong or incorrect measurement algorithm;
- Stability of the measurement device;
- Fault detection algorithm;
- Detection algorithm of inadmissible influences through a user, communication interfaces.

After adoption of the Measurement Instruments Directive (MID) [1], measurement devices software verification is necessary. The Measurement Instruments Directive started in the early nineties, was approved in spring of 2004, and after transition period it became fully functional in autumn of 2006. The MID introduces a "new approach" to the measurement devices software and its verification. Due to MID birth, in 1997 WELMEC (Western European Legal Metrology Cooperation) formed the work group WELMEC-SOFTWARE [2]. The main work object of this group involved forming of essential MDS requirements. Some time later, PTB (Physikalich-

Technische Bundesanstalt) formed another work group, i.e. MID-SOFTWARE, which detailed and concretized WELMEC requirements.

On the basis of MID, WELMEC-SOFTWARE work group formed the essential MDS requirements and classified them into 5 groups:

- Software design and structure;
- Software protection;
- Software conformity;
- Software testability;
- Documentation for type approval.

For three of them (software protection, software confirmation, software testability), 3 requirements levels, i.e. High, Middle, Low were formed (Table 1). On the basis of WELMEC publications, MID-SOFTWARE group offered a new conception, i.e. risk classes (Table 2).

3. ALGORITHM IMPLEMENTATIONS FOR REALIZATION OF MEASUREMENTS DEVICES CHARACTERISTICS

Basically, the software implemented in the measurement device is used for the following [3]:

- Control of measurement converters, processing and storing of data thereof,
- Control of measurement devices functionality.

For data processing of devices measurement converters, data analysis and filter algorithms are often used. They can be implemented in a hardware part of device or in software. In a hardware part, algorithms are designed through application of classical solutions: data registers, multipliers, summators, etc. In a software part, algorithms are implemented through microprocessors firmware. For additional implementations of the algorithm, programmable hardware parts (DSP) are used. Before implementation of the algorithms, suitability of a hardware part of the measurement device must be analyzed. Basically, two forms of microprocessors in measurement device design can be used, i.e. fixed point and floating point arithmetic.

The result of floating point summation or multiplication arithmetic can overrun, therefore, the register must be rounded or truncated. Such a solution at fixed point arithmetic can help using multiplication operation solely, whereas application of summation rounding or truncation of the result will not work. However, floating point arithmetic gives a better dynamic range and greater error of the result. These effects of microprocessors arithmetic must be estimated before implementation of a calculation algorithm.

Most mathematical functions can be implemented using transcendental functions. For example, the FFT computations require generation of complex exponential sequences.

Table 1 **Levels of Measurement Instruments Software**

Category	MID Annex*	Risk of fraud	Software protection level	Software examination level	Degree of Software Conformity
Supply to the customer by mains	MI-001, MI-002, MI-003, MI-004	Middle	*Middle*	*Middle*	*Middle*
		High	*High*	*Middle*	*Middle*
Commercial transactions and services	MI-005, MI-006, MI-007, MI-009	Middle	*Middle*	*Middle*	*Low*
		High	*High*	*Middle*	*Middle*
Environment safety, health	MI-010	Middle	*Middle*	*Middle*	*Low*

* MI-001 – water meters, MI-002 – gas meters, MI-003 – active electrical energy meters and measurement transformers, MI-004 – heat meters, MI-005 –measuring systems for continuous and dynamic measurement of quantities of liquids other than water, MI-006 – automatic weighing instruments, MI-007 – taximeters, MI-008 – material measures, no software, not relevant, MI-009 –dimensional measuring instruments, MI-010 – exhaust gas analyzers

Table 2 **Risk Classes of Measurement Instruments (Germany, PTB)**

Risk class	Software protection level	Software examination level	Degree of Software Conformity
A	*Low*	*Low*	*Low*
B	*Middle*	*Middle*	*Low s*
C	*Middle*	*Middle*	*Middle*
D	*High*	*High*	*Middle*
E	*High*	*High*	*High*

These sequences can be generated by second order digital filter structures or through application of transcendental functions based on truncated polynomial expansions. These types of expansions are often used for measurement devices microcontroller implementations. For example, the sine of number *x* can be approximated using an expansion:

$$\sin(x) \cong x - 0.166667x^3 + 0.008333x^5 - 0.0001984x^7 + 0.0000027x^9 . \quad (1)$$

where argument x is in radians, and its range restricted to the first quadrant – 0 to $\pi/2$. If *x* is outside this range, its sine can be computed by making use of identities $\sin(-x) = -\sin(x)$ or $\sin[(\pi/2)+x] = \sin[(\pi/2)-x]$. Decreasing the number of the components in the formula the approximation error will increase. Figure 1 shows the plots of the sine approximation computed applying just 2 components in formula (1) and the error due to approximation.

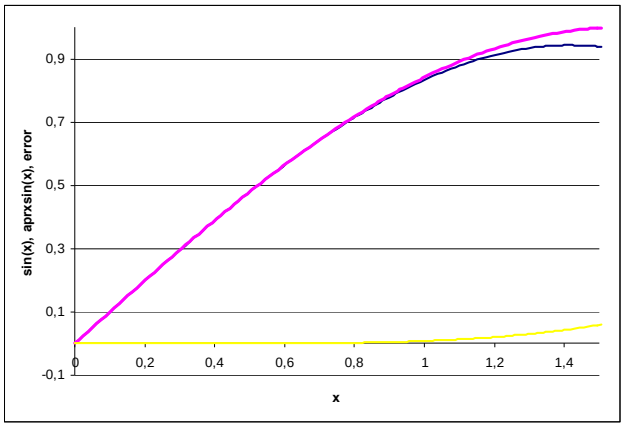

Fig. 1 Plots of the sine value computed using approximation, the actual sine value and error between the actual sine value and approximated

Each measurement device or system can be considered as resistant to faults if the software protection algorithms can identify and correct critical or minor faults. Otherwise, the results received by measurement devices software without protection algorithms can be faulty. It can be considered that the measurement device tolerates faults if software is able to finish measurement successfully after faults have been detected.

