OPTIMAL DESIGN AND IMPLEMENTATION OF A FAST START SCHEME FOR A HIGH REAL-TIME EMBEDDED ELECTROMECHANICAL CONTROL SYSTEM

Xiangfeng Suo[1], Han Xue[1], Yunhui Gao[1]*
[1]School of Computer and Information Engineering, Heihe University, Heihe, Heilongjiang,164300, China
**Corresponding author: Xiangfeng Suo*
Email: suoxiangfeng188@126.com

Abstract: The purpose is to shorten the start-up time of high real-time embedded electromechanical control system, so as to improve the system performance. Based on the theory of software composition and start-up process of high real-time embedded electromechanical control system, in the research of bootloader stage of embedded electromechanical control system, the advantages and disadvantages of two start-up schemes based on NOR flash and NAND (Not AND) flash are analysed. The optimization design of system quick start scheme based on NOR flash is proposed, including the modular design of embedded Linux kernel and the implementation of hybrid file system based on compressed ROM filesystem (CRAMFS) and Yet Another Flash File System (YAFFS2). Finally, the control of the refrigerator in the cooling control system is taken as an example to carry out the simulation experiment. The results show that compared with before optimization, after the optimization of the system quick start scheme, the system in stage 1 is shortened by 2.11s, the stage 2 is shortened by 6.22s, and the total time of the whole starting process is shortened by 6.33s; the theoretical speed of the motor increases from 0 to 1560, while the actual output speed increases from 0 to 1655; the error of the actual output relative to the theoretical speed is small (about 1%), the range of real-time frequency change is [0, 25.2] and the value is in the ideal range. This shows that the optimization design effect of the fast start scheme is ideal, and the scheme can make the motor performance of the system in a stable operation state.

Keywords: High Real-Time; Embedded Electromechanical Control System; Embedded Linux Kernel; Fast Start Scheme; Optimization Design.

1. Introduction

With the continuous development of computer technology and industrial automation, the task and purpose of electromechanical control equipment and control system become more and more complex. Electromechanical control system faces a series of problems, such as complex calculation, large amount of data processing, strict performance index, multi-objective and multi-level control requirements. These problems pose great challenges to power system, industrial production process control system, intelligent robot system, computer integrated manufacturing system, and nuclear power plant safe operation control system [1,2]. In this context, large-scale and single function programmable logic controller cannot meet the needs of high-end electromechanical control field. The embedded electromechanical control system which combines integrated circuit technology, embedded technology and automatic control technology shows great advantages in this field. In some fields, such as industrial process control, digital machine tool, power system, and petrochemical system, embedded electromechanical control system has become the mainstream [3,4].

Embedded electromechanical control system is a modular, reconfigurable, scalable, software and hardware integrated open control system. It can run on various platforms and can be mixed with the current control system, so as to provide users with a unified interactive style [5]. It is characterized by openness. It applies the latest international technology achievements such as electronic technology, communication technology, computer technology and network technology to the field of automatic control system [6]. Through the use of embedded electromechanical control system, common products and technical standards can use standard low-cost equipment, such as a large number of communications, network, graphics, database and other aspects of software, to greatly reduce the development cost and shorten the product development cycle. Although these

technologies are the general technologies in communication, computer, network, software and other industries, they cannot be directly applied in the field of automation control. It needs to be cut, optimized and debugged properly to adapt to the specific requirements of industrial control in terms of anti-interference and reliability [7,8]. At present, the companies that produce PLC products in the market develop the system according to their own standards, and the systems of different manufacturers are not compatible. In order to avoid the commercial and technical risks caused by over reliance on one system supplier, many enterprises usually use multiple different control systems. Due to the closeness, specificity and incompatibility of traditional control system, it is difficult for enterprises to make full use of automation technology, which brings great difficulties to the integration, upgrading and management of internal system. This requires that the development of electromechanical control system is moving towards the direction of standardization, openness, interoperability, portability, reliability and stability [9,10]. Due to the characteristics of embedded system and the application environment requirements of embedded electromechanical control system, the fast start of embedded electromechanical control system has become an important topic in the research of electromechanical control system.

