ISSN 0005–1144
ATKAAF 48(3–4), 99–112 (2007)

Siniša Marijan

Proprietary Embedded Control Electronics for TMK2200 Ultra-low Floor Tramcar

UDK 681.51:629.43
IFAC 5.7.2

Original scientific paper

The paper deals with several proprietary embedded hard real-time systems for ultra-low floor tramcar type TMK2200. The described systems were built by means of platform-based modular hardware and software components. Both hardware and software platforms and components are described. Particular attention is given to vehicle control unit, control electronics for traction units, auxiliary supply units and visualization unit. Furthermore, the concept of tramcar communication networks is presented. The communication between control units is based on CANbus and CANopen communication protocol. Obsolescence problems during development and production phases are also discussed.

Key words: tramcar, hard real-time system, proprietary embedded system

1 INTRODUCTION

This work is related to the development of ultra-low floor tramcar type TMK2200 for the city of Zagreb. The project and production were managed by several KONČAR companies. There were many development teams, both domestic and international, involved in the project. KONČAR – Electrical Engineering Institute was responsible for the development of several proprietary solutions: redundant vehicle control unit, traction units, static converters for auxiliary power supplies and man-machine interface. Additional responsibility was the choice of appropriate communication busses and, accordingly, the system integration of all electronic control units. This paper deals mostly with modular control electronics for the abovementioned proprietary solutions.

The systems, apart from man-machine interface based on PC architecture, were developed around two different proprietary platforms. The described platform architectures recently enabled the development of several, low-volume, different products for the traction and power engineering applications, [1–5]. Such systems, besides fulfilling common real-time demands [6], are supposed to have a long life, i.e. they should be operable for more than two decades with minimum maintenance costs. Accordingly, they can be compared to military, avionic or space solutions. One of the biggest problems during design and maintenance of the long-life embedded systems is caused by component obsolescence. Therefore, this issue is also addressed in the paper.

Section 2 presents tramcar electronic control systems and platform-based architectures used to develop proprietary control units. Section 3 describes proprietary solutions hardware in more details, while section 4 gives highlights on software platform concept. Section 5 addresses vehicle communication networks. Obsolescence problems are discussed in section 6. Section 7 gives some experience facts encountered during the development and production, while section 8 concludes the paper.

2 TRAMCAR ELECTRONIC CONTROL SYSTEMS

Taking into account electronic control units (ECUs) integrated into this vehicle, each standalone unit should be mentioned: main vehicle control unit (VCU), 3 traction control units (TCU), 2 auxiliary power supply control units (ASU), 3 brake control units (BCU), 2 heating/ventilation and air conditioning control units (HVAC), 6 door control units (DCU), 7 bogie control units (BGCU), 1 visualization unit/man-machine interface (MMI). Physical position of the above described units is given in Figure 1. The scope of proprietary solutions is emphasized with green or light gray shading. The delivery of remaining units (red or dark gray) was the responsibility of other suppliers.

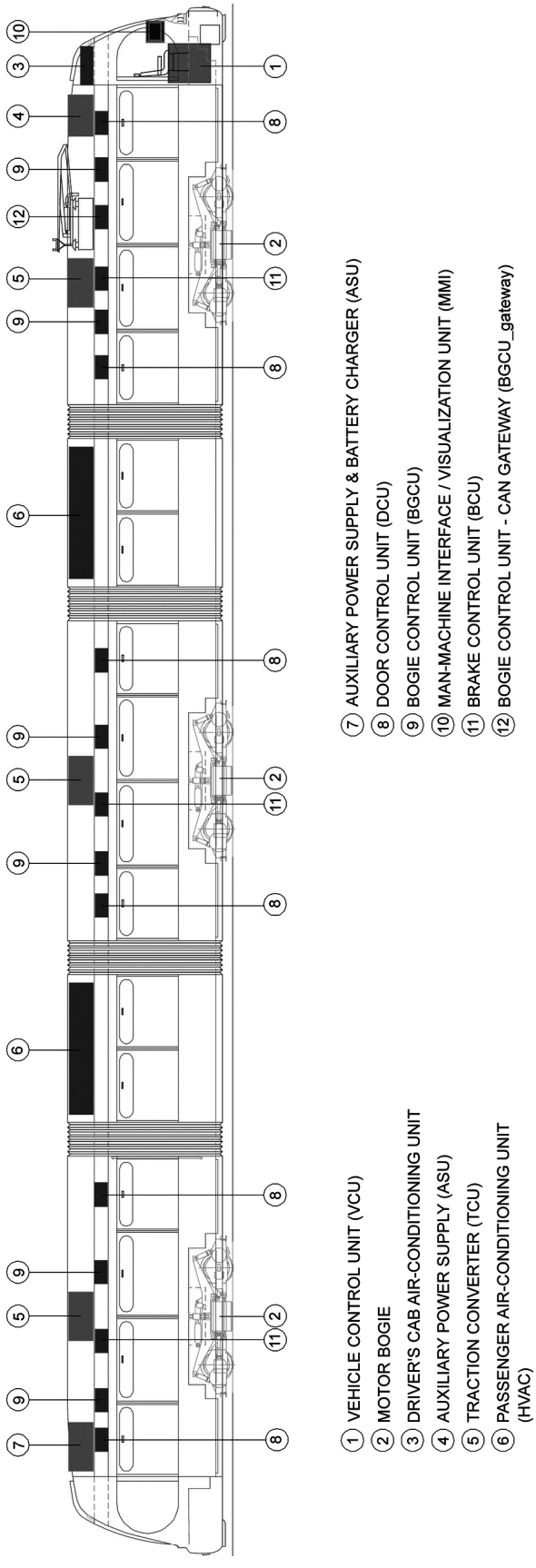

Fig. 1 Physical position of electronic control units (ECUs) in the tramcar

2.1 Platform Architecture for Proprietary Solutions

Each proprietary control unit is based on one of two previously developed platforms. The basic idea during the development of described platform architectures was to build sufficient number of hardware and software components that enable product integrator to almost completely build new system with standard components [7–11]. Apart from the appropriate international standards [12] and industrial experience, the following facts have influenced hardware specification and the development process: proven solutions from previous projects [2–5], modular concept, double or single euro-card format of electronic modules, long time availability of vital electronic components, minimum number of different processors and controllers, simple system extension or reduction, no outsourcing in the development, high reliability, possible integration of COTS (commercial-of-the-shelf) electronic modules into the system, results of the development should be easily adapted to the future similar, or even completely different projects. The result is that all components are completely in-house developed. The only exception is power supplies purchased as custom designed units.