Measurement devices software protection algorithms must be tested in the following way:

- Testing the algorithm at the designing state (analyzing the software code),
- Testing the algorithm at the working state (analyzing the functionality of protection software)

4. Possible stages for estimation of measurement devices software

For measurement devices software investigation, two complementary stages can be used [4]:

- Document investigation;
- Functional investigation.

The main purpose of the documentation investigation is to collect all information about MDS and to estimate it. If there is too few information about some legally relevant software parts and without this information other investigation stages cannot be performed, a decision may be made that measurement devices software cannot be considered as metrologically reliable. For example, if there is no documentation about measurement devices software control commands, then there is no possibility to check the functionality of the measurement device.

The main purpose of the functionality investigation is to check whether measurement devices software behaves according to its documentation.

Investigation of measurement devices software documentation is one of the simplest stages [5]. The data collected from measurement devices software documentation can be used in other two investigation stages. Measurement device's documentation must be investigated by the three following aspects:

- Software's conformity requirement is fulfilled by checking its identification process and identification code in measurement devices software documentation. Besides, if there is an explanation of identification, algorithm estimation must be done to make sure that all legally relevant software is covered by that algorithm;
- Security requirements are fulfilled when information about all measurement devices control commands, data deletion/changing commands or software's updating/changing commands is presented and can be evaluated.
- Testing requirements are fulfilled when all information necessary for measurement devices software verification can be found in its documentation.

For this investigation we can offer a universal questionnaire method (fig. 2). The main point is the following: for each aspect, a questionnaire must be made in such a way that answers for each question can be only YES/NO. When all information about documentation is collected, the decisions concerning its completeness for verification of next stages can be made. If information is insufficient the functionality of measurement devices software cannot be investigated and the measurement device cannot be considered as metrologically reliable.

Hereby information about measurement devices software can be collected. It shall be highlighted that this investigation stage must be used as support for a functionality investigation stage. Selection of questions is a very responsible procedure and it is difficult to formalize it, whereas from an investigator this stage requires a high level of experience and responsibility.

Questions must be applicable to a particular measuring device. For example, if we have a measuring device without a user interface, there is no control panel with a keyboard, but this measuring device has communication interface (for example, RS232), through which it can be remotely controlled; to the questions related to a user interface all answers will be NO and it can lead to wrong conclusions about measurement devices software.

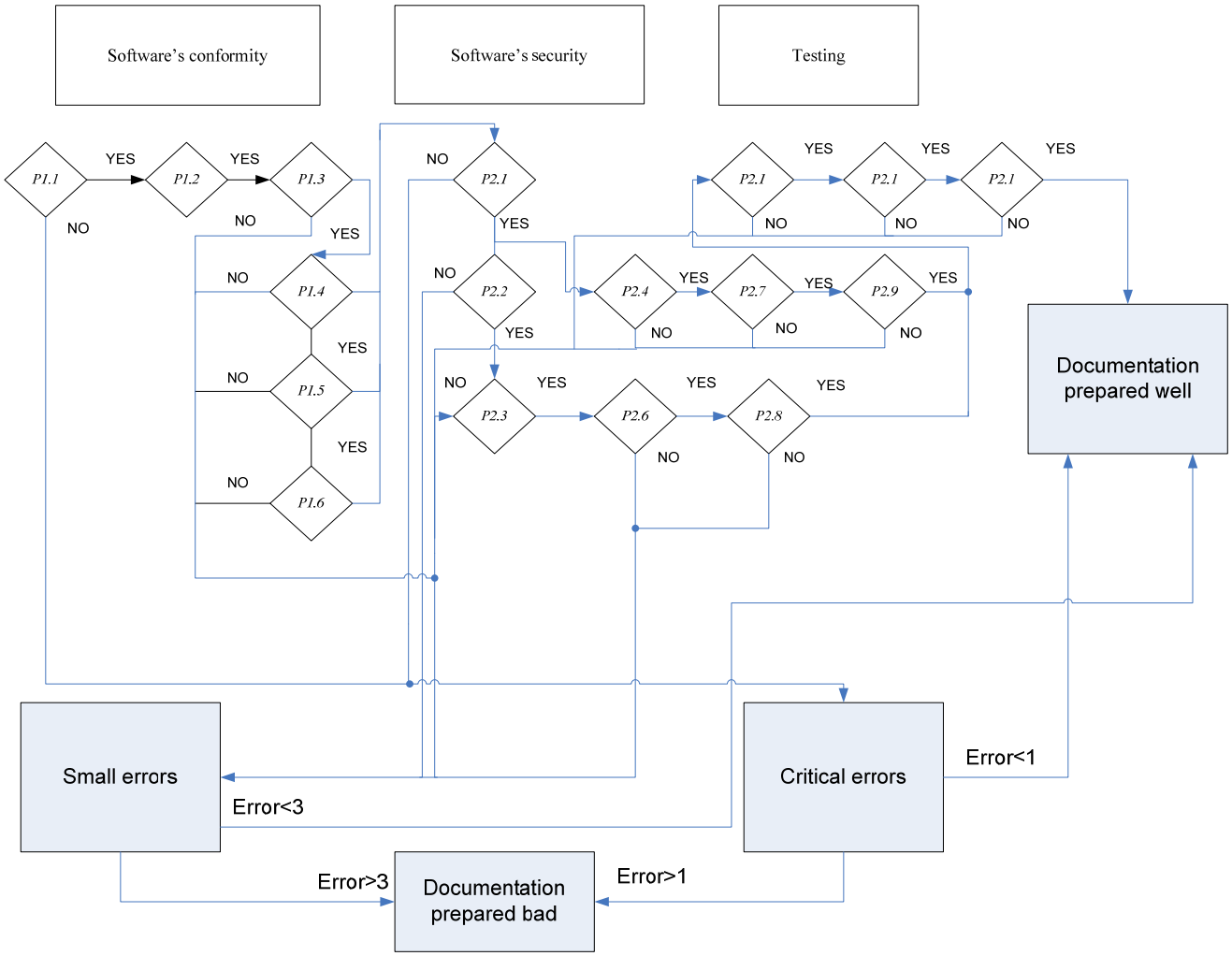

Fig. 2 Universal questionnaire

The main purpose of the functionality investigation is to check whether measurement devices software operates according to the presented documentation.