Based on this, the design of the fast start scheme based on NOR flash is proposed innovatively for the high real-time embedded electromechanical control system, and the kernel and file system are optimized. Moreover, the simulation experiment of the refrigeration system electromechanical group is carried out, which provides the practical basis for the performance improvement and rapid start of the embedded electromechanical control system.

2. Fast Start Scheme and Optimization of High Real-time Embedded Electromechanical Control System

2.1 Software Composition and Start-up Process of High Real-time Embedded Electromechanical Control System

The software structure of embedded electromechanical control system is based on embedded operating system. In the software structure, the information exchange, intelligent control, human-computer interaction, data processing and other functional programs of the control system are based on the embedded operating system to complete the information docking with the underlying hardware [11]. Figure 1 is the software composition of high real-time embedded electromechanical control system.

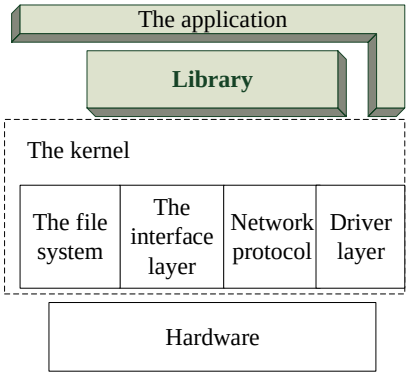

Figure 1: Software composition of high real-time embedded electromechanical control system

The top of hardware layer is the core and key part of embedded operating system. It controls and manages all kinds of hardware in embedded electromechanical control system, provides system call interface for user software, organizes the work of control system reasonably and effectively, and provides support platform for function expansion of control system [12]. The work scope of the kernel includes: device driver, memory space sharing, signal distribution management, I/O access management and process scheduling. For the application software of embedded electromechanical control system, the core of robust and real-time operating system plays an important role in the reliability and stability of the whole embedded control system. According to the requirements of embedded electromechanical control system for real-time, reliability and low cost, embedded RTLinux operating system is selected as the operating system of embedded electromechanical control system.

The next is the start-up process. In the start-up process of the embedded electromechanical control system, there are four main components: bootloader, kernel, root file system and application program. Figure 2 is the start-up process.

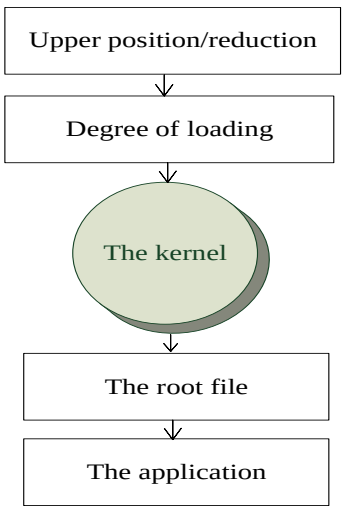

Figure 2: System start-up process

The first software in the process of system start-up is bootloader, which is highly dependent on the hardware of the target board, and is responsible for the initialization of microprocessor, the loading of memory, serial port and kernel image. After the bootloader completes the initialization of the underlying hardware and the loading of the kernel image, it will jump to execute the bootloader code of the kernel.

At the beginning of the kernel, due to different architectures, the start-up code will be very different. It first initializes itself, sets the execution environment of C program code, and then jumps to an architecture independent start_ Kernel function to perform work [13]. This function will initialize the kernel advanced, then load the device driver module, finally install the root file system and start the init process. The rest of the start-up process is handled in user space by the init program on the root file system, depending on the user's specific application requirements.