2.1.1 Platform I Architecture

This platform contains a large amount of electronic modules and accessories, what enables different configuration possibilities. Apart from tramcar control units described in this paper, there are several other products already developed on this platform [2–5].

Processing units were developed around 3 processor architectures. General purpose processing unit (GPPU) with serial communication channels is built around 68 K architecture [13], digital signal processing (DSP) unit for demanding control algorithms is built around C2000 architecture [14–15], while additional communication unit that supports CANbus interface is developed around 8051 architecture [16, 17]. It is obvious that applied processor architectures are matured, established on the market and still have acceptable future roadmaps. Generally, the philosophy during the development of this platform architecture was borrowed from the military industry – use, if possible, only proven and established processors that are expected to have a long life and let others discover the flaws of newly introduced components.

2.1.2 Platform II Architecture

The basic idea during the development of this platform was similar to the first one: component-based hardware and software. As a result, hardware components are also based upon modular principle, but there are no units identical to those used in platform I. Central processing unit is based on 2 identical microcontrollers [20] on the same board. Additional processing capability, if needed, is achieved by means of DSP unit (C2000 architecture). The platform is aimed for developing of embedded control systems such as: auxiliary power supplies, battery chargers, multi-system power converters, ventilation of traction converters etc.

2.1.3 Software Architecture

The platform-based approach enables the application engineer to have a large number of necessary software components available in advance. In this case the product definition phase is much easier, whereas it is easy to identify specific components that are possibly missing and should be developed. Thus, the application programming can start before the complete product hardware has been defined. The demands on software platform were: the application engineer should concentrate on application and not on coding, application engineer must have all the needed software components available in the appropriate graphical form, the whole application must be developed in a block-diagram-user-oriented language, integrated development environment (IDE) should be the same regardless of processor architecture, the only co-operation between application engineer and embedded software developers should occur in the case when a new component is requested or an existing component performs incorrectly. Further details related to the software concept are given in section 4.

3 TRAMCAR CONTROL UNITS BASED ON PLATFORMS

TMK2200 tramcar control units based on the presented platform I are Vehicle Control Unit (VCU) and Traction Control Unit (TCU), while static converters (ASU) are based on platform II.

3.1 Vehicle control unit

Redundant VCU frame consists of two main 19" mounting racks and two smaller additional racks that are used together to house electronic modules, power supplies, cables and motherboards [18]. Electronic modules have standard, double or single-height, euro-card format. VCU is almost completely based on proprietary hardware and software solutions, and is designed for rolling stock applications [12]. The block diagram of the VCU is

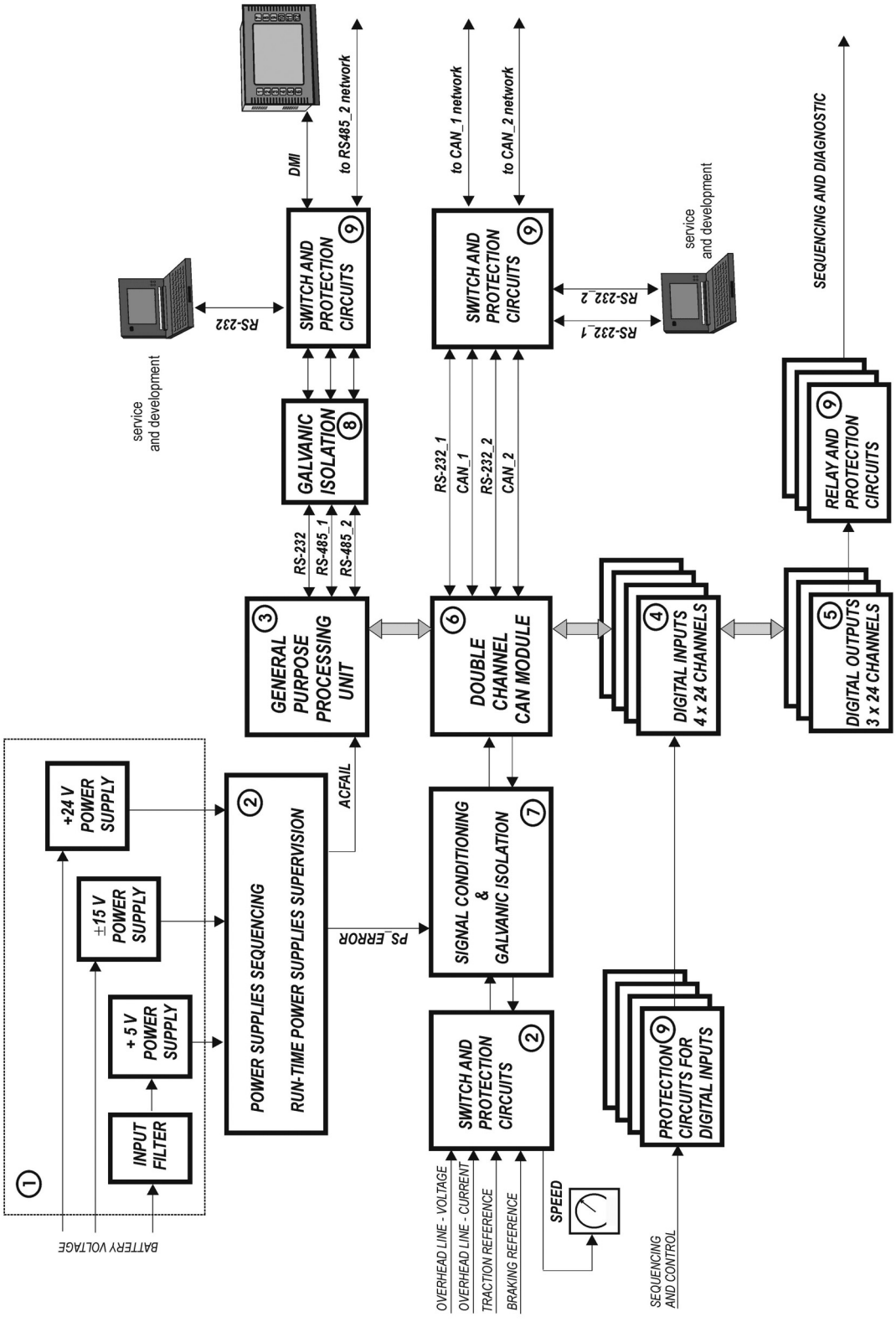

Fig. 2 Block diagram of Vehicle Control Unit (one channel)