What concerns the functionality investigation, four investigation sub-stages could be defined [6]:

1. User interface investigation:
   - Investigation of a user interface menu block scheme using a manufacturer's scheme;
   - Verification of user interface protection;
2. Communication interface investigation;
3. Data processing software investigation:
   - Investigation of measurement devices software functionality when nominal value data sets are used;
   - Investigation of measurement devices software functionality when boundary value data sets are used;
   - Investigation of measurement devices software functionality when dynamically changing value data sets are used;
4. Investigation of software protection algorithms.

User interface investigation

Investigation of a user interface of many conventional measuring devices contains checking of the functionality of buttons and LCD. However, manufacturers claim that a great number of software errors unmask even when measuring devices are under normal conditions of use. As an example, menu navigation of a user interface can be presented [7]. The situation may occur when LCD displays wrong parameters or data. This can lead to misunderstanding or even worse – to economical problems. Further the Measurement Instruments Directive states: the indication of any result must be clear and unambiguous and accompanied by marks and inscriptions required to inform the user on the significance of the result. The presented result must be easy readable under normal conditions of use. In connection to this, the author has proposed an investigation method of a user interface of conventional measuring devices and developed it in details for heat meters as widely spread measuring devices. An automatic procedure realizing the proposed method is described. It could be used in meters manufacturing and/or type approval phases.

Generally, user menu of a conventional measuring device is controlled by means of control buttons. In this case, verification of the user menu functionality can be performed by simulating operation of control facilities and checking output measurement data or parameters directly on the LCD indicator or receiving them via a communication interface and processing on PC.

As a typical example, the verification procedure of user menu of heat meters as widely spread measuring devices is presented hereafter. User interfaces of many heat

meters are controlled by signals called “short” and “long”. “Short” signal is produced by pressing the control button for less than 2 seconds. “Long” signal is produced by pressing the control button for more than 2 seconds. By means of these signals one can switch from one menu item to another and output to LCD different data or parameters, e.g. the quantity of the heat consumed, flow rate of the heat conveying liquid passing through a heat meter, etc.

For automatic control of a user interface by means of PC, external contacts of control buttons are required. To these contacts, an external function generator or DAQ board controlled by PC could be connected. Then controlling the generator or DAQ board, the “short” or “long” signals could be simulated.

The second step of verification would be data acquisition from LCD indicator. The following three techniques are known:

- Taking digital photos of LCD indicator;
- Shooting LCD indicator;
- Processing signals from a communication interface.

The first and the second cases are commonly used in industry, for example, by defect diagnosis of wood splint panels. However, implementation of these approaches in the event of verification of a user interface of a measuring device is the third case based on processing signals from a communication interface.

Usually, the manufacturers of measurement devices provide only a user menu block diagram in the user’s guide. Development of a user menu graph would require few man-hours in comparison with man-hours required for the design project of a heat meter. However, the graph would facilitate verification procedure.

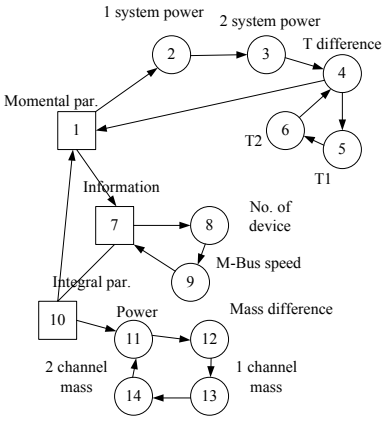

Fig. 3 User interface menu graph

In order to find the paths of the graph the method called “Depth first search” can be used. In this case, all vertices of the graph are treated as new (unvisited). Suppose the search begins from vertex v_0. At the beginning, this vertex is treated as not new (visited). Second step is to find vertex u contiguous to vertex v_0. If vertex u is not new, then finding of the path follows from it. When vertex v_n that has any contiguous vertex is reached we must return to the vertex from which we passed to vertex v_n and continue the path finding. Vertex v_n is depleted. The full path is found when vertex v_0 becomes depleted. In our case, the full paths can be written as an array:

[1 2 3 4 5 6 4 1 7 8 9 7 10 11 12 13 14 11 10 1]

Here, numbers correspond to the menu sections and subsections. All other paths can be found in the same manner choosing different first vertex.

The user menu verification diagram is presented in Figure 4.

PC calculates all paths and creates test sequences for the graph, which describes the user menu diagram. A test generator chooses test sequences randomly. It must be highlighted that the number of sequences will be equal to the number of graph vertices. Control software generates appropriate commands for the chosen test sequence and sends them to an external function generator that elaborates signals for simulation of user menu navigation button signals (“short” or “long” signal). After each pass from one menu subsection to another, indicated data on LCD must be read through a communication interface. Decision making software compares the test sequence sent to the heat meter and data received from it.

If fault is detected, decision making software informs in which menu section it was found. If verification is made during manufacturing of a measuring device, user menu software must be corrected and verified once more. If verification is made at a type approval stage, the decision must be made whether fault is critical or not, i.e. whether user menu fulfills MID requirements or not.