2.2 Fast Start Scheme of High Real-time Embedded Electromechanical Control System

In the research of embedded electromechanical control system bootloader stage, two start-up schemes based on NOR flash and NAND (Not AND) flash [14,15] are analysed. Both of them have certain advantages but obvious disadvantages, as shown in Table 1 below:

Table 1 Advantages and disadvantages of two start-up schemes based on NOR flash and NAND flash

Types of start-up schemes	Advantages	Disadvantages
Based on NOR flash	Fast start-up time	Complex process
Based on NAND flash	Low cost	Taking up a lot of memory space, slow start-up speed

The analysis of the start-up process of the embedded electromechanical control system kernel suggests that the number of statically loaded drivers in the kernel has a great impact on the start-up speed of the system. How to reduce redundant drivers that are not needed by hardware platform, and how to arrange driver static loading or module loading are very important to the start-up speed of kernel [16]. By cutting and configuring the kernel, the kernel can be simplified as much as possible on the premise of meeting the application requirements of embedded electromechanical control system, so as to achieve the purpose of fast start-up in the kernel stage.

JFFS file system is a log structure file system specially designed for NOR flash memory, which has the characteristics of power down reliability, packet loss balance, and data compression [17]. When it is developed to JFFS2 version, it supports NAND flash.

Although JFFS2 file system has been widely used in embedded system, with the continuous development of flash technology, it has gradually exposed some shortcomings, as shown in Table 2 below:

Table 2 Shortcomings of JFFS2 file system

Types of shortcomings	Content
Too long mount time	The mount process of JFFS2 file system needs flash device scanning from the beginning to the end. The mounting time is proportional to the size of flash device and the number of nodes on flash memory.
Poor scalability	Although JFFS2 file system minimizes memory consumption, the memory consumption is directly proportional to the number of I nodes and the number of nodes on flash devices. In practical application, JFFS2 file system can be used in 128M NAND flash.
Arbitrariness of wear balance	JFFS2 file system uses probability method to solve the problem of wear balance, which is difficult to ensure the certainty of wear balance. In some cases, unnecessary block erasure and writing may occur, and sometimes it may lead to the untimely adjustment of wear balance.

Yet Another Flash File System (YAFFS2) file system is an embedded file system specially designed for NAND flash. With the characteristics of loss balance and power-off protection [18], it supports large capacity NAND flash and can be installed quickly. It is the best choice for large capacity NAND flash. YAFFS2 file system has done a lot of work in power failure reliability. However, when the root file system adopts a single file system, the power failure reliability of the embedded system cannot be fully guaranteed. This obviously cannot meet the high reliability requirements of embedded electromechanical control system. The root file system uses the compressed ROM filesystem (CRAMFS) file system to mount the YAFFS 2 file system. Hybrid file system can effectively solve the power failure reliability problem of embedded system.

In addition, the hybrid file system can provide permanent data storage capability for embedded electromechanical control system. It has the characteristics of fast installation, and meets the requirements of fast start of embedded electromechanical control system.

2.3 Optimization Design of Quick Start Scheme

Based on NOR flash, the fast start scheme optimization of high real-time embedded electromechanical control system is completed. First, the storage space address is planned. In the quick start design of this scheme, 8Mb NOR flash is used, and bootloader image and compressed SREC format kernel image are stored in NOR flash [19]. Figure 3 shows the spatial division of NOR flash.