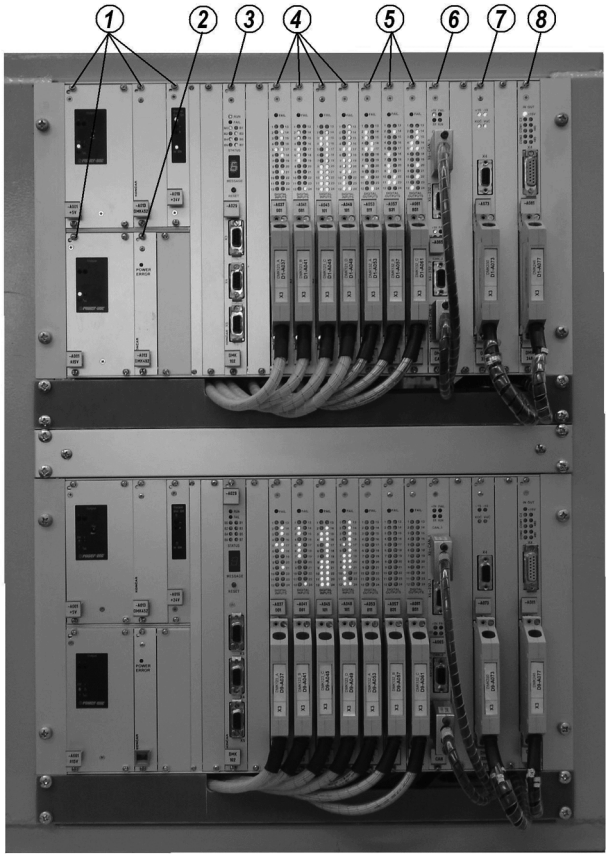

Fig. 3 Vehicle Control Unit (VCU)

given in Figure 2, while the photography of main racks is given in Figure 3. The first rack is used during normal operation as an active system, while the second one is used for system redundancy. In the case of malfunction (failure), there is a possibility to switch to the redundant system, i.e. to continue operation without reducing vehicle functionality. In this way the availability of the whole system is significantly increased. Small racks (not shown in Figure 3) consist of protection circuits, relay interface modules and circuits that enable switching between active and redundant system. These modules are designated with number ⑨ in Figure 2. The following description of modules inside VCU gives, together with Figure 2, a system overview:

① input filter, 5V, ±15V and 24V power supplies;

② module that supervises and sequences power supplies;

③ general purpose processing unit (GPPU) supports all necessary control, sequencing, communication and diagnostic functions. The majority of other modules are connected with GPPU through motherboards;

④ digital input is a universal peripheral module connected to local I/O channel that is controlled by GPPU. It supports up to 24 opto-isolated digital inputs;

⑤ digital output is also a universal peripheral module connected to local I/O channel that is controlled by GPPU. It supports up to 24 opto-isolated digital outputs;

⑥ twin-channel intelligent CAN communication module with several analogue and digital inputs or outputs;

⑦ electronic module for conditioning and galvanic isolation of analogue signals. It is used for measurement of overhead line voltage and current. Analogue signals that determine traction and braking references (obtained from mechanical controller) are also conditioned by means of this module. These signals are, after conditioning, sent to the CAN module where analogue to digital conversion is performed. Furthermore, module supports galvanic separation for analogue signal that drives speed indicator. CAN module produces (D/A conversion) this signal under the control of GPPU;

⑧ module that performs the galvanic isolation for two RS485 communication channels and one RS232 channel, all of them located on GPPU.

As the system was developed to allow future system expansion or reduction, some additional signals and possibilities that are already supported have not been mentioned. Furthermore, as shown in Figure 3, there are several spare slots available for the expansion and adaptation according to the additional requirements in the case of necessity.

Apart from redundant channel that can be used in a case of failure, the high error detection coverage has been required from operational channel to put the system in the fail-safe operation in the case of malfunctioning. Fail-safe means that no undefined system state is allowed, and therefore appropriate actions will activate stopping the vehicle, disconnecting the power line and sending appropriate message to the driver. Therefore, a lot of additional hardware and software mechanisms have been implemented to detect and handle different types of errors [18].

VCU also supports direct communication channel for visualization unit located in the driver's cab [19]. The communication is established over RS-485 physical layer with Modbus communication protocol. The MMI application program supports several modes of operation for both exploatation and maintenance purposes. MMI is based on in-

Fig. 4 Example of Man-Machine Interface (MMI) layer

dustrial PC-compatible computer with XP embedded operating system. The application program (also proprietary solution) is designed as object-oriented multithreaded software, which enables visual elements to be easily changed either at the design stage or during the product lifetime. One of the several available layers is shown in Figure 4.

3.2 Traction Control Unit

TCU is a part of traction, air-cooled, IGBT power converter [2]. Each converter powers two induction traction motors, also developed by KONČAR. A photo of the electronic part of TCU is given in Figure 5. It consists of two racks, (second small rack not shown), that are used to house electronic modules, power supplies and motherboards. Although TCU has 5 processors integrated, the core is GPPU equal to the one applied in VCU. The following modules reside inside TCU:

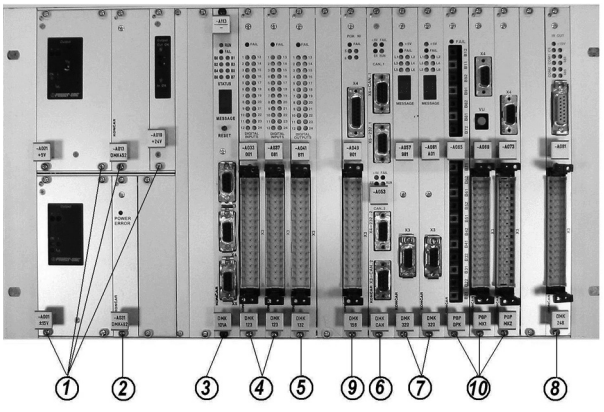

Fig. 5 Traction Control Unit (TCU)

①,②,③,④,⑤,⑥ and ⑧ are identical to those in VCU;

⑦ two identical digital signal processing (DSP) units with several analogue and digital input/output channels. First one controls traction motors, while the second one is used for intelligent control of breaking resistor;

⑨ digital speed measurement module gets signals from incremental encoders and makes them available to the GPPU,

⑩ measurement, control, optical interface, relay and protection modules establish the connection with outside world and power electronics.