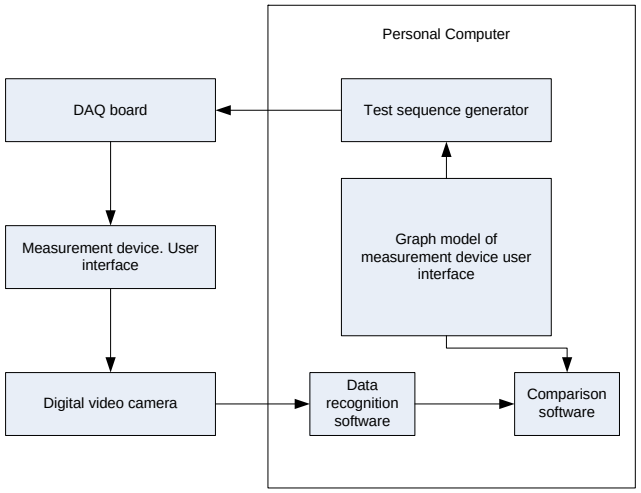

Fig. 4 User menu verification diagram

Communication interface investigation

According to MID, commands received through a communication interface cannot influence measurement device software and measurement data. For the detailed investigation of a communication interface, the specially created command lists presented by the manufacturer must be used. The communication interface verification block scheme presented in figure 5.

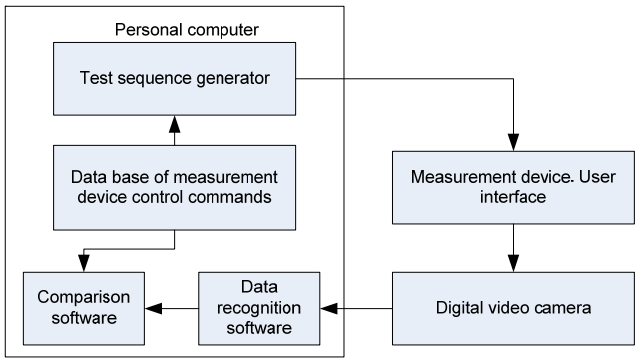

Fig. 5 Communication interface verification block scheme

Investigation of data processing software

The last and most important sub-stage of the measurement instruments software functionality investigation is a verification data processing part of software. For this investigation, a mathematical model of measurement devices data processing software must be created. The investigation of data processing software is based on "black box" principles; a structure scheme of investigation is presented in Fig. 6

In this case, a test data sequence generator presents data sets for a mathematical model, which can be called reference software, and real measurement devices software. Results of a mathematical model and real measurement device are compared. If no errors are observed, measurement devices data processing software is considered as metrologically reliable.

Measurement devices software functionality can be investigated using "black box" principles. Each software module can be described as a function depending on input data. For this investigation, specific test sequences required [8,9,10]. In this work, 4 different input data sets were described:

- Boundary input data sets. For investigation of measurement devices software functionality when measurements are performed using boundary values of measurement converters. For example, measurement of the minimum or maximum temperature (Fig. 7);

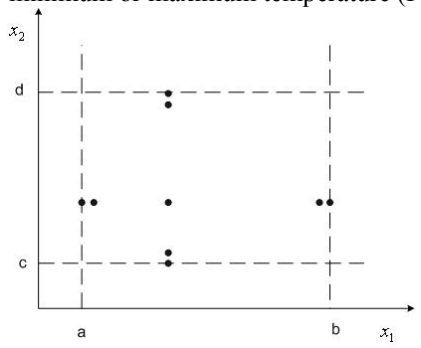

Fig. 7 Boundary data sets

- Faults in boundary data sets show how software protection algorithms respond to single faults. For example, negative temperature of heat conveying liquid (Fig. 8).

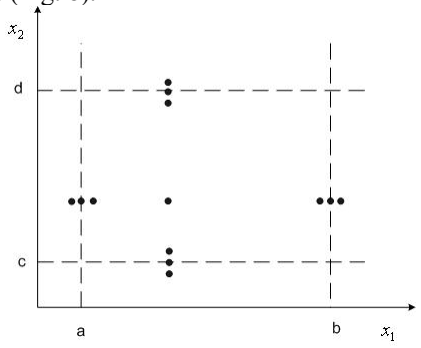

Fig. 8 Faults in boundary data

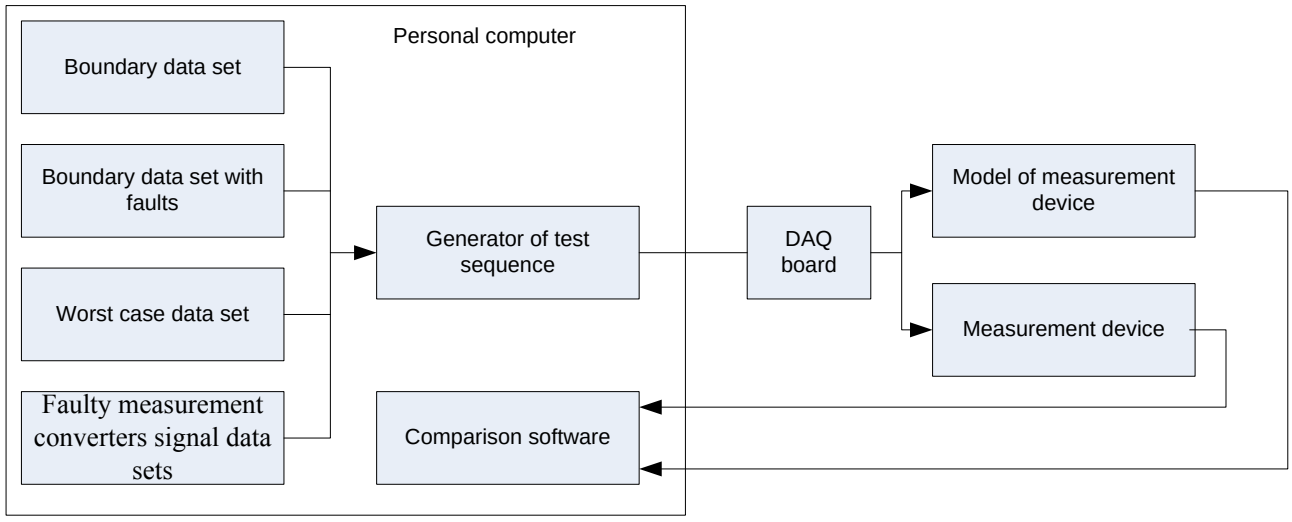

Fig. 6 Investigation of data processing software

- Worst case boundary data sets demonstrate how software protection algorithms react to regular logical faults in measurement data (Fig. 9).