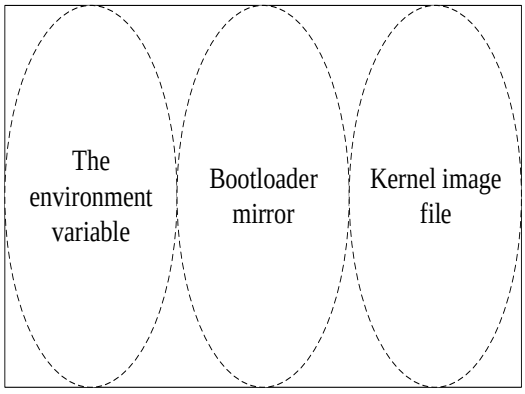

Figure 3: Space division of NOR Flash

Bootloader mirror file starts with the starting physical address and cannot be larger than 1.75m in size. The abnormal reset vector is stored in 256K area at the top of flash, which is used to store the environment variables of Bootloader, and the kernel start-up parameters are also stored in the area. The kernel image file is stored from the bottom of flash. According to Bootloader start-up mode, the region can store kernel images in ELF or SREC format, but the size of the mirror file cannot exceed the abnormal reset vector. Otherwise, Bootloader boot code will be destroyed, which will cause the system to fail to start normally.

Moreover, the driver and kernel optimization of NOR flash. 16 bit NOR flash is used, and the read and write of NOR flash are based on 2 bytes, which reduces the number of write command input and improves the speed of data writing. In the aspect of hardware design, the address line needs shift design. It means that central processing unit (CPU) address line A_0 is set to null, CPU address line A_1 is connected to NOR flash address line A_0, and CPU address line A^2 is connected to NOR flash address line A_1 in turn. Accordingly, in the software design, NOR FLASH read and write should also use 16-bit mode, and input command must also use 16-bit mode command.

The basic operations of NOR flash include reading, writing and erasing. The read operation does not require a driver. CPU can read data directly by accessing any address on chip, just like accessing memory. NOR flash has its own unique properties for write operations. Flash write only allows the corresponding bits to be written from logic 1 to logic 0. In order to write logic 0 to logic 1, it is necessary to erase it first. Otherwise, the write is invalid. Write operation flow and erase process are shown in Figure 4A and Figure 4B below:

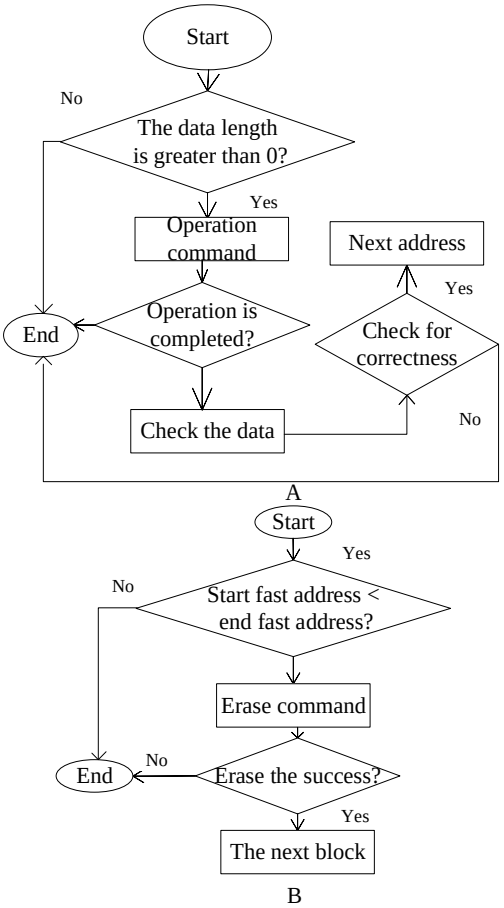

Figure. 4 System write operation flow and erase flow (A: operation flow B: erase flow)

It takes three command cycles for data to be written into NOR flash. After the input is completed, the correctness of the written data needs to be verified to prevent bit swapping. The erasure operation of NOR flash is based on block [20], and the block size is generally 64KB. Block sizes may vary from manufacturer to manufacturer. After the erase operation, the logical value of the corresponding block is reset to 1.