Relay and protection modules are positioned in second rack similar to the one used in VCU. Besides its main function, TCU is also used for some auxiliary services inside a particular part of the vehicle. Furthermore, in the sense of functionality, it is significantly coupled with BCU. It is the TCU that calculates and produces braking reference value for the hydraulic brakes. Apart from that, BCU and TCU together support slide protection functions. Main mounting rack of TCU consists of 17 electronic modules and 11 of them are the same as those used in basic mounting rack of VCU, Figure 3. At the same time 3 of the remaining 6 modules are parts of standard hardware components developed for platform I. It means that only 3 modules were developed specifically for TCU, so it can be concluded that, in this particular project, 82 % of modules are reused. Apart from reusability, the advantage of platform-based modular solutions can be illustrated with the following example. During the final project stage a demand for additional intelligent control of breaking resistor suddenly appeared. The problem has been solved by simply adding additional DSP processing unit. This would not have been possible without modular concept and without spare slots available in mounting rack.

3.3 Static Converters

There are two independent static converters used as auxiliary power supplies (ASU). One of them also supports the »battery charger« function and therefore is equipped with two identical central processing units described in 2.1.2. ASU are used for supplying HVAC units, ventilation of traction converters and all other auxiliary services. The ASU control and communication interface to the outside world is minimized to achieve its simplicity. They are even not members of any existing vehicle communication network.

4 SOFTWARE PLATFORM DETAILS

Software platform consists of 2 main parts: the integrated software development environment (IDE) for application programming and the system software [1–5].

4.1 Application programming

IDE comprises tools that enable application program development and documentation. The basic principles are the same regardless of the hardware, i.e. of the processor type [1–5]. Thus, all processors, controllers and even digital signal processors used in both platforms have the same IDE for the application program development. It is obvious that the same or similar environments (both from the application developer and the system software developer points of view) can help in reducing development costs. The application program is developed in graphical environment by means of block-diagram programming language conceptually similar to IEC 61131-3 FBD (function block diagrams) language. A result of the development is the application program executed according to real time requirements under the control of system software. Table 1 gives a lot of interesting facts about application programs for VCU and TCU implemented in GPPUs. It is obvious that the executable size of application programs and system software is very modest when compared with typical modern high-volume embedded system. It must be emphasized that only application programs for GPPU are considered here. The influence of other processing units (DSP, CAN communication, redundant channel) is not given in Table 1. It can be concluded that an average component is in described applications reused up to 36.6 times (total number of elements used in application program divided by number of different elements used in application program, i.e. 3260/89 = 36.6). Naturally, this value is only statistical, because in-depth analysis has shown that percentage of logical elements used in presented cases ranges from 40 % to 55 %. For example, the logical AND element is within

Table 1 **Software features of General Purpose Processing Unit (GPPU) used in VCU and TCU**

GPPU – SW features	VCU	TCU
Number of IDE available programming components (elements) for application program development	219	219
Number of different components used in application program	89	98
Total number of components used in application program	3260	2202
Re-usage rate of average component (total number of components in application program divided by number of different components used)	36.6	22.4
Re-usage rate of most frequently used components in application program: logical_AND / logical_NOT / logical_OR / arithmetical_SWITCH	365 / 321 / 282 / 240	178 / 161 / 125 / 217
Size of application program expressed in number of developed program modules – graphical representation	47	71
Size of application program expressed in total number of graphical sheets	198	112
Size of application program (kBytes)	216 (ROM) / 49 (RAM) 850 (NVRAM for logging)	121 (ROM) / 43 (RAM) 850 (NVRAM for logging)
Size of application program in assembly language lines of code (LOCs) but without allocated memory for static or dynamic data	35.000	21.000
Assembly language lines of code (LOCs) per program module – average value	744	295
Assembly language lines of code (LOCs) per program component – average value	10.7	9.5
Size of system software (real time kernel and system programs) in assembly language lines of code (LOCs) but without allocated memory for static or dynamic data	11.000	11.000

Table 2 **Distribution of SW components according to application domain**

Application domain of software components (programming elements)	Processor architecture		
	68K (platform I)	C2000 (platforms I and II)	8051 (platforms I and II)
logical functions	9,8 %	17 %	13.8 %
arithmetical functions – single precision	23.7 %	46 %	21.5 %
arithmetical functions – double precision	9.3 %	–	–
data format conversion or packing	2.3 %	2 %	3.1 %
platform related HW dependent components	10.7 %	4 %	12.3 %
general purpose and processing unit dependant components	14.8 %	15 %	18.5 %
conditional program execution	4.7 %	3 %	–
serial data communication (RS485, ModBus, CAN, ...)	12.5 %	–	14.6 %
multiprocessing	4.7 %	4 %	7.7 %
logging, monitoring and diagnostic	6.5 %	6 %	5.3 %
interfacing to operating system	1.8 %	3 %	3.1 %

VCU application program used 365 times, what gives 11 % of total number of used elements, Table 1. For comparison reasons it is worth to mention that commercial IEC 61131 compatible tool ISaGRAF [21] has 94 elementary function blocks available in FBD programming environment.

Table 2 presents the distribution of software components according to function domain, and this is given for each available environment and for both platforms. The results were expected as each application demands similar software components. There are variations according to processor architecture. For example, it is expected that DSP architecture performs more math (46 % elements are used for arithmetical functions) than 8051 architecture (21.5 %).