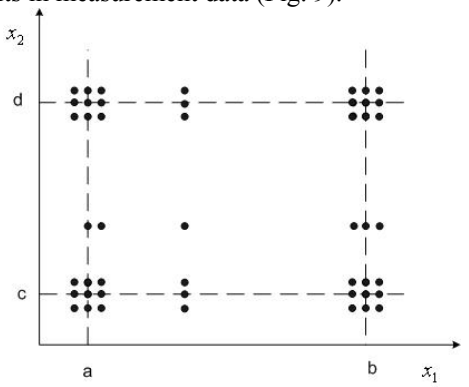

Fig. 9 Worst case boundary data sets

- Faulty measurement converters signal data sets illustrate how software protection algorithms react to illegal signals received from measurement converters (Fig. 10).

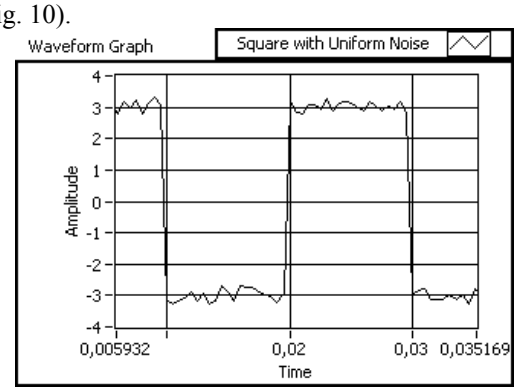

Fig. 10 Faulty measurement converters signal data sets

5 Investigation of the heat meters calculator applying a measurement devices software reliability method

Results calculated by heat meters software can be influenced through communication or user interfaces, temperature or flow sensors. Therefore, for investigation of the heat meters calculator the following experiments shall be accomplished:

1. Investigation of temperature measurement software
2. Investigation of flow measurement software
3. Investigation of measurement data processing software
4. Investigation of software protection algorithms.

For these investigations, specific test data sequences were used. Simulation of temperature sensors was performed using resistor bridge load. Load limits of a standard heat meter are $500 \div 800\ \Omega$. Flow sensors were simulated by external pulse generators. Pulse parameters of a standard heat meter are as follows: pulse repetition frequency within the range is $0 \div 1000$ Hz, pulse amplitude is 3 ± 0.3 V and pulse duty cycle is $20 \div 80$ %.

A block scheme of the used verification equipment is presented in Fig. 11. Resistor bridges R1, R2 simulate temperature sensors. Functional generators G1 and G2 simulate flows of the flow and return liquids. Software, written in the C language and implemented using LabWindows/CVI controls all devices.

Verification is performed in the following order. Control software starts generators and a universal counter which begins calculation of the pulses fed to the heat meter calculator. Pulse generators must be stopped after not less than 1000 pulses are counted up. Then control software reads data from all devices via a communication interface, i.e. temperatures of a flow and return liquids, volumes of the passed liquids, quantity of the heat given up calculated by the heat meters calculator and number of pulses counted by a universal counter.

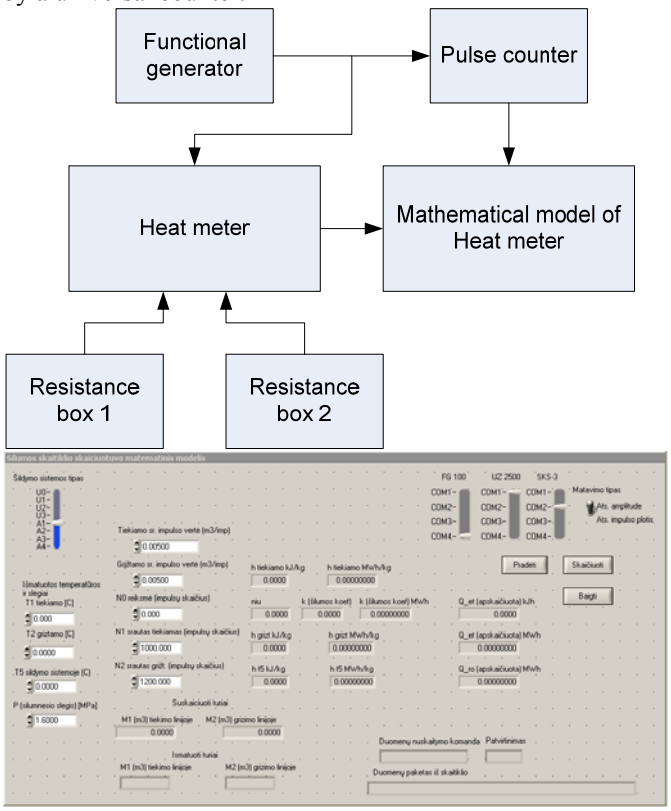

Fig. 11 Verification scheme and a user interface

Then the reference volume of the passed liquid is calculated according to the formula:

$$V = N\,k\,. \qquad (2)$$

Here, N – number of the counted pulses, k – value of one pulse, i.e. pulse/m^3. The reference quantity of the heat given up can be calculated applying the expressions presented in the OIML recommendations [7] i.e.:

$$Q_{ref} = k\Delta\Theta V\,. \qquad (3)$$

Here, Q_{ref} – reference quantity of the heat given up, $\Delta\Theta = \Theta_{flow} - \Theta_{ref}$ temperature difference between the flow and return of the heat exchange circuit. V – volume of the passed liquid calculated using expression (2), k – heat coefficient calculated using the following expression

$$k(p, \Theta_{flow}, \Theta_{return}) = \frac{1}{\nu}\frac{\Delta h}{\Delta\Theta}. \qquad (4)$$