For the optimization of the kernel, the embedded Linux kernel adopts modular design, which can eliminate unnecessary kernel modules according to the specific hardware environment and different application requirements, so as to reduce the kernel volume and system overhead [21]. Tailoring and configuring the kernel is the initial stage in the process of building the kernel. make menuconfig is used to configure the interface and the main menu. It appears in the form of menu in the shell interface of the terminal, and it can extract the default value in the .config file and display it in the configuration menu. After the kernel is started, it can be dynamically loaded into the system through the kernel module [22]. When the module is loaded into the kernel, insmod is used to complete the task. It loads the code and data of the module into the kernel, and then uses the symbol table of the kernel to parse any unresolved symbols in the module. However, the kernel does not modify the module file stored on disk or flash, but only the copy in memory. At this point, insmod can accept some command line options and assign values to integer and string variables in the module before linking the module to the kernel.

Finally, the optimization of embedded Linux file system. Through the miniaturization of root file system and the design of hybrid file system, under the premise of meeting the performance requirements, the root file system mounting process of embedded electromechanical control system is optimized to realize the fast start of the file system mounting process. Miniaturization is mainly to provide necessary application programs for the system through BusyBox [23] after the establishment of basic device files. It provides the whole command set function of Linux through a very small application. It can link glibc or uClibc libraries statically or dynamically. It can modify the default configuration of Busybox and remove unnecessary commands to meet the needs of most applications. Then, the Busybox component is installed to the root file system, which means that the binary Busybox is installed in the / bin directory. The binary file can only be called indirectly through its symbolic link, not directly. Busybox files can execute commands based on the name of the symbolic link that calls them. These symbolic links must be named after the original command, but they can be created in any directory of the executable file. Depending on the purpose of the command, they are usually set up in the / bin, / SBIN, / usr / bin and / usr / SBIN directories.

In the design of hybrid file system, in order to realize the fast start-up of embedded electromechanical control system, the system start-up scheme based on NOR flash is adopted, and the bootloader and kernel image are stored in NOR flash. For the demand of large data storage, NAND is selected as the permanent storage medium. In order to implement the hybrid file system scheme, the kernel needs the support of CRAMFS and YAFFS2 file systems. Figure 5 shows the specific configuration process.

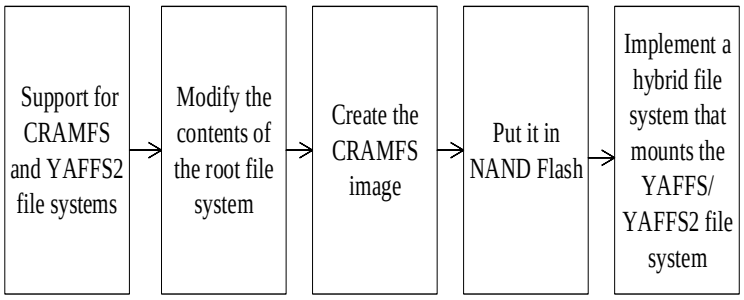

Figure. 5 Configuration process of hybrid file system

After the above work, the basic root file system has been successfully established. However, due to the lack of link library in application development, the root file system cannot execute applications other than BusyBox program.

At this point, the root file system containing glibc's full link library is selected, but glibc's full link library will take up more space. The ldd command is used to list the dynamic link libraries that the application depends on. After which link library components are needed is determined, these link library components and related symbolic links can be copied to the / lib directory of the root file system. After glibc components are installed on the root file system of the target board, applications can use them at run time.

2.4 System Simulation Design

(1) With the control of the refrigerator in the cooling control system as an example, the simulation environment is designed. The embedded electromechanical control system has Ethernet interface and supports TCP / IP protocol [24].

It communicates with remote server through Ethernet. The data collected by the control system is stored on the server in a specific format through Ethernet. The server can analyse the collected data and feedback the corresponding control information, so as to realize remote data collection and control. The operator can also choose local storage. The data collected is stored in local memory through USB (universal serial bus) interface and SDHC (secure digital high capacity) interface. The simulation environment and method are as follows.

Simulation environment: uncompressed vmlinux kernel image and zImage image file compressed based on gzip algorithm [25] are used. Since the configuration of the kernel is the same, the time required to start the kernel is the same from the kernel image decompression to memory to the beginning of the kernel mount of the root file system.