4.2 System software

The system software consists of real-time kernel and system programs. Real-time kernel handles tasks according to the pre-emptive fixed-priority scheduling policy [22–23]. The same scheduling policies were applied for each processing unit used in described projects. With this scheduling policy [22], scheduling a set of n periodic tasks is feasible if:

$$PUB = \sum_{1 \le i \le n} \frac{C_i}{T_i} \le n\left(2^{\frac{1}{n}} - 1\right) \qquad (1)$$

where PUB = processor utilization bound, C_i = computation time of task i, T_i = sample time of task i and n = number of tasks. For example, the current GPPU implementation involves 9 tasks what gives PUB of 0.72. However, it can be shown [24] that the scheduling of particular task set might be feasible even when processor utilization approaches 0.88. Actual processor utilization (CPU_load) of final VCU and TCU application programs is for GPPU around 70 %. For future improvements there are still additional margins possible due to the fact that task periods can be chosen to give theoretic schedulability bound up to 1.00. Precisely, task periods can be harmonic, i.e. each task period is in that case exact integer multiple of the next shorter period [22, 23]. Additional technique that can be used if previously mentioned condition cannot be fulfilled is task skipping, but in a known and predictable way, [25–27]. In this project the application programmer has the possibility to determine for a particular task how many skips are tolerable in a certain time window. Some facts related to the CPU_load measurement and the worst case execution time (WCET) are given in section 9.

5 VEHICLE COMMUNICATION NETWORKS

Tramcar serial communication subsystem was built around three independent CAN busses [28–29] with CANopen [30] protocol and two RS-485 proprietary busses, Figure 6. RS-485 networks are used to connect proprietary equipment, while CAN_1 network connects all tramcar control units (except auxiliary power supplies). CAN_2 connects only VCU with TCUs, thus enabling redundancy on the system level. It is obvious that VCU is responsible for almost all data transactions. Therefore, high demands related to reliability and availability were put on VCU during the development.

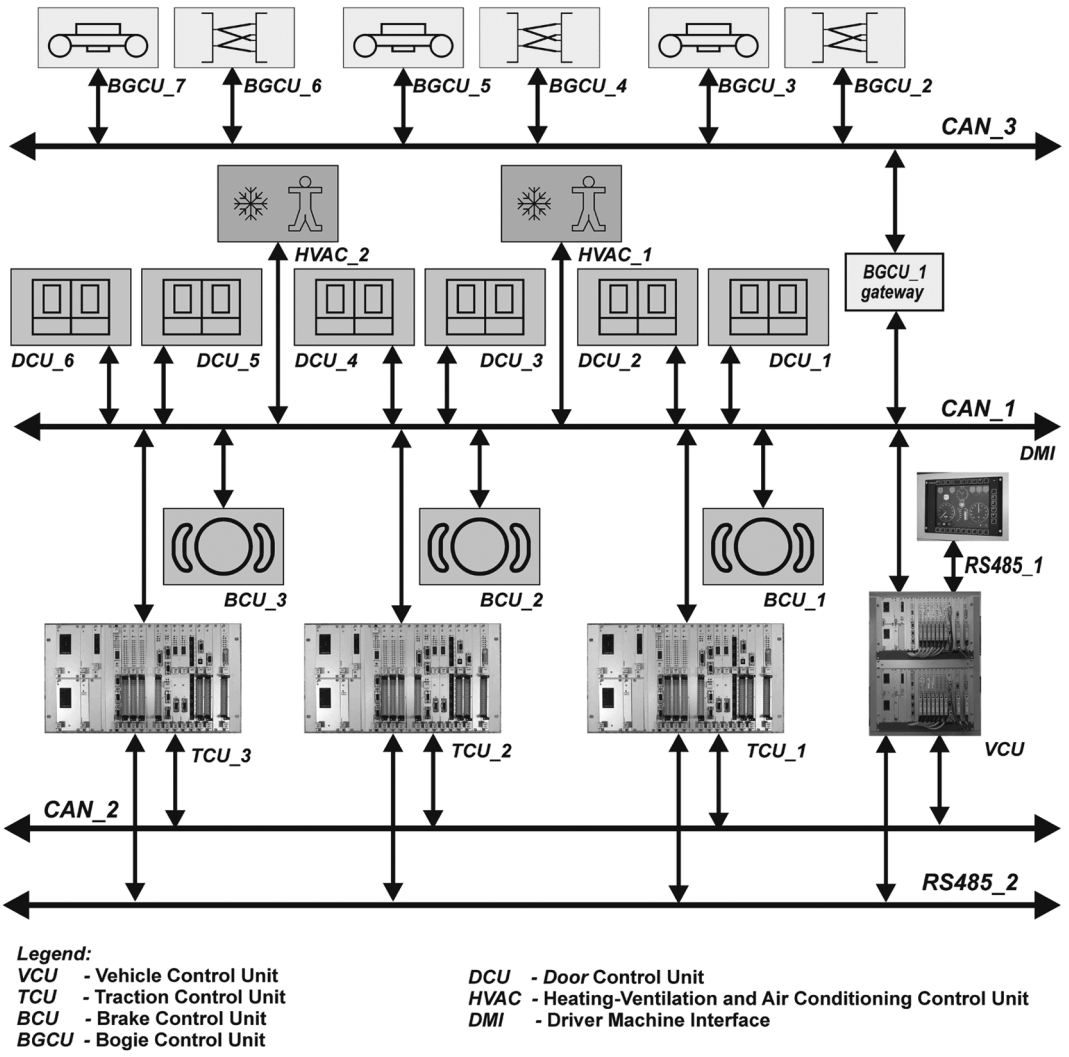

Fig. 6 Vehicle communication networks

CAN is priority-based protocol, and taking into account scheduling algorithm, it supports non-preemptive fixed-priority scheduling. Due to the fact [30] that the appropriate timer (for chosen transmission type 254) initiates Process Data Objects transmission (PDOs) and VCU transmits (Tx) or receives (Rx) all messages on the bus, the CANbus load can easily be calculated, [22], by means of the following equation:

$$\text{CANbus\_load} = \sum_{1 \le i \le n} \frac{ML_i}{T_i}, \qquad (2)$$

where i = PDO number, n = number of TxPDOs on the bus, ML_i = length of i-th message (ms), T_i = time base written in appropriate event timer (ms). For example, CAN_1 supports 53 PDOs and operates with speed of 250 kBits/sec. Each PDO is 8 bytes long, apart from DCU objects that are 2 bytes long (12 PDOs total). Assuming that the length of 8 bytes PDOs is 122 bits, and the length of 2 bytes PDOs is 74 bits, the calculated load for CAN_1 is 45.3 %. During the commissioning 47 %–48 % CANbus_load was measured. The difference is caused by the fact that more than expected stuffing bits are present along with some messages not taken into account (network management, timestamp). The reason for significant number of stuffing bits is partly un-optimized CAN_1 traffic i.e. almost all PDOs have 1–4 spare bytes (filled with zeros) for possible future extensions. The aim is to have the future possibility to add certain amount of new data to PDO content, if necessary. With this concept CANbus_load will, when number of useful signals in PDO is increased, retain the same value, or even be slightly lower. Ad-

ditionally, »empty« parts of PDOs are also processed, but ignored, on control unit's level. This all together makes system expandable, but without affecting real-time related issues (everything is tested and validated in advance). So, sub-optimal design can sometimes save a lot of workload in the case of future changes. Naturally, when additional PDOs would be needed CANbus_load increase should be expected as well as increase in CPU_load for particular control unit that processes PDO related data. More details on tramcar communication networks can be found in [18] and [31].