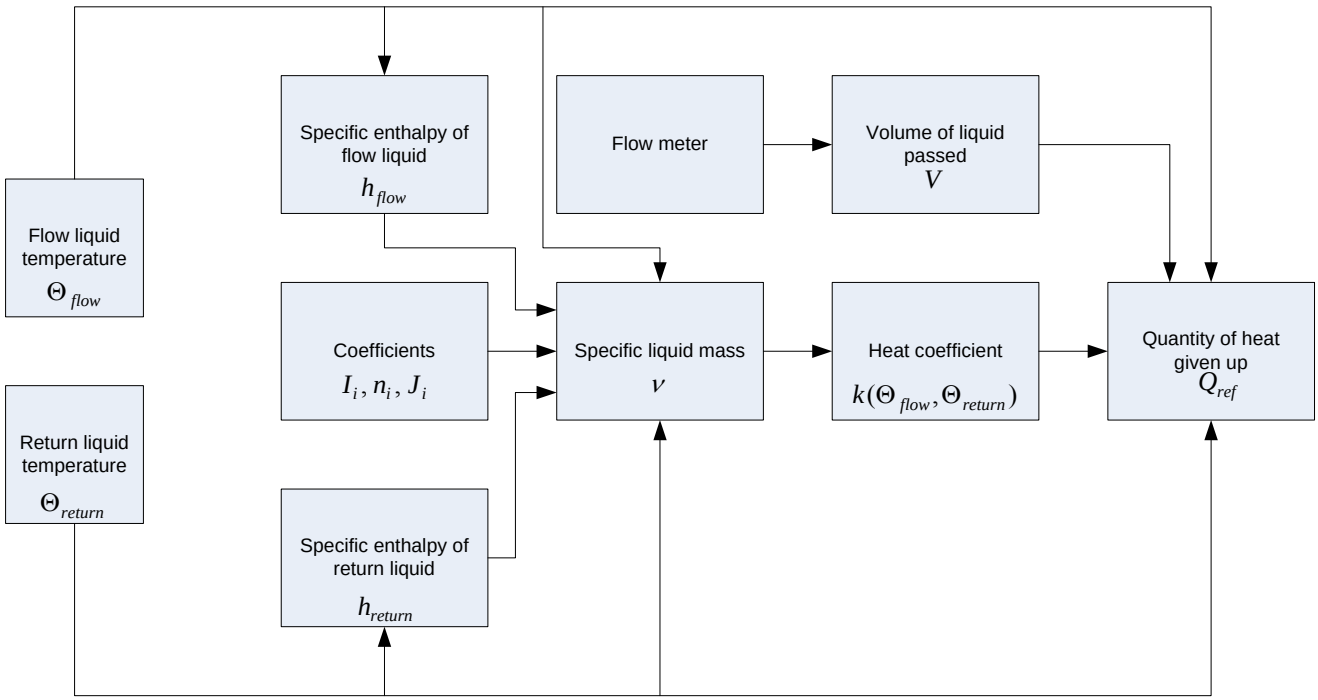

Fig. 12 Mathematical model of the heat meters calculator

Here, $\Delta h = h_{flow} - h_{ref}$ – specific enthalpy difference between the flow and return enthalpies, ν – specific liquid mass, p – pressure of the liquid.

Mathematical model of the heat meters calculator is depicted in Fig 2. After calculations of the reference quantity of the heat given up, relative error of heat meter measurement results is estimated according to the following expression

$$E = \frac{Q_{ref} - Q_m}{Q_{ref}}. \qquad (5)$$

The maximum permissible error of the heat meter calculator is

$$E_c = \pm(0.5 + \Delta\Theta_{\min} / \Delta\Theta). \qquad (6)$$

Here, $\Delta\Theta_{\min}$ – minimum temperature difference – parameter of the specific heat meter.

An experiment was made in order to verify metrological reliability of temperature measurement software. Verification was carried out in the following order:

- Resistor bridges R1, R2 simulate temperature sensors;
- Heat measuring system measures Θ_{1m}, Θ_{2m}, $\Delta\Theta_m$ temperatures;
- Comparison between the calculated Θ_1, Θ_2, $\Delta\Theta$ and measured Θ_{1m}, Θ_{2m}, $\Delta\Theta_m$ temperatures was made.

Results are presented in Table 3. Measurements 1-9 were made using boundary value analysis test cases. Measurements 10-13 were made using "faulty" test cases. Each time when heat measuring systems software receives a "faulty" signal from temperature sensors, it must generate a warning on fault.

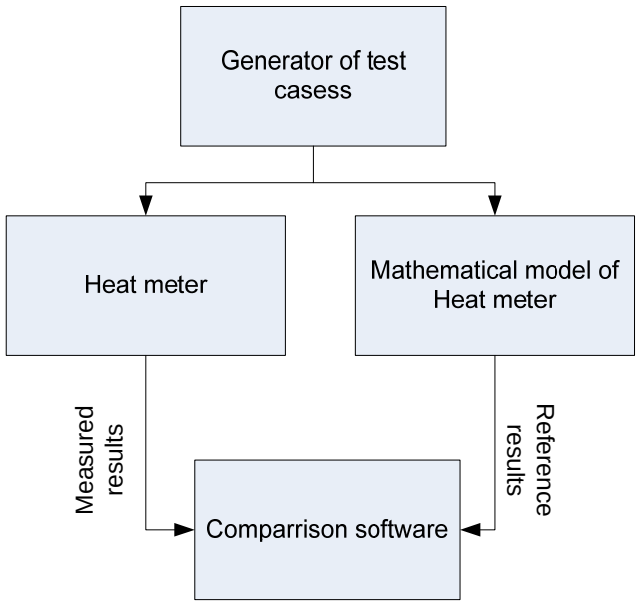

Fig. 13 Experimental scheme

Here, f_{in}= Y shows that "faulty" data was sent to measurement systems software, whereas f_{in}= N demonstrates that correct data was sent to measurement systems software. f_{ms}= Y shows that measurement systems software detected "faulty" data, and f_{ms}= N indicates that software failed.