Simulation method: the time is timed by reading the CPU real-time clock. In bootloader stage, after the initialization of basic hardware, the real-time clock time is read once and printed to the terminal through serial port. When the kernel starts building the virtual file system, the timestamp is printed again.

The last timestamp is after the system starts, and the difference between each timestamp is the start time to be obtained. During the simulation, the start-up process of embedded electromechanical control system is divided into two stages, as shown in Table 3 below:

Table 3 Start-up process of high real-time embedded electromechanical control system

Different stages	Attribute content
Stage 1	From power on of target board to VFS initialization of kernel, because the kernel starts at the same time, the time difference obtained in the verification process is the result of kernel boot optimization.
Stage 2	From the kernel initialization of VFS to the end of system start-up, the time difference obtained in the verification process is the result of optimizing the root file system and system initialization.

(2) Origin 2018Bit is used to analyse the acquired data visually.

3. Optimization Effect of System Performance and Quick Start Scheme

3.1 Comparison of Time Consumption before and after System Optimization

The start-up time consumption of high real-time embedded electromechanical control system before and after optimization is compared, as shown in Figure 6 below:

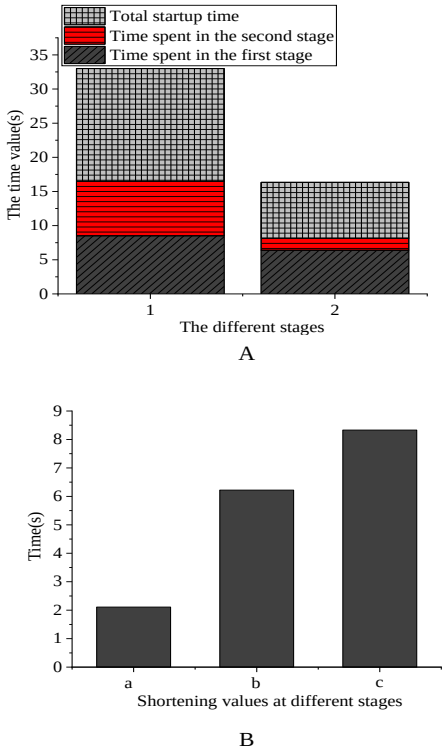

Figure. 6 Comparison of start-up time consumption before and after system optimization (1: stage 1; 2: stage 2; a: shortening value in stage 1; b: shortening value in stage 2; C: total shortening value)

Figure 6 shows that before optimization, the start time of the system in stage 1 is 8.52s, the start time of stage 2 is 7.98s, and the total start time of the system is 16.5s. After optimization, the start-up time of the system in stage 1 is 6.41s, the start-up time in stage 2 is 1.76s, and the total start-up time of the system is 8.17s. The start-up time in stage 1 is shortened by 2.11s, that in stage 2 is shortened by 6.22s, and the total time is shortened by 6.33s. Especially in stage 2, the shortening time is relatively long, which indicates that the optimization design effect of this exploration on the quick start scheme is ideal.

3.2 Comparison of Theoretical Speed and Output Speed of Refrigerator

The theoretical speed and output speed of the refrigerator in the cooling control system are compared. Figure 7A and Figure 7B is the result.