6 OBSOLESCENCE PROBLEMS

Component obsolescence rate within the semiconductor industry is nowadays increasing at non-expected rates. It has the impact to almost all manufacturers of long-life systems such are military, avionic, space, railway and some industrial applications [7, 32–36]. The life expectancy of commercial components is drastically shorter than required by most long-life projects. According to [33], the life expectancy for typical commercial components are: memory devices – 9 months, programmable logic – 1 year, gate arrays – 2 years, microprocessors – 2 years, digital signal processors – 3 years, logic families – 6 years, linear devices – 8 years. Above numbers are statistical data and can vary significantly, e.g. for microprocessors from less than 1 year to more than 20 years. Some manufacturers encourage component obsolescence, particularly processors, in order to enhance market available for their latest products [35]. Due to the fact that 20 % of digital ICs and 10 % of analogue ICs are going to be obsolete every year, product developers and manufacturers are trying to tackle this reality [32, 34]. Typical solutions to cope with obsolescence are: life of type (LOT) buy, component replacement, reverse engineering (emulation), minor redesign and major redesign. Taking into account the development time needed for complex projects there are cases where some components are obsolete before the start of the production.

When dealing with processor obsolescence the situation is even worse. Usually, there is no second source for processor or controller replacement and the only applicable solutions are LOT buy or major redesign. It is obvious that major redesign is a very expensive solution. However, major redesign, if performed correctly, can lead to the improvements of product functional and non-functional features. The worst posible case is end-of-life (EOL) notification for certain processor architecture i.e. for a complete product family. This usually also means the end-of-life for the particular platform. If not handled in advance this often leads to significant economic threats for the company.

6.1 Example

During lifetime of described platforms several obsolescence problems were encountered, and some of them are expected to occur. One rather demanding major redesign of central processing unit used in platform II is presented here. Original unit was developed around two microcontrollers (8051 architecture) where mutual communication was performed through dual-ported RAM [4, 20]. The primary redesign requirements were: same mechanical dimensions, compatible front panel diagnostic and communication facilities, same electrical interfaces and timing characteristics for communicating with peripheral units, and the same controller architecture. At the same time the demand was to improve overall characteristics of the unit and adjust it to the current technology. Therefore, additional requirements were: increased processing power for at least 4 times, CAN communication interface, on-board galvanic isolation of

Fig. 7 Major redesign of processing unit

serial communication channels, the possibility to implement improved PWM algorithms, improved resolution of A/D and D/A channels, improved watchdog and real time clock. The final decision was to use C8051F060 microcontroller [16]. To achieve additional functionality, not needed in a case when only replacement of the old unit is demanded, another microcontroller, XC866, which supports advanced PWM algorithms, was used. XC866 is based on the same architecture [37]. As for the software platform, the following tasks were required: modification of approx. 10 % of programming elements, changes in operating system (3 % of code) and adaptation of IDE to a new controller. Naturally, each application engineer has to modify the application program what usually demands a day without final testing and verification. Therefore, it can be concluded that, due to the same architecture, software related tasks were relatively easy and straightforward to perform. The photo of the both processing units is given in Figure 7. Left side presents the original unit, while the unit to the right is a F3 (form, fit and function) replacement.

7 LESSONS LEARNED DURING THE PROJECT

Several lessons were learned during the development, integration, commissioning and exploitations. Some of them are well known typical conclusions, but some are not so common and are therefore also mentioned below.

a) Industry often follows its own rules during the development of embedded real-time systems. Sometimes they appear to be far away from the solutions proposed by up-to-date academic achievements. Time to market, hard pressures on development teams, cost and management decisions often forces industrial solutions that are not the most advanced products. On the other hand, it is not always easy to identify the most appropriate academic results to be considered for the implementation in the current industrial project because academia tends to suggest general and wide applicable solutions. However, industry can benefit a lot from carefully studied and selected research results.

b) System software engineers tend to offer more functionalities and utilities than really needed by software application engineers, thus increasing demands on system resources. This is usually not a problem if application engineers are experienced and can select only necessary software components.

c) Software application engineers, i.e. engineers that develop the application program (by means of software components), always tend to spend all available processing resources. If system engineers define the processor utilization bound according to the number of tasks and scheduling policies, it will definitely be approached very soon. Further, there is always a certain amount of unnecessary code in the application program that increases CPU_load up to 5 %. This unnecessary code remained during development, commissioning and application program changes.

d) Trends in embedded systems development show that products are more SW related, thus increasing overall development and maintenance SW costs, while at the same time the price of HW is decreasing [8]. Therefore, some authors [38] are encouraging adding additional processing units instead of developing relevant SW based solutions. However, additional processing units usually put additional demands on both system and application SW. In these projects it was concluded that the old rule »make it as simple as possible« is still worth to be considered.