Table 3 Experimental results with boundary value analysis

No.	Θ_1, ºC	Θ_2, ºC	$\Delta\Theta$ ºC	Θ_{1m}, ºC	Θ_{2m}, ºC	$\Delta\Theta_m$ ºC	f_{in}.	f_{ms}
1	80	0	80	79,91	0	79,91	Y	Y
2	80	8	72	79,92	7,59	72,33	N	N
3	80	40	40	79,86	39,52	40,34	N	N
4	80	152	-72	79,9	155,49	-75,59	Y	Y
5	80	157	-77	79,91	156,56	-76,65	Y	Y

6	3	40	-37	2,35	39,55	-37,2	Y	Y
7	8	40	-32	7,76	39,49	-31,73	Y	Y
8	152	40	112	152,07	39,5	112,57	N	N
9	160	40	120	160,8	39,5	121,3	N	N
10	168	40	128	168,21	39,5	128,71	Y	N
11	-8	80	-88	0	79,65	-79,65	Y	Y
12	80	-8	88	79,9	0	79,9	Y	Y
13	80	164	-84	79,97	163,6	-83,63	Y	Y

Experimental results illustrated that measurement systems software detected negative difference between temperatures. At 10 measurements, software detected no faulty value of flow liquid temperature. For detailed verification, the worst case testing case was used. Experimental results are presented in Table 4. Here, only the test cases where measurement systems software failed are provided.

Table 4 Experimental results with worst case testing

No.	Θ_1, ºC	Θ_2, ºC	$\Delta\Theta$ ºC	Θ_{1m}, ºC	Θ_{2m}, ºC	$\Delta\Theta_m$ ºC
1	160	0	160	159,9	0	159,9
2	152	0	152	151,93	0	151,93
6	8	0	8	7,47	0	7,47
17	168	-8	176	167,9	0	167,9
18	160	-8	168	159,9	0	159,9
19	152	-8	160	151,9	0	151,9
21	168	160	8	167,9	159,8	8,1
27	160	-8	168	159,7	0	159,7
28	168	0	168	167,8	0	167,8
31	168	152	16	167,8	151,7	16,1
35	8	-8	16	7,61	0	7,61

As it was expected, heat metering systems software detected fault only at negative temperature differences and minimum temperature values. But it failed with exceeded temperature values and minimum or even negative return liquid temperature values. Such performances of measurement systems software contradict with the essential measurement systems requirements presented in the Measurement Instruments Directive. Besides, it can be stated that measurement results obtained with the measurement system using such software can be falsified or incorrect, i.e. metrologically unreliable.

The standard verification procedure of heat meters covers the estimation of the maximum permissible errors of calculators. As a result, it is not possible to answer the following questions: how the heat meter calculator will respond to the faulty signals received from sensors and how these faults influence measurement results?

In order to answer these questions, an experiment related to verification of heat meters data processing software using the type approved heat meter was carried out.

Primarily, the ramp and random flows were simulated. Parameters of corresponding pulse sequences were:

- Increasing pulse repetition frequency within the range $0 \div 1000$ Hz ,
- Randomly changing pulse repetition frequency within the range $0 \div 1000$ Hz .

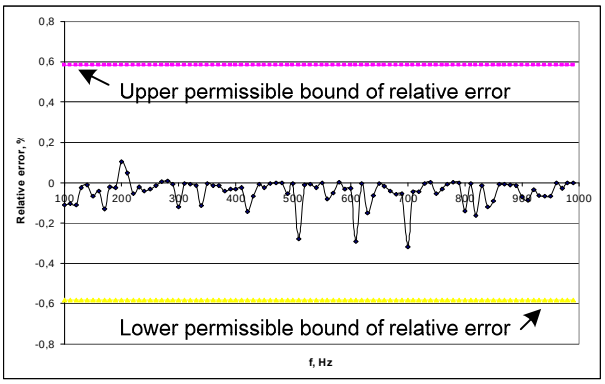

Fig. 14 Relative error of the heat meter calculator with the ramp flow

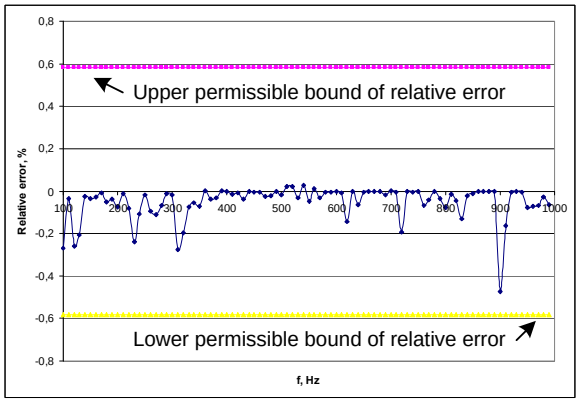

Fig. 15 Relative error of the heat meter calculator with the random flow

Results presented in Fig. 14 and 15 demonstrate that the flow type (ramp or random) have no influence on relative error of the heat meter calculator in a sense that it remains within the maximum permissible error range.

The last experiment showed how heat meters software protection algorithms operate when faulty signals from measurement converters are received. For this experiment, flow sensors signals of the heat meter were chosen. The manufacturer declares that the signal received from the flow sensor can have the following parameters: amplitude can vary $3 \pm 0,3$ V, duty cycle – in bounds of $20 \div 80$ %. According to this, the test sequences were generated:

- Pulse signal ($3 \pm 0,3$ V), with varying amplitude in form of a triangle,
- Pulse signal ($3 \pm 0,3$ V), with varying amplitude in form of a rectangular,
- Pulse signal ($3 \pm 0,3$ V), with varying amplitude in form of a sine,
- Pulse signal ($3 \pm 0,3$ V), with varying amplitude in form of uniform noise,
- Filtered pulse signal.

The figure below demonstrates how varying pulse amplitude affects the calculated volume error of heat conveying liquid. When test sequences of the filtered pulse signal (Test6) were used, the calculated volume error of heat conveying liquid was similar to the maximum permissible error. Thus, heat meters software receiving such a "faulty" signal will work normally and can be

considered as metrologically reliable. With other test sequences (Test2–Test5), the maximum permissible error was exceeded about two times. The experimental results illustrate that the test sequence with varying amplitude in form of uniform noise has the greatest influence on the calculated results of heat meters.