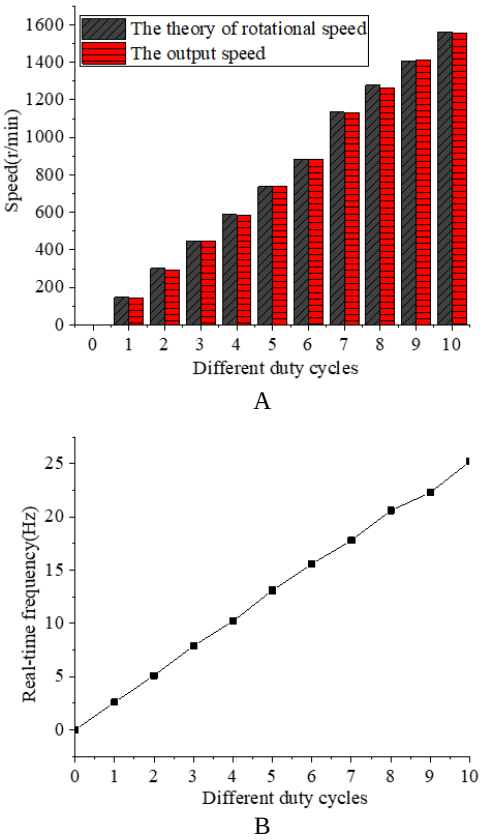

Figure. 7 Comparison results of theoretical speed and output speed of refrigerator

Figure 7 shows that when the duty cycle increases from 0 to 10, the theoretical speed of the motor increases from 0 to 1560, while the actual output speed increases from 0 to 1655. The error of the actual output relative to the theoretical speed is small, about 1%. The change range of real-time frequency is [0,25.2], and the value is also in the ideal range. This shows that the proposed system start-up optimization scheme can make the motor performance of the system in a stable operation state.

4. Conclusion

The software composition and start-up process of the high real-time embedded electromechanical control system are analysed theoretically.

In the research of bootloader stage of embedded electromechanical control system, two start-up schemes based on NOR flash and NAND flash are analysed, and their advantages and disadvantages are compared. Furthermore, based on NOR flash, the fast start scheme optimization of high real-time embedded electromechanical control system is completed. Embedded Linux kernel is optimized by modular design. Based on the support of CRAMFS and YAFFS2 file system, the hybrid file system scheme is implemented. The results show that the optimal design of the proposed quick start scheme can shorten the time in different degrees in stage 1 and stage 2 of the system start-up, and the whole start-up process time is also shortened a lot. In this case, the system can still run stably and achieve the ideal effect. This exploration also has some shortcomings. At present, the design optimization scheme is only suitable for embedded electromechanical control system, and there is no relevant experiment on whether it is suitable for the rapid start of upper application program in specific application environment, which is also the focus of further research in the future.

Acknowledgment

This work was supported by the Basic Scientific Research Expenses of Universities in Heilongjiang Province Supported by the Special Fund Project of Heihe University.

References

[1] Shen L, Barnes M, Preece R, et al. (2016) The effect of VSC HVDC control on AC system electromechanical oscillations and DC system dynamics. IEEE Transactions on Power Delivery, 31(3), 1085-1095.

[2] Antipov A S, et al. (2019) Decomposition Synthesis of the Control System of Electromechanical Objects in Conditions of Incomplete Information. Mechanics of Solids, 54(5), 669-682.

[3] Kherraz N, Haumesser L, Levassort F, et al. (2016) Electromechanical hybrid metamaterial for the control of ultrasonic guided waves. Journal of the Acoustical Society of America, 139(4), 2183-2183.

[4] Acle Y, Freitas F D, Martins N, et al. (2019) Parameter Preserving Model Order Reduction of Large Sparse Small-Signal Electromechanical Stability Power System Models. IEEE Transactions on Power Systems, PP(99), 1-1.

[5] Ye M, Liu Y G, Cheng Y. (2016) Modeling and ratio control of an electromechanical continuously variable transmission. International Journal of Automotive Technology, 17(2), 225-235.

[6] Tumakova E V. (2016) Control of the Effect of External Factors Acting on the Operation of Electromechanical Systems. Measurement Techniques, 59(6), 600-604.

[7] Polyakov A E, Filimonova E M, Chesnokov A V. (2017) Upgrading Automatic Control of a Complex Electromechanical System for Production of Synthetic Yarns and Nonwoven Materials. Fibre Chemistry, 48(4), 1-4.