e) Customers are never completely satisfied. They often put additional demands initially not requested. Mostly these demands can be fulfilled by application program changes. This, however, demands additional testing, validation and verification often on the vehicle level, and at the same time increases CPU_load. As a consequence, measured or calculated values of worst case execution time (WCET) become obsolete very soon. Therefore, it was concluded that the best way to deal with the WCET issues is software measurement technique implemented in the real time kernel. CPU_load is measured by means of appropriate timers all the time, i.e. during normal program execution. This demands more processor resources than necessary, but is stable and does not introduce measurement side effects. Further, the results are logged during runtime, so the best case execution time (BCET), average case execution time (ACET) and WCET are available for further analysis and future system improvements. However, there is no guarantee for WCET obtained this way to be correct, i.e. to have the real WCET available. Therefore, a certain execution time margin should be available. Significant advantage of the concept applied in presented platforms (especially GPPU used in platform I) is temporal predictability of task execution time within the application program. The execution time of about 80 % of software components is independent of input parameters or signals. This is possible because rather simple processor architec-

ture is applied (no caches, pipelining, branch prediction, out-of-order execution), and software components are small and optimized. For example, all test cases have shown the difference between WCET and BCET to be less than 2 %. There are a lot of research activities related to WCET measuring and calculation [39–41] because modern processor architectures cause a lot of challenges for both academia and industry. There are also academic projects [42] where the development of custom processor with time predictable behavior is a project goal. In this project the satisfactory level of predictability was achieved with standard solution, but with the penalty to have lower processing power available. The concept of using processors of modest capabilities (slow processors according to current market trends) showed to be more than adequate if tasks are functionally partitioned among available processing units. It is surprising how many functions can be implemented by means of such processors but with carefully designed and optimized SW components. It has been concluded that the premature optimization [43] of SW components is far from being »the root of all evils«.

f) Apart from customer demands there are always a lot of reasons for application program changing during system life-time such as potential software errors, revisions because of the other equipment, improvements or optimization, maintenance requests etc. As an example, VCU application program had 27 revisions, while TCU application program experienced 12 revisions. None of the changes was due to system errors during lifetime. It can be concluded that the approach to application program development in such low-volume applications is completely different than in high-volume embedded applications, where such number of revisions would be unacceptable.

g) Developers must learn to expect and deal with component obsolescence issues. Usually, it is assumed that this problem is only related to production, purchasing and managing departments. By clever and future-oriented design, some of the problems can be solved much easier, i.e. without major redesign and/or changing the whole product platform.

h) System software developers (the engineers that develop SW platform) should not be geographically dislocated. Sometimes it is a must, but then much lower efficiency and productivity are to be expected. Furthermore, common approach to outsource system SW development should be avoided, if possible. Apart from this, higher level of abstraction in SW development process is necessary. As the rule of thumb the following criteria can be defined: system SW developers should not participate in final product testing and commissioning. This should be done by application engineers only. If system developers are needed on-site during commissioning, then something in the quality of system software or in the project concept is not good enough.

8 CONCLUSION AND FUTURE WORK

The project was successfully completed, and 70 tramcars were delivered. The reliability and availability of presented control units are according to the contract. CANbus proved to be a good choice in light rail applications. It is very robust and reliable. For future projects TTCAN and FlexRay are studied. As for the HW components, the platforms are going to be maintained and expanded to offer more options and possible applications. Faster processors are inevitable, but within current architectures or within most closely related successors.

REFERENCES

[1] S. Marijan, **Control Electronics of TMK2200 Type Tramcar for the City of Zagreb.** Proc. International Symposium on Industrial Electronics, ISIE 2005, volume IV, pp. 1617–1622, Dubrovnik, 2005.

[2] S. Marijan, V. Ćesić, B. Furčić, M. Bilić, N. Težak, I. Bahun, **Digital Control System for the Traction Applications.** Proc. 10th International Power Electronics and Motion Control Conference, EPE-PEMC 2002, Dubrovnik, 2002.

[3] V. Ćesić, M. Kajari, S. Marijan, Z. Jurin, M. Kolić, **Excitation System with Microprocessor-based Twin Channel Voltage Regulator for Synchronous Machines**. Proc. 10th International Power Electronics and Motion Control Conference, EPE-PEMC 2002, Dubrovnik, 2002.

[4] A. Magzan, S. Marijan, J. Ungarov, **Railway Passenger Coach Multi-system Feeding Converter.** Proc. 8th European Conference on Power Electronics and Applications, EPE‘99, Lausanne, 1999.

[5] N. Perić, S. Marijan, M. Kajari, **Microprocessor Based Control System for the Control of Electrical Machines and Processes.** Proc. 6th European Conference on Power Electronics and Applications EPE’95, Sevilla, 1995.

[6] H. Kopetz, **Real-Time Systems, Design Principles for Distributed Embedded Applications.** Kluwer Academic Publishers, 1999.

[7] S. Marijan, I. Petrović, **Platform Based Development of Embedded Systems for Traction and Power Engineering Applications – Experiences and Challenges.** To be published in Proc. 5th IEEE International Conference on Industrial Informatics, INDIN’07, Vienna, 2007.

[8] L. D. J. Eggermont (editor), **Embedded Systems Road Map 2002, Vision on Technology on the Future of PROGRESS".** Technology Foundation (STW), Ne therland, 2002.

[9] I. Crnkovic et al., **Component-based Development of Safety-critical Vehicular Systems.** Report ISSN 1404-3041, ISRN MDH-MRTC-190/2005-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, September 2005.

[10] J. P. Gonzalez-Zugasti, K. N. Otto, J. D. Baker, **Assesing Value in Platformed Product Family Design.** Res. Eng. Design, 13 (2001) pp. 30–41, 2001.

[11] L. P. Chao, K. Ishii, **Product Platform Design and the Gate Model: Lessons from the Industry Case Studies.** Proc. ASME International Mechanical Engineering Congress, IMECE '04, Anaheim, California, 2004.

[12] ..., **EN 50155, Railway Applications – Electronic Equipment Used on Rolling Stock.** European Committee for Electrotechnical Standardization, August, 2001.

[13] ..., **MC68302UM/AD, Integrated Multiprotocol Processor User's Manual.** Freescale Semiconductor Inc., 1995. (http://www.freescale.com)

[14] ..., **TMS320F240 DSP Controller.** Texas Instruments, Inc, sprs042e.pdf, 2002. (http://www.ti.com)

[15] ..., **TMS320LF2407A DSP Controller.** Texas Instruments, Inc, sprs145ke.pdf, 2005. (http://www.ti.com)

[16] ..., **C8051F060/1/2/3/4/5/6/7 Mixed Signal ISP Flash MCU Family.** Silicon Laboratories, c8051f06x.pdf, 2004. (http://www.silabs.com)

[17] ..., **C505xx 8-bit Single Chip Microcontroller.** Data sheet, Infineon technologies, 12/2000. Available: http://www.infineon.com

[18] S. Marijan, **Vehicle Control Unit for the Light Rail Applications.** Proc. 13th International Conference on Electrical Drives and Power Electronics, EDPE 2005, Dubrovnik, 2005.