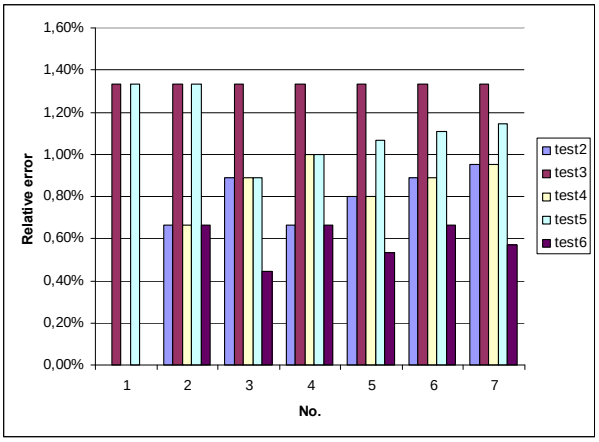

Fig. 16 Calculated volume error of heat conveying liquid

The last experiment was made with "faulty" conveying liquid temperature values. Nominal values of flow and return liquids are 80 and 40 ^{o}C, whereas a standard temperature measurement range of heat meters is 0-160 ^{o}C. Results are presented in Table 1 and Figure 7. Analysis of the experiment results highlighted that in the cases of particular combinations of temperature values (but still permissible values) of conveying liquids the quantity of heat given up is calculated wrongly, and the main thing is that the software gives no report about the faults.

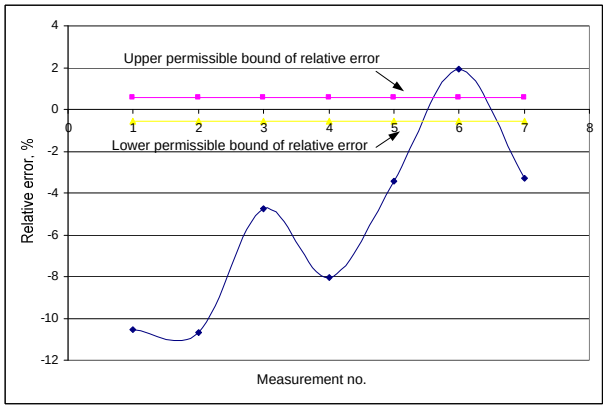

Fig. 17 Relative error of the heat meter calculator with faulty temperature values

Consequently, experiments with the type approved heat meter pointed out that in some cases, especially when parameters of the signals (pulse amplitude and duty cycle) received from sensors reach marginal but still permissible values, the value of heat given up is calculated by the heat meter calculator wrongly. It would be difficult or even impossible to get such results by the standard heat meters verification procedure. And this shows that after adoption of the Measuring Instruments Directive and starting verification of data processing software of heat meters some of them fail to pass the type approval stage.

Conclusions

1. A method for estimation of measurement devices software metrological reliability has been developed. The method enables to test the implemented algorithms, functionality and documentation of the measurement device.
2. Analysis of potential errors that can influence measurement devices software has shown that the method of measurement devices reliability must contain the following two stages: estimation of measurement devices documentation and estimation of measurement devices software functionality.
3. Experimental results have shown that the proposed method is appropriate for estimation of metrological reliability of measurement devices software.

Examination has confirmed that simulation of heat meters measurement converters signals identifies logical faults of the software protection algorithm:

- Logical faults in the software protection algorithm when temperature sensors measure higher than the maximum permissible temperature of flow liquid or negative temperature of return liquid.
- Incorrect volume result and "no faults detected" by the software protection algorithm when simulating a "corrupted" signal of a flow sensor.

References

[1] Directive 2004/22/ec of the European Parliament and of the Council of 31 March 2004 on Measuring Instruments. *Official Journal of the European Union*. – 2004. [europa.eu.int].

[2] Software Requirements on the Basis of the Measuring Instruments Directive. *WELMEC guide 7.1*. – 1999. [www.welmec.org].

[3] **Lazić, Lj., Velašević, D., Mastorakis, N., A** framework of integrated and optimized software testing process, WSEAS TRANSACTIONS on COMPUTERS, Issue 1, Volume 2, 15-23, January 2003.

[4] **Čitavičius A., Knyva V., Knyva M.** Investigation of Heat meters Software Functionality. *Digest of Conference on precision electromagnetic measurements* (CPEM 2006), Torino, Italy. – 2006, P.418-420.

[5] **Knyva V., Knyva M.** Investigation of Heat Meters Data Processing Software *Proceedings of the 2007 WSEAS Int. Conference on Circuits, Systems, Signal and Telecommunications*, Gold Coast, Australia, January 17-19, 2007

[6] **Knyva V., Knyva M.** Problems of Heat Meters Software Verification. *WSEAS Transaction on Systems*. – Athens:WSEAS Press, Issue 5, Volume 6, 2007

[7] **Čitavičius A., Knyva V., Knyva M.** Verification of User Interface of Supply to the Customer by Mains Measuring Devices. *WSEAS Transaction on Systems*. – Athens:WSEAS Press, 2006. – Vol. 5, No. 10, P. 2450-2455.

[8] **Francisco J. G., Veronica R., Virtudes T.** Pseudo-random sequence generators based on cellular automata and bent functions . *WSEAS Transaction on Compuiter Research*. – Athens:WSEAS Press, Issue 5, Volume 6, 2008

[9] **Cox M. G., Harris P. M.** Design and use of reference data sets for testing scientific software. *Analytica Chimica Acta*. – 1999. – Vol. 380, No. 2. – , P. 339-351.

[10]**Niculescu E., Purcaru D., Niculescu M.** Worst Case Analysis of the Analog Circuits, *Proceedings of the 11th WSEAS International Conference on CIRCUITS*, Agios Nikolaos, Crete Island, Greece, July 23-25, 20