[8] Du W, Bi J, Cao J, et al. (2016) A Method to Examine the Impact of Grid Connection of the DFIGs on Power System Electromechanical Oscillation Modes. IEEE Transactions on Power Systems, 31(5), 1-10.

[9] Ku J E. (2018) Superconvergence of least-squares methods for a coupled system of elliptic equations - ScienceDirect. Computers & Mathematics with Applications, 75(6), 2059-2070.

[10] Mohla D. (2017) The History and Benefits of Standardization in Power Distribution Systems [Standards News]. IEEE Industry Applications Magazine, 23(5), 70-80.

[11] Khizhnyakov Y N, Yuzhakov A A, Besukladnikov I I, et al. (2018) Adaptive Fuzzy Control of Tracking Electromechanical Systems. Russian Electrical Engineering, 89(11), 648-651.

[12] Luvisotto M, Pang Z, Dzung D, et al. (2017) Physical Layer Design of High-Performance Wireless Transmission for Critical Control Applications. Industrial Informatics, IEEE Transactions on, 13(6), 2844-2854.

[13] Ma L, Zhang C, Yu B, et al. (2017) An empirical study on the effects of code visibility on program testability. Software Quality Journal, 25(3), 951-978.

[14] Kim C H, Lee S, Woo S Y, et al. (2018) Demonstration of Unsupervised Learning With Spike-Timing-Dependent Plasticity Using a TFT-Type NOR Flash Memory Array. IEEE Transactions on Electron Devices, PP(5), 1-7.

[15] Chen R, Yi W, Liu D, et al. (2017) Heating Dispersal for Self-Healing NAND Flash Memory[J]. IEEE Transactions on Computers, 66(2), 361-367.

[16] Hikichi S E, Salgado E G, Beijo L A. (2017) Forecasting number of ISO 14001

certifications in the Americas using ARIMA models. Journal of Cleaner Production, 147(MAR.20), 242-253.

[17] Sha H M, Chen X, Zhuge Q, et al. (2016) A New Design of In-Memory File System Based on File Virtual Address Framework. IEEE Transactions on Computers, 65(10), 1-1.

[18] Zheng L, Fang W, Liu J, et al. (2016) A user-visible solid-state storage system with software-defined fusion methods for PCM and NAND flash. Journal of Systems Architecture, 71, 44-61.

[19] Yu K M, Wang Y, Wang C. (2017) Smooth geometry generation in additive manufacturing file format: problem study and new formulation. Rapid Prototyping Journal, 23(1), 34-43.

[20] Fang H K, Chang-Liao K S, Cheng C H, et al. (2020) Operation Characteristics of Gate-All-Around Junctionless Flash Memory Devices with SiN/ZrO-Based Stacked Trapping Layer. IEEE Transactions on Electron Devices, 67(9), 3626-3631.

[21] Velasco A D, Montrucchio B, Rebaudengo M. (2016) KITO tool: A fault injection environment in Linux kernel data structures. Microelectronics Reliability, 60(May), 153-162.

[22] Gong Y, Wang M, He J. (2016) The behavior of hydrophobic-core/hydrophilic-shell structured microgels at an interface: from Mickering emulsion to colloidosomes with dual-level controlled permeability[J]. Rsc Advances, 6(97), 95067-95072.

[23] Braz L, Gheyi R, Mongiovi M, et al. (2016) A change-centric approach to compile configurable systems with #ifdefs. Acm Sigplan Notices, 52(3), 109-119.

[24] Samain J, Carofiglio G, Muscariello L, et al. (2017) Dynamic Adaptive Video Streaming: Towards a systematic comparison of ICN and TCP/IP. IEEE Transactions on Multimedia, 19(10), 2166-2181.

[25] Arnavut Z, Kamal A E. (2018) Special section on enabling technologies for network-based applications. Computers & Electrical Engineering, 66, 48-49.