[19] M. Bilić, S. Marijan, M. Šprah, **Driver Machine Interface for Rail and Light Rail Applications.** Proc. 13th International Conference on Electrical Drives and Power Electronics, EDPE 2005, Dubrovnik, 2005.

[20] ..., **SAB 80C515/80C535 8-bit Single Chip Microcontroller Family.** User's manual, Siemens, 08/95, 1995.

[21] ..., **ISaGRAF User's Guide.** Ver. 3.5, ICS Triplex IsaGRAF Inc., 2004.

[22] C. L. Liu, J. W. Layland, **Scheduling Algorithms for Multiprogramming in a Hard-real-time Environment.** Journal ACM, vol. 20, pp. 46–61, Jan. 1973.

[23] L. Sha et al., **Real Time Scheduling Theory: A Historical Perspective.** Real-Time Systems, vol. 28, issue (2–3), pp. 101–105, November 2004.

[24] J. Lehoczky, L. Sha, Y. Deng, **The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour.** Proc. of the 10th IEEE Real-Time Systems Symposium, pp. 166–171, 1989.

[25] G. Bernat, A. Burns, A. Llamos, **Weakly-hard Real-time Systems.** IEEE Transactions on Computers, 50 (4), pp. 308–321, April 2001.

[26] G. Koren, D. Shasha, **Skip-over: Algorithms and Complexity for Overloaded Systems that Allow Skips.** Proc. of the 10th IEEE Real-Time Systems Symposium, pp. 110–117, 1995.

[27] G. C. Buttazzo, **Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications.** Springer, 2005.

[28] ..., **ISO 11898-1, Road Vehicles – Controller Area Network (CAN), Part 1: Data Link Layer and Physical Signalling.** International Organization for Standardization, 2003.

[29] ..., **ISO 11898-2, Road Vehicles – Controller Area Network (CAN), Part 2: High-speed Medium Access Unit.** International Organization for Standardization, 2003.

[30] ..., **CiA ds301, CANopen: Application Layer and Communication Profile.** CAN in Automation Draft Standard 301, version 4.02, 2002.

[31] S. Marijan, M. Bilić, K. Ivanuš, **CANopen Implementation in Zagreb Tramcar.** Proc. 11th International CAN Conference, iCC 2006, Stockholm, Sept. 2006.

[32] K. Hayward, M. Codner, **Avionics and Mission Systems – A Key Element in Delivering Through-life Capability.** Whitehall Report 3-06, The Royal United Services Institute, 2006.

[33] J. Parkinson, **Take a Hybrid Approach to Component Obsolescence.** COTS Journal, pp. 20–27, Jan. 2001.

[34] ..., **GEAD Technical Commission, Electronic, Electrical and Electromechanical Component Obsolescence – Management Guidelines.** Report GIFAS/5203/2005, French Aerospace Industry, 2005.

[35] ..., **Processor Obsolescence in Avionics.** Report ASSC/330/2/165, Avionic Systems Standardisation Committee, 1999.

[36] K. Hunt, S. Haug, **Challenges of Component Obsolescence.** Proc. 2001 Royal Aeronautical Society Conference, London, 2001.

[37] ..., **XC866 8-bit Single Chip Microcontroller.** Data sheet, Infineon technologies, 12/2006. (http://www.infineon.com)

[38] J. Ganssle, **Subtract Software Costs by Adding CPUs** (http://www.embedded.com/showArticle.jhtml?articleID=161600589; accessed May 25, 2007).

[39] R. Kirner, P. Puschner, **Discussion of Misconceptions about WCET Analysis.** Proc. 3rd International Workshop on Worst-Case Execution Time Analysis, WCET'2003, Polytechnic Institute of Porto, Portugal, pp. 61–64, 2003.

[40] R. Kirner, P. Puschner, **Classification of WCET Analysis Techniques.** Proc. 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC' 05, pp. 190–199, 2005.

[41] N. Hillary, K. Madsen, **You Can't Control what you Can't Measure, or Why it's Close to Impossible to**

Guarantee Real-Time Software Performance on a CPU with On-Chip Cache. Proc. 2nd International workshop on worst-case execution time analysis, WCET 2002, TU Wien, 2002.

[42] M. Delvai, W. Huber, P. Puschner, A. Steininger, **Processor Support for Temporal Predictability – the SPEAR Design Example.** Proc. 15th Euromicro Conference on Real-Time Systems, ECRTS'03, pp. 169–176, 2003.

[43] R. Hayde, **The Fallacy of Premature Optimization.** Ubiquity – an ACM IT Magazine and Forum (http://www.acm.org/ubiquity/views/v7i24_fallacy.html; accessed May 28, 2007).

Vlastita rješenja elektroničkih ugradbenih upravljačkih sustava niskopodnog tramvaja TMK2200. U radu su prikazana vlastita rješenja elektroničkih ugradbenih upravljačkih sustava primijenjena na niskopodnom tramvaju TMK2200. Sustavi su zasnovani na generičkim platformama koje, zahvaljujući modularnim programskim i sklopovskim komponentama, omogućuju razvoj različitih proizvoda. Osim koncepcije platformi detaljnije su prikazani centralni upravljački sustav, upravljačka elektronika pretvarača glavnog pogona, upravljačka elektronika pomoćnih pogona te rješenje sučelja korisnika i vozila. Opisana je i koncepcija komunikacije svih uređaja u vozilu koja je zasnovana na CAN sabirnicama i CANopen komunikacijskom protokolu. Problem zastarjelosti elektroničkih komponenata ugrožava ne samo proizvodnju već i održavanje razvijenih sustava. Zbog toga su u članku opisani rizici i moguća rješenja, a za ilustraciju je prikazan konkretan primjer problema zbog zastarjelosti ključnih komponenata procesorskog modula.

Ključne riječi: tramvaj, sustavi za rad u stvarnom vremenu s nametnutim vremenskim ograničenjima, vlastita rješenja elektroničkih ugradbenih sustava

AUTHOR'S ADDRESS

Siniša Marijan, B.Sc., E.E.

Končar Electrical Engineering Institute
Fallerovo šetalište 22, HR-10002, Zagreb, Croatia
Tel: ++385 1 3655301 101
Fax: ++385 1 3667 309
sinisa.marijan@koncar-institut.hr

Received: 2007-09-12